Recoverable Timestamping Approach For Concurrency Control In Distributed Database

Rinki chauhan
Department of CSE
Manav Rachna International University
Faridabad, India
iknirs@gmail.com

Suman Malik
Department of CSE
Manav Rachna International University
Faridabad, India
suman.sheoran14@gmail.com

R K Rathy
Department of CSE
Manav Rachna International University
Faridabad, India
rathy.citm@yahoo.co.in

Preeti Bhati
Department of CSE
Manav Rachna International University
Faridabad, India
versatilepreeti@yahoo.com

Abstract—A distributed database consists of different number of sites which are interconnected by a communication network. In this environment in absence of proper synchronization among different transaction may lead to inconsistency of databases. A new approach for timestamp ordering problem in serializable schedules is presented. Strict 2PL does not allow all possible serializable schedules and so does not result high throughput. The main advantages of the approach are the ability to enforce the execution of transaction to be recoverable and the high achievable performance of concurrent execution in distributed databases.

Keywords-Concurrency control, distributed database, timestamp, recovery, transaction.

I. INTRODUCTION

Distributed database systems (DDBS) are systems that have their data distributed and replicated over several locations or sites unlike centralized databases where one copy of the data is stored. Both types of databases have same problem of access control, such as concurrent user access.

Concurrency control is a method of managing concurrent access of transactions on a particular data item such that the consistency of the database is maintained. Consistency means when any transaction comes for execution the database is in consistent state and when it leaves the system the database should be in consistent state. Also, the result produced by the transaction should be correct. This problem becomes complex in distributed databases since the data is not stored at one place. The user can access the data from any site and the controlling mechanism at other site may not recognize it instantly.

The notion of a transaction is of fundamental importance to concurrency control. In a distributed database environment, a transaction may access data stored at more than one site. Each transaction is divided into a number of sub-transactions, one for each site at which data accessed by the transaction is stored.

Recovery is an equally important issue to be considered for handling transactions in today's scenario. Considering practical cases, failures are bound to occur in every system. To overcome from those failures recovery measures are needed. We have mixed the recovery measures with the concurrency control and have proposed an algorithm in this paper. The algorithm uses timestamp ordering for controlling the concurrency of the distributed database along with a special vector named as the commit vector used for recovering the data.

II. DISTRIBUTED TRANSACTION-PROCESSING MODEL

For understanding how a concurrency control algorithm operates we present a simple model of Distributed Database Management System in this section in Fig. 1. A distributed database management system (DDBMS) is a collection of sites interconnected by a network.

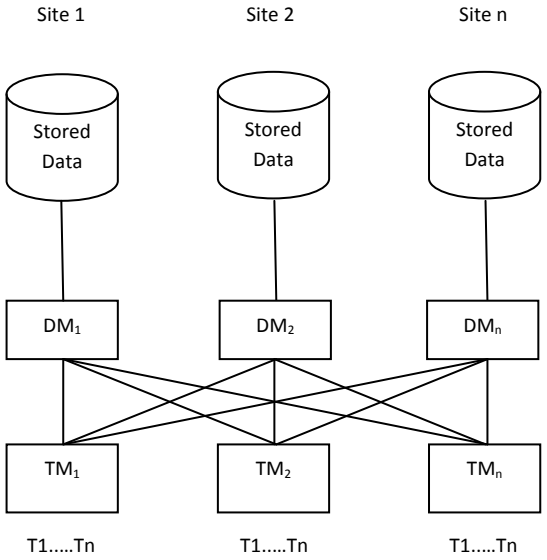

Figure 1. DDBMS Transaction Processing Model

Each site running on one or both of the following software modules: a transaction manager (TM) and data managers (DM). TM supervises the interactions between users and DDBMS while DMs manage the actual database. Here, the communication network is considered to be reliable. This means that if site A sends a message to site B that message reaches to site B without any error or vice-versa.

III. DIVISION OF TRANSACTIONS

In DDBMS the transactions issued by the user need to be divided into subtransactions if the data required by the transaction is not available at the site it is issued. The subtransactions will be sent to the sites at which the required data is present. Also the commit point of the transaction is divided along with the subtransactions. This can be illustrated easily with the following example. Consider 3 sites sit1, site2 and site3 where data item A is stored on site2 and data item B is stored on site3 as shown in Fig. 2.

