Using Feedback Control Scheduling and Data Versions to enhance the Quality of Data in RTDBSs*

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

In Real-Time DataBase Systems (RTDBSs), both the integrity constraints of the database and the temporal constraints of the transactions must be respected. However, the workload in these systems may be unpredictable, then RTDBSs may become overloaded and it seems to be difficult for the transactions both to meet their deadlines and to keep the database consistency. To address this problem, different research groups focus their work on quality of service (QoS) guarantee using the Feedback Control Real-Time Scheduling (FCS). In this paper, we have extended the FCS architecture by using a multi-versions data notion. This technique allows to execute transactions on time using fresh and precise data. We have considered two data management 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. We have carried out intensive simulations to compare the performances of these policies and a policy using only one data version. We describe our architecture and we report results of our simulations.

1. Introduction

Real-Time Databases Systems (RTDBS) are systems proposed to maintain both the logical consistency of the database (integrity constraints) and its temporal consistency (each transaction have to meet its deadline). Since the workload in theses 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). In current research toward the design of more powerful behavior of RTDBS under unpredictable workloads, different research groups focus their work on QoS guarantee. Their research are often based on feedback control real-time scheduling theory. However, due to the high services demand, and even with these services guarantees, many transactions may miss their deadlines. 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 many transactions to 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 limit the deadline miss ratio. At the same time, our work aims to ensure a certain freshness for the data accessed by timely transactions, even in the presence of unpredictable workloads. In this paper, 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. Section 5 shows the details of the simulation settings and the evaluation results. We conclude this paper by a discussion about this work and by the presentation of our future works.

2. Related works

In RTDBS, new techniques have been designed to manage real-time transactions [1] [9]. These techniques use feedback control scheduling theory 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 [7] [2]. In [2], Amirijoo et al. have proposed two algorithms: FCS-IC-1 and FCS-IC-2 (Feedback Control

* Real-Time DataBase Systems

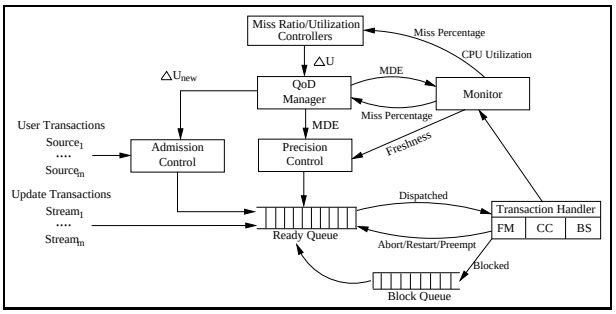


Figure 1. A Feedback Control Scheduling Architecture (FCSA).

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 [8], the authors suggest a QoS-sensitive approach, called QMF (a QoS-sensitive approach for Miss Ratio and Freshness guarantees) using a dynamic scheme to balance the user transactions and the update transactions workloads. To provide differentiated services in terms of miss ratio, they have extended in [5] 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.

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.

3.1. Data model

Data objects are classified into either real-time or non real-time data. A non real-time data is a classical data found in most of databases, whereas a real-time data has a validity interval beyond which it becomes useless. These data may change continuously to reflect the real world state (for example, the current temperature value). Each real-time data has a timestamp indicating the last observation of the real world state. In our model, we consider only real-time data.

Many versions of a real-time data item may be stored in the database and the number of versions considered may be either fixed or dynamically adjusted. To store a version, data freshness and the MDE parameters (see section 3.3) are taken into account.

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.

3.3. Performance metrics

Three main performance metrics are considered: $MissRatio$ (MR), $DataFreshness$ (DF), and $DataError$ (DE) [3].

1. $MissRatio$: the transactions miss ratio is defined as follows:

$$MR = 100 \times \frac{\#Tardy}{\#Terminated}(\%) \qquad (1)$$

   where $\#Tardy$ denotes the number of transactions that have missed their deadline, and $\#Terminated$ is the number of terminated transactions.

2. $DataFreshness$: in RTDBS, data can become outdated. To measure the freshness of a data item d_i in a RTDB, the notion of absolute validity interval (AVI) is used. A data version is related to a timestamp indicating the latest observation of this data item in the real world. d_i is considered temporally consistent (or fresh) if $(CurrentTime - Timestamp(d_i) \leqslant AVI(d_i))$. The database freshness can also be measured. It represents the ratio between fresh data and all the data in the database.
3. $DataError$: it represents the deviation between the current data value (CurrentValue(d_i)) and the updated value (UpdateValue(d_i)). The upper bound of the error is given by the maximum data

1 Database Administrator

2 transactions that have missed their deadlines

error (denoted MDE). DE of data version d_i is defined as:

$$DE_i = 100 \times \frac{CurrentValue(d_i) - UpdateValue(d_i)}{CurrentValue(d_i)}(\%) \quad (2)$$

We note that the quality of data (QoD) depends on its freshness and on DE, whereas the quality of transaction (QoT) depends on the Miss Ratio.

