Enhancement of Generalized Earliest Deadline First Policy

Mourad Kaddes[1,2], Laurent Amanton[1], Alexandre Berred[3], Bruno Sadeg[1] and Majed Abdouli[2]
[1]*LITIS, Le Havre University, 25 rue Philipe Lebon, 76600 Lehavre, France*
[2]*MIR@CL, Pôle Technologique de Sfax, Sfax University, BP 242, 3021 Sfax, Tunisia*
[3]*LMAH, Le Havre University, 25 rue Philipe Lebon 76600, Lehavre, France*

Keywords: Real-time Database System, Transaction Processing, Scheduling Policy, Success Ratio, Transaction Importance.

Abstract: Scheduling transactions in real-time database systems (RTDBSs) is more complex than scheduling tasks in real-time systems. In fact, the RTDBS must guarantee the database logical consistency, on one hand, and it must schedule the transactions in order to meet their deadlines, on the other hand. The main policy used to schedule transactions in RTDBSs is Earliest Deadline First (EDF). However, it is well-known that EDF is not efficient for scheduling transactions in overload conditions. Consequently, different scheduling protocols: AED, AEVD, APP, AEDF-Co, GEDF... were proposed to improve the system performances in firm RTDBSs, *i.e.* late transactions are considered useless. Generalized Earliest Deadline First (GEDF) is a new scheduling protocol in which transaction priority is assigned according to both deadlines and a parameter, called SPriority, which expresses the importance of transactions. In this paper, an RTDBSs analysis is presented. The accuracy of GEDF scheduling policy and the influence of database workload on the system performances is investigated. This study enabled us to describe the complete behavior of the transaction success ratio. Moreover, based on intensive simulations, we have derived the optimal values of the system parameters which improve the success ratio without modifying GEDF protocol.

1 INTRODUCTION

A real-time database systems (RTDBSs) can be considered as a combination of a traditional database system and a real-time system. RTDBSs have to satisfy both temporal consistency and logical consistency of the database, *i.e.* they must guarantee the transactions *ACID* [1] properties on one hand, and they must schedule the transactions in order to meet their individual deadlines, on the other hand (Ramamritham et al., 2004).

Different scheduling algorithms are proposed in the literature according to the type of knowledge used (see (Ramamritham et al., 2004; Han et al., 2012)). The most performance studies use EDF scheduling policy which is based on a priority assignment according to the deadlines, *i.e.* the shortest is the transaction deadline, the highest is its priority. However, with EDF, successful transactions are prioritized in favor of transactions which are close to their deadlines, *i.e.* successful transactions are not necessarily the most important transactions in the system. Moreover, it is well-known that EDF is not efficient to schedule transactions (or tasks) in overload conditions, leading to the degradation of the system performances. This is due to the assignment of high priorities to transactions that finally might miss their deadlines. These high-priority transactions also waste system resources and delay other transactions (Yu et al., 1994). To overcome these disadvantages, the study dealt with in (Haritsa et al., 1991) introduced an extension of EDF (called Adaptive Earliest Deadline: AED). AED stabilizes the overload performance of EDF through an adaptive admission control mechanism in an RTDBSs environment. In this method, the incoming transactions are assigned to either *hit* or *miss* group. Using a feedback mechanism, the capacity of the hit group is adjusted dynamically to improve the performances. Transactions in miss group receive processing only if the hit group is empty. AED is later extended by Pang et *al*. (Pang et al., 1992) where they proposed AEVD (adaptive earliest virtual deadline) protocol which adresses the fairness issue in an over-

[1]Atomicity, Consistency, Isolation, Durability

Kaddes M., Amanton L., Berred A., Sadeg B. and Abdouli M..
Enhancement of Generalized Earliest Deadline First Policy.
DOI: 10.5220/0004448802310238
In *Proceedings of the 15th International Conference on Enterprise Information Systems* (ICEIS-2013), pages 231-238
ISBN: 978-989-8565-59-4

loaded system. In AEVD, the virtual deadlines are computed based on both arrival times and deadlines. Since transactions with longer execution times will arrive earlier relative to their deadlines, AEVD can raise their priorities in a more rapid pace as their durations in the system increase. Consequently, longer transactions can exceed the priorities of shorter transactions that have earlier deadlines but later arrival times. To resolve some weaknesses of AEVD, Datta et *al*. (Datta et al., 1996) have introduced priority based scheduling policy, called AAP (Adaptive Access Parameter) method where they use explicit admission control. In (Han et al., 2012), authors have proposed a new scheduling algorithms called Adaptive Earliest Deadline First Co-Scheduling (AEDF-Co). In AEDF-Co, a dynamic scheduling approach is adopted to adaptively schedule the update and application jobs based on their deadlines. The performance goal of AEDF-Co is to determine a schedule for given sets of periodic application and update transactions such that the deadline constraints of all the application transactions are satisfied and at the same time to maximize the quality of data (QoD) of the real-time data objects.

