Journal of Computer Science 1 (4): 471-476, 2005
ISSN 1549-3636
© Science Publications, 2005

Real-time Optimistic Concurrency Control based on Transaction Finish Degree

[1]Han Qilong , [1,2]Hao Zhongxiao
[1]Department of Computer Science, Harbin Institute of Technology, Harbin 150000, China
[2]Department of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China

Abstract: Concurrency control is one of the main issues in the studies of real-time database systems. Optimistic concurrency control algorithms have the attractive properties of being non-blocking and deadlock-free. However, they have the problems of late conflict detection and transaction restarts. Although the number of transaction restarts is reduced by dynamic adjustment of serialization order in real-time database systems, they are still some unnecessary transaction restarts. In this study, we propose a new method called Transaction Finish Degree (TFD) and a new Multiversion Optimistic Concurrency Control algorithm based on TFD (MVOCC-TFD), which can reduce the number of unnecessary restarts. Theoretical analysis and experimental results demonstrate that the new algorithm can outperform the previous ones.

Key words: real-time database, concurrency, scheduling algorithm, transaction finish degree

INTRODUCTION

Real-Time Database Systems (RTDBS) are transaction processing systems that attempt to satisfy the timing constraints associated with each incoming transaction. In RTDBS, the primary performance criterion is timeliness level, not average response time or throughput. Thus, scheduling of transactions is driven by priority considerations rather than fairness considerations. Many researches have been devoted to design appropriate concurrency control algorithms for RTDBS. Most concurrency control algorithms for RTDBS are based on one of the following basic concurrency control mechanisms: locking[1-3] or optimistic concurrency control[4-6].

Optimistic concurrency control protocols have the nice properties of being non-blocking and deadlock-free. These properties make them especially attractive for RTDBS. As conflict resolution between transactions is delayed until transactions are close to completion, there will be more information available for making the choice in resolving the conflict. However, the problem with optimistic concurrency control protocols is the late conflict detection, which leads to huge overhead because some near-to-complete transactions have to be restarted. So it is important to design new methods to minimize the number of transaction restarts. The OCC-DA[7], OCC-TI[8] and OCC-DAT[9,10] concurrency control protocols are based on Dynamic Adjustment of Serialization Order (OCC-DASO), avoiding some unnecessary restarts. Hence, the number of transaction restarts with these protocols is smaller than that with other optimistic concurrency control protocols, such as OCC-BC[11], OCC-WAIT[7] and WAIT-X[1,7]. Unfortunately, there are still some unnecessary restarts with these protocols, especially the near-to-complete transaction restarts.

In this study, we propose a new method, called Transaction Finish Degree (TFD), which can further avoid the near-to-complete transaction restarts. Based on TFD, we also develop a multiversion optimistic concurrency control protocol, called MVOCC-TFD. With the new protocol, the number of transaction restarts is smaller than that with OCC-DASO.

PROBLEMS WITH OCC-DASO

In this section, we will motivate our work by illustrating the problems associated with OCC-DASO algorithms. The validation algorithm of OCC-DASO can be simply written as:

$$RS(T_v) \cap WS(T_a) \neq \emptyset,\ T_v \rightarrow T_a,$$
$$WS(T_v) \cap RS(T_a) \neq \emptyset,\ T_a \rightarrow T_v,$$
$$WS(T_v) \cap WS(T_a) \neq \emptyset,\ T_v \rightarrow T_a .$$

Although these algorithms provide dynamic adjustment execution order to decrease the number of unnecessary transaction restarts, they do not resolve the problem of near-to-finished transaction restarts and can not work under serious conflicts condition.

We will use two examples to illustrate such problems. Before that, we first introduce a set of

Corresponding Author: Han Qi-Long, Department of Computer Science, Harbin Institute of Technology, Harbin 150000, China Phone: 86-451-13936101522

notations. We use $r_i[x]$ and $w_i[x]$ to denote read and write operation, respectively, on data object *x* by transaction T_i and let d_i, c_i and v_i denote deadline, commitment and validation of transaction T_i. Each Transaction T has a Read Set, RS(T) and a Write Set, WS(T), when T starts its execution.

