A Concurrency Control Algorithm for Firm Real-Time Database Systems

Seok Jae Lee, Jae Ryong Shin, Seok Il Song, Jae Soo Yoo, and Ki Hyung Cho

Department of Computer & Communication Eng., Chungbuk National University, 48 Gaesin-dong, Cheongju, Chungbuk, Korea, 361-763
{sjlee, jrshin, prince}@netdb.chungbuk.ac.kr,
{yjs, khjoe}@cbucc.chungbuk.ac.kr

Abstract. Unlike a conventional database system, whose main objective is to provide fast average response time, Real-time database systems (RTDBS) may be evaluated based on how often transactions miss their deadline, the average lateness or tardiness of late transactions, the cost incurred in transactions missing their deadlines. Therefore, in RTDBS, transactions should be scheduled according to their criticalness and the tightness of their deadlines, even if this means sacrificing fairness and system throughput. And it always must guarantee preceding process of a high priority transaction (HPT) as the 2PL-HP (two phase locking with high priority) method. 2PL-HP resolves a conflict through aborting or blocking of a low priority transaction (LPT). If HPT is eliminated in a system because of its deadline missing, an unnecessary aborting or blocking of LPT is occurred. To resolve the problem, AVCC (alternate version concurrency control) algorithm that outperforms 2PL-HP was proposed. However, AVCC must always create the alternative version and have additionally a technique to manage complex alternative versions. In this paper, we propose a new efficient scheduling algorithm, called Multi-level EFDF that combines EFDF (earliest feasible deadline first) and Multilevel Queue scheduling algorithm very ably, and a concurrency control algorithm, called 2PL-FT that prevents wastes of needless resources and eliminates an unnecessary aborting or blocking of LPT. We show through the performance evaluation that our algorithm achieves good performance over the other existing methods proposed earlier.

1 Introduction

Real-time database systems (RTDBS) are database systems whose transactions are associated with timing constraints such as deadlines. Therefore transaction needs to be completed by a certain deadline. Besides meeting transaction timing constraints, RTDBS need to observe data consistency constraints as well. That is to say, unlike conventional database systems, whose main objective is to provide fast average response time, RTDBS may be evaluated based on how often transactions miss their deadline, the average lateness or tardiness of late transactions, the cost incurred in transactions missing their deadlines.

Therefore, in RTDBS, transactions should be scheduled according to their criticalness and the tightness of their deadlines, even if this means sacrificing fairness and system throughput. And it always must guarantee preceding process of a high priority

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 524–532, 2003.

transaction (HPT) as 2PL-HP (two phase locking with high priority). 2PL-HP resolves a conflict through aborting or blocking of a low priority transaction (LPT). However, if HPT is eliminated in a system because of its deadline missing, an unnecessary aborting or blocking of LPT is occurred. To resolve the problem, AVCC (alternate version concurrency control) that outperforms 2PL-HP in simulations for firm real-time transaction was proposed. However, the algorithm must always create the alternative version and have additionally a technique to manage complex alternative versions.

The objective of this paper is to present a new concurrency control algorithm, called 2PL-FT. In addition, we propose a new scheduling algorithm, called Multi-level EFDF that combines EFDF (earliest feasible deadline first) and Multilevel Queue scheduling algorithm very ably.

The rest of the paper is organized as follows. In Sect. 2, we review related works and Sect. 3 presents our proposed scheduling and concurrency control algorithms. And Sect. 4 shows experimental results. Finally, Sect. 5 gives concluding remarks.

2 Related Works

Concurrency Control algorithms for RTDBS are in general some extensions or combinations of traditional concurrency control techniques, i.e. two phase locking (2PL), optimistic concurrency control (OCC) or timestamp ordering (TO), which guarantee a serialization order among conflicting transactions. Most of these algorithms were developed only for firm deadline RTDBS which is the simplest environment [4]. The most important problem faced in RTDBS concerns the degradation of system performance due to aborts and restarts of transactions. These restarts are caused by concurrency control protocols trying to resolve conflicts between transactions[2].

In non real-time (conventional) database systems all transactions have the same priorities. Blockings and restarts are the usual approaches to resolve access conflicts. 2PL and OCC are representative techniques used for blocking and restart to resolve data conflicts, respectively. It is clear that 2PL-HP has wasted restart and wasted wait due to the semantics of the firm deadline while OCC has wasted execution. A wasted restart happens if HPT aborts LPT and then HPT is discarded as it misses its deadline. In other words, a transaction which is later discarded can cause restarts. A wasted wait happens if LPT waits for the commit of HPT and later HPT is discarded as it misses its deadline. In other words, a transaction which is later discarded can cause wait of a conflicting LPT. And a wasted execution happens when LPT in the validation phase is restarted due to a conflicting HPT which has not finished yet.

