Management of QoS and Data Freshness in RTDBSs* using Feedback Control Scheduling and Data Versions

Emna Bouazizi, Claude Duvallet, Bruno Sadeg
LIH, Université du Havre, 25 rue Philippe Lebon
BP 540, F-76058 LE HAVRE Cedex
{Emna.Bouazizi,Claude.Duvallet,Bruno.Sadeg}@univ-lehavre.fr

Abstract

These recent years, a lot of real-time applications are becoming increasingly sophisticated in their data needs, resulting in a greater demand for real-time data services. Real-time database systems (RTDBS) can manage these applications, but the workload in these systems is unpredictable, then RTDBS may become overloaded. A lot of work dealing with quality of service (QoS) has been done to control the transient overshoot. They are based on feedback control real-time scheduling theory. In this paper, we propose a work which allows to execute transactions before their deadlines while using the fresh data. We have extended the feedback-based miss ratio control, by using a multi versions data architecture to guarantee a set of requirements on the behavior of RTDBS and we have considered two data managment policies. In the first policy, we have limited the maximum number of data versions and this number is the same for all the data items. In the second policy, the number is dynamically adjusted for each data item.

1 Introduction

Currently, the demand for real-time database services has increased in most applications where it is desirable to execute transactions within their deadlines. They also have to use precise and fresh data in order to reflect the continuously changing external environment.

To support these applications, some techniques based on the Quality of Service (QoS) guarantee have been proposed, we discuss some issues in section 2. In this paper, we propose to use data versions, based on feedback control real-time scheduling theory, to guarantee a set of requirements on the behavior of RTDBS and to maximize the number of transactions which meet their deadlines. We have based our work on results already found in order to take into account the QoS in RTDBS using the Feedback Control Real-Time Scheduling (FCS) [6] [7]. Our contribution consists of adding the notion of data versions to manage the transactions in order to limit the data access conflicts between transactions, and then to enhance the conccurency.

The main objective of our approach is to limit the deadline miss ratio. In this article, we begin by a presentation of existing approaches on which is based our work (see section 2). Then, in section 3, we present the model we consider. In section 4, we describe our architecture for using multi versions data. The benefits of our architecture are discussed in section 5. Section 6 shows the details of the simulation settings and the evaluation results. We conclude this article by a discussion about this work and by the presentation of our future works.

2 Approaches related to QoS

In RTDBS, new techniques have been designed to manage real-time transactions [1] [8]. These techniques use feedback control scheduling theory [6] and imprecise computation in order to provide an acceptable RTDBS behavior. Many works are based on the QoS specification to control the transient overshoot of RTDB [6] [2].

In [2], Amirijoo et al have proposed two algorithms: FCS-IC-1 and FCS-IC-2 (Feedback Control Scheduling - Imprecise Computing). These algorithms have been proposed to dynamically balance the workload and the quality of the data and transactions. The authors employ three feedback control scheduling policies, called FC-M (Feedback Miss Ratio Control), FC-U (Feedback Utilisation Control) and FC-UM (Feedback: Integrated Utilisation Miss Ratio Control), to control the user transactions quality in the presence of unpredictable workload and inaccurate execution time estimations. In [7], the authors suggest a QoS-sensitive approach, called QMF (a QoS-sensitive approach for Miss Ratio and Freshness guarantees) using a dynamic scheme

*Real Time DataBase Systems

to balance the user transactions and the update transactions workloads. To provide differentiated services in terms of miss ratio, they have extended in [4] their QMF approach which can support a single class of miss ratio and freshness guarantees in RTDB. They call the new approach QMF-Diff (QMF with Differentiated Services) . A DBA[1] can explicitly specify the required database QoS including the miss ratio differentiation among the service classes.

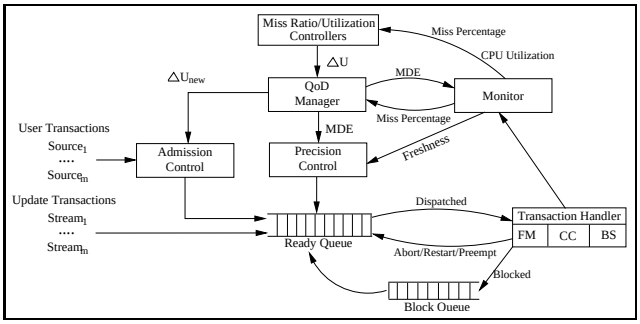


Figure 1. A FCS Architecture

3 Real-Time Database model

We consider firm RTDBS model, in which tardy transactions[2] 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.

3.1 Data model

Data objects are classified into either real-time or non real-time data. In our model, we consider mainly real-time data.