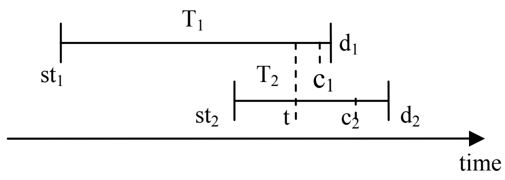

Fig. 1: OCC-DASO data conflict decision

Example 1: Consider transactions T_1 and T_2,
T_1: $r_1[a]w_1[b]w_1[c]r_1[d]$
T_2: $r_2[b]r_2[c]$.
H_1: $r_1[a]\ w_1[b]\ r_2[b]\ w_1[c]r_2[c]v_2\ c_2\ r_1[d]v_1\ c_1$.

The execution profile of two concurrently executing transactions, T_1 and T_2, is shown in Fig. 1. T_1 has a start time st_1 and deadline d_1, similarly T_2 has a start time st_2 and deadline d_2. Suppose at time $t = v_2$, when transaction T_1 is close to completion, transaction T_2 reaches its validation point and detects a conflict with T_1. Following the OCC-DASO algorithms, the serialization order is adjusted as T_2T_1, and then transaction T_1 has no chance of meeting its deadline.

Example 2: Consider transactions T_1 and T_2,
T_1: $r_1[a]w_1[b]w_1[c]$
T_2: $r_2[b]r_2[c]w_2[a]$
H_1: $r_1[a]\ w_1[b]\ r_2[b]\ r_2[c]\ w_2[a]v_2\ w_1[c]v_1$.

The validation transaction T_2 has serious conflict with active transaction T_1 for $RS(T_2) \cap WS(T_1) \neq \emptyset$ and $WS(T_2) \cap RS(T_1)\ )\neq \emptyset$. The OCC-DASO algorithms can not deal with this condition.

The above examples show the problems of OCC-DASO algorithms. To overcome these problems, we propose a new method, which can resolve the problem of near-to-complete transaction restarts and handle the serious conflict transaction scheduling problem as well.

A NEW REAL-TIME MULTIVERSION CONCURRENCY CONCTROL PROTOCOL

As explained in the previous sections, although the OCC-DASO algorithms highlight some major strengths of optimistic concurrency control in real-time database systems, there remains potential for improving its performance. In this section, we present a new multiversion optimistic concurrency control algorithms based on the transaction finish degree, called MVOCC-TFD.

Transaction finish degree and multiversion: The objectives of concurrency control in RTDBS are to avoid inconsistent retrievals and to preserve the correct state of the database. Serializability is the definition of correctness for concurrency control in database systems. To describe serializability, we use similar definitions as presented in[12].

Definition 1: A Multiversion(MV) History H is serial if for any two transactions, T_i and T_j, that appear in H, either all of T_is operations precede all of T_js or vice versa.

Definition 2: A serial MV history H is 1-serial (or one-copy serial) iff for all i, j and some data item x, if T_i reads the value of x created by T_j, then i = j, or T_j is the last transaction preceding T_i that writes into any version of x.

Definition 3: An MV history H is one-copy serializable (or 1SR) if its committed projection, C(H), is equivalent to a 1-serial MV history, where C(H) is the history obtained from H by deleting all operations that do not belong to committed transactions in H.

In addition, we introduce the concept of transaction finish degree and its relevant properties. A set of notations are used in the following definitions.

st: Starting time of a transaction
t: Current time
ft: Estimated accomplishing time of a transaction
dt: Deadline of a transaction
T_y: Validation transaction

Definition 4: Value function V(T) is the value of system about current time when a transaction accomplished. Formally, $V(T)=c(w_1(t\text{-}st)\text{-}w_2dt)$, where w_1 and w_2 are the weights.

