ISSN:0975-9646

Proficient Method towards Concurrency Control in Distributed Database

Anil Ahir

Sr. Lecturer ,Babasaheb Gawde Institute of Technology,
Mumbai, India

Abstract-Majority of the research in database management system focuses primarily on centralized database systems. However, with the demand for higher performance and higher availability, database systems have moved from centralized to distributed architectures. As the development and maturity of the popular centralized database system moves towards the distributed approach, the challenges and roles start becoming more complex and complicated. The discussion revolves around the variety of protocols, their working, their advantages and their disadvantages in a distributed environment.Concurrency control is an integral part of database systems. In literature various techniques have been discussed which are used to prevent, detect and resolve the deadlocks. In this report I have analyzed various concurrency control protocol and the deadlock detection and resolution techniques.

Keywords:Timestamp, classification, threshold, testing.

1. INTRODUCTION

The concurrency control in distributed DBMS's is an important research problem. In a distributed database system, data in the system can be accessed by different users at the same time. If these concurrent accesses are not controlled properly, the database will become inconsistent during multiple concurrent updates to the database. Such inconsistencies may give rise to several problems The term 'Distributed' in the concept of 'Distributed Databases' clearly means the environment of the database system that it is not at one place or location but exist in at physically autonomous locations with some interconnection in the logical plan. The 'Distributed Database' can be thought of as a collection of multiple, logically interrelated databases distributed over a computer network in such a way the distribution is clear to the users. The fact that the database is located at multiple sites obviously means that it promotes availability of data to multiple users concurrently without losing the integrity and accuracy of the database. Distributed database as equated to the Centralized database varies mainly in the way the data is truly stored and located. Hence the data is not present at one place it shows the concept of Fragmentation and Replication [9]

Several issues could be generated if concurrency of transaction is not handle technically For example, it migh
- lead to the lost update problem in a funds transfe transaction. It may be, however, almost impossible t execute a task, such as funds transfer usually consisting c a sequence of several atomic operations, withou
- temporarily violating the integrity constraints. Howeve these atomic operations can be grouped into time_units c consistency called transactions. Thus if a transaction start with a consistent database and the transaction is run b itself to completion, it is guaranteed to produce a consister database.

2. DISTRIBUTED DATABASE SYSTEM

2.1Summary on Distributed database Systems

A distributed database management system (DDBMS) manages the database as if it were all stored on the same computer. A distributed database is a database in which shares of the database are stored on multiple computers within a network. Users have admittance to the share of the database at their location so that they can access the data relevant to their tasks without interfering with the work of others.The DDBMS synchronizes all the data periodically and, in cases where multiple users must access the same data, ensures that updates and deletes performed on the data at one location will be automatically reflected in the data stored elsewhere.

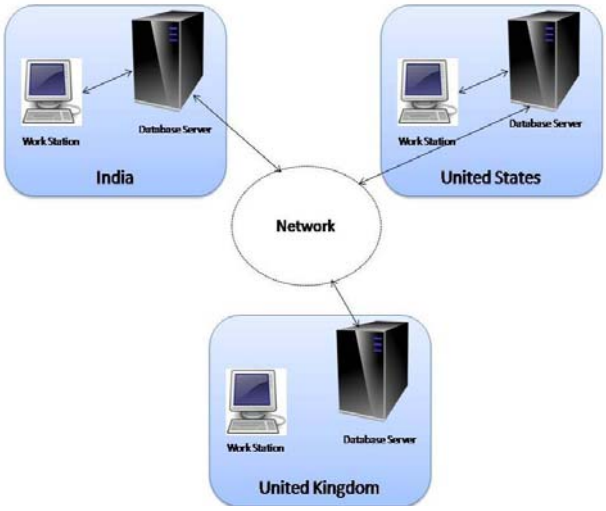

FIG. 1 Distributed database conceptual diagram

2.2Serializability

When more than one transaction is executed by the operating system in a multiprogramming environment, there are possibilities that instructions of one transaction are interleaved with some other transaction.[10]

Schedule: A chronological execution sequence of transaction is called schedule. A schedule can have many transactions in it, each comprising of number of instructions/tasks.

