Comparison of two worst-case response time analysis methods for real-time transactions

A. Rahni, K. Traore, E. Grolleau and M. Richard
LISI/ENSMA
Téléport 2, 1 Av. Clément Ader
BP 40109, 86961 Futuroscope Chasseneuil Cedex
{rahnia,traore,grolleau,richardm}@ensma.fr

Abstract

This paper presents a comparison of two worst case response time analysis methods in the context of transactions. In the general context of tasks with offsets (general transactions), only exponential methods are known to calculate the exact worst-case response time of a task. We focus more precisely on monotonic transactions. In this context, we present the fast and tight analysis, proposed in [7, 6], and the analysis technique of monotonic transaction that we have proposed in [14]. We compare them on a case study and on several configurations generated randomly.

Keywords: Response Time, Transactions

1. Introduction

The Response-Time Analysis [1] is an essential analysis technique that can be used to perform schedulability tests. Tindell proposed in [11] a new model of tasks with offset (transactions) extending the model of Liu and Layland [5]. Since the transactions are non-concrete(the transaction release times are not fixed a priori), the main problems is to determine the worst case configuration for a task under analysis (its critical instant). Tindell showed that the critical instant for a task under analysis (τ_{ua}) occurs when one task of higher priority in each transaction is released at the same time as τ_{ua}.

An exact calculation method has been proposed in [10], but has an exponential complexity and is intractable for realistic task systems; Tindell [11] has proposed a pseudo-polynomial approximation method providing an upper bound of the worst-case response-time. Later, this approach has been improved in [4, 6, 7, 8]

In the sequel, we present the model of tasks with offsets (a.k.a. transaction), then we present the best known approximation method proposed by Turja and Nolin [8]. Section 4 presents the monotonic transactions exact analysis [13] and these two methods are used on the same tasks system. In the last section their performance are compared on randomly generated transaction systems.

2. Model of transactions

A tasks system Γ is composed of a set of $|\Gamma|$ transactions Γ_i, with $1 \leq i \leq |\Gamma_i|$ (where $|\Gamma_i|$ is the number of elements in the set Γ_i).

$$\begin{aligned} \Gamma &: \left\{\Gamma_1, \Gamma_2, .., \Gamma_{|\Gamma|}\right\} \\ \Gamma_i &: \left\{\tau_{i1}, \tau_{i2}, ..., \tau_{i|\Gamma_i|}, T_i\right\} \\ \tau_{ij} &: < C_{ij}, O_{ij}, D_{ij}, J_{ij}, B_{ij}, P_{ij} > \end{aligned}$$

Each transaction Γ_i (see figure 1) consists of a set of $|\Gamma_i|$ tasks τ_{ij} released at the same period T_i , with $0 < j \leq |\Gamma|$. Without loss of generality, we suppose that the tasks are ordered in the set by increasing offset. A task τ_{ij} is defined by : a worst-case execution time (WCET) C_{ij}, an offset O_{ij} related to the release date of the transaction Γ_i, a relative deadline D_{ij}, a maximum jitter J_{ij} (the activation time of task τ_{ij} may occur at any time between $t_0 + O_{ij}$ and $t_0 + O_{ij} + J_{ij}$, where t_0 is the release date of the transaction Γ_i, a maximum blocking factor B_{ij} due to lower priority tasks (e.g. priority ceiling protocol [9]), and P_{ij} is its priority (we assume a fixed-priority scheduling policy). The figure 1 presents an example of transaction Γ_i composed of three tasks with period $T_i = 16$.
Let us note $hp_i(\tau_{ua})$ the set of indices of the tasks of Γ_i with a priority higher than the priority of a task under analysis τ_{ua}, assuming that the priorities of the tasks are unique.

3 Fast and Tight Analysis

This method provides an efficient implementation to calculate an upper-bound of the worst-case response times [7]. The main idea is to represent the periodic interference function statically, and during

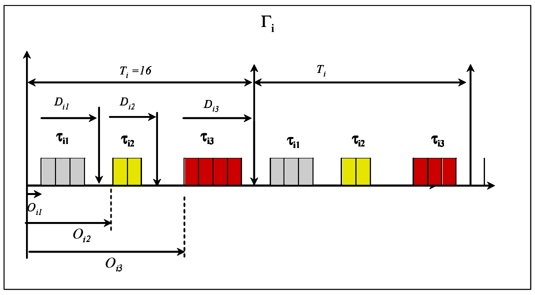

Figure 1. Example of transaction.