According to the value function, Critical attribute (Ca) of a transaction may be higher, normal or lower. If $V(T)\geq C1$, then Ca is higher, else if $C1>V(T)>C2$, Ca is Normal, otherwise Ca is Lower. C1, C2 is the threshold value.

Definition 5: Deferrable time (sdt) is the time interval between deadline and estimated accomplishing time of a transaction. Formally, sdt = dt – ft. All the operations of a validation transaction have been accomplished at validation time, hence, $sdt_v = dt_v - t$.

Definition 6: At current time t, the finished ratio of transaction T**,** FR(T), is defined by the ratio of the time interval between deadline and t and the time interval between t and staring time. Formally, FR(T) = (dt- t)/(t-st).

Definition 7: Conflict transactions set $CTS(T_v)$ contains all transactions which have conflicts with T_v. The conflict transactions set can be divided into two classes in terms of their relative deadline to that of T_v, namely $CHS(T_v)$ and $CLS(T_v)$

$$CHS(T_v) = \{T \mid T\in CTS(T_v), dt_T < dt_{Tv}\} \text{ and}$$
$$CLS(T_v) = \{T \mid T\in CTS(T_v), dt_T \geq dt_{Tv}\}$$

Definition 8: Transaction Finish Degree (TFD) of T_v is the ratio of $FR(T_v)$ to $FR(T_a)$, where, $T_a \in CHS(T_v)$.

Definition 9: For any given transactions T_i and T_j ($i \neq j$), T_i is serious conflict with T_j if

$$RS(T_i) \cap WS(T_j) \neq \emptyset \text{ and}$$
$$WS(T_i) \cap RS(T_j) \neq \emptyset$$

MVOCC-TFD protocol: Our protocol is based on OCC-DASO and we assume that each transaction must meet the following three conditions.

- A real-time transaction can commit wile it does not conflict with other transaction.
- A real-time transaction T misses its deadline iff $sdt_T \leq 0$
- A transaction can be delayed to commit after validation phase.

We designed four rules based on the notion of TFD, as presented below.

- R1. For each transaction T, its read set RS(T) and its write set WS(T) are declared when it starts.
- R2. The validation phase is divided into preparation phase and adjustment phase. In preparation phase, TFD values are computed and serious conflict is checked. The reordering of transaction commitment is performed in adjustment phase.
- R3. If $TFD_v>1$, there is no serious conflict and the active transaction T_a is near-to-completed, then commitment order is adjusted to T_a, T_v.
- R4. If $TFD_v\leq1$, there is no serious conflict and the active transaction T_a is not near-to-completed, then Ca of the validation transaction and the active conflict transaction must be checked. If $Ca_v\geq Ca_a$, then the commitment order is adjusted to T_v, T_a, else T_a, T_v.
- The next two rules describe how the serious conflict is resolved by multiversion method.
- R5. If T_v is serious conflict with active transaction T_a and $RS(T_v) \cap RS(T_a)=\emptyset$, then T_v reads the data item version written by transaction T_a and T_v commits immediately after T_a.
- R6. If T_v is serious conflict with active transaction T_a and $RS(T_v) \cap RS(T_a)\neq\emptyset$, we assume the interaction set is $RR=\{x|x\in WS(T_k), T_k\in CTS(T)\}$, then T_k is adjusted to the last transaction preceding T_v and T_a. According to the relation of T_v and T_a, transactions can be continuously committed as T_k, T_a, T_v or T_k, T_v, T_a.

Correctness and properties

Lemma 1: The MVOCC-TFD protocol does not further delay the commitment of validation transaction than the OCC-DASO.

Proof: The MVOCC-TFD protocol differs from OCC-DASO method in the introduction of TFD and serious conflict checking. By definition of TFD, the active transactions whose deadline earlier than that of validation transaction are first considered. Further, the new protocol can resolve the serious conflict problem while the OCC-DASO method cannot. Therefore, the commitment of validation transaction is not being delayed by MVOCC-TFD protocol.

Lemma 2: MVOCC-TFD is a protocol in favor of near-to-complete transaction.

