An Efficient Approach Concurrency Control in Database Management System: A Performance Analysis

Radha Krishna Rambol, Zafar Imam and N. Ahmad

University Department Statistics and Computer Applications
T. M. Bhagalpur University, Bhagalpur-812007, India

Abstract
The traditional pessimistic lock-based concurrency control mechanism which focuses on the data's consistency and the transactions' concurrency cannot meet the demand that the in-time database systems make on the temporal consistency. This paper presents a new concurrency control method which is based on the locking, multi-version and single phase commit protocol concurrency control mechanisms, after the improvement of the concurrency control protocol for in-time transactions. Furthermore, we have adopted a method which is based on different concurrency control mechanisms according to the idiographic situation. In this way it can effectively improve the concurrency of transactions and increase the amount of the transactions completed within the deadline and through this database can make efficient.

Keywords
Commit protocol, Deadline, Concurrency control, EDBMS, DBMS

1. Introduction

Recently several algorithms have been proposed for concurrency control in a Database Management System (DBMS) (see ref. [1 - 14]). The various research efforts are examining the concurrency control algorithms developed for DBMSs (see ref. [1 - 15]). The Efficient Database Management Systems (EDBMS) have shown the timing constraint of both data and transactions. The traditional lock-based pessimistic concurrency control mechanisms two phase high priority locking protocol can guarantee the transactions serializable, so as to powerfully guarantee the consistency of data. However, because of a high rate of restart of transactions, it cannot satisfy the need of the time management database systems very well. Whereas the optimistic concurrency control mechanism believes that the probability of any two concurrent transactions requesting the same database is seldom. The Multi-version and single phase commit Concurrency Control Protocol are kind of the optimistic concurrency control mechanisms which makes the transaction has a large degree of concurrency by maintaining multiple versions of data items with conformation of transaction in-time constraints. So it is more suitable for efficient database management systems where the transaction has a low rate of restart and delay of cut-off time but a high degree of concurrency. A transaction is normally considered as a program unit that must be executed in its atomicity standard.

The module of a database management system (DBMS) that is responsible for transaction execution is a transaction manager. An objective in most database management systems is to execute multiple transactions concurrently. Generally, transactions could interfere with one another and, as a result, could cause the database to be inconsistent. The techniques that have been developed to ensure the consistency of a database in the midst of concurrent transaction execution are called concurrency control techniques ([2], [3]). This is based on locking protocol system, which is one of the best protocols in database management system. When transactions are executed in a multilevel environment in addition to consistency, it must be ensured that the access control policy enforced by the system is not violated and transactions executing on behalf of higher level users do not interfere with those executing at a lower level. While transaction management techniques are relatively mature for traditional database applications (such as banking and business data processing), it is only recently that concurrency control techniques are being examined for a multilevel database environment. In other words, the developments in multilevel database concurrency control are more than a decade behind the developments in database concurrency control.

Furthermore, during recent years, database concurrency control has progressed beyond traditional applications and techniques are now being developed for advanced applications. Such applications involve heterogeneous environments, real-time processing, long duration transactions, and collaborative computing environments. Furthermore, theory of database concurrency control is sufficiently developed for traditional database applications. Therefore, much needs to be done on concurrency control for multilevel database applications. A new concurrency control method which is based on the locking and multi-version with single phase commit option used to maintain the concurrency control mechanisms. In this way it can effectively improve the concurrency of transactions and increase the amount of the transactions completed within the deadline.

Manuscript received July 5, 2013
Manuscript revised July 20, 2013

2. Review of Transactions

In the multi-version systems of the, the real-time transactions can be divided into three categories according to the multi-version concurrency control mechanism:
Read-only Transaction (T_r): Always read the data elements that are the maximum timestamp with a less than or equal T_r in timestamp *T*S (T_r). In other words, the read-only transaction reads the most recent version of the data before it, so reading-reading conflict or the reading-writing conflict will not happen. T_r never fails (see ref. [8]).
Write-only Transaction (T_w): The old data elements are not modified. But a new version of data elements will be created which is given by T_w as timestamp *T*S (T_w). So writing-writing conflict will not happen and T_w will not be blocked by other transactions ([9]).
Data-processing Transaction (T_p): It not only reads the data elements, but also writes a new version of data elements. So writing- writing conflict between T_p's is likely to happen ([7]). From the above analysis we can see that in the multi-version systems of database writing-writing conflict between the data-processing transactions must be effectively resolved in order to improve the system's performance. This leads to propose a new concurrency control mechanism. This mechanism uses the concurrency control methods of combining the optimistic multi-version with single phase commit protocol and the pessimistic blockade two phase high priority protocols so as to increase the rate of success of the transaction.

