Estimating Execution Time Probability Distributions in Component-based Real-Time Systems

Ricardo Perrone[1], Raimundo Macêdo[1], George Lima[1] , Verônica Lima[2]

[1] Distributed Systems Laboratory (LaSiD)
Computer Science Department - UFBA

[2]Statistics Department - UFBA

{perrones,macedo,gmlima,cadena}@ufba.br

Abstract. *Many component-based real-time systems have recently been proposed as a solution to modular and easily maintainable distributed real-time systems. This paper proposes a methodology for estimating probability distributions of execution times in the context of such component-based distributed real time systems, where no access to component internal code is assumed. In order to evaluate the proposed methodology, experiments have been conducted with components implemented over CIAO, and the related probability distributions estimated. The collected experimental data show that the proposed approach is indeed a good approximation for component execution time probability distributions.*

1. Introduction

The requirements of modern real-time distributed systems, such as more flexibility, interoperability, and cost savings, have motivated both the use of software-intensive solutions and the exploitation of hardware and software COTS (Commercial Off-The-Shelf). In this context, distributed software components appear as a promising technology, since their modular and uncoupled approach used for implementing and combining components leads to reusable and easily maintainable systems. Therefore, the use of real-time component-based middleware for the construction of real-time distributed systems has deserved a great deal of attention from the research community in the last years [Wang et al. 2004].

Whereas it is widely acknowledged that applications can benefit from the adoption of such modular solutions, where COTS can be integrated into a system provided that their implementations comply rigorously with the specified interfaces, the lack of knowledge of the component implementation (black-box approach) makes it difficult to verify the timeliness guarantees of the whole system - which may turn out to be a barrier to use such an approach in the real-time arena.

A conventional way of verifying the timeliness guarantees of a safety-critical real-time system is to calculate the feasibility of meeting the deadline of each of the related tasks, even when all tasks are activated concurrently (worst-case response times). To realize that analysis, one must first calculate the worst case execution

time (WCET) of each task in isolation and then combine such WCETs in formulas that capture the worst-case response times for each task. In order to precisely calculate the WCET of a given task, one must take into account each instruction execution time of the task worst case execution path (moreover, variations of instruction execution times due to mechanisms such as memory caches and pipelines should be also considered).

Unfortunately, when COTS components are used, it is not always possible to apply such an approach (when the component code is not available, but just its interface). Hence, such COTS-based real-time systems have limited applicability on safety-critical applications where missing deadlines can cause great losses. On the other hand, there are other real-time scenarios, such as multimedia, telecommunication, and some industrial applications, where missing a deadline causes only a quality of service degradation, but it is still tolerable given that the probability of such misses is below a certain limit. Therefore, finding alternative ways of estimating response times of such COTS-based real-time systems is an important challenge to be faced by the research community.

This paper tackles this challenge by proposing and validating a methodology for estimating probability distributions of execution times in the context of a component-based distributed real-time system. These estimations can help designers to verify the timeliness behavior of component-based systems by applying alternative models of timeliness analysis [Kim et al. 2005, Manolache et al. 2001]. The ultimate motivation of the proposed approach is to apply it to estimate response times of services developed in the context of ARCOS, a component-based framework implemented atop CIAO and devoted to the construction of industrial control and supervision distributed systems [Andrade and Macêdo 2007].

In order to evaluate the proposed methodology, experiments have been conducted with components implemented over CIAO, and the related probability distributions estimated. The analysis of performance data shows that the proposed approach is indeed a good approximation for estimating component execution time probability distributions.

The remainder of this paper is structured as follows. In section 2 related work is discussed. In section 3 the assumed system model and the new approach for estimating the component execution time probability distributions are presented. In section 4, a case study is used to illustrate and validate the efficacy of the presented approach. Finally, in section 5 conclusions are drawn and future work pointed out.

2. Related Work

Most work in the filed of time analysis of real-time systems is on estimating WCET and the related techniques can be broadly divided into static, probabilistic, and hybrid. The publication list on static analysis is considerably large where the focus is on analyzing the execution paths of the application code to derive the values of WCET for a given hardware platform. There are some recent work that apply such an approach to component-based real-time systems [Estévez-Ayres et al. 2005, Ballabriga et al. 2007, Fredriksson 2006, Moss and Muller 2004]. Usually static approaches produce over-pessimistic estimations and/or need the knowledge of the

application code. This has motivated the development of probabilistic and hybrid estimation techniques, which are more related to our work.