Proof: According to R3 of the MVOCC-TFD protocol, the near-to-complete transaction is committed first. Therefore, the protocol is in favor of the transaction that is near to completion.

Lemma 3: All the transactions which can be scheduled by OCC-DASO can be scheduled by MVOCC-TFD.

Proof: Since the MVOCC-TFD protocol is based on the OCC-DASO method, the new protocol contains the three validation rules of OCC-DASO method.

Theorem 1: MVOCC-TFD can avoid more unnecessary restarts than OCC-DASO.

Proof: First, by lemma 1, the commitment of validation transaction is not being delayed by MVOCC-TFD protocol. Then, by lemma 3, if OCC-DASO needs not restart transactions, then MVOCC-TFD can also avoid restarts in the same context. Moreover, MVOCC-TFD can avoid the serious conflict transaction restarts, which cannot be avoided by OCC-DASO.

Theorem 2: For each execution generated by MVOCC-TFD protocol, there is an equivalent execution result of a serialization.

Proof: If there is no serious conflict among the transactions, the theorem can be supported by the OCC-DASO serialization theory. Otherwise, if the serious conflict occurs among transactions, we consider two cases covered by R6 and R7. The read and write operations of the transactions are restricted by multiversion theory. Therefore, its committed projection is equivalent to a 1-serial MV history and thus the algorithm can produce serializable histories.

Examples: We will use the two examples show how MVOCC-TFD protocol can avoid unnecessary restarts. For example 1, we first compute the value of TFD.

$$TFD = \frac{FR(T_2)}{FR(T_1)} = \frac{\left.dt_2 - t \middle/ t - st_2\right.}{\left.dt_1 - t \middle/ t - st_1\right.} > 1,\ T_1 \in CHS(T_2).$$

Following the MVOCC-TFD protocol, the commitment order is T_1, T_2. Hence,

H: $r_1[a]\ w_1[b]\ r_2[b]\ w_1[c] r_2[c] v_2\ r_1[d] v_1\ c_1\ v_2\ c_2$.

There is no transaction restart and both transactions can meet their deadlines.

For example 2, the MVOCC-TFD protocol can schedule T_1 and T_2 correctly after checking the TFD and the interaction set of T_1 and T_2.

PERFORMANCE EVALUATION

In this section, we present simulation results to show the performance of our protocol MVOCC-TFD compared with three other concurrency control protocols.

Simulation model: We have carried out a set of experiments in order to demonstrate feasibility of our algorithms in practice. RTMMDBTP is architecture for real-time, main-memory database management systems. This platform consists of a main-memory database and optimistic concurrency control protocols. All experiments were conducted on the RTMMDBTP prototype database and were executed on a 2.8GHz Pentium4 processor with 512MB main memory using Windows2000 operating system. In this study, a transaction is discarded immediately after it misses its deadline. All parameters are shown in Table 1.

Table 1: Parameters

Parameter	Description	Value
DBSize	Number of data objects in database	9000
ArrRate	Average arrival rate of transactions	100-500
Deadline	All transactions are firm transactions	Firm
CPUTime	CPU computation time	10ms
MinSlack	Minimum slack factor	2
MaxSlack	Maximum slack factor	4
WOP	Write only probability	0-100%
ROP	Read only probability	0-100%

The primary performance metric is the percentage of transactions which miss their deadlines, denoted as Miss Ratio (MR). We will compare the MR of MVOCC-TFD protocol with that of OCC-DA, OCC-TI and OCC-DATI protocols. We use the following formula for deadline-assignment to a transaction:

Deadline = st(T) + et(T)*lock_factor(T), where

- st(T) and et(T) denote the starting time and estimated execution time, respectively
- lock_factor(T) = 1 − (the number of data items accessed by T) / (the total number of data items needed by T)