Serial Schedule: A schedule in which transaction are aligned in such a way that one transaction is executed first. When the first transaction completes its cycle then next transaction is executed. Transactions are ordered one after other. This type of schedule is called serial schedule as transactions are executed in a serial manner.

2.3 Deadlock

In computer science, **deadlock** refers to a specific condition when two or more processes are each waiting for another to release a resource, or more than two processes are waiting for resources in a circular chain (see Necessary conditions). Deadlock is a common problem in multiprocessing where many processes share a specific type of mutually exclusive resource known as a software, or soft, lock. Computers intended for the time-sharing and/or real-time markets are often equipped with a hardware lock (or hard lock) which guarantees exclusive access to processes, forcing serialization. Deadlocks are particularly troubling because there is no general solution to avoid (soft) deadlocks.[9]

This situation may be understood by an analogy with two people who are drawing diagrams, with only one pencil and one ruler between them. If one person takes the pencil and the other takes the ruler, a deadlock occurs when the person with the pencil needs the ruler and the person with the ruler needs the pencil, before he can give up the ruler. Both requests can't be satisfied, so a deadlock occurs.[9]

2.3.1 Dead lock prevention

Deadlocks can be prevented by ensuring that at least one of the following four conditions occur:

- Removing the mutual exclusion condition means that no process may have exclusive access to a resource. This proves impossible for resources that cannot be spooled, and even with spooled resources deadlock could still occur. Algorithms that avoid mutual exclusion are called non-blocking synchronization algorithms.
- The "hold and wait" conditions may be removed by requiring processes to request all the resources they will need before starting up (or before embarking upon a particular set of operations); this advance knowledge is frequently difficult to satisfy and, in any case, is an inefficient use of resources. Another way is to require processes to release all their resources before requesting all the resources they will need. This too is often impractical. (Such algorithms, such as serializing tokens, are known as the all-or-none algorithms.)
- A "no preemption" (lockout) condition may also be difficult or impossible to avoid as a process has to be able to have a resource for a certain amount of time, or the processing outcome may be inconsistent or thrashing may occur. However, inability to enforce preemption may interfere with a priority algorithm. (Note: Preemption of a "locked out" resource generally implies a rollback, and is to be avoided, since it is very costly in overhead.) Algorithms that allow preemption include lock-free and wait-free algorithms and optimistic concurrency control.
- The circular wait condition: Algorithms that avoid circular waits include "disable interrupts during critical time_unittions" , and "use a hierarchy to determine a partial ordering of resources" (where no obvious hierarchy exists, even the memory address of resources has been used to determine ordering) and Dijkstra's solution.[9]

3. CONCURRENCY CONTROL TECHNIQUES FOR DISTRIBUTED DATABASE

3.1 Priority Based Locking

In the proposed protocol we assume that the database is fully replicated. The system consists of N number of nodes consisting of different transactions. Each node sends its read/write request to a global node. Global node assigns the requested lock to the transaction having the higher priority. A waiting queue is maintained in which the transactions waiting for a lock are placed. First, we find the total execution time of the transactions performing read/write operation on the basis of timestamp based priority. Then we find the total execution time of transactions on the basis of read transactions being assigned a higher priority. Then we compare the results from both the cases.

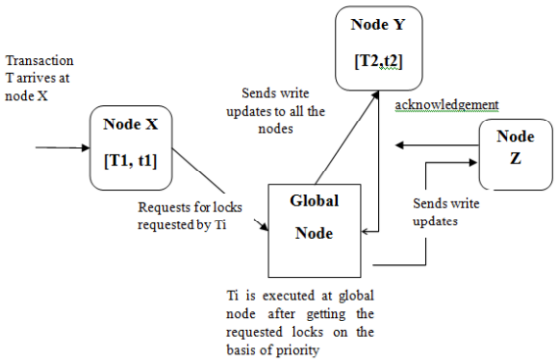

FIG. 2 Distributed Locking Concurrency Control Protocol

Each transaction is assigned a unique timestamp whenever it enters the system. When a transaction T1 arrives at node X with timestamp t1 and T2 arrives at node Y with the timestamp t2, the following steps are performed (shown in Fig 2):