In this paper, we propose an improvement of the GEDF scheduling policy presented by Semghouni et *al*. (Semghouni et al., 2007). To this purpose, we accurately study the GEDF scheduling policy and particularly study (*i*) the influence of the weight of the SPriority (see formula 1), (*ii*) the rank assigned to weight of transaction in the SPriority formula (see formula 3) on RTDBSs performances according to the workload, under the main concurrency and scheduling protocol 2PL-HP (Two phase locking high priority). GEDF is based on a weight technique, *i.e*. a weight is assigned to a transaction according to its importance in the system. GEDF proposes to overcome the shortcoming of EDF and is considered as a generalization of EDF due to its flexibility and its adaptability to the system workload conditions. To show the improvement of GEDF scheduling policy on RTDBSs performances, the system performances according to the transactions success ratio with different values of the weight of the Spriority and the transactions priority parameters are analyzed. Then the optimal values of these parameters according to the workload are deduced. To this purpose, we have conducted intensive Monte Carlo simulations on the RTDBSs simulator we have developed. The simulator is based on components generally encountered in RTDBSs (Kim and Son, 1996; Ramamritham et al., 2004).

The remainder of this paper is organized as follows. In Section 2, we briefly present GEDF policy and the simulator components. We present the metrics used in section 3. Section 4 presents the Monte Carlo simulation experiments and shows how we can improve the success ratio of GEDF by choosing the optimal values of influent factors according to the system status. Finally, in section 5, we conclude the paper and discuss some aspects of our future work.

2 SYSTEM MODEL AND SIMULATOR

2.1 System Model

Only firm real-time transactions are considered and classified into update and user transactions. Update transactions are periodic and only write temporal data which capture the continuously state changing environment. We assume that an update transaction is responsible for updating a single temporal data item in the system. Each temporal data item is updated following a *more-less* approach where the period of an update transaction is assigned to be more than half of the validity interval of the temporal data (Xiong and Ramamritham, 2004). User transactions can read or write non-temporal data and only read temporal data. User transactions arrive in the system according to a Poisson process with an average rate λ. The number of operations generated for each user transaction is uniformly distributed in the user transaction size interval (denoted $User_{SInerval}$). Data accessed by the operations of the transaction are randomly generated and built according to the level of data conflicts (for more detail see (Semghouni et al., 2008)).

GEDF is a dynamic scheduling policy where transactions are processed in an order determined by their priorities, *i.e*. the next transaction to run is the transaction with the highest priority in the active queue. The priority is assigned according to both the deadline which expresses the criticality of time and the SPriority which expresses the importance of the transaction. We consider that the zero value of the *Priority* ($Priority = 0$), corresponds to the highest priority in the system. Transaction T is assigned a priority by the formula:

$$Priority(T) = (1-a) \times Deadline(T) + a \times SPriority(T) \quad (1)$$

where :

- *SPriority*. System priority is a parameter related to each transaction. It expresses the degree of importance of the task(s) executed by a transaction and defines its rank among all the transactions in

the system. Two weight functions are used according to the transaction class to assign the SPriority value and are described in what follows.

- $0 \leq a \leq 1$, is the weight given to *SPriority* in the priority formula.

1- Update Transactions Class. Let *MaxPeriode* be the longest period among the periods of update transactions. The SPriority of an update transaction T is computed according to the following formula:

$$SPriority_{update} = N \times \frac{Periode_T}{MaxPeriode} \quad (2)$$

2- User Transactions Class. The user transaction importance SPriority uses criteria based on both the transaction "write" set operations and the transaction "read" set operations. A user transaction T is assigned a SPriority value by the following formula:

$$SPriority_{user} = MaxValue - \gamma \times Weight_T - (1-\gamma) \times DBA_{Value} \quad (3)$$

where

1. $Weight_T$ denotes the weight of the current user transaction and is given by

$$Weight_T = (\sum_{i=1}^{n} Wread_{TD} + \sum_{j=1}^{m} Wwrite_{NTD} - \sum_{k=1}^{l} Wread_{NTD}) \quad (4)$$

   where

   (a) $Wread_{TD}$, $Wwrite_{NTD}$ and $Wread_{NTD}$ denote respectively the weight assigned to a *read operation* of a temporal data, the weight assigned to a *write operation* of a non-temporal data and the weight assigned to a *read operation* of a non-temporal data (see the transaction characteristics in Table 1).