3.2 Transaction model

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.

[1] Database Administrator

[2] transactions that have missed their deadlines

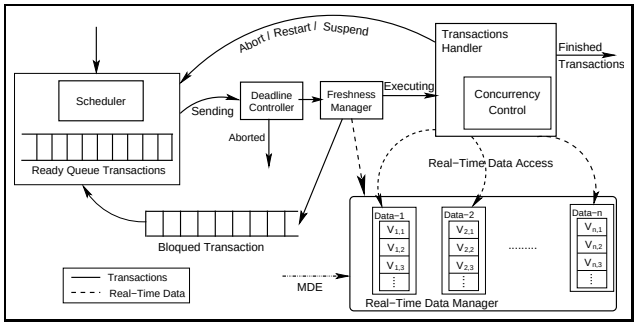


Figure 2. MVD Architecture

4 Multi Versions Data-Feedback Control Scheduling Architecture: MVD-FCSA

The MVD-FCSA is shown in Figure 2. We have based our work on the work done by Amirijoo et al. in RTDBS [2]. In this work, the authors have used FCSA (see Figure 1). In our work, we have added a new notion which consists of the creation of data versions as soon as a conflict (read-write) occurs between transactions. In our system, we extend the FCS architecture by using the notion of Multi Versions Data. Figure 2 depicts our MVD-FCS architecture where the solid arrows represent the transaction flows and the dotted arrows represent the real-time data flows.

4.1 Scheduling real-time transactions

Transactions are scheduled in the ready queue according to their priority. The priority of a transaction depends on both its deadline and its type (update or user transaction). Hence, we merge EDF policy with respect to type and priority.

4.2 Deadline Controller

It controls transaction validity [5] [3]. If the current time is greater than transaction deadline, transaction will be aborted. Otherwise, Deadline Controller (DC) makes a second verification, where a transaction is accepted only if the sum of the minimal execution time and the current time is lower than transaction deadline, otherwise the transaction will be aborted. If the two verifications steps succeed, then the transaction is transferred to the Freshness Manager (FM).

4.3 Freshness Manager

It checks the freshness of acceded data just before a transaction commits. This way, the data accessed by committed transactions are always fresh at commit time. If the

accessed data is fresh, transaction can be executed and it is sent to the transactions handler. Otherwise, if the accessed data item is currently stale or will be before the deadline of the transaction, FM blocks the user transaction. The blocked transaction will be transferred from the blocked queue to the ready queue as soon as the corresponding update has committed.

4.4 Real-Time Data Manager

The main objective of this component is 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, we use two approaches that use the MVD technique.

In our first approach, we use a fixed number of data versions. When update transactions want to modify a real time data, a new data version is created. Most conflict cases come from incompatible access patterns when update transaction want to modify data item that is accessed by user transaction. One of these transactions must be aborted and restarted according to the used conccurency control protocol. Then the risk that transactions miss their deadline increases. MVD notion is used to alleviate this risk. We keep all data values that correspond to different versions of the same data item. The maximum number of versions is limited and is fixed in advance by the DBA according to QoS requirement level.

In our second approach, we dynamically adjusted number of data versions. For each data, we have a queue of versions . The queue is continually updated in order to limit the number of data versions by supressing/adding versions, based on both freshness and MDE criterion The Size of each Versions Queue, denoted SVQ, is dynamically adjusted, and is defined as follows:

$$SVQ = IntegerPart[\frac{AVI_j}{Period_i}] \quad (1)$$

where $Period_i$ is the periodicity of $transaction_i$, and AVI_j is the absolute validity interval of d_j.

4.5 Concurrency Control with MVD

Database consistency can be maintained using concurrency control protocols. Here, we use 2PL-HP (Two Phase Locking-High Priority Protocol) where if the higher priority user transaction reads a data item, then update transaction (with lower priority) will be blocked. Otherwise, the user transaction is aborted and restarted. Consequently, the 2PL-HP might increase the execution time of blocked/aborted transactions. This leads to the transaction to miss their deadline. To address this problem, we have proposed the MVD technique that allows user transactions to access an old data version when update transaction writes a new data version , and an adapted 2PL-HP when the maximum number of versions is reached. (cf. Algorithm 1).

Algorithm 1: 2PL-HP adapted protocol for MVD

Tr_{up} : update transaction (write access).
Tr_{user} : user transaction (read access).
$Gr(Tr_{user})$: user transactions group accessing a data version.
$Gr_i(Tr_{user})$: n^{th} user transactions group.
nb_group : number of transactions group.

```
begin
    if Priority (Tr_up) < Priority (Gr(Tr_user)) then
        Tr_up blocked waiting the release of version
    else
        for i from de 1 to nb_groups do
            if Priority (Tr_up) > Priority (Gr_i(Tr_user))
            then
                abort and restart of Gr_i(Tr_user)
            endif
        endfor
    endif
end
```

NOTE: the priority of transactions group correspond to the highest transaction priority among all transactions in this group.

