A study of Concurrency control techniques in Distributed database

Yogita Yashveer Raghav, Assistant professor, K.R Mangalam University, SohnaRoad, Gurgaon
ygtraghav@gmail.com

Abstract: ***This paper surveys the basic concurrency control techniques in Distributed database system. Distributed database is the collection of the databases stored at different sites in the interrelated manner over the network. Concurrency control techniques are required because parallel transactions are to be executed at the same time. In this paper i have discussed various concurrency control techniques, their advantages and disadvantages and make comparative study between them.***

Replication of data states, that the same copy of data has been maintained in different sites so that if one data sites fails, the same data can be recovered by other sites and execution of the query can be done by taking the data from other sites. But to use the replica we need to maintain the concurrency control techniques, otherwise inconsistency can occur in the execution of the transactions and incorrect value can occur.

Keywords: Distributed Database, Deadlock, transaction, Concurrency, locking, Cascading rollback, time stamp ordering, and performance.

1 Introduction:

In Distributed database system data is scattered at various data sites in the form of fragmentation. For faster accessing of data, not to depend on any single data site at the time of failure data can be recovered and accessed by another data sites. Replication is to be introduced. It means replica is to be maintained at different data sites so consistency needs to be maintained between replicas otherwise some other issues can be occurred.

As we all know that ACID properties need to follow by every transaction which are explained as follows

Atomicity: This property says that either all or none.it means that either the transaction should commit completely or it should rollback if any failure occur.

Consistency: This property means that transaction should be in the consistent state before or after the transaction.

Isolation: It means that transaction executes in concurrent manner but each and every transaction should assume that they are executing in isolation**.** Any transaction should not interfere another transaction.

Durability: This property says that transactions should be recoverable from the failures.

ISSN : 2348-5612 © URR

2. Problems of Concurrent Transaction: Due to concurrent execution of the transactions, some issues may be arising that are discussed following:

2.1. Lost and update: it means that any transaction's update can be lost due to concurrent execution of the transaction that is explained in following example

Transaction T1	Transaction T2
R(A)	
A=A+10	R(A)
	A=A+200
	W(A)
W(A)	

Figure 1

In the example Transaction T1 reads the value of Data item Initially we assume that Data item A's value is 100, update it and add 10 in it.to before writing in local buffer, control is pre-empted by transaction 2 and that reads the value of A that is stored in database i.e. 100.and update and add in it 200.Now A's Value is 300 for transaction T2.Then control is transferred to transaction 1 and write the value 110 in local buffer of transaction 1. Now the problem is that for same data item A, both transactions have the different value, which is inconsistent. That should not happen in transaction

2.2 Dirty Read: it means that any transaction's updated value such as transaction 1's updated value of A is to be read by transaction 2, and before committing Transaction 1, it fails and rollback. Now

© UNIVERSAL RESEARCH REPORTS | REFEREED | PEER REVIEWED
ISSN : 2348 - 5612 | Volume : 05 , Issue : 03 | January – March 2018

Transaction 2 has been committed so cannot be rollback. This problem is called the dirty read problem. That should not happen in transaction for maintaining the consistency.

Roll back

Transaction T1	Transaction T2
R(A)	
A=A+10	
W(A)	
	R(A)
	A=A+200
	W(A)
Failure Occur	
	Commit

Figure 2

2.3 Unrepeated read (W R Conflict): If a transaction T1 reads an item value twice and the item is changed by another transaction T2 in between the two read operation of the same item. Initially A has 100.so transaction T1 reads it and Transaction T2 preempts the control update the Data item A and add 200 into it.A becomes 300.Then Transaction T1 reads A.Now it has become 300.Now it is also a issue because Every transaction should execute in isolation.