   (b) n, m, and l are the numbers of operations ($Read_{TD}$/ $Write_{NTD}$/ $Read_{NTD}$) in each user transaction.

2. $\gamma \in]0,1]$ is the rank assigned to the transaction weight in the SPriority formula (see Table 1).
3. DBA_{Value} is a uniform random variable between 0 and $(MaxValue - N)$, *i.e.* $Random(MaxValue - N)$. We recall that N is the value that divides the SPriority interval $[0, MaxValue]$ according to transactions class, *i.e.* $SPriority_{update} \in]0, N]$ and $SPriority_{user} \in]N, MaxValue]$.
4. $Maximum(\gamma \times Weight_T - (1-\gamma) \times DBA_{Value}) \leq MaxValue - N$, because the user transactions $SPriority$ belongs to $]N, MaxValue]$.

2.2 Simulator

User transactions are submitted to the system following a Poisson process (see Figure 1) with an average rate λ into the active queue. The *deadline controller (DC)* supervises the transactions' deadlines, and informs the *transaction scheduler (TS)* when a transaction misses its deadline in order to abort it. *Freshness manager (FM)* exploits the absolute validity interval (*avi*) to check the freshness of a data item before a user transaction accesses it and blocks all user transactions which read stale temporal data. Transactions data conflicts are resolved by the *Concurrency controller (CC)* according to transactions priorities. *CC* informs *TS* in the following cases: (a) when a transaction is finished (committed) and its results are validated in the database, (b) when a transaction is blocked waiting for a conflict resolution, (c) when a transaction is restarted, following the commit of other transactions, (d) when a transaction is rejected because its restart is impossible, *i.e.* its best execution time is higher than its deadline minus the current time ($BET_T > DT - currenttime$), (e) or when a transaction is transferred from the blocked queue to the active queue, *i.e.* its data conflicts are resolved.

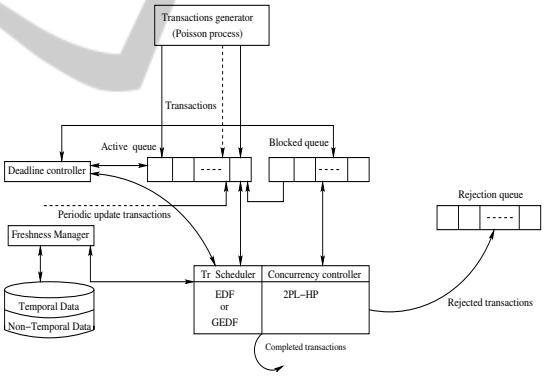


Figure 1: Simulator architecture.

3 SYSTEM PERFORMANCE METRICS

To measure the system performances, we consider transaction success ratio as the main metric. The success ratio is given by:

$$SRatio = \frac{CommitT}{SubmittedT},$$

where $CommitT$ indicates the number of transactions committed by their deadlines, and $SubmittedT$ indicates all submitted transactions to the system in the sampling period. We divide this metric into two parts according to the class of transactions:

- Success ratio of update transactions:

$$SRatio_{Update} = \frac{CommitT_{Update}}{SubmittedT_{Update}}.$$

This ratio indicates the number of update transactions committed by their deadline. It represents the consistency level of temporal data in the database.

- Success ratio of user transactions:

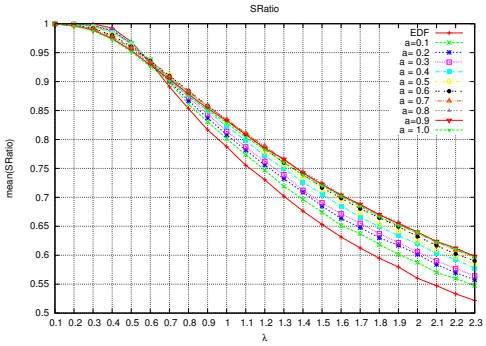
$$SRatio_{User} = \frac{CommitT_{User}}{SubmittedT_{User}}.$$

This ratio indicates the number of user transactions committed by their deadlines.