4. Multi-Versions Data-Feedback Control Scheduling Architecture

The multi-versions data - feedback control scheduling architecture 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 Feedback Control Scheduling Architecture (see Figure 1). In our work, we have added a new notion which consists on creating of data versions when a conflict (read-write) occurs between transactions. In this section, we give an overview of this technique and we present our approach. As shown in Figure 1, a RTDBS consists, among others, of an admission controller, a ready queue, a blocked queue, and a transaction handler.

For the QoS management, a monitor, a miss ratio controller, an utilisation controller and a QoD manager are added to the system in order to adjust its performances and to control the information flows. An admission controller (AC) is used to avoid system overload. Transactions handler provides a platform for managing transactions. Transactions are scheduled by a basic scheduler in the ready queue using the EDF scheduling policy [6]. At each sampling period, the monitor samples the system performance data from the transaction manager and sends it to the controller. The miss ratio and utilisation controller generates signals based on the sampled miss ratio and utilisation data [3]. 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 its type and priority. A low priority transaction can be scheduled if there is no ready transaction with higher priority to schedule. Note that since

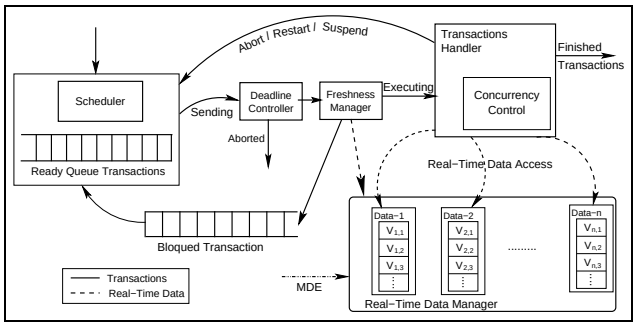


Figure 2. MVD Architecture.

the updated data are needed by user transactions, update transactions receive a higher priority than user transactions.

4.2. Deadline Controller

To control transaction validity [6] [4], the deadline controller (denoted DC), uses three controlled variables: transaction deadline, current time and minimal execution time. If the sum of the minimal execution time and the current time is lower than transaction deadline, then transaction is accepted, otherwise the transaction will be rejected. If this verification step succeed, then the transaction is transferred to data freshness manager (FM).

4.3. Freshness Manager

The freshness manager checks if the transactions served from the deadline controller access fresh data. freshness manager (FM) is used to provide better QoS in RTDBS where several transactions access to the same fresh data. 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, the transaction can be executed and then 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, then 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. In our system architecture, the FM checks the freshness of accessed data, i.e., AVI is not reached and remains fresh until the end of the transaction execution. Therefore, we must verify that AVI of accessed data is greater than the transaction deadline. To this purpose, we use MVD architecture. So, the freshness condition must be considered

by checking the freshness of the most recent data versions that are not write-locked by other transaction(s).

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 approachs that use the MVD technique. In the first approach, the maximum number of data versions is limited and is the same for all the data. In the second experiments, the number of versions is dynamically adjusted.

4.4.1. MVD with fixed number of data versions An update transaction always writes a RT data item while a user transaction reads this data item. In general, in RTDBS, only update transactions may modify real-time data. 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 concurrency control protocol. Then the risk that transactions miss their deadline increases. MVD technique 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.

4.4.2. MVD with dynamically adjusted number of data versions RTDB usually monitor the current real-world state using periodic updates. In this paper, we investigate several important problems to guarantee the desirable quality of real-time data services, in terms of timeliness, freshness, precision and mostly the decreasing of transaction miss deadline. 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 freshness and MDE criterion. The size of each versions queue, denoted SVQ, is dynamically adjusted, and it is defined as follows:

$$SVQ = IntegerPart[\frac{AVI_j}{Period_i}] \quad (3)$$

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

4.5. Concurrency Control with MVD

One of the most important issues in the design of RTDBS is the concurrency control component. It's objectives are (i) to control the interaction among concurrently update and user transactions and (ii) to maintain the database consistency.