One of the first results on probabilistic WCET estimation was by [Edgar and Burns 2001]. By carrying out measurements, execution time values are sampled based on which WCET is estimated using Extreme value statistics. The authors assume that the sampled data follow a Gumbel distribution and are independent and identically distributed. Although this is a black-box approach, the assumption on data independence may limit the applicability of this approach for practical systems. The work by [Bernat et al. 2002] can be seen as a hybrid of the static and probabilistic approaches. They determine the execution time profiles of blocks of code by, for instance, measurements. Execution time profiles are actually a table that contains the execution time observed and its frequency. Then, they obtain a WCET value (with an associated probability) by convolving and combining these profiles. Certain data and execution paths dependencies are taken into consideration, although the fine-grained instrumentation of application code is necessary. A scheme to store execution time profiles of component-based systems is also presented by [Nolte et al. 2003]. They argue that the components of the system should keep monitoring themselves. Based on this scheme, a hybrid WCET estimation technique is proposed [Möller et al. 2005].

Instead of estimating a value for WCET, this paper focus on deriving the execution time probability distributions in component-based systems. Such distributions have been used in modern real-time scheduling mechanisms [Kim et al. 2005, Manolache et al. 2001]. Similar to probabilistic and hybrid approaches, the proposed estimation method is based on measurements. However, we do not require the instrumentation of application code to carry out the measurements. We assume, for example, that there may be components developed by third-part development teams, so the knowledge of application code may not be available. This makes static or hybrid estimation approaches clearly unsuitable. Moreover, unlike some probabilistic approaches [Edgar and Burns 2001], we do not assume any particular distribution and may consider complex systems, where data independence may not be ensured.

3. An Approach for Estimating the Execution Time Distribution

It is assumed a system made up of components that communicate through the related interfaces, known as *ports*. Such components are implemented by processes and communicate by exchanging messages through communication channels. Processes and channels are assumed to be timely: process steps and message delivery are carried out within bounded times. However, such bounds or deadlines may be missed from time to time. Hence, it is assumed a soft real-time system.

The proposed method is based on measurements, where a *monitor component* is responsible for measuring the response-time for executing a service provided by another component. It also measures the response time to call a null-code service, which is called round-trip time. Thus, response time (R) and round-trip time (RT) are defined as the variables of interest. It is important to emphasize that no knowledge about the application code is being used. Application code is seen as black-box

entities.

The main goal of the proposed approach is to estimate the execution time probability distribution, where the execution time of a component service can be seen as the variable $C = R - RT$. To realize that, the monitor component is designed to measure several values of R and RT. Before detailing the approach, some definitions are introduced in the following.

Let S_r and S_{rt} denote two collections of measured values that contain response times and round-trip times obtained by the monitor, respectively. The elements of a collection are not necessarily distinct. The set of distinct values in a collection S is given by $D(S)$. The number of times a value v is observed in a collection S is denoted by $\eta(v, S)$. Also, define the function $f(v, S)$ that gives the relative frequency of value v in a collection S. For example, if the monitor measured the same value $r \in S_r$ three times in 10 measurements, $f(r, S_r) = 0.3$. More precisely,

$$f(v, S) = \frac{\eta(v, S)}{\sum_{\forall u \in D(S)} \eta(u, S)} \tag{1}$$

It is possible to derive the probability distribution of C by computing the joint probability distribution

$$P(C = c) = \sum P(R = r, RT = rt) \qquad \forall r \in S_r, rt \in S_{rt} : c = r - rt \tag{2}$$

Nevertheless, not all combinations of values in $r \in S_r$ and $rt \in S_{rt}$ are possible since it is known that $P(C \leq 0) = 0$. More generally, assuming that there must exist a lower bound $c_{min} > 0$ on the execution time, it is necessary to consider those combinations that satisfy $r - rt \geq c_{min}$. It is important to point out that non-valid values of $r - rt$ may appear because the measurements in S_r and S_{rt} are carried out independently. Thus, let us define a collection of possible values for C as

$$S_c = \{r - rt | r - rt \geq c_{min}, r \in S_r, rt \in S_{rt}\} \tag{3}$$

Assuming $c_{min} \approx 0$ is simple but non-realistic. In order to provide better estimations when deriving the probability distribution of C, it is convenient to use better estimations for c_{min}. Since larger values of execution times are of more interest, the following procedure can be used to estimate c_{min}:

