Identification of execution modes for real-time systems using cluster analysis

Kevin Zagalo, Liliana Cucu-Grosjean, Avner Bar-Hen

Identification of execution modes for real-time systems using cluster analysis

Kevin Zagalo
Inria
Paris, France
kevin.zagalo@inria.fr

Liliana Cucu-Grosjean
Inria
Paris, France
liliana.cucu@inria.fr

Avner Bar-Hen
Cnam
Paris, France
avner@cnam.fr

***Abstract*—Estimating bounds for the execution or response times of a task is a central concern for real-time designers. Several solutions exist, and probabilistic approaches estimate such bounds by building appropriate probability distributions. Those methods are safe, but they may be pessimistic and rely on strong hypothesis such as independence between tasks.**

The worst case execution times of tasks are hard to estimate because their measurements are usually disturbed by the system itelf. In general measures are done in *isolation*, however dependencies between tasks are rarely modeled in that case. By isolation, we mean that a task (or program) is executed without any type of interference coming from other executed tasks.

In this paper we propose a statistical analysis of measured response times based on clustering analysis, *i.e.*, building classes of response times that may identify executing modes for a given set of tasks. This work is a first step towards a multivariate analysis that may explicitly identify dependence structures.

***Index Terms*—Gaussian mixtures, real-time systems, timing analysis.**

I. Introduction

The natural design process of a real-time system is based on building the *schedules* of its tasks. A schedule is a sequence of tasks instances for which the execution instants are defined. These instances are activated according to some time constraints. If there exists at least one schedule with all time constraints satisfied, then the set of tasks is said *schedulable*.

In this work, we consider a system proven to be schedulable and in which tasks share some resources and some global variables. Their executions are not independent, and they interfere with each other. Indeed, a task instance leaves the cache memory in a certain state, and that state will generate a certain amount of cache misses for the other tasks instances. In the same way, the operating system overhead can act differently in various configurations of the system. In this work, we consider all those interferences as *noise*.

The contribution of this paper is to provides a method to build classes of noise, each corresponding to a *state* of the system. We believe that our method could be use to adapt deadlines and/or execution times to each *mode* of the system.

More formally, for a given mode m , we want to be able to *decompose* an execution/response time x in term of a *statistical mode* $\mu^{(m)}$ *i.e.*, the value that appears most often, and a noise B_m , allowing to write

$$x = \mu^{(m)} + B_m$$

where B_m follows a normal distribution with zero mean and deviation parameterized by $\sigma^{(m)}$.

We call *execution mode* or *mode* a state of a real-time system, in which there is a *relevant* general behavior. The *relevant* term have no formal definition yet, nevertheless model-based clustering provides some tools that we intend to use later to give more intuition on how we can measure this relevance.

For example, when a drone is in a take-off *mode*, the tasks defining a flight management functionality use the same values for global variables defining that particular state of the systems, and, as consequence, a certain *joint behavior* in that state. Unlike the given definitions in [1], [2], a mode is more of an abstract state, in the sense that it may not always correspond to a physical state, neither a criticality level. It can for example correspond to a certain configuration in which cache misses are high, or in which operating systems are more invasive.

We think that modes can be well described by the execution times associated. Thus, our goal is to provide a method to identify modes of a real-time system just by looking at response times. Our purpose is to extract, represent and exploit those sequences of execution modes, and to show that supposing dynamic behavior is a way to model dependencies and introduce predictability in real-time systems.

II. Related work

An important step in building a schedule of a system is estimating the execution times of its tasks. In this section, we provide a review of the main results of probabilistic timing analysis on estimating such bounds.

The existing work is mainly concentrated on bounding estimators of worst case execution times, or response times of independent tasks, and their evolution while some parameters change.

In [3], the authors show that a probabilistic approach may approximate cache miss rates, on both single core and multicore systems. Cache miss being a factor of randomness, one has to take it into account when studying a system built on processors with cache memories. Our study does not model those as [4], but current results provide arguments to introduce them later.

The cited analyses are *static*. By static we mean that the analysis consider models for the program and the processor without measuring any execution of the programs on the processors.