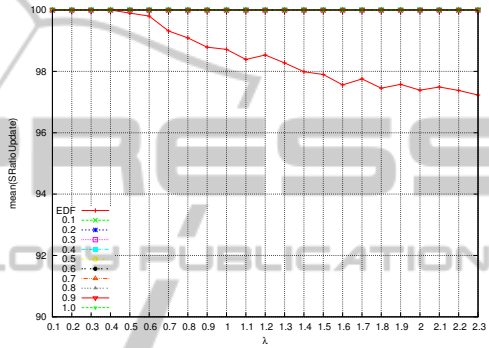
4 RESULTS

In (Semghouni et al., 2007), the authors have shown that GEDF scheduling policy gives better results than EDF, notably when the system is overloaded. In the present paper, we propose to improve the performances of GEDF scheduling policy. To achieve our goal, we analyse the influence of the *SPriority* weight and the rank assigned to transaction weight on the GEDF behavior and on the system performances. For this, we varied the values of the parameter a in the Formula 1 and we have assigned the value $\frac{4}{5} = 0.8$ to γ parameter in order to minimize the effect of the *DBA*, *i.e.* database administrator interaction, in the SPriority Formula 3 in the first step. In the second step, we varied the value of the parameter γ in the SPriority and we have fixed the value of a to 0.4 and then to 0.9 to show the influence of the rank assigned to transaction weight on the GEDF behavior and the system performances. In the third step, we varied the value of a and γ jointly to get the optimal system performances. The assigned values used in simulations are $a = 0, 0.1, 0.2, ..., 1.0$, $\gamma = 0.1, 0.2, 0.4, 0.5, 0.6, 0.8$. This variation of parameters allows to deduce the appropriate assigned values to the parameters a and γ according to the system workload.

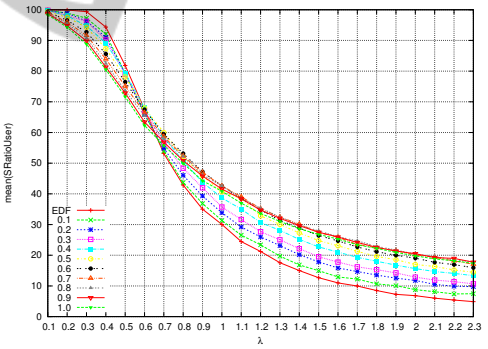
We carried out Monte Carlo simulations that allows us to study the transactions' success ratio behavior and the system quality of service. According to the system parameters given on Tables 1 and 2, we repeat the experiment 1000 times in each simulation in order to obtain a sample of 1000 values for the performances.



(a) Success ratio of transactions when using EDF Vs GEDF$_{a=0.1,0.2,..,1.0}$



(b) Success ratio of update transactions when using EDF Vs GEDF$_{a=0.1,0.2,..,1.0}$



(c) Success ratio of user transactions when using EDF Vs GEDF$_{a=0.1,0.2,..,1.0}$

Figure 2: Influence of the SPriority weight parameter according to system workload.

4.1 Influence of SPriority Weight on System Performances According to the System Workload

To show the importance of influence of SPriority weight we have varied the values of the parameter a form 0 to 1 using step 0.1 under different system workloads. We note that the value of γ parameter is set to $\frac{4}{5}$ in order to minimize the influence of DBA.

Figure 2(b) shows that the best performances for

Table 1: Simulation parameters used for transaction characteristics.

Transaction characteristics		
Notation	Definition	Values
φ	Probability to execute a "Read" or a "Write" operation.	$\varphi(Read) = 2/3$, $\varphi(Write) = 1 - \varphi(Read) = 1/3$.
$User_{SInerval}$	User transaction size interval.	[5, 45] combined operations.
$Update_{size}$	Number of operations in an update transaction.	1 write operation.
SPriority	Intervals of SPriority.	$SPriority_{Update} \in [0,16]$ and $SPriority_{User} \in]16,80]$.

Table 2: Simulation parameters used for system characteristics.

System characteristics		
Notation	Definition	Values
Quantum	Execution capacity in one clock cycle.	20 Tasks/clock cycle
Task	Atomic action.	one Read or Write operation.
ReadTime	Consumption of a read operation.	1 quantum unit.
WriteTime	Consumption of a write operation.	2 quantum units.
Time	Duration of one experiment.	1000 clock cycles.
DBSize	Number of data in the DB.	300.
TD-size	Number of temporal data in the DB.	15% $\times$ DBSize

update transactions are obtained with GEDF scheduling policy: the success ratio is maximal, *i.e.* 100% with different values of a, where $a \in]0, 1]$. Moreover, we have obtained the same performances on the update success ratio results for all system workload conditions (Figure 2(b)). We can conclude that increasing the user transactions number has no significant effect on the update transactions performances.