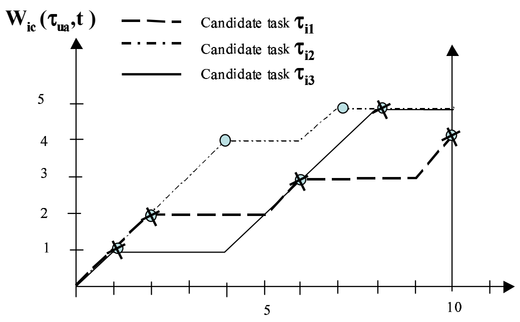

Figure 2. Interference of transaction.

the response-time calculation, to use a simple lookup function in order to compute its value. The interference imposed by the transaction Γ_i on a task under analysis τ_{ua} during a busy period of length t starting at the release of τ_{ua} and corresponding to the release of τ_{ic} is noted $W_{ic}(\tau_{ua}, t)$ (τ_{ic} is then called the critical instant candidate in Γ_i). In order to simplify, we suppose no Jitter in the transaction (i.e $J_{ij} = 0$).

$$\begin{aligned}
W_{ic}(\tau_{ua}, t) &= \sum_{j \in hp_i(\tau_{ua})} \left(\left(\left\lfloor \frac{t^*}{T_i} \right\rfloor + 1 \right) * C_{ij} - x_{ijc}(t) \right) \\
t^* &= t - \Phi_{ijc} \\
\Phi_{ijc} &= (T_i + (O_{ij} - O_{ic})) \ \% \ T_i \\
x_{ijc}(t) &= \begin{cases} 0 \text{ for } t^* < 0 \\ \max(0, C_{ij} - (t^* \ \% \ T_i)) \text{ otherwise} \end{cases}
\end{aligned}$$

Φ_{ijc} is the phase between τ_{ic} and τ_{ij}. $x_{ijc}(t)$ corresponds to the part of the task τ_{ij} that cannot be executed in the time interval of length t (note that this part is the main difference between the methods presented in [4] and [7]).
In order to find the critical instant, one would have to compare every combination of critical instant candidates, making the exact test intractable. The fast and tight analysis method consists of creating a global interference function $W_i(\tau_{ua}, t)$ for each transaction Γ_i, in choosing the maximum value of each interference function.

$$W_i(\tau_{ua}, t) = \max_{\forall c \in hp_i(\tau_{ua})} W_{ic}(\tau_{ua}, t)$$

The figure 2 shows the graphical representation of the interference of a transaction : each curve represents the interference function for each critical instant candidate. Since the transaction is in normal form, the derivative of each curve is either 0 or 1. $W_i(\tau_{ua}, t)$ that has to be computed is the maximum of all the curves. The efficient implementation proposed in [7] stores the set of points P_{ic}, where each point $P_{ic}[k]$ has an x (representing time) and a y (representing interference) coordinate. These points correspond to the convex corners of the curve $W_{ic}(\tau_{ua}, t)$.
The calculation of the upper bound on the worst-case response time R_{ua} of τ_{ua} is obtained by an iterative fix-point lookup.

$$\begin{aligned}
R^0_{ua} &= C_{ua} \\
R^{(n+1)}_{ua} &= C_{ua} + \sum_{\Gamma_i \in \Gamma} (W_i(\tau_{ua}, R^n_{ua})) \qquad (1)
\end{aligned}$$

where $W_i(\tau_{ua}, t)$ is deduced from the static representation of the transactions interferences.

3.1 Normal form of a transaction

Without loss of generality, we consider that all the tasks of Γ_i have a higher priority than the task under analysis τ_{ua}. Since some tasks of a transaction may have to overlap, issuing in an interference function which derivative would be greater than 1, a normal form of the transaction is first obtained using three operations: order, merge, and split [7]. For each critical instant candidate τ_{ic}, the transaction Γ_i is put in normal-form. We start with all the tasks numbered in increasing Φ_{ijc} (phase between τ_{ij} and τ_{ic}). Thus the task τ_{ic} in the original transaction is named τ_{i1} at the beginning of the normal-form processing.
The underlying idea behind the merge operation is that if two tasks τ_{ij} and τ_{ij+1} overlap, then the longest busy period initiated by τ_{ij} is always including the longest busy period initiated by τ_{ij+1}. The split operation is used when a part of a task has to be executed during the next period of the transaction : in this case the spilling task is splitted into two tasks. The spilling part is taken into account as a task with an offset equal to 0 in the second period of the transaction. Thus, since the tasks are numbered according to the increasing value of Φ_{ijc}, the tasks can be renumbered (ordering operation) in the second period of the transaction. These operations are used until no task in the transaction is forced to overlap on the next one, and until no task is forced to spill in the second period of the transaction.
Note that the first period of a transaction may differ in the number of tasks from the second period due to the spilling tasks.
Note that if a jitter has to be taken into account, this operation has to be done for every critical instant

candidate.