- Node X and node Y requests from the global node the locks for all the entities referenced by the transaction T1 and T2.
- The global node checks all the requested locks. If some entity is already locked by another transaction, then the request is queued. There is a queue for each entity and the request waits in one queue at a time. If the request is for read lock then the global node grants the read_lock to all the requested transactions at same time because the proposed method works on assigning higher priority to the transactions requesting for read lock.
- When the transactions get all its read locks, they are executed at the global node (the execution can also take place at node X, but that may require more messages). The execution time of read transactions is calculated as the maximum time of all the transactions requesting for read lock. The values of read set are read from the database, necessary computations are carried out and the locks are released when the transaction completes. The global node then assigns the lock to the next transaction waiting for write lock in the queue. Necessary computations are carried out

and the values of the write set are written in the database at the global node.

- The values of the write set are transmitted by the global node to all other nodes.
- Each node receives the new write set and updates the database; then an acknowledgment is sent back to the global node.
- When the global node receives acknowledgments from all other nodes in the system, it knows that transaction T1 has been completed at all nodes. The global node releases the locks and starts processing the next transaction [7]

3.2 Threshold and Classification Based Locking

- All the transaction is classified according to its execution time.
- Classification is done on basis of Threshold Time value (T)
- Those transactions satisfy the T will group into once.
- Groups are classified on the basis of T value, every next group will follow 2 Times the T value of its successor group
- All transaction will be grouped according to its execution time.
- Classification will be on until all transaction is grouped.
- Once the classification is done the scheduler now keep track on ET, An ET is an threshold time given to any group for execution.
- After expiry of ET a next group is selected.
- If no transaction left, ET restart and next group is selected. Until no transaction left in any group

4. Computing Execution Time On The Basis Of Threshold And Classification.

Consider an Example to test this algorithm
T1(r)=2time_unit,
T2(r)=2time_unit,T3(w)=3time_unit,T4(r)=3time_unit, T5(r)=4time_unit, T6(w)=5time_unit, T7(r)=6time_unit, T8(r)=6time_unit,T9(w)=2time_unit,T10(r)=3time_unit
According to algorithm
We need to select T value
Let Value of T=2

- Scheduler will now classify the above transaction according to T, where T=2time_unit.

Hence all above transaction will now follow the below given group

- First Group=T1,T2,T9----------→ less than 2 time_unit's (T)
- Time_unitond Group=T3,T4,T5,T10----> more than 2 and less than 4time_unit's (2T)

T class (<=2 Time_unit)	2Tclass (<=4 and >2 Time_unit)	4T class (<=8 and >4 Time_unit)
GROUP A	**GROUP B**	**GROUP C**
T1 Read 2tu	T3 Write 3tu	T6 Write 5tu
T2 Read 2tu	T4 Read 3tu	T7 Read 6tu
T9 Read 2tu	T5 Read 4tu	T8 Read 6tu
	T10 Read 3tu	

- Third Group=T6,T7,T8----------→more than 4 and less than 8time_unit's (4T)

Now ET need to be select, Let ET time is 6 time unit
Therefore now for execution, each group will be executed not more than 6 Time_units.
Within ET time limit if no transaction left in a group new transaction from higher adjacent group will be selected.
While executing Read transaction, the read process can run parallel hence more than one read transaction can be executed, hence when multiple read instruction are executed , the transaction with higher execution time is taken under consideration.

For First ET cycle.
From Group A & B T1+T2+T9+T3=5time_unit

T class (<=2 Time_unit)	2Tclass (<=4 and >2 Time_unit)	4T class (<=8 and >4 Time_unit)
GROUP A	**GROUP B**	**GROUP C**
T1 Read 2tu	T3 Write 3tu	T6 Write 5tu
T2 Read 2tu	T4 Read 3tu	T7 Read 6tu
T9 Read 2tu	T5 Read 4tu	T8 Read 6tu
	T10 Read 3tu	

At first ET cycle T1,T2,T9 are read instruction hence they are executed parallel with higher time unit that is 2, hence all 3 transcation will execute in 2time_unit hence, still time_unit remain in ET1 therefore higher adjacent group is selected and T3 which is write instruction all fall below remain ET1 time. Hence T3 is executed separately. Later 1 time_unit remain and next instruction in group B is of 3 time_unit hence ET1 is terminated