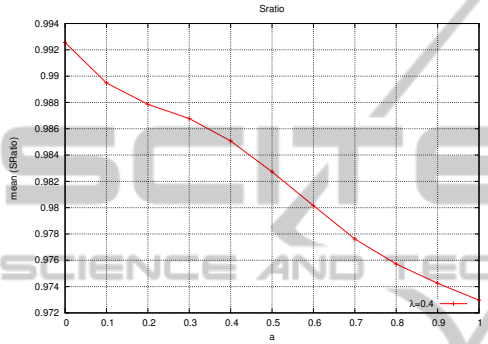
With EDF, there is no difference between the two classes of transactions, since only the deadline is taken into account. Thus, user transactions can be scheduled prior to update transactions if their deadlines are imminent, which affects (decreases) the success ratio of update transactions and degrades the temporal data consistency in the database. This affects also considerably the success ratio of user transactions, especially when the system workload is heavy.

In what follows, we study the user transactions performances on three intervals: $\lambda \in [0.1, 0.6[$ (light workload to average workload), $\lambda \in [0.6, 0.8]$ (average workload) and $\lambda \in]0.8, 2.3]$ (high workload). Hence we explore the effects of SPriority weight a on the success ratio : we compare the results obtained under different values of a according to system workload. Figures 2(a), 2(b) and 2(c) illustrate graphically this comparison.

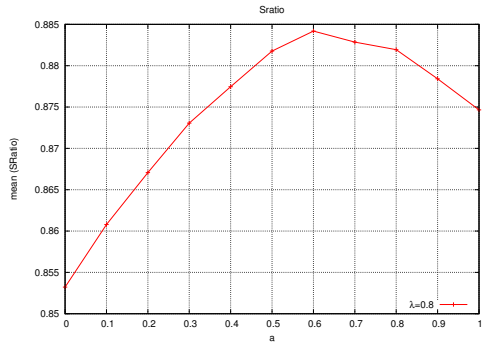
When λ is in the interval $[0.1, 0.6[$, *i.e.* the system is not overloaded, EDF gives better performances on user transactions $SRatio_{User}$ than GEDF, for different values of a (see Figure 2(c)). Indeed, when using GEDF scheduling policy, the lower priority transactions must wait for the commit of the higher priority transactions to be executed even if their deadlines are imminent. This has a negative effect when the system workload is light, which reduces the chances of lower priority transactions to commit, *i.e.* the user transactions' success ratio decreases. For an average system workload, *i.e.* the value of λ is in the interval $[0.6, 0.8[$, we can see that GEDF provides better results than EDF according to the variations of the SPriority weight parameter. The inflection points corresponding to those situations can be seen in Figure 2(a) and on firgure 2(c). When the system workload is heavy, *i.e.* $\lambda \in]0.8, 2.3]$, the situation is completely reversed in favor of GEDF that provides better performances than EDF. We note that inflection points are jointly related to a and the best performances are given by different values of a according to the workload.

In the following, we discuss and compare the success *SRatio* obtained under different values of SPriority weight when using 2PL-HP protocol (see Figure 3). When we look at *SRatio* obtained under different values of SPriority weight parameter, we deduce that the form of the graphic depends on the load of system (see Figure 3). When λ is in the interval [0.1, 0.6[, EDF gives the best *SRatio*, *i.e.* $SRatio_{EDF} > SRatio_{GEDF}$ with different values of the a parameter. Particularly, when λ is in the interval [0.1, 0.4[, and when a increases, *SRatio* decreases.

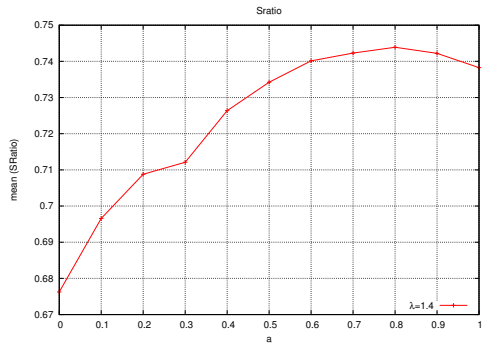
When λ is equal to 0.6, the best *SRatio* is provided by GEDF for $a = 0.4$. The *SRatio* increases giving the best *SRatio* when $a = 0.4$ and then decreases. When λ is in interval [0.7, 1.1], *SRatio* provided by GEDF is better than the *SRatio* provided by EDF for all values of a. In the same way, *SRatio* increases given the best *SRatio* when $a = 0.5$ and then decreases. When λ is in the interval [1.2, 2.3), the peak of SRatio changes from $a = 0.6$ to $a = 0.9$, simultaneously with the increasing of λ. The difference between $SRatio_{EDF}$ and $SRatio_{GEDF}$, achieves to 9% and 15% between $SRatio_{EDF_{user}}$ and $SRatio_{GEDF_{user}}$ when $\lambda = 1.4$.