Some other methods referred to as *measurement-based probabilistic time analysis*, *e.g.* [5]–[10], find the best-fitted parameters for extreme value distributions, computing maximum likelihood estimators and non-parametric tests to do so. One may find a complete survey of these results in [11], while open problems are underlined in [12].

Some work refers to mode changes as [1], [2], however to our best knowledge, no existing work refers to the identification of execution modes of the studied real-time system by using statistical analysis.

III. Task model

In this section we define our model of tasks (Section III-A) and a motivation of clustering methods on response time measurements (Section III-B) .

A. Response time measurements

We consider a set of N tasks, scheduled on one processor according to a preemptive fixed priority policy, Rate Monotonic. We consider n measurement points. For any task $\tau_j, \forall j \in \{1, \cdots, N\}$, we denote by $r_{i,j}$ the measured value of the response time of τ_j at a measurement point i with $1 \leq i \leq n$. This notation is possible by keeping track of the schedule of all tasks.

A reader may notice that our notation is placing the number i of the measurement points in first position within the index of the response time $r_{i,j}$ and the number j of the task in the second position. This notation is explained by the fact that we group the response time values within our solution by measurement points presented in Section IV.

We provide an example in Fig. 1, where the evolution (*i.e.*, variation over time) of three response times is presented (the tasks belong to the case study described in Section III-B).The green curve corresponds to the highest priority task and the red to the lowest priority task. The horizontal axis indicates the time evolution from left (beginning of the schedule) to right (the end of the schedule). The vertical axis indicates the value of the response time.

B. Motivational case study

We motivate our proposed solution by using response time measurements of programs of an autopilot. More precisely, in this paper the measurements are done on a modified version of the autopilot PX4 v1.9.2 [1] in the context of the CEOS[2] project. The programs are executed on the micro-controller Pixhawk [3], running on the operating system NuttX. Its CPU is a single core processor ARM Cortex M4. The studied data corresponds to hardware in the loop simulations of drone flights.

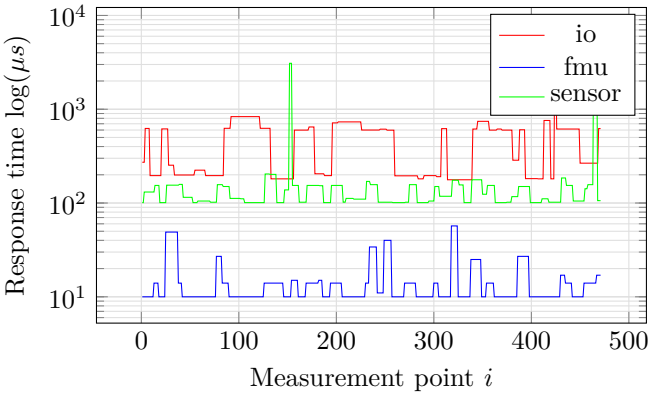


Fig. 1: Evolution of response times in a drone's autopilot.

For the sake of the presentation we select four tasks (a.k.a, modules within the autopilot context): *sensors*, *fmu*, *io* and *attitude* [4]. Fig. 1 corresponds to the response times of three tasks of PX4 over a small interval of time, Fig. 2 corresponds to the whole flight for one task (*attitude*), Fig. 3 corresponds to the distribution of the module *altitude* and Fig. 4 corresponds to a projection of the four modules.

From these figures and especially Fig. 2, we may notice a mixture of several distributions, and an evolution in what we may consider to be three clusters. We chose the term *mode* because it has both a statistical and a real-time systems meaning. It describes the same concept, even though the statistics provide a more abstract representation of its real-time equivalent. Obviously our work make these two concepts converge. Modes describe several types of *interference* and *backlogs*, quantifying dependencies, and more generally evolution of tasks in a joint behavior.

Our purpose is to study the multiple causes of this evolution and we want our method to be as general as possible. In order to obtain such generality, different sources of variability will be considered as one noisy variable. In the next section we propose a possible approximation for the observed distributions associated to modes.