AVCC proposed in [3] always outperforms 2PL-HP in simulations for firm deadlines. Because HPT can proceed without aborting conflicting LPTs, they use a deferred update policy which updates on local copies of a data item and makes them global at commit time. It only needs to stop LPT until the completion (commit or abort) of the conflicting HPT. If HPT is discarded by missing its deadline it can execute the stopped LPT by resuming it. This is what it refers to as the stop/resume deferred restart policy. But AVCC always have to maintain the stopped version and

initiate the restarted version of LPT. Also each transaction might have multiple DR (deferred restart) versions and a single IR (immediate restart) version.

3 The Proposed Concurrency Control Algorithm

In this section, we introduce architecture of a new real-time database model and describe a feasibility test method, scheduling and concurrency control algorithms in detail.

3.1 Architecture

Fig. 1 shows architecture of a new real-time database model. The model is composed of a module to initialize information that is required to test feasibility of transactions, a real-time scheduling module, and a real-time concurrency control module. In an application, the number of supplied functions and transactions is not infinite. Therefore, it is possible to classify transactions into multiple types. In the proposed model, we separated transactions according to the types, and assigned a real-time property such as firm or soft and criticalness to each transaction.

In the init module, it acquires a real-time property and criticalness according to the type of a new transaction and decides the expected computation time by the average of total computation times of the transaction and the average of total system loads. New transactions are classified in a level by criticalness and their order is scheduled by the deadline in each level. The CPU scheduler allocates the CPU to the transaction with the earliest feasible deadline in the highest level. Therefore, it guarantees that important transactions are executed in the first place.

All firm real-time transactions that are released are dependent on the feasibility test. Through the test, a firm real-time transaction that has already missed or is about to miss its deadline is eliminated in a system. And the firm HPT that is real-time transaction with the higher priority at the time of conflict is dependent on the feasibility test. If the firm HPT is not feasible, the infeasible HPT is eliminated from the system. Therefore, it prevents wastes of needless resources and eliminates an unnecessary aborting or blocking of LPT that is a transaction with the lower priority.

3.2 Feasibility Test

The feasibility test is that it compares an expected computation time (Ei) of a transaction Ti, with its deadline (Di). For example, the feasibility test of a transaction Ti is executed as follows:

$$(D_i - t) \geq (E_i - C_i) \tag{1}$$

Where t is the current time and C_i is the real computed time of T_i until now. The E_i is computed based on the average computation time C_{avg}, the minimum computation time

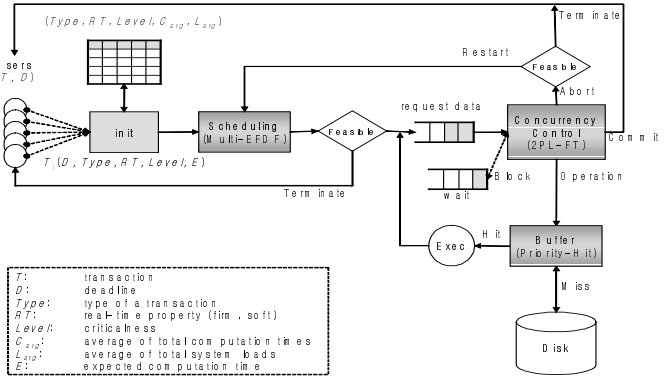

Fig. 1. Architecture of our model

C_{min}, the average system load L_{avg} and the current system load L_i. We propose two methods for computation of E_i.

$$E_i = \frac{(C_{\min} + C_{avg})}{2} \times \frac{L_i}{L_{avg}} \quad \ldots \ (Mid\text{-}type) \tag{2}$$

$$E_i = \max\left(C_{\min}, C_{\min} \times \frac{L_i}{L_{avg}} \right) \ldots \ (Min\text{-}type) \tag{3}$$

In the first method, called *Mid-type*, we use the intermediate value of C_{min} and C_{avg}. In the second method, called *Min-type*, we use a value that is less than or equal to C_{min}. In each method, we use (L_i/L_{avg}) for reflecting fluctuation rates of the system load. In case of *Mid-type*, transactions that its C_i is less than E_i can be damaged though the number of transactions is very few. But *Min-type* method overcomes that problem though the elimination ratio for infeasible transactions is decreased. Therefore we use *Mid-type* method for non-critical transactions and *Min-type* method for critical transactions. Note that non-critical transactions and critical transactions can be classified by a level of transactions.