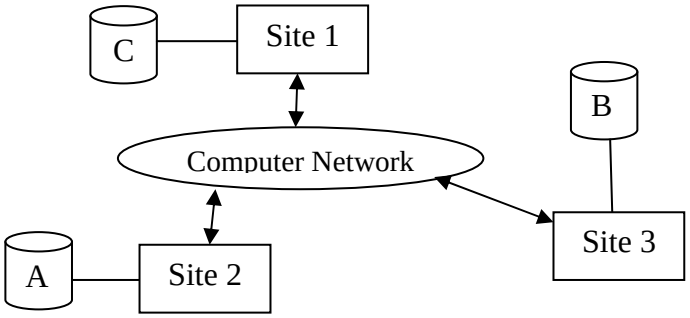

Figure 2. Scenario of a Distributed Database

At site 1 if a transaction arrives like T1- r(A)W(A)W(B)C1

Then subtransations of T1 will be

T11 - r(A)W(A)C11and

T12 - W(B)C12

Subtransaction T11 will be send to site 2 and subtransaction T12 will be send to site3. After execution of these subtansaction the results will be sent back to the site at which the transaction was issued.

IV. ENHANCED RCTO ALORITHM

The assumptions of our work are same as that proposed in NEW-RCTO algorithm[7]. The enhanced RCTO also follows basic timestamp ordering except in commit operation, and (2) for every data item x, there are two vectors: (i) the write vector, wv, records the write timestamp for each write/read operation, and (ii) the commit vector, cv, records commit operation for each transaction only if the timestamp of received commit operation does not equal the timestamp of the first stored element in wv. A rotate function shifts the elements of wv such that the first element in wv shifts to the last location and the remaining elements move up as in Fig. 3.

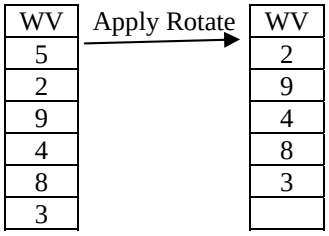

Figure 3. Example of Rotating Function

ENHANCED-RCTO algorithm consists of three different phases: (1) read phase, (2) write phase, and (3) commit phase. Before defining these phases, the timestamp of commit operation for different transaction on the same object maintains a relationship as depicted in the following code:

1. if ts(Ci) = ts(wv[i])
{
2. Execute Ci,
3. Delete the contents of wv[i],
4. Move up the remaining values of wv
}

The above code operates simultaneously in both wv and cv vectors in order to get a high speed in processing of commit operation. Fig. 4 depicts a schedule execution which consists of the following operations: C3C4C5 is a set of three commit transactions, t1,t2,and t3, stored in commit vector, while W2(x) W4(x) W1(x) is a set of write operation on the same object x stored in write vector for the three transactions. Since C4 and W4 are stored in the same row with the same index, they will be deleted from wv and cv simultaneously. The remaining w2, c3, w1 and c5 operations will be moved up one row. Fig. 4 Write vector and commit vector status after matching w4 c4

Index	wv	cv
1	2	C3
2	4	C4
3	1	C5

Index	wv	cv
1	2	C3
2	1	C5

Figure 4. Read operation R2 arrived and then multiplied by random value

A. Read Phase

Whenever read operation, Ri, arrives, the following code will be executed:

1. For each received read operation i
{
2. Let z = i * random number

3. If ts(Ri) is already in wv
then
4. store z instead of Ri;
else
5. store z at the end of wv
}

Index	wv	cv
1	2	C3
2	1	C5

→

Index	wv	cv
1	2*random no	C3
2	1	C5

Figure 5. Read operation R2 arrived and then multiplied by random value

Fig. 5 depicts the read phase with the last data stored in Fig. 4. Suppose the next arrival operation is R2(X), then apply read phase will result the right table. Since the timestamp of read operation is already stored in wv, the contents of wv is multiplied by a random value. The main advantage of this phase is to distinguish the read operation from the write operation which will not cause any delay in execution of the read operation. Fig. 5 Read operation R2 arrived and then multiplied by random value.

