Multi-Versions Data for improvement of QoS in RTDBS[*]

Emna Bouazizi, Claude Duvallet, Bruno Sadeg
LIH, Université du Havre, France.
{Emna.Bouazizi,Claude.Duvallet,Bruno.Sadeg}@univ-lehavre.fr

Abstract

In current research toward the design of more powerful behavior of RTDBS under unpredictable workloads, different research groups focus their work on QoS (Quality of Service) guarantee. Their research are often based on feedback control real-time scheduling theory. In this paper, we propose a technique which allows to execute transactions on time using fresh and precise data while taking into account the global size of the database. We have extended the feedback-based miss ratio control by both using multi-versions data, and proposing a data management policy combining (1) limitation of the versions number and (2) dynamic adjustment of this limit according to a maximum database size parameter.

1 Introduction

In previous years, a lot of work has been done on RTDBS [8][9], which are systems that are designed to manage applications where it is desirable to execute transactions timely using fresh and precise data [1]. Since the workload in this systems is unpredictable, the system may become quickly overloaded, leading to the decrease of the well-known RTDBS performance criterion (the number of transactions that complete before their deadline).

To support these applications, some techniques based on QoS guarantee have been proposed to control the transient overshoot. They are often based on feedback control real-time scheduling theory [6][7]. Up to now, the major drawback is that in case of conflicts between transactions, some transactions are blocked, or aborted and restarted. This may lead to the transactions miss deadlines. To address this problem, in this paper, we have extended the feedback-based miss ratio control by using a multi-versions data architecture. This limits data access conflicts between transactions, enhancing then the concurrency and limits the deadline miss ratio.

The main objective of our approach is to maximize the number of transactions which meet their deadlines. At the same time, our work aims to support a certain freshness for the data accessed by timely transactions under a condition: the fixed maximum size of the RTDB. To this purpose, we merge two approaches previously proposed in [4] and [5]. In the new mixed approach, the number of versions is dynamically adjusted, but does not have to exceed a threshold which is a maximum data versions number, and also does not have to exceed a fixed threshold which is the maximum database size. The remaining of the paper is organized as follows. Section 2 describes the real-time database model. In Section 3, our proposed model is described. Some simulation results are given in Section 4. In Section 5, we conclude the paper and give some perspectives.

[*]Real Time Database Systems

2 Real-Time Database model

We consider firm RTDBS model, in which tardy transactions[1] are aborted because they are useless after their deadline, and we consider a main memory database model in which the CPU is the main system resource taken into account. Data objects are classified into either real-time or non real-time data. In our model, we consider mainly real-time data.

Transactions can be classified into two classes: update transactions and user transactions. Update transactions are used to update the values of real-time data in order to reflect the state of real world. Update transactions execute periodically and have only to write real-time data. User transactions, representing user requests, arrive aperiodically and may read real-time data, and read or write non real-time data.

3 Multi-Versions Data-Feedback Control Scheduling Architecture

We have based our work on [2][3]. We have extended the FCS architecture by exploiting several versions of real-time data, and then proposed the Multi-Versions Data-Feedback Control Scheduling (MVD-FCS) approach [4][5]. In this

[1]Transactions that have missed their deadlines.

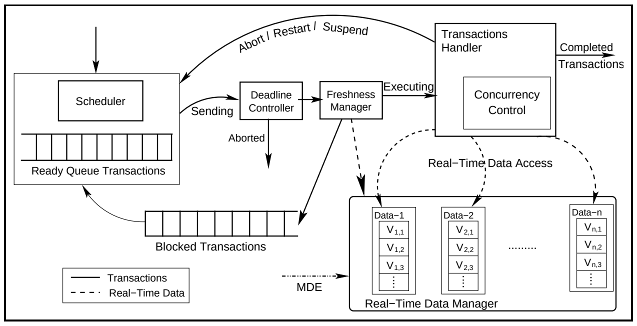


Figure 1. Multi-Versions Data - Feedback Control Scheduling Architecture

section, we present a new approach which enhances MVD-FCSA depicted in Figure 1 where the solid arrows represent the transaction flows and the dotted arrows represent the real-time data flows.