3.3 Multi-level EFDF

The proposed Multi-level EFDF scheduling algorithm that combines EFDF (earliest feasible deadline first) and Multilevel Queue scheduling algorithm very ably eliminates a transaction that is about to miss its deadline and allocates CPU to the firm real-time transaction with the earliest deadline in the highest level. Accordingly transactions in lower levels must be in the ready queue until higher levels are empty. And if the priority of a new transaction is higher than the priority of the currently running transaction, CPU can be preempted by the new transaction. Fig. 2 shows the architecture of Multi-level EFDF.

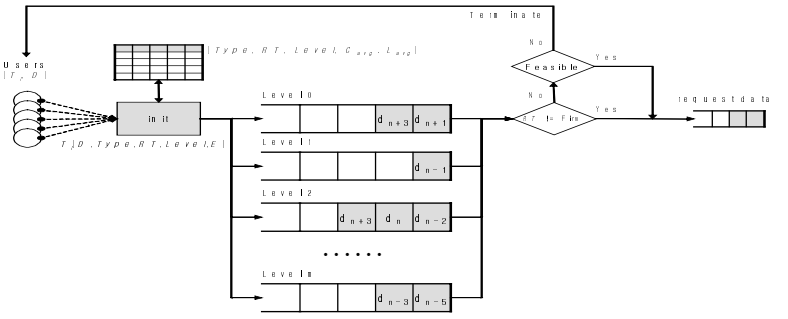

Fig. 2. Architecture of Multi-level EFDF

3.4 2PL-FT (Two Phase Locking with Feasibility Test)

The proposed real-time concurrency control algorithm, called 2PL-FT, eliminates an unnecessary aborting or blocking of LPT. And then it prevents wastes of needless resources and reduces the number of missed deadlines. To this end, the process of our algorithm is as follows:

(1) If a conflict occurs between a transaction T_H that already is locking on a data object O and a transaction T_R that is requesting the lock, it chooses as candidate transaction T_C the HPT between T_H and T_R.
(2) It decides whether T_C is about to miss its deadline or not (It is feasibility test for T_C).
(3) If T_C is infeasible, T_C is immediately eliminated in the system and LPT is continuously executed or acquires the lock. Table 1 shows the plan of the conflict resolution in detail.
(4) Otherwise T_C acquires the lock or is continuously executed. In contrast, LPT is restarted or blocked.

Table 1. Method to solve the conflict

Conditions		T_C is feasible	T_C is infeasible
$T_C = T_H$	T_H	Continuously executed	Eliminated
	T_R	Blocked	Acquires the lock
$T_C = T_R$	T_H	Restarted	Continuously executed
	T_R	Acquires the lock	Eliminated

For example, if T_C that is requesting the lock (T_R) is infeasible, the transaction is immediately eliminated in the system and LPT (T_H) is executed continuously. And if T_C with the lock (T_H) is infeasible, the transaction is eliminated and LPT that is requesting the lock (T_R) is executed.

Fig. 3 shows architecture of 2PL-FT, respectively.