(a) Success ratio of transactions when $\lambda = 0.4$ (light workload).



(b) Success ratio of transactions when $\lambda = 0.8$ (average load).



(c) Success ratio of transactions when $\lambda = 1.4$ (overload).

Figure 3: Influence of SPriority weight on transactions with 2PL-HP protocol when $\gamma = 0.8$.

4.2 Influence of the Rank assigned to Transaction Weight (RTW) in the SPriority on the System Performances According to the Workload

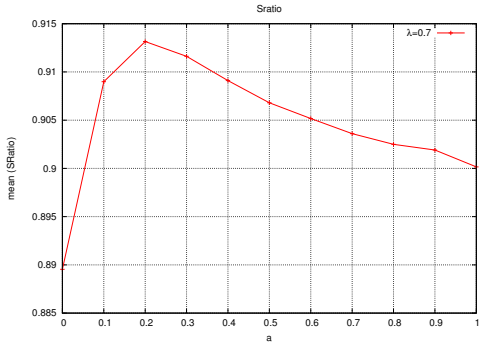
In order to study the influence of the rank assigned to transaction weight, we analyze firstly the success ratio of transactions obtained under different values of RTW. Then we accurate our analysis by studying the succes ratio of user transactions under different values of RTW and SPriority weight.

In the previous section, we have studied the influence of SPriority weight according to the workload when RTW value γ is equal to 0.8. We have seen that when $\lambda < 0.6$, EDF gives the best *SRatio* whereas when $\lambda \geq 0.6$, GEDF gives the best *SRatio*. Particularly when $\lambda > 1.2$, the best *SRatio* is given by GEDF when $a = 0.8$. To analyze the impact of γ, we have done the same analysis with $\gamma = 0.2$ (see Figure 4). In the same way, when $\lambda < 0.6$, the best *SRatio* is given by EDF and when $\lambda \geq 0.6$, GEDF gives a better result than EDF. In fact, when λ is in [0.6, 0.8[, the best *SRatio* is obtained when $a = 0.2$ (Figure 4(a)). When the workload increases, the value of a increases too, but does not exceed 0.5 (Figure 4(b)).

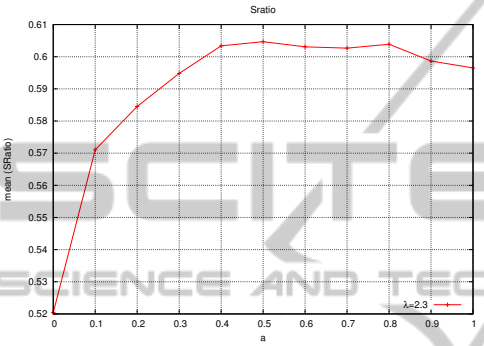
In the following section, we compare the success ratio of user transactions under different values of RTW and SPriority weight ($\gamma = 0.1, 0.2, 0.4, 0.5, 0.6, 0.8$, $a = 0.1, 0.2, ..., 1.0$) according to system workload. In all simulations, we have seen that EDF is more efficient than GEDF when $\lambda < 0.6$. Hence, we discuss and compare the success $SRatio_{user}$ only in average and high workload and how it is possible to exploit the flexibility of GEDF in order to improve the systems performances. Figures 5(a) and 5(b) illustrate graphically this comparison.

The choice of RTW affects significantly the success ratio of transactions when the weight of SPriority is small, *i.e.* $a \leq 0.5$. In fact, the range between the different values of $SRatio_{user}$ under various RTW values reaches 8.5% (see Figure 5(a)).Whereas, when the weight of SPriority is important, *i.e.*, when $a > 0.7$, the influence of RTW on the success ratio of transaction decreases considerably. The range between the different $SRatio_{user}$ under various RTW values does not exceed 1.76%, (Figure 5(b)).

To get the optimal $Sratio_{user}$ under different workloads of systems, we have made a combination between all values of RTW and those of SPriority weight. Table 4 in appendix gives the best and worst values to assigne to the γ under various SPriority weights according to the system workload.



(a) Success ratio of transactions when $\lambda = 0.7$ (average workload).



(b) Success ratio of transactions when $\lambda = 2.3$ (overload).

Figure 4: Influence of SPriority weight on transactions with 2PL-HP protocol when $\gamma = 0.2$.

