Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 13 Issue 2 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

A Comprehensive Concurrency Control Technique for Real-Time Database System

By Md. Anisur Rahman & Md. Sahadat Hossain

Dhaka University of Engineering & Technology Gazipur, Bangladesh

Abstract - Real-time database must maintain the Temporal Consistency of the data which cannot be achieved with the conventional concurrency control techniques as they focus only on the consistency of the data. Different protocols exhibit good performance on different situations. But a single technique is inadequate to meet the demand of real-time database. To improve the concurrency control technique for real-time transactions, this paper will present a comprehensive technique that coordinates multi-version, OCC Sacrifice, Speculative Concurrency Control and 2PL-HP protocols. The presented technique uses best suited protocol based on the contention of transactions. Thus it can significantly improve the concurrency of transactions as well as increase the number of transactions.

Keywords : multi-version, speculative concurrency control, real-time transaction, temporal consistency.

GJCST-C Classification : H.2.8

Strictly as per the compliance and regulations of:

A Comprehensive Concurrency Control Technique for Real-Time Database System

Md. Anisur Rahman [α] & Md. Sahadat Hossain [σ]

Abstract - Real-time database must maintain the Temporal Consistency of the data which cannot be achieved with the conventional concurrency control techniques as they focus only on the consistency of the data. Different protocols exhibit good performance on different situations. But a single technique is inadequate to meet the demand of real-time database. To improve the concurrency control technique for real-time transactions, this paper will present a comprehensive technique that coordinates multi-version, OCC Sacrifice, Speculative Concurrency Control and 2PL-HP protocols. The presented technique uses best suited protocol based on the contention of transactions. Thus it can significantly improve the concurrency of transactions as well as increase the number of transactions.

Keywords : multi-version, speculative concurrency control, real-time transaction, temporal consistency.

I. Introduction

Real-time database systems are identified as having timing constraints and can be found in applications such as defence systems, Internet and multimedia applications, industrial automation, programmed stock trading and air traffic control etc.

The timing constraints of real-time database are typically specified in the form of deadlines that require a transaction to be completed by a specified time. Failure to meet a deadline can cause the results to lose their value, and in some cases a result produced too late may be useless or even harmful. So unlike traditional database Real-time databases (RTDBMS) must maintain Temporal Consistency of data. Temporal Consistency requires two main requirements: Absolute validity and relative consistency. Absolute validity is the notion of consistency between environment and its reflection in the database. Relative consistency is the notion of consistency of the data that are used to derive new data. The correctness of the system depends not only on the logical results but also on the time used to produce these results, as the transactions for their concurrent implementation has own timing constrains and dependence. That is Real-time systems are to ensure completion of more transactions within the deadline.

The conventional pessimistic concurrency control mechanism based on locking e.g. two phase locking with higher priority (2PL-HP) can assure the transactions serializability [1], so as to strongly assure the consistency of data. However, because of a high rate of restart of transactions, it cannot satisfy the need of the real-time database systems very well. The optimistic concurrency control techniques assume that the probability of any two concurrent transactions requesting the same data is not often. So it allows all operations to be performed directly whenever transactions request. But these transactions must pass through the validation checking before they are allowed to be committed to database.

The Multi-version Time-stamp Ordering (MVTO) technique is one type of optimistic concurrency control (OCC)mechanism which provides a large degree of concurrency for the transactions by maintaining multiple versions of data items [1]. So it is more appropriate for real-time database systems where the transaction has a low rate of restart and delay of cut-off time but a high degree of concurrency. It ensures transactions serializability using Time-stamp Ordering mechanism.