Transaction T1	Transaction T2
R(A)	
	R(A)
	A=A+200
	W(A)
R(A)	

Figure 3

2.4 Incorrect Summary Problem: If one transaction is calculating aggregate summary function on a number of records, while other transaction is updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated-result in incorrect summary. Initially A=100, Y=200:

Transaction T1	Transaction T2	
	Sum =0	
	R(A)	
	Sum=sum+A	Sum=100
	R(B)	
	Sum=sum+Y	Sum=300
R(Y)		
Y=y+100		
W(Y)		

Figure 4

These are the some problems that need to be handled otherwise inconsistency will occur.

3. Solution of the concurrent Execution of transaction problem: Solution of the problem is locking. Locking is the mechanism by which any transaction cannot read or write data until it acquires an appropriate lock on it.
Lock can be of two types: Exclusive and shared
Shared Lock: If any data item needs to be read only, then we need to acquire shared lock. In shared lock we cannot write or update the value of data item.
Exclusive Lock: If any data item needs to be read or write then we need to acquire exclusive lock.

3.1 Two-Phase Locking Protocol

Principles- This Protocol used to ensure seriabilzability by forcing some restriction on transaction as Any transaction is not allowed to obtain new locks till it had released a lock this restriction called two phase locking(2pl). This protocol called 2pl because it has two principal phases as in figure (5).

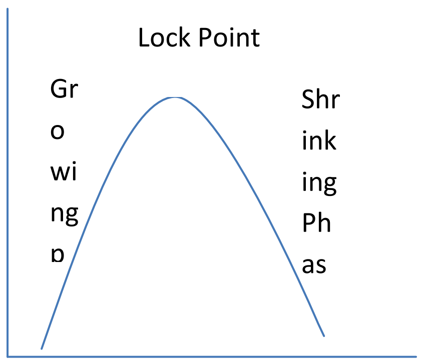

Figure 5

The first phase is known as the growing phase; in which a transaction acquires all the locks it needs. In this phase transaction cannot release the lock. The second phase is known as the shrinking phase, where the process releases the locks but cannot acquire the lock. If a process fails to acquire all the locks during the first phase, then it is necessary to release all of them, wait, and then start over. [12] This protocol ensures conflict– serializable schedules.

Discussion- This protocol may be good in case of absence of any information about the transactions or the database. There are three types of phase locking protocol as in figure (1): strict two-phase locking, rigorous two-phase locking and conservative locking.

3.2 Strict two-phase Locking

In Basic 2PL the problem of deadlock and cascading rollback occurs. Strict two phase Locking has solved the problem of Cascading rollback by following given strategy, but deadlock is not solved by Strict 2PL.

[1] In this protocol any transaction does not release any of its exclusive write lock until after it commits or aborts. Other transactions cannot access to locked item until current transaction has committed. Transaction must hold all its exclusive locks till it commits or aborts and no cascading rollback takes place. Read lock of transaction can be released when transaction terminates (commits its results) but write must be maintained until after commitment or abortion of transaction.

3.3 Rigorous two-phase Locking

[1] It is more restrictive variation of strict 2PL.But this technique also is not successful for solving the problem of deadlock. All locks (read and write) are held until after transaction commits or aborts. Drawbacks of these protocols are Deadlock and starvation which occurs when a transaction cannot proceed for an indefinite period of time while other transactions in the system continue normally.

3.4 Conservative two phase Locking

In Conservative two phase locking any transaction before starting its execution acquires the locks first on the entire data item which are required in execution. Then starts execution. This technique has solved the problem of deadlock. But it is not the suitable and good solution for solving the problem.

1. It has proved obstacle in concurrent execution of transactions.

2. Resources are not fully utilized in this technique.

4. Wait-Die & Wound-Wait Algorithms:

These protocols are based on the timestamp value. Timestamp is the value or time when any transaction enters in the system.

4.1 WAIT-DIE Rule: [1] If Ti requests a lock on a data item which is already locked by Tj, then Ti is permitted to wait iff TS(Ti)<TS(Tj).
If TS(Ti) > TS(Tj), then Ti is aborted and restarted with the same timestamp.