B. Write Phase

Whenever write operation, Wi, arrives, the following code will be executed:

1. For each received write operation i
{
2. If ts(Wi) is not stored in wv then
3. Save ts(Wi) in wv
Else
4. Save i * random number in wv
}

Index	wv	cv
1	2	

→

Index	wv	cv
1	2	
2	3	

Figure 6. Write vector and commit vector status after executing W3

The write step enforces the write operation to store its timestamp in wv. Fig. 6 shows the contents of wv and cv after a new operation arrived, i.e. W3(x). The initial data stored is shown in the left side of Fig. 6 and the right side is the result after W3 arrived. Fig. 6 Write vector and commit vector status after executing W3.

Index	wv	cv
1	3	4
2	5	1

→

index	wv	cv
1	3*Random no	4
2	5	

Figure 7. Write vector and commit vector status after executing W3 when read operation accessing same data is already present

In Fig. 7 status of write vector and commit vector is depicted when W3(x) arrives after arrival of R3(x). the NEW-RCTO algorithm states nothing about the arrival of wrte operation after the read operation arrival for the same data item.

C. Commit Phase

This phase is the main phase that enforces all commit operation either to compare the ts(Ci) with the first element in
wv or to keep commit operation in cv. The following code represents the commit phase:

1. Let k is the index of the first empty available position in cv.

2. For each received Commit operation Ci
{
3. If (ts(Ci) = wv[0]) or (ts(Ci) =wv[k])
{
4. Execute Ci;
5. Delete wv[0] Or Delete wv[k];
6. Move up all the remaining values in wv and cv
simultaneously
}
else
{
7. Record Ci at the first empty cell in cv
}
}

Index	wv	cv
1	2	
2	3	

→

Index	wv	cv
1	2	
2		

Figure 8. C3 commit arrived and then delete W3 from wv[2]

The main advantage of commit phase is to delay any premature commit and enforce the commit of transaction of write operation to be executed before any commit of different transaction for read operation. Fig. 6 shows the commit phase after C3 arrived. Since W3 is already stored in wv, the commit operation executes and delete wv contents -i.e.W3. Fig. 6 C3 commit arrived and then delete W3 from wv.

V. CONCLUSION

This paper has proposed enhanced-RCTO into concurrency control in distributed database. This offers a recoverable
execution of transactions. As shown in different applications, enhanced-RCTO guaranteed the execution output by the scheduler to the data manager to be recoverable. All write/read operations are executed without any delay.

REFERENCES

[1] P. A. Bernstein, V. Hadzilacos, N. Good-Man, "Concurrency control and recovery in database systems", Addison Wesley, 1987.
[2] T. Kristian, S. J. Christian, T. S. Richard, "Effective time stamping in databases, a time center", Technical report, 1998.
[3] Elmasri, Navathe, "Fundamentals of database systems", Addison Wesley, 2nd edition 1994.
[4] Ramon Lawrence, "Advanced database seminar-concurrency control", Lecture notes, 2001
[5] D. Jeffrey, Ullman, "Principles of database and knowledge-base systems", W.H. Freeman & Company, 1988.
[6] A. James, A. Fekete, N. Lynch, M. Merrittt, W. Weihl, "A theory of timestamp-based concurrency control for nested transactions", Proceedings of the 14th VLDB Conference, 1988.
[7] Hassan M. Najadat “A New Approach for Recoverable Timestamp Ordering Schedule” WASET Publication Year:2006 Issue 13

Rinki Chauhan received B.Tech degree in Information Technology from U.P. Technical University in 2009 and is persuing M.Tech. in CSE form Manav Rachna International University, Faridabad. Presently, he is working in Computer Engineering department in M.V.N Institute of Engg &Technology, Palwal. Her areas of interest are Database and Wireless Networks.

Suman Malik received B.Tech degree in Computer Science from M.D. University in 2006 and is pursuing M.Tech. in CSE form Manav Rachna International University, Faridabad. Presently, she is working in Computer Engineering department in M.V.N Institute of Engg &Technology, Palwal. Her areas of interest are Web development and Data warehousing.

Dr. R. K. Rathy is working as Sr. Professor in Manav Rachna International University, Faridabad. He is having a long experience of IIT Kanpur and Meerut for more than 40 years. He has guided many graduates, post graduate and Ph.D pursuing students in their corresponding works.

Preeti Bhati received B.Tech degree in Computer Science from M.D. University in 2007 and is pursuing M.Tech. in CSE form Manav Rachna International University, Faridabad. Presently, she is working in Computer Engineering department in M.V.N Institute of Engg &Technology, Palwal. Her areas of interest are Artificial Intelligence, Networking and Data warehousing.