In OCC Broadcast Commit (OCC-BC) protocol, which is another OCC method, when a transaction commits, it notifies its intentions to all other currently running transactions [5]. Each of these running transactions carries out a check to test whether it has any conflicts with the committing transaction. If any conflicts are detected, the transaction carrying out the check immediately aborts itself and restarts. Note that there is no need for a committing transaction to check for conflicts with already committed transactions, because if it were in conflicts with any of the committed transactions, it would have already been aborted. Thus, in OCC-BC once a transaction reaches its validation phase, it ensures its commitment. Compared to OCC Forward protocol, it encounters earlier restarts and less wasted computations. Therefore this protocol should perform better than the OCC-forward protocol in meeting task deadlines. However, a problem with this protocol is that it does not consider the priorities of transactions. On the other hand, it may be possible to achieve better performance by explicitly considering the priorities of the transactions.

OCC Sacrifice (OCC -S) is another type of Optimistic method that considers the priority of a transaction in the validation check phase to determine which transaction(s) should be restarted [5]. Transaction with higher priority commits and

Author α : Department of CSE, Dhaka University of Engineering & Technology Gazipur, Bangladesh.
E-mails : anisur.rahman.duet@gmail.com, sahadat39@yahoo.com

causes to restart conflicting transactions. This method reduces wasted computation providing early restart of conflicting transactions.

In Speculative Concurrency Control (SCC) technique, conflicts are checked at every read and write operation. Whenever conflicts are detected a new version of each of the conflicting transactions is initiated. The primary version executes as any transaction would execute under an OCC protocol, ignoring the conflicts that develop. Meanwhile the new version executes as any transaction would do under a pessimistic protocol-subjected to locking and restarts. Improving the concurrency control protocol, this paper will present a new concurrency control method which adopts multi-version, OCC Sacrifice, SCC or 2PL-HP depending on the contention of transaction in the system. In this way, it can effectively improve the concurrency of transactions and increase the amount of the transactions completed within the deadline. The feasible analysis denotes that this new method is better than the traditional one on performance.

II. The Description of the Proposed Technique

Several concurrency control techniques exhibits better performance in different idiographic situations. In some cases locking protocols shows better performance but fails to meet the requirement in some other cases. So we have classified the type of transactions as well as consider the actual condition of the contention based on the time required to executer that transaction. Transactions in the Real-time database can be split into three categories according to Multi-version Time-stamp Ordering concurrency control method depending on the type of operations they performs [1]. They are:

a) Read-Only Transactions (Txn-R)

This type of transaction always read the data elements that is the maximum Timestamp with a less than or equal Txn-R in Time-stamp TS (Txn-R). That is Txn-R gets the most recent version of the data before it, so reading-reading conflicts or reading -writing conflicts do not occur, and Txn-R are always succeeded.

b) Write-Only Transactions (Txn-W)

For this transaction type, the old data elements are not modified. Just a new version of data element will be created which is given by Txn-W as timestamp TS (Txn-W). So writing-writing conflicts do not occur and Txn-W will not be blocked by another transactions.

c) Update transactions (Txn-U)

This type of transaction not only reads the data elements but also writes a new version of data elements. So, writing-writing conflicts between update transactions most likely to be occurred Now, it is necessary to resolve the conflicts between the update transactions effectively to provide enhanced concurrency. The proposed method considers the contention of transactions in order to adopt best suited technique for that situation.

Contention is the number of transactions that are running in the system or waiting in concurrency control queue of Transaction manager to be executed. Conflicts among Update transactions (Txn-U) are resolved according to following rules:

If Contention is low, adopt OCC Sacrifice method. Allow all transactions (Txn-U) to be executed freely without any checking. When a transaction Txn-U_i reaches its validation stage, Txn-U_i checks for the conflicts with currently executing Txn-Us by means of the Read-set and Write-set of transactions. In the real-time database system Execution-Time (ET) of a transaction is predictable [1]. Let Conflict -Set (CS) be the set of Txn-Us that are conflicting with Txn-U_i. Now Txn-U_i restarts with rolling back its operations already performed, If $ET(\text{Txn-}U_i) < \sum_j ET(\text{Txn-}U_j)$, For all Txn-$U_j \in CS$; otherwise Txn-$U_i$ commits and For all Txn-$U_j \in CS$ restart with updated data item.