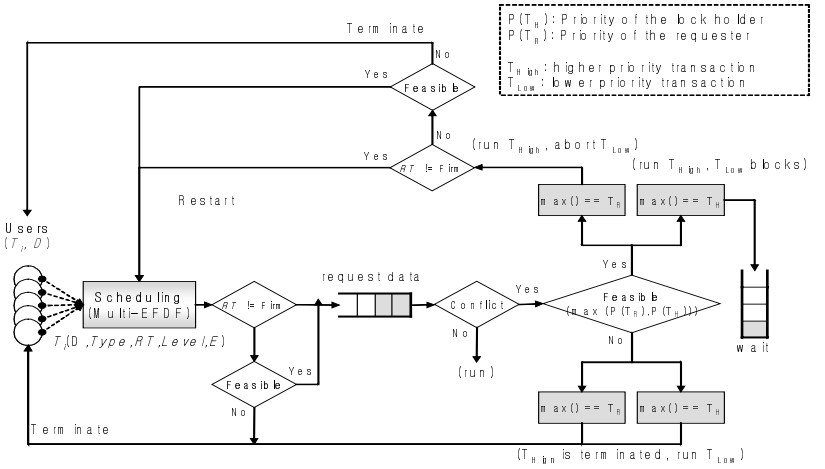

Fig. 3. Architecture of 2PL-FT

Fig. 3 shows that all firm real-time transactions that are released and firm HPT that is conflicted with LPT are dependent on the feasibility test. Through the test, (1) a firm real-time transaction that has already missed or is about to miss its deadline is eliminated in the system. (2) And if a firm HPT is infeasible, the HPT is eliminated too from the system. In contrast, LPT is not terminated or blocked but is executed continuously or acquires the lock.

4 Performance Evaluations

In this section we evaluate the performance of the proposed 2PL-FT algorithm with respect to restarting ratio and deadline missing ratio of transactions. The experiments were performed on a Pentium-III with 256Mbytes of main memory. The input and design parameters are shown in Table 2. These parameters are based on that of AVCC.

Table 2. The input and design parameters

Parameters	Description	Value
DBSize	The number of pages in the databases	1,000 pages
CPUtime	The processing time of a page	10 ms
NumLevels	The number of levels	20 levels
NumTypes	The number of transaction types	100 types
TransSize	The number of pages accessed per transaction	8~24 pages
Slackfactor	The tightness/slackness of deadlines	100~600 %

First of all, we evaluate an efficiency of feasibility test method with *Mid-type* or *Min-type* that are method for computing the expected computation time (E_i). Fig. 4 shows transaction ratio with missed deadline according as transaction arrival ratio is increased. In Fig. 4, *Mid-type* uses the intermediate value of C_{min} and C_{avg} to compute E_i and *Min-type* uses a value that is less than or equal to C_{min}. The "Average only" uses C_{avg} only.

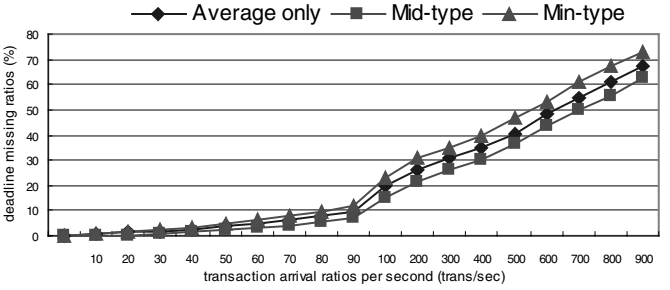

Fig. 4. Deadline missing ratios by feasibility test methods

The *Mid-type* and Average only show better performance than *Min-type* method. Because *Min-type* method applies the minimum execution time to the feasibility test, it cannot eliminate well firm real-time transactions that are about to miss their deadline.

Therefore it cannot fully prevent wastes of needless resources and eliminate an unnecessary aborting or blocking of LPT. Nevertheless we use *Min-type* for critical transactions because it is wastes of needless resources that the firm real-time transaction with the execution time less than the minimum value reflected fluctuation rate of the system load is executed uselessly. On the other hand *Mid-type* decreases the deadline missing ratio of firm real-time transactions very well. The reason is that many of firm real-time transactions that cannot be committed normally in their deadline are discarded.

From now on, we investigate restarting transaction ratios and deadline missing ratios of 2PL-FT and AVCC algorithms. In these experiments we use the feasibility test with *Mid-type* method only. Fig. 5 shows that the restarting ratio of 2PL-FT is about 38% less than that of AVCC, because AVCC always have to maintain the stopped version and initiate the restarted version of LPT conflicted with HPT. And then, we can see that the heavier resource competition, the more increased the restarting transaction ratio of AVCC. However, in 2PL-FT, LPT is restarted in case HPT is only feasible. Finally, deadline missing ratios of two algorithms are shown in Fig. 6 The deadline missing ratio of 2PL-FT is about 27% less than that of AVCC. The reason is as follows:

(1) Unnecessary restarting transactions of AVCC obstruct harmonious execution of normal transactions. On the other hand, infeasible transactions eliminated by 2PL-FT help an execution of the other transactions.

(2) AVCC with stop/restart version of LPT prevents an unnecessary aborting of LPT, whereas 2PL-FT with feasibility test of HPT prevents not only an unnecessary aborting of LPT but a blocking of LPT.