3. The Description of New Approach

After the usage of the new method, T_r, T_w will not fail. The resolution of the conflicts among transactions in T_p will be based on the following principles.
In multi-version systems of efficient database, the transaction priority P(*T*) is mainly determined by Deadline(*T*), in our paper work, we used the following formula for deadline calculation:

$$D(T) = A(T) + SF(T) * E(T) \qquad (1)$$

In equation (1) where *D(T)* and *A(T)* are the deadline and arrival time of transaction *T*, respectively, and *E(T*) is the expected execution time of the largest possible transaction (a transaction accessing 1.5 **TransSize* pages). *SF(T)* is a *slack factor* that varies uniformly over the range set by the workload parameters *LSF (*Low slack factor*)* and *HSF (*High slack factor*)*, and it determines the tightness/slackness of deadlines. The *Arrival Rate* parameter specifies the mean rate of transaction arrivals. The number of pages accessed by a transaction varies uniformly between 0.5 and 1.5 times *TransSize*. Page requests are generated from a uniform distribution (without replacement) spanning the entire database. *WriteProb* gives the probability of a page that is read being also updated ([11-14]). In multi-version systems of efficient database, the transaction priority P(*T*) is mainly determined by Deadline(*T*), that is, $\forall A_1, A_2 \in A$, located

$$\text{Deadline}(T_1) >= \text{Deadline}(T_2),$$

then

$$\text{P}(T_1) <= \text{P}(T_2),$$

the high-priority transaction will gain the priority of implementation; order O on behalf of data elements of the conflict, $\forall A_i, A_j \in T_p$, $\text{P}(T_i) < \text{P}(T_j)$, T_i, T_j in the conflict of O, then T_i ,T_j will be in following manner:

- T_i, T_j can deal with the different data items O without disturbing each other, then O will be divided into smaller data items, change the block size of data, and allow low-priority transaction to inherit high-priority transaction's priority, and assume $\text{P}(T_i) = \text{P}(T_j)$ which makes that T_i, T_j can perform simultaneously with helping each other. Of course, this approach may have some errors, so these errors must be calculated before the submitting phase, if they are in the permitting extent in systems of the efficient database to generate a new version of data items and set O's timestamp by the final submission of the transaction. This method can effectively reduce the blocking time in the conflict of the low-priority transaction and the rate of restart in reducing the transaction.
- O has the atomicity which cannot be divided further, T_i, T_j are likely to have conflicts and if using the optimistic mechanism of the commit confirmation, the rollback of transaction is almost inevitable, so in this case blockade mechanism is still used to save resources, and at the same time, the traditional two phase high priority lock means of resolving conflicts have to cause low-priority transactions to restart, which will make the number of transactions for delaying deadline increase. Therefore, according to the mechanisms of commit confirmation improvements will be made as follows:
- In efficient database systems, Execution Time (*T*) of transaction is predictable. When the requested data is possessed by T_i, $\text{P}(T_i) < \text{P}(T_j)$, instead of immediate restart of T_i, first of all weigh the conditions before making a decision:

DeadTime = min (Deadline (T_i), Deadline (T_j));

// Order DeadTime as the less one between Deadline (T_i) and Deadline (T_j);

If ((Current time + T_i's remained time of the execution + Execution Time (T_j)) < DeadTime)

Then
{

P(T_i) = P(T_j);
// Low-priority transaction inherits the priority of high-priority one, which improves T_i's priority, thus accelerating their implementation;

Implementation of the T_i comes to the end;

// T_j waits for T_i's completion and the release of the lock on O;

T_j obtains the lock on O and starts the implementation until the completion;
}

Else

if ((current time + T_i's remained time of the execution + Execution Time (T_j)) < Deadline(T_i)
&& Deadline (T_i) > Deadline (T_j))

Then

{

T_i dies; // T_i releases the lock on O;

T_j obtains the lock on O and starts the implementation until the completion;

// T_i waits for T_j's completion and the release of the lock on O;

Ti obtains the lock on O and starts the implementation until the completion;