1. Find the minimum value $rt_u \in S_{rt}$ such that the probability $P(RT \leq rt_u) \geq p$, where p, $0 < p < 1$, is a threshold on the desired probability given by the user and $P(RT \leq rt_u)$ can be computed as

$$P(RT \leq rt_u) = \sum_{\forall rt \in D(S_{rt}) : rt \leq rt_u} f(rt, S_{rt}) \tag{4}$$

2. Find the minimum value $r_{mim} \in S_r$ such that $r_{min} - rt_u > 0$.
3. Define $c_{min} = r_{min} - rt_u$.

Once c_{min} is found, collection S_c and function (1) produce the desired distribution of C. In order to illustrate the proposed estimation approach, let

$S_r = \{1, 2, 3, 6, 6, 7\}$ and $S_{rt} = \{1, 2, 3, 3, 3, 4\}$. Also, consider the desired threshold $p = 0.8$. In this case, $rt_u = 3$ since $P(RT \leq 2) = 1/3$ and $P(RT \leq 3) = 5/6 \approx 0.83$. Thus, $r_{min} = 6$ and so the values of C cannot be less than $c_{min} = 3$. Therefore, $S_c = \{3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 6\}$.

It is important to emphasize that the above definition of rt_u is used to derive the lower bound c_{min}. Discarding values of C below this bound is useful since it improves the quality of the estimation and does not compromise the estimation procedure. Indeed, the discarded values are too low and can be only observed for too high and very unlikely round-trip time values. Since it is of more interest to better estimate higher values of C, this discarding strategy is also a safe approach. The next section shows a case study that illustrates the effectiveness of the proposed estimation method.

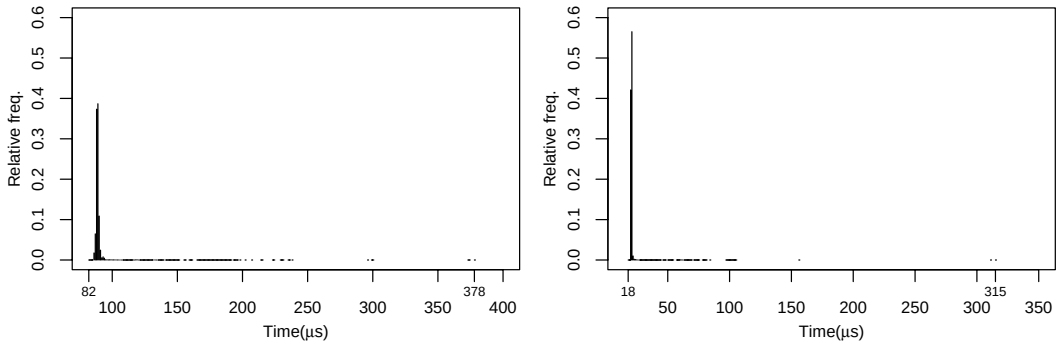
4. Case Study and Evaluation

We applied the proposed method in an experimental test-bed. The component model used was CIAO, which is in line with the CORBA CCM lightweight specification [OMG 2006]. CIAO provides suitable execution environments to host the components, named *containers*. The services provided by the containers include security, data persistence, and component life-cycle operations. The default configuration of CIAO was used during installation, and the following ORB parameters configured: FIFO scheduling policy, direct priority mappings to native priorities, policy of priorities declared by server and static threads for each component installed. We configured this component platform (version 0.6.0) on top of Linux 2.6.20-16 running on a 1.83 GHz Intel Centrino Duo machine with 1GB of RAM and 2 MB cache. This run-time environment was set up with `maxcpus` = 1 (only one core was considered) and `runlevel` = 1 (only basic Linux kernel services were enabled). These setup parameters were necessary to minimize possible interferences during measurements.