In the FCSA [2], two actions are considered important for improving the MVD-FCSA service:

- In case of conflict between transactions, when a higher priority transaction uses the data item, transactions with lower priority will be blocked.
- FM blocks user transactions if the accessed data are stale.

To enhance this protocol and to minimize the transaction miss ratio, we have proposed the MVD-FCSA, which consists of the creation of data versions as soon as conflicts (read-write) occur between transactions.

The majority of MVD-FCSA components exist in the classical feedback control scheduling architecture [2], but they are adapted as follows: the eeal-time transactions scheduler, the deadline controller (DC) and the freshness manager (FM) are presented in [4] .

The Real-Time Data Manager is used to guarantee the data freshness and to enhance the deadline miss ratio even in the presence of conflicts and unpredictable workloads. To achieve this goals, in [5] we have used a MVD-FCSA with a fixed number of data versions.This number is fixed in advance by the DBA according to QoS requirement level, and it is the same for each real-time data. In [4], we have enhanced this approach by allowing the dynamic adjustment of the versions number. For each data, we have a versions queue. The queue is continually updated in order to limit the number of data versions by supressing/adding versions, based on both the data freshness and MDE criterion.

In this paper, we have extended this last approach by taking into account the database size constraint. We merge the two approach described in [4] and [5] , i.e. the number of data versions is dynamically adjusted and does not have to exceed the fixed threshold representing the number of data versions, and we have considered in the same time a threshold representing the database size. In this new approach, a data item will be accessed only if it's version number is lower than the maximum database size. This way, RTDB size constraints are respected. The respect of the threshold of the RTDB size is a practical factor for RTDBS specification.

The database consistency can be maintained using the Concurrency Control. We use 2PL-HP (Two Phase Locking-High Priority Protocol) where if a higher priority transaction accesses a data item, then other transactions (with lower priority) will be blocked. Otherwise, the transaction is aborted and restarted. Consequently, the 2PL-HP might increase the execution time of transactions. This leads the transaction to miss their deadlines. To address this problem, i.e. to alleviate this risk, we propose (1) the MVD technique that allows user transactions to access a less recent data version when update transaction writes a new data version, and (2) an adapted 2PL-HP when the maximum number of versions is reached. The priority is applied on transactions group that accessed to the same data version [4]. The priority of transactions group corresponds to the highest transaction priority among all transactions in this group.

4 Simulations and results

4.1 Simulations

The simulated workload consists of update and user transactions, which access data. Update transactions occupy approximately 50% of the workload. The period of update transaction ($Period_i$) is uniformly distributed and estimated execution time is given by: ExecutionTime = NbOfOperations * OpExecTime, where NbOfOperations and the OpExecTime represent respectively the number of operation in the transaction T_i and the execution time of an operation. The model consists of eight components. We have used two transactions generators, an update transactions generator (UpdateTransGen) which generates update transactions and a user transactions generator (UserTransGen) which generates user transactions. The workload model characterizes transactions in terms of the number of read/write operations. Update transactions can only write one data. Transactions are scheduled in a ready queue according to their priority. The priority assignment formula is given by P(T_i)= (-1) * deadline (T_i). Deadline controller (DC) uses three controlled variables: transaction deadline (deadline), current time (StartTime) and minimal execution time (ExecutionTime). The deadline formula is calculated as follows: deadline (T_i) = StartTime + ExecutionTime * (1 + SlakTime), where SlakTime is a constant that provides control over tightness/slackness of transaction deadlines.

Parameter	Meaning	Value
NbOfOperations	Number of operations in an user transaction	[1, 5]
OpExecTime	Execution time of an operation	1s
$Period_i$	Periodicity of update transaction	[1000ms, 5000ms]

Table 1. Parameters of Simulation

In the sets of experiments, we have varied the database size and the maximum number of data versions for each data item. The database size represents the number of versions. Table 1 summarizes simulations parameters.