}

Else

// Abandon the blockade mechanism, adopt the mechanism of commit confirmation in order to maximize the number of transactions completed within the deadline

{

T_i dies; // T_i releases the lock on O;

Start T_j's implementation;

// There is no data locked any longer

Start T_i's implementation;

// T_j, T_i simultaneously start implementation;

If T_j finds the risk when commit is confirmed

Then

{

T_i dies;
// Finding the risk, unconditionally put an end to low-priority transactions to ensure that high-priority transactions can be successfully implemented;

commit T_j;
}

Else
{
commit T_j;

commit T_i;
// Operations of T_i, T_j's reading and writing can just performed in time to a serial implementation, this can be successfully completed by the commit conformation;

}
Endif
}
Endif
Endif

As for the transactions' conflicts between data elements which cannot be further divided, if the order of the implementation of transactions can still meet the need of their deadline, the priority-inheritance mechanism can be used without changing the order of the implementation of transactions , then use the blockade mechanism to make transactions serialized to save the expense, or abandon the blockade mechanism to use the mechanism of commit conformation so as to minimize the rate of transactions' delaying deadline.

4. Performance Analysis

Through the testing comparison between the new and traditional methods, shows how transaction's different inter-arrival time affects the transaction's restart [16]. From the above description we can see, as the interval increases,

the rate of restart becomes small due to the less opportunity of conflicts. But when the interval is not long, the new method is significantly better than traditional one. The performance of efficient database systems has a fundamentally different target compared with the traditional one. The efficient database systems require the number of transactions completed within the deadline for the largest rather than the number of concurrent transaction for execution to maintain the largest [4].

The new method makes read-only and write-only transactions never fail and avoids their unnecessary restart. It effectively saves the system's expense and improves the system's throughput. As for the data-processing transactions, the new method makes the same data elements operate on different data items of the transaction without interfering but collaborating with each other by changing dynamically the data elements of the block size; when it comes to the data elements which cannot be divided, according to the idiographic situation this method can adaptively use the blockade mechanism and the mechanisms of valid confirmation for the implementation of the conflicts and create 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 situations the new method can flexibly take advantage of the traditional concurrency control mechanism with multiple versions, blockade, and commit confirmation, it can improve 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.

5. Conclusion

As the efficient database systems have a strong time constraint for the transactions and data, and the traditional concurrency control mechanisms cannot meet their needs very well. After the improvement of the concurrency control protocol, this new approach will present a new concurrency control method. With strong self adaptability, this method is able to use different concurrency control mechanisms according to different situations. Overall performance may be improved by 20-30%. It can also effectively improve the performance of system.

Acknowledgements

The authors are grateful to the Editor in Chief and the learned Reviewers for their valuable comments and suggestions to improve the manuscript.

References

[1] J. E. Armendariz. Design and Implementation of Database Replication Protocols in the MADIS Architecture. PhD thesis, Universidad Publican de Navarra, Pamplona Spain, Feb, 2006.

[2] Bernstein, P. A., V. Hadzilacos, and Goodman, N., 1987, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley Publishing Company.

[3] Keefe, T., W.T. Tsai, and J. Srivastava, 1989, *Database Concurrency Control in Multilevel, Secure Database Management Systems*, Technical Report 89-73, University of Minnesota (a version also published in the Proceedings of the 6th IEEE Data Engineering Conference).

[4] Keefe, T., and W.T. Tsai, 1990, "Multiversion Concurrency Control for Multilevel Secure Database Systems Proceedings of the IEEE Symposium on Security and Privacy", Oakland, CA.

[5] Maimone, W,, and I. Greenberg, 1990, "Single-level Multiversion Schedulers for Multilevel Secure Database Systems, "Proceedings of the 6th Computer Security Applications Conference, Tucson, AZ.

[6] Jajodia, S., and B. Kogan, 1990, "Transaction Processing in Multilevel Secure Databases Using the Replicated Architecture," Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA.

[7] Costich, O., and J. McDermott, 1992, "A Multilevel Transaction Problem for Multilevel Secure Database Systems and Its Solution for the Replicated Architecture," Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA.

[8] Costich, O., and S. Jajodia, 1992, "Maintaining Multilevel Transaction Atomicity in MLS Database Systems with Kernelized Architecture," Proceedings of the 6th IFIP Working Conference in Database Security, Vancouver, British Columbia.

[9] K. Ramamritham. Real-time databases. Distributed and Parallel Databases, 1(2):199.226, 1993.