IV. Gaussian mixture models

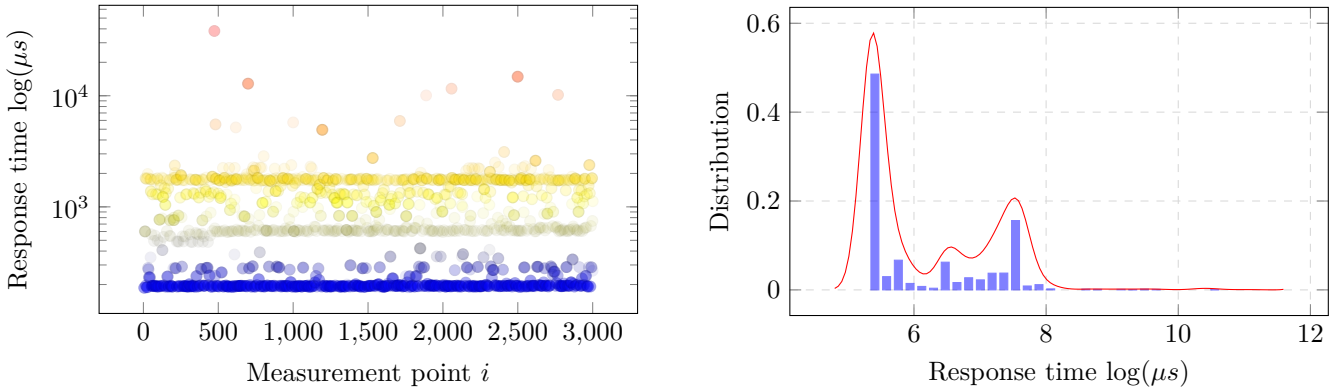
The purpose of clustering is to classify the response time measurements into several abstract classes corresponding to execution modes, without requiring any additional information. For that, we consider as a first possible approximation the log-normal distribution, *i.e.*, log-response times are considered normally distributed for each execution mode.

[1]https://github.com/PX4/Firmware/tree/v1.9.2

[2]https://www.ceos-systems.com/

[3]https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk.html

[4]https://dev.px4.io/master/en/middleware/modules_main.html



(a) log-Response times against activation times.

(b) Distribution of log-response times and GMM estimation.

Fig. 2: Variation of the response time of a the *attitude* task, and its distribution.

Assuming that each of these distributions can be approximated by Gaussian mixture (thus *multimodal*), several models may indicate to which component of the mixture a measurement is more likely to belong to. We use here the Gaussian mixture model (GMM).

Let $\{\mathbf{r}_i\}_{i=1}^n$ be a N-multivariate data set obtained, in our case, by grouping the response time values of the tasks by measurement points. We use GMM to identify M Gaussian distributions for which our data are more likely to be fitted. More formally, we are looking for weight coefficients $\{\pi_m\}_m$, modes [5] $\{\boldsymbol{\mu}_m\}_m$, and covariance matrices $\{\boldsymbol{\Sigma}_m\}_m$ such that the *likelihood*

$$\mathcal{L}\left(\{\pi_m, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\}_m\right) = \prod_{i=1}^{n} \sum_{m=1}^{M} \pi_m \phi(\mathbf{r}_i; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \qquad (1)$$

is maximal for a fixed sample $\{\mathbf{r}_i\}_{i=1}^n$, and where $\phi(\mathbf{r}_i; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$ is the density function of a multivariate Gaussian distribution defined by

$$(2\pi)^{-\frac{N}{2}} \det(\boldsymbol{\Sigma}_m)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{r}_i - \boldsymbol{\mu}_m)^\top \boldsymbol{\Sigma}_m^{-1}(\mathbf{r}_i - \boldsymbol{\mu}_m)\right\}$$

Such parameters are called maximum likelihood estimators. They are, by construction, the optimal parameters providing a Gaussian mixture such that we could simulate another sample without doing any measures, see Fig. 2b and Fig. 3b. We compute them by using the *Expectation-Maximization* algorithm on (1). See [13] for more details.