3.2 Example

We apply this method on the transaction Γ_i that contains twelve tasks with no jitter ($J_{ij} = 0$). and the task τ_{ua} with a WCET $C_{ua} = 9$ and a lower priority than all the tasks of Γ_i.

$$\Gamma_i := \{< \tau_{i1}\tau_{i2}, ..., \tau_{i12} >, 60\}$$

$$\begin{array}{ll} \tau_{i1} :< 3, 1, D_{i1}, 0, 0, 1 > & \tau_{i7} :< 2, 36, D_{i7}, 0, 0, 7 > \\ \tau_{i2} :< 4, 9, D_{i2}, 0, 0, 2 > & \tau_{i8} :< 5, 43, D_{i8}, 0, 0, 8 > \\ \tau_{i3} :< 2, 11, D_{i3}, 0, 0, 3 > & \tau_{i9} :< 3, 46, D_{i9}, 0, 0, 9 > \\ \tau_{i4} :< 3, 20, D_{i4}, 0, 0, 4 > & \tau_{i10} :< 1, 49, D_{i10}, 0, 0, 10 > \\ \tau_{i5} :< 4, 29, D_{i5}, 0, 0, 5 > & \tau_{i11} :< 4, 56, D_{i11}, 0, 0, 11 > \\ \tau_{i6} :< 5, 31, D_{i6}, 0, 0, 6 > & \tau_{i12} :< 2, 57, D_{i12}, 0, 0, 12 > \end{array}$$

Obtaining the normal-form for Γ_i for the critical instant candidate τ_{i1}: the three operations (order, merge and spill), merge τ_{i3} in τ_{i2}, τ_{i6} and τ_{i7} in τ_{i5}, τ_{i9} and τ_{i10} are merged in τ_{i8}, and τ_{i11} is merged in τ_{i12}. The last task spills in the next period, thus the second period of the transaction (and the following) will have a additional task The resulting transaction in normal-form, for the first period is:

$$\begin{array}{ll} \tau_{i1} :< 3, 0, D_{i1}, 0, 0, 1 > & \tau_{i2} :< 6, 8, D_{i2}, 0, 0, 2 > \\ \tau_{i3} :< 3, 19, D_{i3}, 0, 0, 3 > & \tau_{i4} :< 11, 28, D_{i4}, 0, 0, 4 > \\ \tau_{i5} :< 9, 42, D_{i5}, 0, 0, 5 > & \tau_{i6} :< 5, 55, D_{i6}, 0, 0, 6 > \end{array}$$

For the second period, the spilling time of the original τ_{i12} is taken into account in the first task τ_{i1} of the second period of the transaction, obtaining a WCET of 4.

$$\begin{array}{ll} \tau_{i1} :=< 4, 0, D_{i1}, 0, 0, 1 > & \tau_{i2} :=< 6, 8, D_{i2}, 0, 0, 2 > \\ \tau_{i3} :=< 3, 19, D_{i3}, 0, 0, 3 > & \tau_{i4} :=< 11, 28, D_{i4}, 0, 0, 4 > \\ \tau_{i5} :=< 9, 42, D_{i5}, 0, 0, 5 > & \tau_{i6} :=< 5, 55, D_{i6}, 0, 0, 6 > \end{array}$$

The same operation is done for all the task candidates τ_{ic} for $c = 2..12$. The upper bound of the worst-case response time is then obtained using formula 1:

$$\begin{array}{lll} \text{Iteration 0} & : & R^0_{ua} = 9 \\ \text{Iteration 1} & : & t = 9 : R^1_{ua} = 9 + 11 = 20 \\ \text{Iteration 2} & : & t = 20 : R^2_{ua} = 9 + 20 = 29 \\ \text{Iteration 3} & : & t = 29 : R^3_{ua} = 9 + 29 = 38 \\ \text{Iteration 4} & : & t = 38 : R^4_{ua} = 9 + 29 = 38 \end{array}$$

4 Monotonic Transactions

In this section we present the different steps of monotonic transaction analysis [13, 14, 12]. Monotonic transaction analysis relies on transactions for which one interference curve (for one candidate) is always greater or equal than the other curves. In this way, it's close to the concept of accumulatively monotonic generalized multiframe task sets [2]. In this case, if a transaction Γ_i is monotonic then the critical instant occurs when the task under analysis is released at the same time as the first task of the pattern of the normal form of Γ_i. Therefore, for the case where all the transactions of the task system are monotonic for a task under analysis, the computed worst-case response time is exact.