[10] Bernstein, P. A., V. Hadzilacos, and Goodman, N., 1987, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley Publishing Company.

[11] J. Haritsa, M. Carey, and M. Livny, "Earliest-Deadline Scheduling for Real-Time Database Systems," Proc. 12th IEEE Real-Time Systems Symp., San Antonio, Tex., Dec. 1991.

[12] J.R Haritsa., M. Carey and M. Livny, "Data access scheduling in firm realtime database systems", Journal of RTS, vol.4, no 3, p. 203-241, 1992.

[13] K. Ramamritham, J. Stankovic, P. Shiah, "Efficient Scheduling Algorithms for Real-Time Multiprocessor Systems," IEEE Transactions on Parallel and Distributed Systems, Vol. 1, No. 2, pp. 184-194, April 1990.

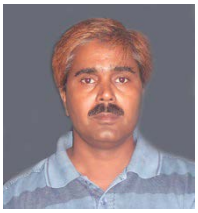
[14] L. Baccouche and I. Balti "Distributed real-time transaction execution control", technical report INSAT 01_05, INSAT, Computer Science Department, January 2005.

[15] Rambol, R. K., Ahmad, N. and Sharma, B. K., "Efficient Data Accessing through Heterogeneous ERP Solution", International Journal of Computer Applications, Vol. 53 (6), pp.14–17, 2012.

[16] Farooq, S. U., Quadri, S. M. K. and Ahmad, N., "Software Measurements and Metrics: Role in Effective Software Testing", International Journal of Engineering Science and Technology, Vol. 3 (1), pp. 671 – 680, 2011.

Radha Krishna Rambola is a research scholar in University Department of Statistics and Computer Applications at Tilka Manjhi Bhagalpur University (TMBU), Bhagalpur, India. He is working as an Asst. Professor at Asia Pacific Institute of Information Technology SD India. Which is International Engineering College affiliated from Staffordshire University (UK). He is having Over 13 years of experience in teaching and industry with Engineering technology uses Languages, System Analysis and Design, Software Development Project, Project Management, Database Management System and Engineering related subjects with deep experience of ERP. He is B.E (Computer Science and Engineering) from University of Madras in 1998, M. Tech (CSE) from Allahabad Agricultural Deemed University, Allahabad (In year 2006). His interest area of research is Database management system, Concurrency Control and Data Mining. Rambola is IBM DB2 & Web Sphere certified. He is a member of IEEE.

Md Zafar Imam is a research scholar in University Department of Statistics and Computer Applications at Tilka Manjhi Bhagalpur University (TMBU), Bhagalpur, India. He is working as a teaching faculty in Bachelor of Information Technology in Marwari College in the university of Tilka Manjhi Bhagalpur University, Bhagalpur, India, since 2010. He graduated from Tilka Manjhi Bhagalpur university with a B. Sc in Physics in 2001, MCA in 2008. He is currently a Ph.D. candidate in University Department of Statistics and Computer Applications, at Tilka Manjhi Bhagalpur University, Bhagalpur. His main research interest includes software testing, and software reliability analysis.



Nesar Ahmad is an Associate Professor and Head in University Department of Statistics and Computer Applications at Tilka Manjhi Bhagalpur University (TMBU), Bhagalpur, India. He received the B. Sc. degree in Mathematics from Bihar University, India, in 1984 and the M. Sc., M. Phil., and Ph. D. degree in Statistics from Aligarh Muslim University, Aligarh, India, in 1987, 1990, and 1993, respectively. From 1995 to 1996 he was a research associate of UGC/CSIR at Aligarh Muslim University, Aligarh. He has been a Lecturer in Statistics from January 2006 to December 2009 at the University of the South Pacific, Suva, Fiji Islands. After working five months as a lecturer at Poona College, Pune, he joined the University Department of Statistics and Computer Applications at TMBU, Bhagalpur, India in 1996. He has about 17 years of experience in teaching and research. His research interests include life testing, statistical modeling, reliability analysis, software testing, software reliability engineering and optimization. He has published more than 50 papers in journals, and conferences in these areas. He is in the editorial board of International Journal of Scientific and Statistical Computing (IJSSC) and Journal of Convergence Information Technology (JCIT). Ahmad is life member of Indian Science Congress and Bihar Journal of Mathematical Society, and regular member of Aligarh Statistical Association.