4.2 Results

Since in our approach we have only extend the transactions flows of the classical FCSA, the performance metric in our experiments is the success ratio. The graphical results show the miss ratio of transactions when using MVD-FCSA. We have evaluated the behavior of the system by varying a set of parameters:

1. The threshold of data versions number.
2. The threshold of database size.
3. The number of transactions.

4.2.1 Experiment 1:

We use our mixed approach (dynamic adjustment of data versions with maximum fixed number). In Figure 2, we have fixed a threshold of data versions number (equal to 4 versions) and we have varied the database size (500, 750, 1000). In Figure 3, we have also varied the database size, while the threshold of data versions number is equal to 6.

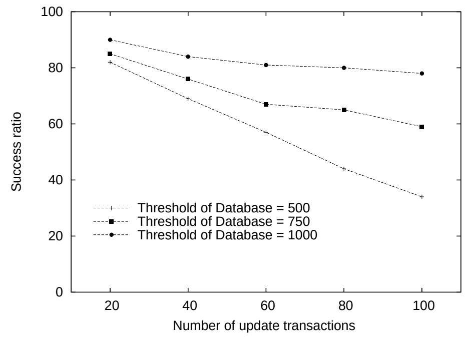
Figures 2 et 3 show the effects of varying the database size. The resulting success ratio increases according to the increase of the database size.

4.2.2 Experiment 2:

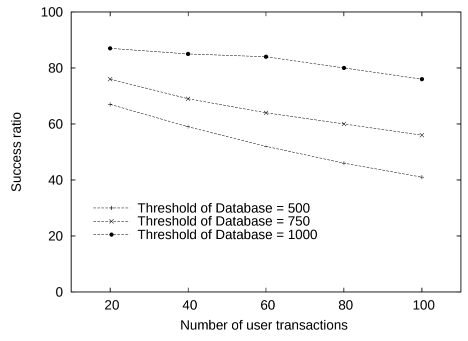
We also use the mixed approach of MVD-FCSA. We have fixed the database size and we have varied the threshold of data versions number. Compared to the effect of using a maximum of six versions, the use of four versions shows a relatively high success ratio, as shown in Figures 4.

4.3 Discussions

We have compared the system performances, in terms of miss ratio, by varying the database size and by varying the maximum number of data versions. All experiments simulation show that:



(a) Update transactions



(b) User transactions

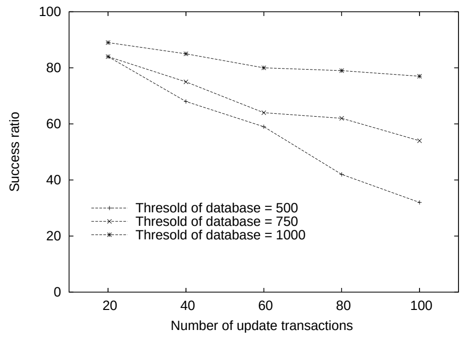
Figure 2. Simulation results when using the mixed approach of MVD-FCSA (maximum number of versions = 4).

1. MVD-FCSA minimizes transactions miss deadline (compared to the classical FCSA).
2. The success ratio increases according to the decrease of the number of versions.
3. The success ratio increases according to the increase of the database size.
4. The increase of the versions number is the most significant and influential criterion on success ratio.

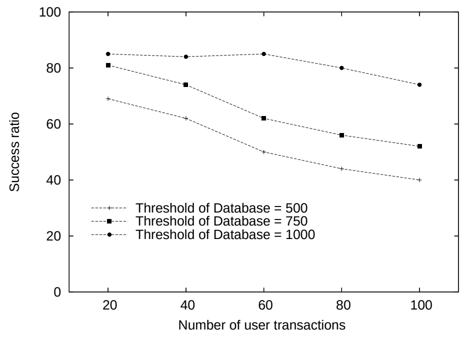
5 Conclusion and future work

In this paper, we have presented the multi-versions data-feedback control scheduling architecture for QoS management. We have used multi-versions data with dynamically adjusted number of versions while taking into account the RTDB size constraint.