5 Benefits of MVD-FCSA

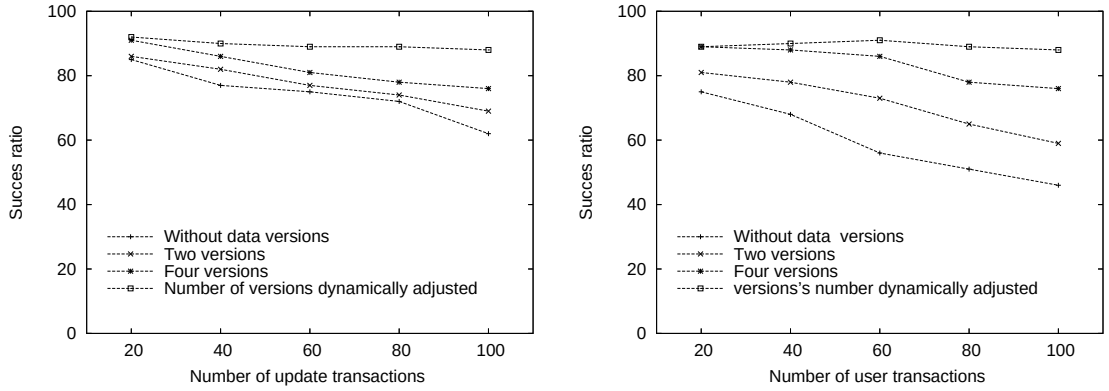
MVD-FCSA is a good solution for QoS guarantees. By comparison with the classic feedback control scheduling architecture, it allows to limit the deadline miss ratio, to support freshness for the data accessed by timely transactions (even in the presence of unpredictable workloads). So the data used by committed transactions are always 100% fresh (at commit time). MVD-FCSA permits also to guarantee the quality of data (QoD: precision and freshness) and quality of transaction (QoT) which are enhanced by alleviating the risk of transaction miss deadline, and therefore it enhance the QoS.

6 Simulations and results

The simulated workload consists of update and user transactions, which access data.

As shown in Figure 3, when a data item is without versions, with two versions and with four versions, the resulting success ratio increase according to the increasing of the number of versions.

Compared to the effect of using MVD-FCSA with fixed number of versions, using of MVD-FCSA with dynamically



(a) For update transactions

(b) For user transactions

Figure 3. Simulation results of the MVD-FCSA

adjusted number of data versions shows a relatively high success ratio as shown in Figure 3.

We also compared the system performances, in terms of miss ratio, by using the transactions types. In comparison with the four curves showed in Figure 3(a), the curves in Figure3(b) are more near. This is because, in our approach, the update transactions have generally, the higher priority. That's why, in Figure 3, the increasing of number of versions is the most significant and influential criterion on success ratio.

It has been shown that MVD-FCSA minimize the transaction miss deadlines. We have also seen that the transactions miss ratio and the number of versions increase together.

In MVD-FCSA, data freshness and data precision are also required. This way, real-time data managment guarantee both the QoD and the QoT by decreasing the transactions miss ratio. The experiments show that MVD-FCSA is useful and particularly successful with dynamically adjusted number of data versions, notably when the number of transactions increases.

7 Conclusion and future work

In this work, we have presented an enhancement of the feedback control scheduling architecture for quality of service management in which we use multi versions data. This improvement is used to minimize the number of conflicts by minimizing the number of aborted transactions when using an adapted 2PL-HP concurrency control protocol.

We plan to extend this work in several ways. We will consider other aspects to study different components of the feedback control scheduling architecture for quality of service management in RTDBS. Among them, will deal with imprecise computing [2] applying to video contents.

8 Acknowledgement

This works is supported by the ACI project #1055 (French Research Ministry).

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] C. Date. *An Introduction to Database Systems*. Addison-Wesley, 1985.

[4] K. Kang, S. Son, and J. Stankovic. Service Differentiation in Real-Time Main Memory Databases. In *5th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, Washington D.C., April 29 - May 01 2002.

[5] C. Liu and J. Leyland. Scheduling Algorithms for Multiprogramming in Hard Real-Time Environment. *Journal of the ACM*, 20(1):46–61, 1973.

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

[7] C. Lu, J. Sankovic, , G. Tao, and S. Son. Feedback control real-time scheduling: Framework, modeling and algorithms. *Journal of Real-Time Systems*, 23(1/2), 2002.

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