Table 3 shows the optimal values of a and γ according to the workload. To show the gain obtained by using these optimal values, we have compared $SRatio_{user_{GEDF}}$ when γ and a are assigned the following optimal values (see Table 3); $\gamma = 0.8$, $a = 0.9$; $\gamma = 0.2$, $a = 0.1$ and EDF (see Figure 6).

We note that when the system is moderately loaded, the best $SRatio_{user}$ is given with a small value of SPriority and with a small value of RTW. When the workload increases, we have to increase the weight of SPriority and RTW in order to obtain the best $SRatio_{user}$. This is illustrated in table 3.

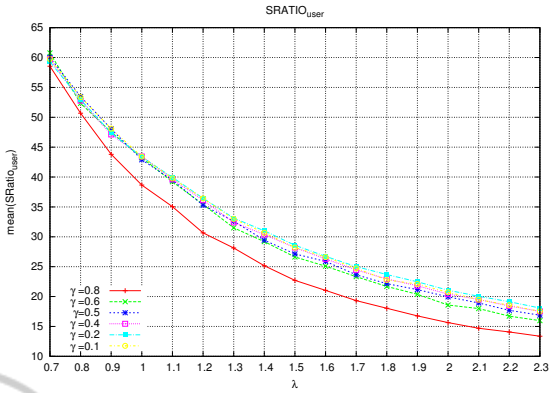
Table 3: Influence of RTW and SPriority weight on $SRatio_{user}$.

$\lambda \leq 0.6$	$\lambda \in [0.7, 1.1[$	$\lambda \in [1.1, 1.2[$
EDF	a= 0.3, $\gamma = 0.2$	a=0.4, $\gamma = 0.2$

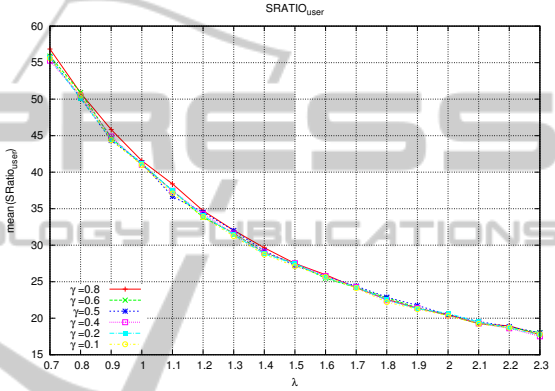
$\lambda \in [1.2, 1.4[$	$\lambda \in [1.4, 2.2[$	$\lambda \in [2.2, 2.3[$
a=0.5 ,$\gamma = 0.5$	a=0.6, $\gamma = 0.5$	a=0.7,$\gamma = 0.5$

5 CONCLUSIONS

GEDF is a multi-parameters scheduling policy based on the well-known EDF policy. Its parameters have to be defined by taking into account the system characteristics such as the workload. The analysis of the system performances we have conducted in this paper allows us to evaluate the optimal values for the parameters in order to significantly increase the success ratio of the transactions which is the main criterion used in RTDBSs. In a future work, we plan to study the influence of other parameters such as size



(a) influence of γ on $SRatio_{user}$ when $a = 0.4$.



(b) influence of γ on $SRatio_{user}$ when $a = 0.9$.

Figure 5: Influence of γ in $SRatio_{user}$.

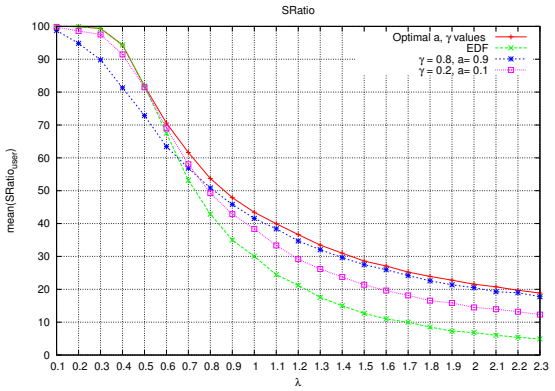


Figure 6: Success ratio of user transactions with optimal values of RTW and SPriority weight according to workload.

of user transactions, proportion of temporal data in the database and database size. We also plan to provide an adapted GEDF protocol analysis to different extended transactions models such as nested transaction.

REFERENCES

Datta, A., Mukherjee, S., Konana, P., Viguier, I. R., and Bajaj, A. (1996). Multiclass transaction scheduling and overload management in firm real-time database systems. *Information Systems*, 21(1):29 – 54. Real-Time Database Systems.