In our case, this previous definition supposes that the observations $\mathbf{r}_i$ are *conditionally* independent. By that, we mean that if we know which task is activated at measurement i, there is statistical independence. Though, it does not imply that we require the tasks to be functionally independent.

An example is shown in Fig.1. Each step i correspond to a jump of one of the curves, each j to a particular curve, meaning that $r_{i,j}$ corresponds to a constant step of the task j, and $\mathbf{r}_i = [\tilde{r}_{i,1}, \ldots, \tilde{r}_{i,N}]^\top$, where $\tilde{r}_{i,j} = \log(r_{i,j})$. In other words, we see a response time as if it was a signal.

An example of a task set composed of only one task is given in Fig. 3. Each color represents a mode identified by GMM. The histogram and the dotted lines represent the empirical distributions, while the continuous ones represent the estimated distributions of each cluster. The frame on the right shows that our estimators are tight to the data, meaning that our Gaussian assumption is reasonable on this case. To get a more precise idea of the representation in multivariate analysis, a 3d-projection of the modes identified by the GMM is provided in Fig. 4. This representation is a projection : normally the estimated distribution lives in a $N = 4$ dimensional space.

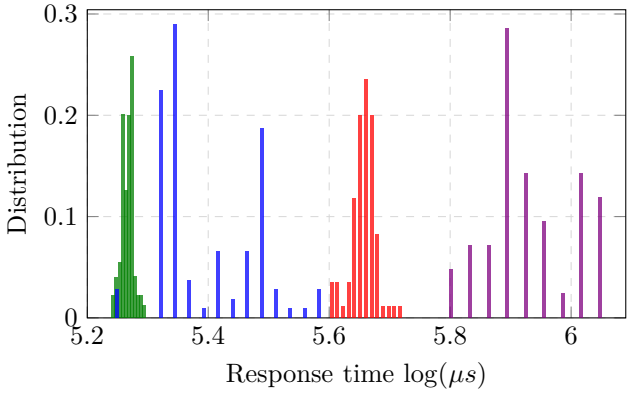
V. Conclusions and future work

In this paper we present first encouraging results confirming that capturing in $\boldsymbol{\Sigma}_m$ the dependencies between consecutive executions of tasks may be possible by identifying execution modes. We propose a first statistical model of such modes by assuming that the distribution associated to each mode is Gaussian and estimated their parameters with maximum likelihood estimators. By applying this reasoning to response time measurements of an autopilot drone, we validate our initial hypothesis of associating Gaussian distributions to execution modes response times.

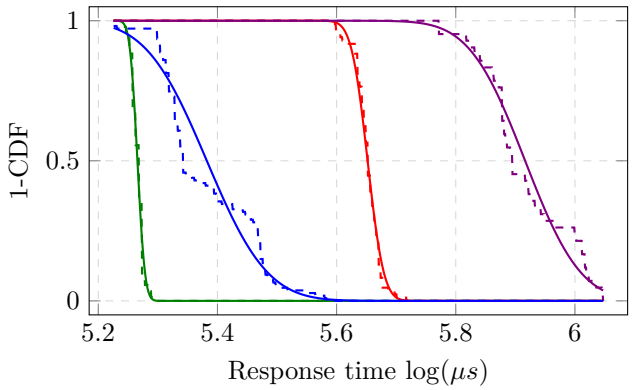
One challenge underlined by this paper is the identification of the best number M of modes from measurements analysis and the skeleton of the programs, which will indeed show that the cluster are *relevant*. Including the values for the input variables of each task may, also help to identify the number of nodes. This latter working hypothesis is currently under study for our data from drone flights.

Applying GMM opens the way to a fair amount of new analyses. Our long term objective is quantifying dependencies using *copulas* (see some insight in [14]), that may encapsulate the dependence structures of each mode. This solution could allow to make more accurate estimations of response times.

[5] In this case, we will refer to the statistical definition of *modes*.



(a) Classified log-response times histograms normalized for each mode.



(b) Empirical (dotted) and estimated (continuous) distribution functions.

Fig. 3: Univariate 4-clustering on the *attitude* task. Each color corresponds to a mode.