We implemented a monitored and the monitor components, both hosted in the same container. The monitored component provides a service that simulates a kind of data analyzer, which keeps sampled data records and provides predictions on the analyzed data behavior. Since the semantics of this component service is not relevant here, it will not be further described. To obtain round-trip times, one third component was hosted latter in the same container with no internal code implementation (*null-service*). In order to minimize the interference in the measured times of the monitored component, its execution priority was set to the maximum value.

To carry out the necessary measurements, the monitor did n calls through the connections with the analyzer and the *null-service*. For each completed call, the times measured were recorded. Both services use an identical port (named facet in the CIAO terminology) and the calling parameters were randomly generated by the monitor based in a range of values of interest. The measurements performed by the monitor were not correlated whatsoever. In our experiments, first the values of variable R were measured and then those of RT. The calling frequencies used in those measurements were set to 25MHz, 50MHz, 75MHz and 100MHz so that there were $n/4$ measurements for each calling frequency. In order to measure these

values, we used the class *High Resolution Timer* provided by the framework ACE [Schmidt and Huston 2003], on top of which CIAO was implemented. Thus, time was measured in microseconds (μs). The number n of measurements should be defined by the user and must be large enough so that the collected values of R and RT are representative. In our experiments, $n = 2 \times 10^6$.



(a) Response Time (b) Round Trip

Figure 1. Probability distributions of R and RT.

Figure 1 plots the probability distributions derived from the collected values in S_r and S_{rt}. As can be seen, these distributions exhibit similar behaviors although the dispersion of S_r is higher. This is due to the fact that they are more subject to higher interference when compared to S_{rt}. Given that we did not use a real-time operating system, the observed dispersion is not so high.

Table 1. Descriptive analysis.

	R	RT	C	C'
Mode	89	21	68	66
Median	89	21	68	66
Average	88.8	20.62	68.20	65.87
Std. dev.	2.317528	1.143541	2.381033	1.699584
min.	82	18	60	61
max.	378	315	360	352

Given the collections S_r and S_{rt}, the proposed estimation method was carried out. Table 1 shows some descriptive analysis for the obtained results. In order to evaluate the proposed method, we also measured the execution time of the code of the monitored component service. Thus, it was necessary to instrument the monitored service internal code. Measured execution times are represented by variable C' in the table. Also, it can be noted that there is a higher standard deviation for R when compared with RT. This is caused by the already mentioned higher dispersion of S_r as compared to S_{rt}.

Figures 2 and 3 illustrate better the effectiveness of the proposed approach. As can be seen, the probability distributions of C and C' and their respective mass functions are very similar. In figure 2 the distribution of C exhibits a right-shifted curve when compared with C' distribution. It is due to the influence

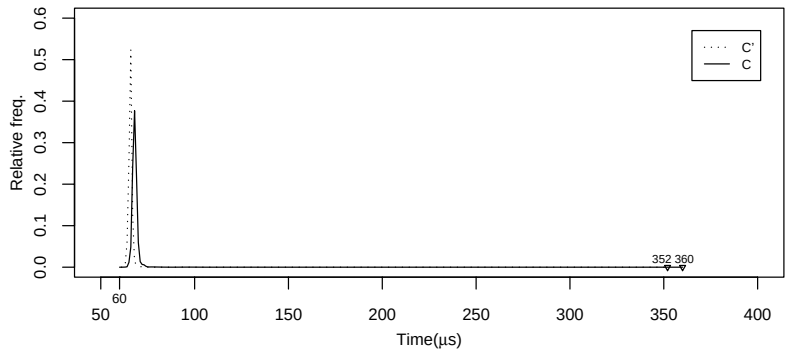


Figure 2. Distributions of variables C and C'

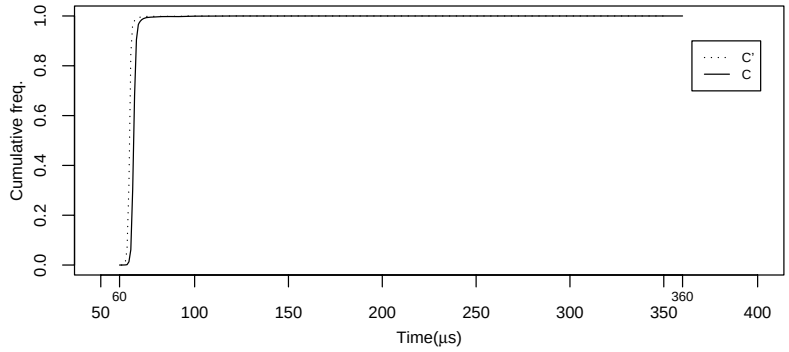


Figure 3. Cumulative Distribution of variables C and C'.