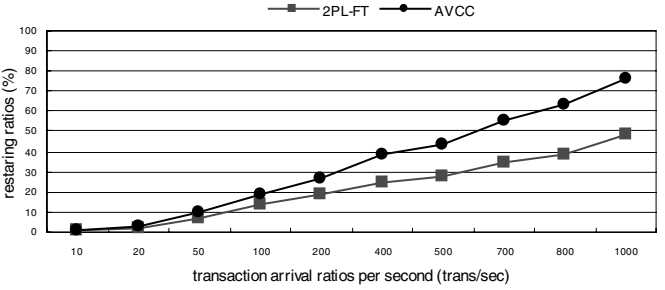

Fig. 5. Restarting transaction ratios

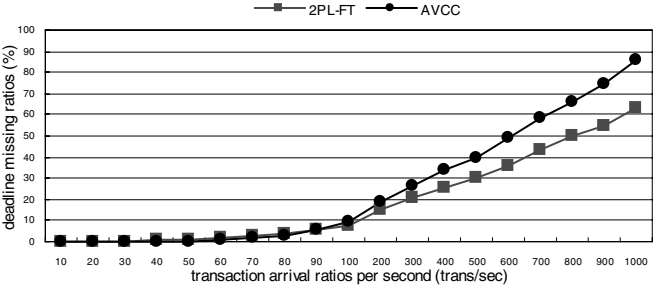

Fig. 6. Deadline missing ratios

5 Conclusions

In AVCC, HPT can proceed without aborting conflicting LPTs. They use a deferred update policy which updates on local copies of a data item and makes them global at commit time. But AVCC always have to maintain the stopped version and initiate the restarted version of LPT. Also each transaction might have multiple DR versions and a single IR version. In addition, AVCC must have additionally a technique to manage complex alternative versions.

In this paper, we have proposed a new efficient real-time concurrency control algorithm, called 2PL-FT that need not have alternative versions and a scheduling algorithm, called Multi-level EFDF that assigns the highest priority to the transaction with the earliest deadline in the highest level. And we have proposed the feasibility test to prevent wastes of needless resources. We have showed through the performance evaluation that the deadline missing ratio of 2PL-FT is about 27% less than that of AVCC. In the future research, we will adapt our idea to optimistic concurrency control algorithms such as OPT-BC, OPT-Sacrifice, OPT-Wait, Wait-50, Wait-X.

Acknowledgment. This work was supported by Korea Research Foundation Grant (KRF-2002-074-DS2501).

References

1. Abbott, R., and Garcia-Molina, H.: Scheduling Real-Time Transactions: A Performance Evaluation, Proceedings of the 14th Conference on Very Large Database Systems (1988)
2. Ben Kao and Hector Garcia-Molina: An Overview of Real-Time Database Systems, in Sang H. Son, editor, Advances in Real-Time Systems, chapter 19, Prentice Hall (1995)
3. D. Hong: Alternative Version concurrency Control Method for firm real-time database systems, Korea Information Processing Society, Vol. 5, No. 6 (1998) 1377–1389
4. D. Hong, Sharma Chakravarthy, Theodore Johnson: Locking Based Concurrency Control for Integrated Real-Time Database Systems, RTDB '96 (1996) 138–143
5. Haritsa, J. R., Carey, M., Livny, M.: Data Access Scheduling in Firm Real-Time Database Systems, Journal of Real-Time Systems, No. 4 (1992) 203–241
6. Haritsa, J. R., Carey, M., Livny, M.: Value-Based Scheduling in Real-Time Database Systems, The VLDB Journal, Vol. 2, No. 2 (1993) 117–152
7. Haritsa, J. R., Livny, M., Carey, M.: Earliest Deadline Scheduling for Real-Time Database Systems, Proceedings of IEEE Real-Time System Symposium (1991) 232–242
8. Haritsa, J. R., Livny, M., Carey, M.: On Being Optimistic about Real-Time Constraints, Proceedings of the 9th ACM Symposium on Principles of Database Systems (1990)
9. Huang, J., Stankovic, J., Ramamritham, K., Towsley, D.: Priority Inheritance in Soft Real-Time Database, Journal of Real-Time Systems, Vol. 4, No. 3 (1992) 243–268
10. Piotr Krzyzagorski: Concurrency Control in Real-Time Database Systems, EDBT Ph.D. Workshop (2000)
11. Ulusoy, O.: Processing of Real-Time Transactions in a Replicated Database Systems, Journal of Distributed and Parallel Database, Vol. 2, No. 4 (1994) 405–436