Since there is only one possible candidate in a monotonic transaction, there is only one normal-form to compute.

4.1 Finding the monotonic pattern

Let Γ_i^* be the normal form of the transaction Γ_i where $\Gamma_i^* :< \left\{\tau^*_{i1}, \tau^*_{i2}, ..., \tau^*_{i|\Gamma^*_i|}, T_i\right\}, T_i >$ and $\Gamma_i :< \left\{\tau_{i1}, \tau_{i2}, ..., \tau_{i|\Gamma_i|}, T_i\right\}, T_i >$. Let us note:

$$\alpha_{ij} = O^*_{i(j+1)} - (O^*_{ij} + C^*_{ij}) \; for \; 1 \leq j < |\Gamma^*_i|$$

$$\alpha_{i|\Gamma^*_i|} = (T_i + O^*_{i1}) - (O^*_{i|\Gamma^*_i|} + C^*_{i|\Gamma^*_i|})$$

where $\alpha_{ij} > 1$ since Γ^*_i is in normal form.
Note that since it's not necessary to statically store the interference function, there is no need to make a difference between the first and the second period of the transaction.

Γ_i is a monotonic transaction for the task τ_{ua} (we consider that all the tasks of Γ_i have a higher priority than the one of τ_{ua}) if the WCET of the tasks of Γ^*_i have decreasing values while the idle slots α_{ij} have increasing values i.e: $C^*_{i(p+1)} \leq C^*_{ip}$ for all $1 \leq p < |\Gamma^*_i|$ and $\alpha_{ip} \leq \alpha_{i(p+1)}$ for all $1 \leq p < |\Gamma^*_i|$.
Γ_i is monotonic if we can find a monotonic pattern in Γ^*_i by rotating the tasks of Γ^*_i. We know that for a monotonic pattern the first task has the highest WCET. In order to look for a monotonic pattern, we start by inventorying all the tasks with the maximum WCET. Then, we consider alternatively each of these tasks τ^*_{ik} as the first task of the transaction Γ^*_i by rotating the tasks of Γ^*_i; and we verify if the conditions of monotony (on C^*_{ij} and α_{ij}) are respected; if so, Γ_i is monotonic and τ^*_{ik} become the first task of Γ^*_i.

4.2 Example

We apply this method on the same example as the one we used for the fast and tight analysis.
We find Γ^*_i the normal form of the transaction Γ_i by applying the operations of normalization process:

$$\Gamma^*_i : \{< \tau^*_{i1}, \tau^*_{i2}, ..., \tau^*_{i5} >, 60\}$$

$$\begin{array}{ll} \tau^*_{i1} :< 6, 9, D_{i1}, 0, 0, 1 > & \tau^*_{i2} :< 3, 20, D_{i2}, 0, 0, 2 > \\ \tau^*_{i3} :< 11, 29, D_{i3}, 0, 0, 3 > & \tau^*_{i4} :< 9, 43, D_{i4}, 0, 0, 4 > \\ \tau^*_{i5} :< 9, 56, D_{i5}, 0, 0, 5 > & \end{array}$$

We have: $\alpha^*_{i1} = 5, \alpha^*_{i2} = 6, \alpha^*_{i3} = 3, \alpha^*_{i4} = 4, \alpha^*_{i5} = 4$
There is a monotonic pattern starting from task τ^*_{i3} where:

$$C^*_{i3} \geq C^*_{i4} \geq C^*_{i5} \geq C^*_{i1} \geq C^*_{i2}$$
$$\alpha^*_{i3} \leq \alpha^*_{i4} \leq \alpha^*_{i5} \leq \alpha^*_{i1} \leq \alpha^*_{i2}$$

Hence the transaction is monotonic, and the critical instant of τ_{ua} corresponds to the release of the task τ^*_{i3}, we apply the iterative fix-point lookup in order to calculate the worst-case response time of τ_{ua}.
In the case of monotonic transactions, the two methods presented provide the same exact worst-case response time with the same number of iterations in the process of calculation, because in every iteration, the task that initiates the critical instant is the same. The only difference between these two methods comes from the stage of static representation in the fast and tight analysis (stage A) and the research of the monotonic pattern for the method of monotonic transaction (stage B). Stage A for n tasks requires at least n normal-form processing, plus computing the static tables. Stage B requires only one normal-form processing operation and a linear complexity test in order to check the conditions related to C^*_i and α^*_i.