Simulation results show that MVD-FCSA with dynamically adjusted number of data versions may be applied efficiently in RTDBS, i.e more transactions meet their dead-

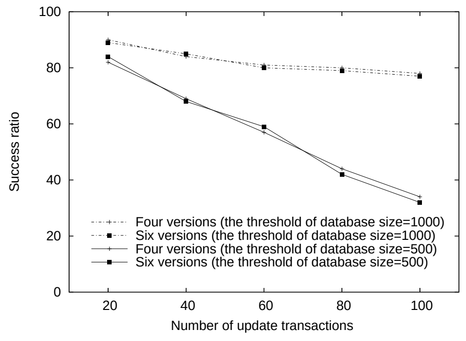


(a) Update transactions

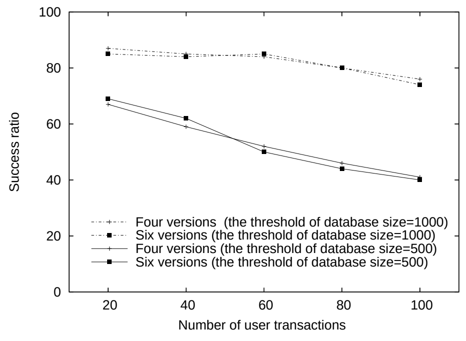


(b) User transactions

Figure 3. Simulation results when using the mixed approach of MVD-FCSA (maximum number of versions = 6).



(a) Update transactions



(b) User transactions

Figure 4. Simulation results of using the mixed approach of MVD-FCSA: varying the number of versions and the threshold of database size.

lines. We note that the respect of the threshold of the RTDB size might be a practical factor for RTDBS specification.

We plan to extend this work in several ways. We will take into account the data *importance*. Indeed, in case of a small threshold of the RTDB size, all data beyond the threshold value are not accessed whatever their importance. Further, we also plan to extend our work to manage derived data and to consider other aspects to study different components of the feedback control scheduling architecture for QoS management in RTDBS.

References

[1] R. Abbott and H. Garcia-Molina. Scheduling Real-Time Transactions: A Performance Evaluation. *International Journal of Distributed and Parallel Databases*, 1(2), 1988.

[2] M. Amirijoo, J. Hansson, and S. H. Son. Algorithms for Managing Real-time Data Services Using Imprecise Computation. In *Proceedings of International Conference on Real-Time and Embedded Computing Systems and Applications (RTCSA)*, Taiwan, 2003.

[3] M. Amirijoo, J. Hansson, and S. H. Son. Error-Driven QoS Management in Imprecise Real-Time Databases. In *Proceedings of* 15^{th} *Euromicro Conference on Real-Time Systems (ECRTS)*, Portugal, 2003.

[4] E. Bouazizi, C. Duvallet, and B. Sadeg. Management of QoS and Data Freshness in RTDBSs using Feedback Control Scheduling and Data Versions. In 8^{th} *IEEE International Symposium on Object-oriented Real-time distributed Computing (IEEE-ISORC'05)*, Washington, 2005.

[5] E. Bouazizi, B. Sadeg, and C. Duvallet. Ordonnancement contrôlé par rétroaction dans les SGBD temps réel. In 13^{th} *conference on Real-Time Systems (RTS)*, Paris, France, 2005.

[6] C. Lu. *Feedback Control Real-Time Scheduling*. PhD thesis, University of Virginia, May 2001.

[7] C. Lu, J. Stankovich, G. Tao, and S. Son. Feedback Control Real-Time Scheduling: Framework, Modeling and Algorithms. *Real-Time Systems*, 23(1/2):85–126, 2002.

[8] K. Ramamritham. Real-Time Databases. *Journal of Distributed and Parallel Databases*, 1(2):199–226, 1993.

[9] K. Ramamritham, S. Son, and L. DiPippo. Real-Time Databases and Data Services. *Real-Time Systems*, 28:179–215, 2004.