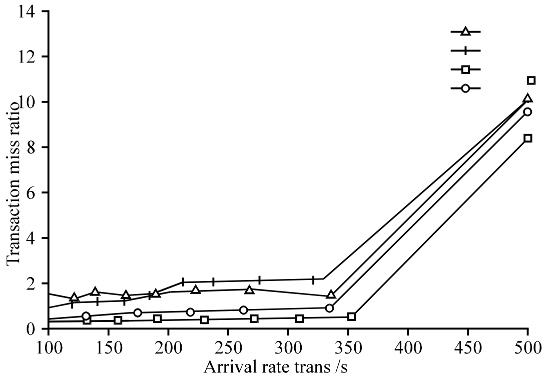

Fig. 2: Fraction of 10% write transaction

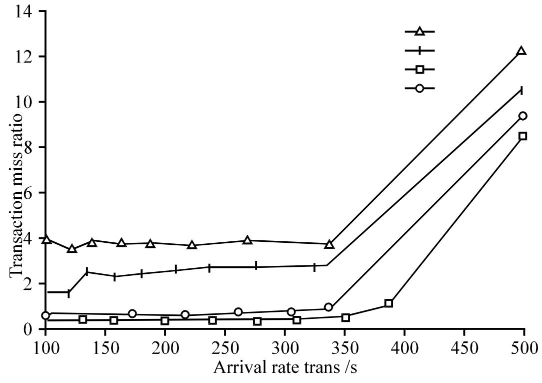

Fig. 3: Fraction of 40% write transaction

Performance analysis: In the first set of experiments, a fixed fraction of write transactions has been used while varying the arrival rate from 100 to 500 transactions per second. In Fig. 2 and Fig. 3, the fraction of write transactions is 10 and 40%.

From results in Fig. 2 and Fig. 3, we can conclude that the more transactions arrived per second, the higher transaction restarts ratio. We can also observe that the restarts ratio is higher when increasing the fraction of write transactions. Since OCC-DA and OCC-TI do not have effective methods to manage write operations, they are adversely affected when the write transactions fraction is increased. Thanks for introducing timestamps; the OCC-DATI protocol can reduce write conflicts to some extent. The MVOCC-TFD protocol outperforms all the other protocols. It can reduce the unnecessary restarts of near-to-completed transactions because of the multiversion mechanism.

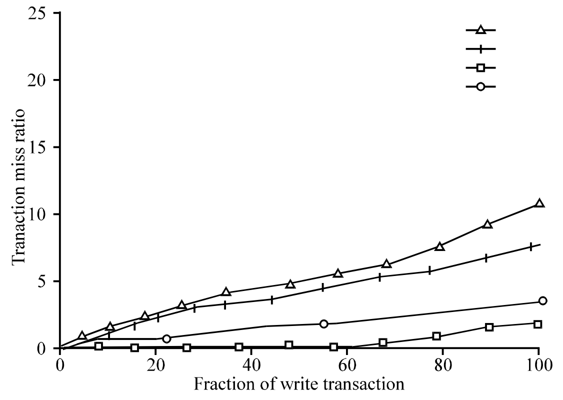

Fig. 4: Arrival rate 200 trans/s

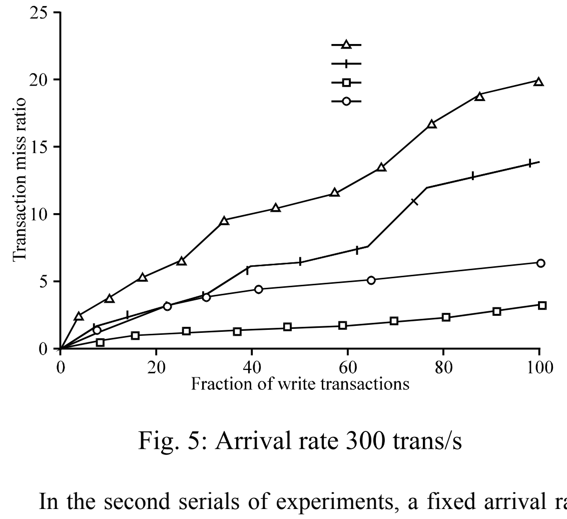

Fig. 5: Arrival rate 300 trans/s

In the second serials of experiments, a fixed arrival rate of transactions has been used with the fraction of write transactions varying from 10 to 100%. Figure 4 and 5 show that the performance of MVOCC-TFD is better than OCC-DA, OCC-TI and OCC-DATI protocol. When there are read only transactions, all four protocols can schedule effectively. While increasing the write transactions, the performance of MVOCC-TFD is better than others.