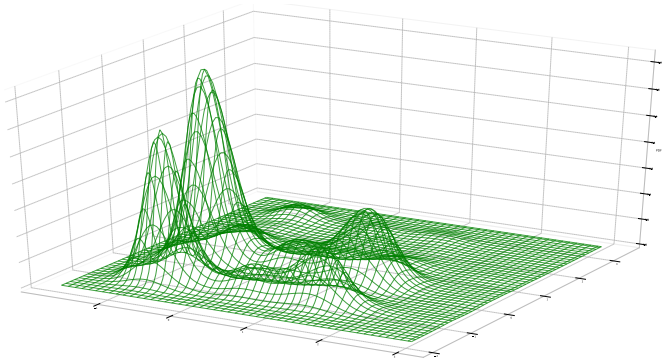


Fig. 4: Gaussian mixture 3d-representation of four programs' log-response times.

Another long term objective is the usage of modes to describe the system's state in a dynamical way. We believe that Markov models on modes space states can be used to encapsulate the statistical behavior of each mode.

Last and not least, estimating worst case response times by applying the measurement-based probabilistic timing analysis method remains possible. The distribution of the maximum of a normal distribution can be approximated by a Gumbel distribution, and indeed, using the classical method coupled with a Gaussian copula for each mode could be interesting to propagate those dependencies to existing results.

References

[1] P. Pedro and A. Burns, "Schedulability analysis for mode changes in flexible real-time systems," in *Proceeding. 10th EUROMICRO Workshop on Real-Time Systems (Cat. No. 98EX168)*. IEEE, 1998, pp. 172–179.

[2] J. Real and A. Crespo, "Mode change protocols for real-time systems: A survey and a new proposal," *Real-time systems*, vol. 26, no. 2, pp. 161–197, 2004.

[3] R. I. Davis, J. Whitham, and D. Maxim, "Static probabilistic timing analysis for multicore processors with shared cache," in *Proceedings of the Real-Time Scheduling Open Problems Seminar (RTSOPS)*, 2013, pp. 3–5.

[4] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network," in *2012 Proceedings IEEE INFOCOM Workshops*. IEEE, 2012, pp. 310–315.

[5] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quiñones, and F. J. Cazorla, "Measurement-based probabilistic timing analysis for multi-path programs," in *2012 24th euromicro conference on real-time systems*. IEEE, 2012, pp. 91–101.

[6] D. Maxim and L. Cucu-Grosjean, "Response time analysis for fixed-priority tasks with multiple probabilistic parameters," in *2013 IEEE 34th Real-Time Systems Symposium*. IEEE, 2013, pp. 224–235.

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

[8] F. J. Cazorla, T. Vardanega, E. Quiñones, and J. Abella, "Upper-bounding program execution time with extreme value theory," in *13th International Workshop on Worst-Case Execution Time Analysis*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[9] F. Wartel, L. Kosmidis, C. Lo, B. Triquet, E. Quinones, J. Abella, A. Gogonel, A. Baldovin, E. Mezzetti, L. Cucu *et al.*, "Measurement-based probabilistic timing analysis: Lessons from an integrated-modular avionics case study," in *2013 8th IEEE International Symposium on Industrial Embedded Systems (SIES)*. IEEE, 2013, pp. 241–248.

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

[11] R. I. Davis and L. Cucu-Grosjean, "A survey of probabilistic timing analysis techniques for real-time systems," *Leibniz Transactions on Embedded Systems*, vol. 6, no. 1, pp. 03–1, 2019.

[12] 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," *IEEE Embedded Systems Letters*, vol. 9, no. 3, pp. 69–72, 2017.

[13] C. Bouveyron, G. Celeux, T. B. Murphy, and A. E. Raftery, *Model-Based Clustering and Classification for Data Science: With Applications in R*. Cambridge University Press, 2019, vol. 50.

[14] G. Bernat, A. Burns, and M. Newby, "Probabilistic timing analysis: An approach using copulas," *Journal of Embedded Computing*, vol. 1, no. 2, pp. 179–194, 2005.