5. Performance comparison and future works

We have implemented the two methods in order to compare their respective performance. The figure 3 shows that the time used by the method proposed in [8]increases linearly with the number of transactions, while the method proposed in [14] is less sensitive to the size of the system when only monotonic transactions are involved. The tests have been led on a Pentium IV processor, for sets of 20 configurations per number of transactions. The transactions are monotonic, and contain 15 tasks, while the workload is fixed around 0.8. The random generation system is based on the UUniFast algorithm presented in [3].
The bound on the worst-case response time is the same for both methods, since at this stage, montonicity is more a characterization allowing an optimization than a method by itself (it has still to be coupled with the method of [8] because in a system of transactions, the transactions don't have to be all monotonic).
In future works, we will use the monotonicity property as a basement to introduce a new evaluation method in order to decrease the pessimism of [8] for the upper bound on the worst-case response times.

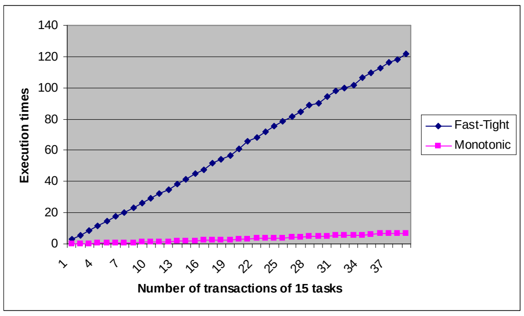

Figure 3. Execution time

References

[1] N. Audsley, A. Burns, R. Davis, K. Tindell, and A. Wellings. Fixed priority pre-emptive scheduling: An historical perspective. *Real-Time Systems 8*, pages 129–154, 1995.

[2] S. Baruah, D. Chen, S. Gorinsky, and A. Mok. Generalized multiframe tasks. *The International Journal of Time-Critical Computing Systems*, (17):5–22, 1999.

[3] E. Bini and G. Buttazzo. Biasing effects in schedulablity measures. *IEEE Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS04), Catania, Italy*, (16), July 2004.

[4] J. P. Gutierrez and M. G. Harbour. Schedulability analysis for tasks with static and dynamic offsets. *Proc IEEE Real-time System Symposium (RTSS)*, (19), December 1998.

[5] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in real-time environnement. *Journal of ACM*, 1(20):46–61, October 1973.

[6] J. Maki-Turja and M. Nolin. Faster response time analysis of tasks with offsets. *Proc 10th IEEE Real-Time Technology and Applications Symposium (RTAS)*, May 2004.

[7] J. Maki-Turja and M. Nolin. Tighter response time analysis of tasks with offsets. *Proc 10th International Conference on Real-Time Computing and Applications (RTCSA04*, August 2004.

[8] J. Maki-Turja and M. Nolin. Fast and tight response-times for tasks with offsets. *17th EUROMICRO Conference on Real-Time Systems IEEE Palma de Mallorca Spain*, July 2005.

[9] L. Sha, R. Rajkumar, and J. Lehoczky. Priority inheritance protocols : an approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, 1990.

[10] K. Tindell. Using offset information to analyse static priority pre-emptively scheduled task sets. *Technical Report YCD-182,Dept of Computer Science, Unoversity of York, England*, 1992.

[11] K. Tindell. Adding time-offsets to schedulability analysis. *Technical Report YCS 221, Dept of Computer Science, University of York, England*, January 1994.

[12] K. Traore. *Analyse et Validation des Applications Temps Réel en Présence de Transactions : Application au Pilotage d'un Drone Miniature.* Thèse, ENSMA-Université Poitiers, 2007.

[13] K. Traore, E. Grolleau, and F. Cottet. Characterization and analysis of tasks with offsets: Monotonic transactions. *Proc 12th International Conference on Embedded and Real-Time Computing Systems and*

Applications. RTCSA'06, (12), August 16-18th Sydney, Australia 2006.

[14] K. Traore, E. Grolleau, A. Rahni, and M. Richard. Response-time analysis of tasks with offsets. *12th IEEE International Conference on Emerging Technologies and Factory Automation ETFA'06*, September 2006.