CONCLUSIONS

Although the number of transaction restarts is reduced by dynamic adjustment of serialization order, there are still some unnecessary restarts. In this study, we propose a new method called transaction finish degree and a new optimistic concurrency control protocol MVOCC-TFD. It reduces the number of near-to-completed transactions restart. In addition, by adopting multiversion mechanism, MVOCC-TFD resolved the serious conflict problem which cannot be handled by OCC-DASO. Furthermore, a wealth of detailed experiments show the number of transaction restarts with MVOCC-TFD is less than that with OCC-DASO. To conclude, our proposed MVOCC-TFD protocol outperforms OCC-DASO.

The described in this study can be extended in several ways. First, we have not considered the nested transactions and distributed requirements, although these may be possible from the application specification. Second, we have restricted ourselves by not distinguishing temporal and non-temporal data management. By exploiting the semantic information in transactions and the type of data they access, the protocol could be extended to provide a higher degree of concurrency. Finally, in this study, we considered the problem of real-time concurrency control in a database system. There are other issues need to be considered in designing a comprehensive RTDBS, including architectural issues, recovery and data models. We will integrate these issues in our future research plan.

REFERENCES

1. Kao, B. and H. Garcia-Molina, 1995. An overview of real-time database systems. Advances in Real-Time Systems. Prentice Hall, pp: 463-486.
2. Huang, J., J. Stankovic, K. Ramamritham and D. Towsley, 1991. On using Priority inheritance in real-time databases. In Proc. of the 12th IEEE Real-Time Systems Symposium. IEEE Computer Society Press, San Antonio, Texas, USA, pp: 210-221.
3. Lam, K.W. and S.L. Hung, 1997. Integrated concurrency control protocol for hard real-time database systems. Computers and Digital Techniques, 7: 214-218

4. Chiu, A，B. Kao and K. Yiu Lam, 1997. Comparing two-phase locking and optimistic concurrency control protocols in multiprocessor Parallel and Distributed Real-Time Systems, 4: 141-148.
5. Huang, J., J. A. Stankovic and K. Ramamritham, 1991. Experimental evaluation of real-time optimistic concurrency control schemes. In Proc. the 17th VLDB Conf., pp: 35~46.
6. Lee, B. and Hwang, 2002. Optimistic concurrency control based on timestamp interval for broadcast environment. In Proc. the 6th East European Conference in Advances in Databases and Information Systems.
7. Haritsa, J. R., M. J. Carey and M. Livny, 1990. Dynamic real-time optimistic concurrency control. In Proc. 11th Real-Time Symposium, pp: 94-103
8. Lee, J., 1994. Concurrent Control Algorithms for Real-time Database Systems, PhD thesis, Faculty of the School of Eng. and Applied Sci., Univ. Virginia.
9. Lindstrom, J., 2003. Optimistic concurrency control methods for real-time database systems. Ph.D Thesis. Univ. Helsinki, Finland.
11. Haritsa, J. R., M. J. Carey and M. Livny, 1990. On being optimistic about real-time constraints. In: Proc. 9th ACM Symposium on Principles of Database Systems, pp: 331-343.
10. Konana, P., J. Lee and S. Ram, 1997. Updating timestamp interval for dynamic adjustment of serialization order in optimistic concurrency control-time interval protocol. Inform. Processing Lett.,63: 189-193.
12. Bernstein, P. A., V. Hadzilacos and N. Goodman, 1987. Concurrency control and Recovery in Database Systems. Addison-Wesley.