If Contention is medium, adopt SCC method. When a transaction Txn-U_i begins to execute, it issues Exclusive Write (EW) lock on data object. If it completes the work with an data object D, it produces a new-image of that object D_n and converts EW lock into Speculative-Write (SPW) lock. After producing D_n, Txn-U_i allows any transaction Txn-U_j requesting for D to get speculative execution Txn-U_j begins speculative execution with new-image D_n and old image D_o. After the completion of its operation, Txn-U_j commits speculative execution with D_n or D_o according to the abort or commit of Txn-U_i respectively.

If Contention is high, adopt 2PL-HP that ensures more transactions to be completed within the deadline. The transaction priority P (Txn-U) is principally determined by deadline of the transactions. So as in Real time database system we can have the deadline of the transactions and take it as the priority for that transaction. That is for all Txn-U_j, Txn-$U_j \in T$, whenever deadline (Txn-U_i) $>=$ deadline (Txn-U_j), then P (Txn-U_i) $<=$ P(Txn-U_j), the higher-priority transaction will get the priority of execution. When a transaction Txn-U_j begins to execute, it tries to issue write lock on data object D. If D is already occupied by another transaction Txn-U_i , Txn-U_i have to sacrifice and restart. If P (Txn-U_j) $>$ P(Txn-U_j) otherwise Txn-U_j waits for D to be free by the completion or abort of Txn-U_i.

III. Analysis of the Performance

Through the comparison testing between the new and traditional method, Fig.1 and Fig.2 shows how transactions different inter-arrival time affects the transactions restart. Analyzing the comparison figure Fig.1, It can be easily understood that, as the interval between transaction arrival

increases, Contention decreases, the rate of restart of transaction becomes small due to the less opportunity of conflicts. But when the interval is not long, Contention increases the new method is considerably better than conventional one. The performance of real-time database systems has a fundamentally different target compared with the traditional database systems. The real-time database systems require the more number of transactions to be completed within the deadline of the transactions rather than the number of concurrent transaction for execution to maintain the largest concurrency. The new method makes read-only and write- only transactions never fail and avoids their unnecessary restart using multi-version method. It effectively saves the systems expense and improves the systems through put. As for the update transactions, the new technique makes the same data element to be operated by more transactions without interfering by another one by using suitable method to resolve conflicts, according to the contention of transactions in the system. This method can adaptively use the optimistic mechanism and speculation based mechanisms for the implementation of the resolution of conflicts and creates a new version of data elements to improve the concurrency degree of transaction and the amount of transactions completed before deadline. In summary, in different contention size, the new method can flexibly take advantage of the traditional concurrency control mechanism with multiple versions, OCC Sacrifice, Speculative Concurrency Control and 2PL-HP; it can get better the concurrency of the system, save effectively the expense of the system. Compared with the traditional concurrency control mechanisms, the improved one is better on performance.

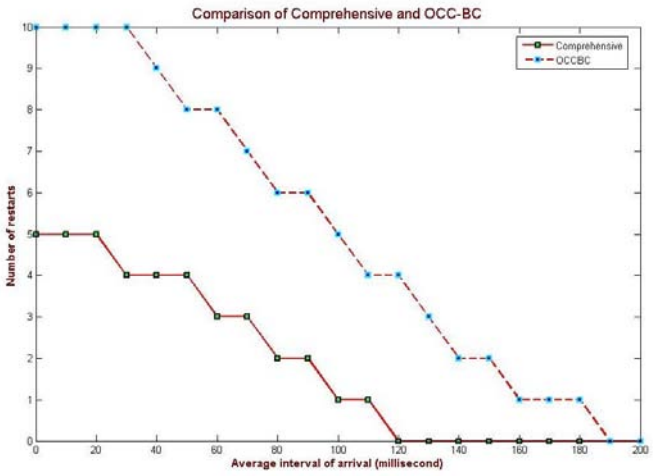

Figure 1 : Restarts vs. Average interval of arrival (For 10 Transactions)