1. if TS(Ti) < TS(Tj) then Ti waits else Ti dies
2. non-preemptive: Ti never preempts Tj
3. prefers younger transactions

4.2 WOUND-WAIT Rule: [1] If Ti requests a lock on a data item which is already locked by Tj , then Ti is permitted to wait iff TS(Ti)>TS(Tj). If TS(Ti)<TS(Tj), then Tj is aborted and the lock is granted to Ti.

1. if TS(Ti)<TS(Tj) then Tj is wounded else Ti waits
2. preemptive: Ti preempts Tj if it is younger
3. prefers older transactions

4.3 Basic Timestamp Ordering Algorithm

[1] Time Stamp can be used to determine the out datedness of a request with respect to the data object it is operating on and to order events with respect to one another. Timestamp is a unique identifier used to identify a transaction. In this algorithm transaction is ordered based on their timestamp values. The timestamp-ordering protocol ensures serializability among transaction in their conflicting Read and Write operations. [11]. This is the responsibility of the

Protocol system that the conflicting pair of tasks should be executed according to the timestamp values of the transactions.

Rule:

```
1. Transaction T issue R (A) operation.
        If W(TS) > TS (T)
        Then rollback T
        Otherwise
        Execute R (A) successfully
        and set R(TS) (A) = max {R(TS) (A), TS (T)}.
2. Transaction T issues W (A) operation.
        If RTS (A) > TS (T)
        Then rollback T
        If WTS (A) > TS (T)
        Then rollback T
        Otherwise
        Execute W (A) successfully
        and set WTS (A) =TS (T).
```

5. Conclusion: In this paper we have described that Distributed database system is considered to be more reliable. It is really mandatory for database to perform the ACID properties. I have also discussed problems of concurrent execution of transactions such as lost update problem, dirty read, unrepeated read and incorrect summary problem. Then I have discussed concurrency control algorithms such as, basic 2PL, Strict 2PL rigorous 2PL, Conservative 2PL, Wait and die, Wound and wait and timestamp ordering,. Now a day's distributed database systems becomes very important for computer science. Many organizations are now deploying distributed database systems.

References:

[1] *Manoj Kumar Sah 1, Vinod Kumar 2, Ashish Tiwari 3,Security and Concurrency Control in Distributed Database System, International Journal of scientific research and management(IJSRM), Volume 2,Issue 11 Pages 1839-1845 ,2014*

[2] D. G. Shin, and K. B. Irani, "Fragmenting relations horizontally using knowledge based approach," IEEE Transactions on Software Engineering (TSE), Vol. 17, No. 9, pp. 872–883, 1991.

[3] E. S. Abuelyaman, "An optimized scheme for vertical partitioning of a distributed database," Int. Journal of Computer Science & Network Security,Vol. 8, No.1, 2008.

[3] Gupta Dhiraj and Gupta V.K., Approaches for Deadlock Detection and Deadlock Prevention for Distributed, Res. J. Recent Sci., 1(ISC-2011), 422-425 (2012).

[4] Michael J. Carey Miron Livny ,Distributed Concurrency Control Performance: A Study of Algorithms, Distribution, and Replication, Computer Sciences Department University of Wisconsin Madison, WI 53706.

[5]Md. Tabrez Quasim, (2013). An Efficient Approach For Concurrency Control In Distributed Database System. Indian Streams Research Journal, Vol. III, Issue. IX

[6] Bernstein P and Goodman N "Timestamp-Based Algorithms for Concurrency Control in Distributed Database Systems," Proc. 6th VLDB Cot& Mexico City, Mexico, Oct.1980.

[7] M. AlFares et al, "Vertical Partitioning for Database Design: A Grouping Algorithm", in Proc. International Conference on Software Engineering and Data Engineering (SEDE), 2007, pp. 218-223.

[8] Naser s. Barghouti and gail e. Kaiser, "Concurrency Control in Advanced Database Applications", ACM Computing Surveys, Vol 23, No 3, September 1991.