Han, S., Lam, K.-y., Wang, J., Son, S. H., and Mok, A. K. (2012). Adaptive co-scheduling for periodic application and update transactions in real-time database systems. *J. Syst. Softw.*, 85(8):1729–1743.

Haritsa, J. R., Livny, M., and Carey, M. J. (1991). Earliest Deadline Scheduling for Real-Time Database Systems. In *IEEE Real-Time Systems Symposium*, pages 232–243.

Kim, Y.-K. and Son, S. H. (1996). Supporting predictability in real-time database systems. In *Proceedings of the 2nd IEEE Real-Time Technology and Applications Symposium (RTAS '96)*, RTAS '96, pages 38–, Washington, DC, USA. IEEE Computer Society.

Pang, H., Livny, M., and Carey, M. J. (1992). Transaction scheduling in multiclass real-time database systems. In *IEEE Real-Time Systems Symposium*, pages 23–34.

Ramamritham, K., Son, S. H., and Dipippo, L. C. (2004). Real-time databases and data services. *Real-Time Syst.*, 28(2-3):179–215.

Semghouni, S., Amanton, L., Sadeg, B., and Berred, A. (2007). On new scheduling policy for the improvement of firm rtdbss performances. *Data Knowl. Eng.*, 63(2):414–432.

Semghouni, S., Sadeg, B., Berred, A., and Amanton, L. (2008). The Behavior of Transactions Success Ratio in Firm Real-Time Database Systems. *Journal of Advances in Computer Science and Engineering*, 2(2):133–163. 30 pages.

Xiong, M. and Ramamritham, K. (2004). Deriving deadlines and periods for real-time update transactions. *IEEE Trans. Computers*, 53(5):567–583.

Yu, P., Wu, K.-L., Lin, K.-J., and Son, S. (1994). On real-time databases: concurrency control and scheduling. *Proceedings of the IEEE*, 82(1):140 –157.

APPENDIX

Table 4: Influence of RTW and SPriority weight on $SRatio_{user}$.

a	λ	Best γ	Worst γ	maximal range
0.1	$\lambda \in [0.7, 2.3[$	0.1	0.8	8.15% ($\lambda = 1.0$)
0.2	$\lambda \in [0.7, 0.8[$	0.2	0.8	9.23% ($\lambda = 1.1$)
	$\lambda \in [0.8, 2.3[$	0.1	0.8	
0.3	$\lambda \in [0.7, 0.8[$	0.4	0.8	8.35% ($\lambda = 1.2$)
	$\lambda \in [0.8, 0.9[$	0.2	0.8	
	$\lambda \in [0.9, 2.3[$	0.1	0.8	
0.4	$\lambda \in [0.7, 0.8[$	0.6	0.8	5.88% ($\lambda = 1.5$)
	$\lambda \in [0.8, 1.0[$	0.5	0.8	
	$\lambda \in [1.0, 1.1[$	0.4	0.8	
	$\lambda \in [1.1, 1.5[$	0.2	0.8	
	$\lambda \in [1.5, 2.3[$	0.1	0.8	
0.5	$\lambda \in [0.7, 0.8[$	0.8	0.2	4.55% ($\lambda = 2.0$)
	$\lambda \in [0.8, 1.1[$	0.6	0.8	
	$\lambda \in [1.1, 1.5[$	0.5	0.8	
	$\lambda \in [1.5, 1.9[$	0.4	0.8	
	$\lambda \in [1.9, 2.1[$	0.2	0.8	
	$\lambda \in [2.1, 2.3[$	0.1	0.8	
0.6	$\lambda \in [0.7, 0.9[$	0.8	0.2	2.79% ($\lambda = 1.9$)
	$\lambda \in [0.9, 1.2[$	0.6	0.8	
	$\lambda \in [1.2, 1.9[$	0.5	0.8	
	$\lambda \in]1.9, 2.3]$	0.4	0.8	
0.7	$\lambda \in [0.7, 1.0$	0.8	0.2	1.76% ($\lambda = 0.9$)
	$\lambda \in [1.0, 1.5[$	0.6,0.5	0.2	
	$\lambda \in [1.5, 2.3[$	0.5,0.4	0.8	
0.8	$\lambda \in [0.7, 1.5[$	0.8	0.2	1.22% ($\lambda = 1.2$)
	$\lambda \in [1.5, 2.3]$	0.6, 0.5	0.8, 0.2	
0.9	$\lambda \in [0.7, 1.7[$	0.8	0.2	1.72% ($\lambda = 1.1$)
	$\lambda \in [1.7, 2.3[$	-	-	($dif < 0.3$)%