For second ET cycle.
From Group B, T4+T5+T10=4time_unit

T class (<=2 Time_unit)	2Tclass (<=4 and >2 Time_unit)	4T class (<=8 and >4 Time_unit)
GROUP A	**GROUP B**	**GROUP C**
T1 Read 2tu	T3 Write 3tu	T6 Write 5tu
T2 Read 2tu	T4 Read 3tu	T7 Read 6tu
T9 Read 2tu	T5 Read 4tu	T8 Read 6tu
	T10 Read 3tu	

At this cycle T4, T5 and T10 are executed parallel because they are read instruction ant T5 has higher tu value among them of 4tu. Therefore no other transaction left in same group and 2 time_unit left in ET2 hence higher adjacent group is selected but no transaction available less than 2 time_unit. hence ET2 is terminated.

For third ET cycle
From Group C,T6=5time_unit

T class (<=2 Time_unit)	2Tclass (<=4 and >2 Time_unit)	4T class (<=8 and >4 Time_unit)
GROUP A	**GROUP B**	**GROUP C**
T1 Read 2tu	T3 Write 3tu	T6 Write 5tu
T2 Read 2tu	T4 Read 3tu	T7 Read 6tu
T9 Read 2tu	T5 Read 4tu	T8 Read 6tu
	T10 Read 3tu	

At this Cycle of ET only T6 which is write instruction is executed isolated, hence no other transaction falls in remaining time of ET3

For Fourth ET cycle

From Group C,T7+T8=6time_unit

T class (<=2 Time_unit)	2Tclass (<=4 and >2 Time_unit)	4T class (<=8 and >4 Time_unit)
GROUP A	GROUP B	GROUP C
T1 Read 2tu	T3 Write 3tu	T6 Write 5tu
T2 Read 2tu	T4 Read 3tu	T7 Read 6tu
T9 Read 2tu	T5 Read 4tu	T8 Read 6tu
	T10 Read 3tu	

All 10 transaction is executed in ET1+ET2+ET3+ET4=5+4+5+6= 20 Time_unit.

CONCLUSION

If the same example is to executed on basis of time priority3.1 above shown it would take 25 Time_unit to finish

REFERENCES

[1] M. Kaur and H. Kaur, "Concurrency Control in Distributed Database System", International journal of Advanced Research in Computer Science and Software Engineering, vol. 3,Issue 7, July 2013, pp. 1413-1417

[2] Gupta V.K., Gupta Dhiraj and ShuklaDatta, "Concurrency Control and Time_uniturity issues of Distributed Databases Transaction", Research Journal of Engineering Science, vol. 1(2),Aug 2012, pp. 70-73

[3] A. Yadav and A. Agarwal, "An Approach for Concurrency Control in Distributed Database System", International Journal of Computer Science & Communication, vol. 1, no. 1, Jan 2010, pp 137-141

[4] S. Jitendra and Gupta V.K, "Concurrency Issues of Distributed Advance Transaction Process", ReseachJournal of Recent Sciences, vol. 1, Feb 2012, pp 426-429

[5] K. Ganesh, K. Ajit and A. Umesh, "Concept and techniques of transaction processing of Distributed Database management system", International Journal of Computer Architecture and Mobility, vol-1, issue-8, June 2013, pp 1-4

[6] Svetlana ZhelyazkovaVasileva and AleksandarPetrovMilev, "Models of 2PL Algorithm with Timestamp Ordering for Distributed Transaction Concurrency Control", International Journal of Soft Computing and Engineering,vol-3, issue-4, September 2013, pp. 247-252

[7] MinakshiSangwan, "Priority-Based Locking for Concurrency Control in Distributed Databases", International Journal of Engineering Trends and Technology, vol-12, number-4, June 2014, pp. 170-175

[8] Swati Gupta and MeenuVijarania, "Analysis for Deadlock Detection and Resolution Techniques in Distributed Database", International Journal of Advanced Research in Computer Science and Software Engineering, vol-3, issue-7, July 2013, pp. 399-403

[9] Anjali Ganesh Jivani "An Insight into Concurrency Control Protocols of Distributed Databases" International Journal of Emerging Science and Engineering (IJESE), Volume-1, Issue-12, October 2013