caused by the joint probability computation when the equation (2) was applied - i.e. $P\left(r = 89 \wedge rt = 21\right) \cong 21.86\%$. However, this right-shift does not compromise the results since the estimated probabilities in C distribution are related to time values that are higher than time values related to the curve of probabilities in C' distribution. The same occurs with comulative distribution of C in figure 3, that exhibits a right-shift (e.g. $P\left(C \leq 100\right) = 99.93\%$ and $P\left(C' \leq 100\right) = 99.99\%$).

Finally, it is should be noticed that the estimated probability distribution provides a simple way of estimating probabilistic WCET, for instance, $360\mu s$ with probabilistic guarantee $P\left(C < 360\right) \cong 99.999\%$). Likewise, the probability distribution produced can be used in probabilistic models like in [Kim et al. 2005] or in [Manolache et al. 2001] where the ratio of missed deadlines are computed per task, and the analysis uses a pseudo-continuous distribution based on probabilistic density curves.

5. Conclusion

This paper proposed and evaluated a methodology for estimating component execution time probability distributions of COTS-based real-time systems. To evaluate the proposed methodology, component execution time probability distributions were estimated from interactions via the component interface and from instrumented code inside the component, bypassing the component interface. The comparison of these distributions showed that the proposed approach is indeed a good approximation for estimating component execution time probability distributions. Future

work includes complementing the proposed methodology with mechanisms to derive probabilistic worst-case execution times of components and related probabilistic scheduling feasibility tests.

References

Andrade, S. and Macêdo, R. (2007). Engineering components for flexible and interoperable real-time distributed supervision and control systems. In *12th IEEE Conf. on Emerging Technologies and Factory Automation*, Patras - Greece.

Ballabriga, C., Cassé, H., and Sainrat, P. (2007). Wcet computation on software components by partial static analysis. Junior Researcher Workshop on Real-Time Computing. IRIT - Université de Toulouse, France.

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

Edgar, S. and Burns, A. (2001). Statistical analysis of wcet for scheduling. In *22nd IEEE Real-Time Systems Symposium*, page 215, Washington, DC, USA.

Estévez-Ayres, I., García-Valls, M., and Basanta-Val, P. (2005). Enabling wcet-based composition of service-based real-time applications. *SIGBED Review*, 2:25–29.

Fredriksson, J. (2006). Increasing accuracy of property predictions for embedded real-time components. In *18th Euromicro Conf. on Real-Time Systems*.

Kim, K., Diaz, J. L., Lopez, J. M., Bello, L. L., Lee, C.-G., and Min, S. L. (2005). An exact stochastic analysis of priority-driven periodic real-time systems and its approximations. *IEEE Trans. Comput.*, 54(11):1460–1466.

Manolache, S., Eles, P., and Peng, Z. (2001). Memory and time-efficient schedulability analysis of task sets with stochastic execution time. In *13th Euromicro Conference on Real-Time Systems*, page 19, Washington, DC, USA.

Möller, A., Peake, I., Nolin, M., Fredriksson, J., and Schmidt, H. (2005). Component-based context-dependent hybrid property prediction. In *Workshop on Dependable Software Intensive Embedded systems*.

Moss, A. and Muller, H. (2004). Model generation for temporal properties of reactive components. In *1st International Workshop on Software Analysis and Development for Pervasive Systems*, pages 12–19.

Nolte, T., Möller, A., and Nolin, M. (2003). Using components to facilitate stochastic schedulability analysis. In *24th IEEE Real-Time System Symp - WiP Session*, Cancun, Mexico.

OMG (2006). *CORBA Component Model Specification*.

Schmidt, D. C. and Huston, S. D. (2003). *C++ Network Programming: Systematic Reuse with ACE and Frameworks*. Addison-Wesley Longman.

Wang, N., Gill, C., Subramonian, V., and Schmidt, D. C. (2004). *Configuring Real-Time Aspects in Component Middleware*, pages 1520–1537. SpringerLink.