Algorithm 1: 2PL-HP adapted protocol for MVD with fixed data version number

Tr_{up} : update transaction (write access).
Tr_{user} : user transaction (read access).
$Gr(Tr_{user})$: user transactions group accessing to the same data version.
$Gr_i(Tr_{user})$: i^{th} user transactions group.
nb_groups : number of transactions groups accessing a data version.

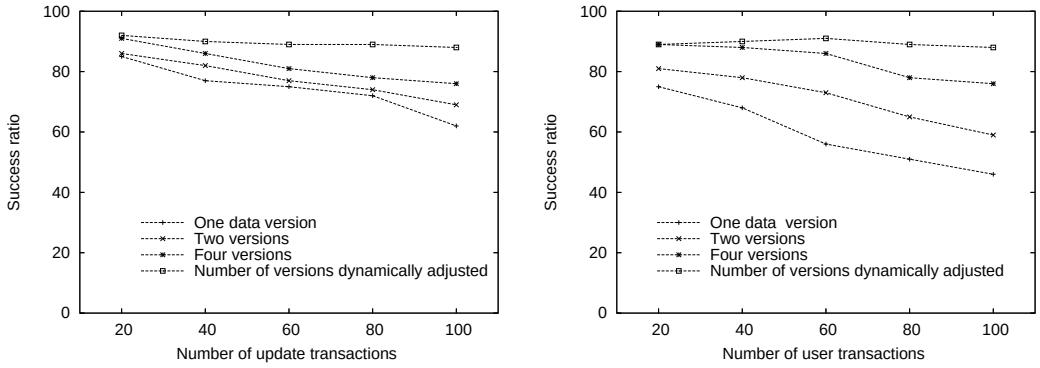
```
begin
    if Priority (Tr_up) < Priority (Gr(Tr_user)) then
        Tr_up is blocked waiting the release of version
    else
        for i from de 1 to nb_groups do
            if Priority (Tr_up) > Priority
            (Gr_i(Tr_user)) then
                abort and restart Gr_i(Tr_user)
       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

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 an update transaction (with lower priority), which wants to update the same data, 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 the transaction to miss their deadline. To address this problem, i.e. to aleviate this risk, we propose the MVD technique that allows user transactions to access an old data version when update transaction writes a new data version on the one hand, and an adapted 2PL-HP when the maximum number of versions is reached on the other hand. We present here the adapted 2PL-HP protocol adaptation (cf. Algorithm 1)[3].

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

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]
DBsize	Database size	300

Table 1. Parameters of Simulation



(a) For update transactions

(b) For user transactions

Figure 3. Simulation results of the MVD-FCSA.

5. Simulations and results

5.1. Simulations

We have studied and evaluated the behavior of an RTDBS according to a set of performance metrics. The performance evaluation is undertaken by a set of simulation experiments, where values of a set of parameters have been varied. Table 1 summarizes simulations parameters.

The simulated workload consists of update and user transactions, which access data. Update transactions present 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 operations in the transaction T_i and the execution time of an operation. The model consists of eight components. We have used two transaction generators, an update transaction generator (UpdateTransGen) which generates update transactions and an user transaction generator (UserTransGen) which generates user transactions. The workload model characterizes transaction in terms of the number of read/write operations. An update transaction can only write one data. Transactions are scheduled by a scheduler (Scheduler) in 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.

To check the freshness of accessed data, the freshness manager (FM) uses the AVI parameter. Transactions handler (TH) controls the execution of transactions. Concurrency controller (CC) uses the adapted 2PL-HP protocol to control the interaction between transactions. The real-time data management (RTDM) is the most important component of our model.

5.2. Summary of results and discussions

Figure 3 shows transaction success ratio, when a data item has a single version, two versions and 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 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 Figure 3(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 minimizes the transaction miss deadlines.

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.

5.3. Benefits of MVD-FCSA

MVD-FCSA is a good solution for QoS guarantees. By comparison with the classical feedback control scheduling architecture, it allows to limit the deadline miss ratio, to ensure freshness of 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 enhances the QoS.

6. 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. According to the experimental results, our simulations show that MVD-FCSA is useful and particularly successful with dynamically adjusted number of data versions.

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, we will deal with imprecise computing approach [2] applying to video contents. Derived data management is another interesting extension because derived data is of particular interest for some real-time database applications such as e-commerce and online stock trading systems.

References

[1] R. K. Abbott and H. Garcia-Molina. Scheduling real-time transactions: A performance evaluation. *ACM Trans. Database Syst.*, 17(3):513–560, 1992.

[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)*, pages 136–157, 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)*, pages 63–72, Portugal, 2003.

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

[5] 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*, pages 119–128, Washington D.C., April 29 - May 01 2002.

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

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

[8] 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.

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