IV. Conclusions

As the real-time database systems have a tight time constraints for the transactions as well as data, and the very well. So there is a demand of comprehensive traditional concurrency control mechanisms cannot satisfy their needs method to meet the requirements of the real time database system. By improving the concurrency control protocols, this paper has presented a comprehensive concurrency control technique that highly reduces the rate of abortion as well as considers the timing constraints of the transactions. With a strong self-adaptability, this method is able to use best suited method among different concurrency control mechanisms according to different situations of contention in the system. It can also effectively improve the performance of system providing higher concurrency considering deadline. The next step is to do further test and evaluation so that the other protocols can be justified with respect to comprehensive method that would be a good verification which is left as well as verification in the actual environment so as to refine and improve the algorithm.

References Références Referencias

1. S P. Wu, Z. Pang, "Research on the Improvement of the Concurrency Control Protocol for Real-Time Transaction", Proceedings of International Conference on Machine Vision and Human-Machine Interface, pp. 146-148, 2010.
2. K. M. Prakash Lingam, "Freezing as a correctness measure for multi-version time-stamp ordering protocol", Proceedings of2nd International Conference on Computer Engineering and Technology, vol.3, pp.312-316, 2010.
3. M. Hedayati, S. H. Kamali, R. Shakerian, M. Rahmani , "Evaluation of performance concurrency control algorithm for secure firm real-time database systems via simulation model", Proceeding of the International conference on Networking and Information Technology, pp. 260-264, 2010.
4. M. Laiho , F. Laux, "Implementing Optimistic Concurrency control for persistence Middleware using row version verification", Proceeding of the Second International Conference on Advances in databases, knowledge and Data Applications, pp. 45-50, 2010.
5. Rajib Mall, IISC, Bangalore "Real Time systems: Theory and Practice" , Ch-7, ISBN 10:8131771016
6. H. G. Molina, J. D. Ullman, J. Widom, "The Implementation of Database System", China Machine Press, 2002:369-377.
7. X. M. Yang, Y. X. Jun, "The comparative study on the implementation of concurrency control", Computer Application Research, 2006, (6):19-22.

8. C.Park, S.Park, S H. Son "Multi-version locking protocol with Freezing for secure Real-Time Database Systems", IEEE Transactions on Knowledge and Data Engineering, Vol. 14, no.5, pp.1141-1154, Oct -2002.
9. Fekete, D. Liarokapis, E. O'Neil, P. O'Neil and D. Shasha, "Making snapshot isolation serializable," ACM Trans. Database Syst., vol. 30,no. 2, pp. 492-528, 2005.
10. S. Jorwekar, A. Fekete, K. Ramamritham, and S. Sudarshan, "Automating the detection of snapshot isolation anomalies," Proceedings of the VLDB'07, 2007, pp. 1263-1274.
11. C. Park and S. Park, "Alternative Correctness Criteria for Multi-version Concurrency Control and a Locking Protocol via Freezing," Proc. Int'l Database Eng. and Applications Symp. pp. 73-81, Aug. 1997.
12. P.A. Bernstein and N. Goodman, "Multi-version Concurrency Control-Theory and Algorithms," ACM Trans. Database Systems, vol. 8, no. 4, pp. 465-483, Dec. 1983.
13. P. Bernstein, V. Hadzilacos, and N. Goodman, "Concurrency Control and Recovery in Database Systems," Addison-Wesley, 1987.
14. C. Park, S. Park, S H. Son, "Multi-version Locking Protocol with Freezing for Secure Real-Time Database Systems," IEEE Transactions on Knowledge and Data Engineering, vol. 14, no. 5, pp. 1141-1154, Oct. 2002.