Efficient Concurrency Control Mechanism for Distributed Databases

Parul Tomar
Department of Computer Engineering
YMCA University of Science and Technology
Faridabad, India
Email id: Ptomar_p@hotmail.com

Suruchi
Department of Computer Engineering
YMCA University of Science and Technology
Faridabad, India

Abstract – **Distributed database is not a new term in the current world. A distributed database is a set of several databases that are correlated to each other in a logical fashion over the network. Data is distributed as well as replicated at various locations in order to increase the availability. During recent years, data distribution has been evaluated as an important issue for databases. Since the data is distributed, it becomes necessary that all the copies be synchronized at the time of any data update, thus leaving all the copies with consistent data. This research paper will present a consistent concurrency algorithm which presents a solution to consistency problem of distributed databases**.

Keywords – Concurrency control, Database, Distributed databases, Timestamp, Transaction, Voting .

NOMENCLATURE

DB: Database
DBM: Database Manager
DBMS: Database Management System
DDBMS: Distributed Database Management System
DTM: Distributed Transaction Manager

I. INTRODUCTION

Database (DB) can be defined as collection of data with a well-defined structure and purpose defining the related activities [1,2,3]. Database management system (DBMS) is a software that control the database and is used to maintain and utilize large collections of data [4,5].
During recent years "distribution" has been evaluated as a critical and important issue for databases. A distributed database is a set of several parts that correlate with each other logically over a network of interconnected computers. The data in a database can be distributed across multiple physical locations. Data can be distributed either through Fragmentation or Replication. Because of the distribution of database, multiple users can access it without interfering with each other [6,7].
There are various advantages of having distributed databases such as Local autonomy or site autonomy, Protection of valuable data, improved performance etc [8,9,10].
Even though distributed systems are found in many applications, designing them is a difficult task. There are many issues that need to be considered during its implementation. Depending on the application requirements key characteristics of a distributed system are Heterogeneity, Security, Openness, Concurrency, Scalability, Fault Tolerance, and Transparency[4,11].

II. REQUIREMENTS OF DISTRIBUTED DATABASES

The implementation of the distributed system is very complex. In order to get the desired result, various issues needs to be considered. The complexities should not worry the user of the distributed system from using it. Users of the system should not be aware of these complexities. This property of the distributed system is called its transparency.

III. COMPONENTS OF DISTRIBUTED DATABASES

Distributed DBMS consists of various components. Every database is managed by a database manager. Database Manager is one of the main components of distributed database [12]. The responsibility of database manager is to deal with a portion of the distributed database. A database manager may be a centralized DBMS.

Another main component is the User Request Interface. It is a client program which acts as an interface to the Distributed Transaction Manager. "A Distributed Transaction Manager is a program that translates requests from the user and converts them into actionable requests for the database manager". Components diagram of a DDBMS is shown in the Fig.1.

978-9-3805-4421-2/16/$31.00

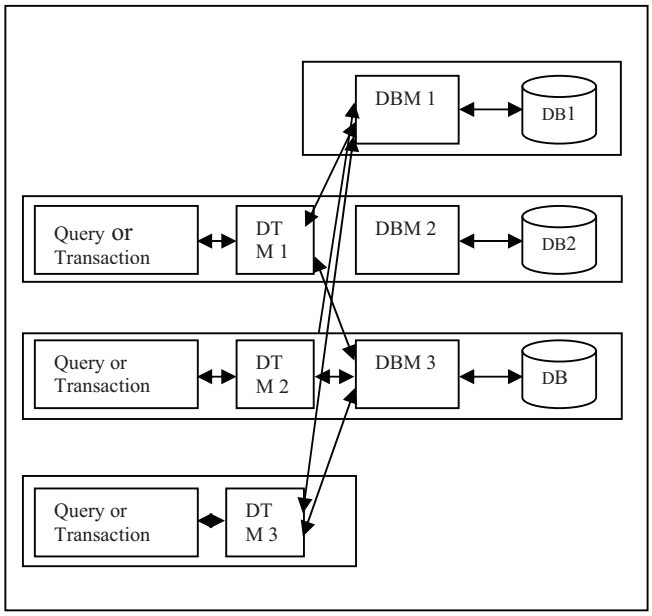

Fig. 1 Components Of Distributed Systems

IV. CONCURRENCY CONTROL IN DDBMS

Concurrency control is the process of maintain the consistency of database in case of concurrent access in a multiuser database management system (DBMS). Each user uses the database in a multi-programmed manner without knowing the presence of other user. The main goal of concurrency control is to prevent database updates performed by one user from interfering with database updates performed by other user [13]. The concurrency control is more difficult in case of distributed DBMS (DDBMS) because users may access data stored at different replicated copies of original data stored at central site. Concurrency control mechanism at one site cannot instantaneously know about interactions at other site [14, 15]. In distributed databases concurrency control and recovery is more difficult as many problems arise in a distributed DBMS environment which are not present in case of centralized DBMS environment. These include the following:

- Dealing with multiple copies of the data items
- Failure of individual sites
- Failure of communication links
- Distributed commit
- Distributed deadlock

Different Concurrency Control Mechanism in DDBMS

The most important concurrency control protocols can be categorized as follows: rollback based, Timestamp based and wait / Lock based Protocols [16, 17].

Wait: In this technique, one of the conflicting transaction waits for the completion of the actions of other transaction.

Timestamp: Timestamp is a technique in which ordering of the transaction is done. A unique timestamp value is assigned to each transaction. All the conflicting actions of the transactions execute according to their timestamp values.

Rollback: Rollback is nothing but moving back to the original point of start. If two transactions conflict, either some actions or the complete transaction is rolled back to the starting point. This approach is also called optimistic because it is expected that conflicts are such that only a few transactions would rollback.

Locking: The traditional approach to concurrency control is based on locking. In this method that is based on the allocation of data to transaction, when a transaction wants to access data for writing or reading it should send a corresponding lock request to a section called lock manager.

Voting[18,19,20]: In the voting method, a lock request is sent to all sites that includes a copy of the data item. Each copy maintains its own lock and can grant or deny the request for it. A transaction can hold a lock only if it gets the majority of votes i.e. atleast (n/2+1) votes. As the decision of locking is dependent on the votes of all the involved sites, this methods is truly a distributed concurrency control method. This method also has certain disadvantages which are as follows:

- Consistency problem: Though majority protocol is used but all the sites are not updated at all the times so our database is not in a consistent state.
- Dirty Read: If a query request arrives on any site which is not updated, then query will read the old value of the data item.

This research paper will solve these problems of voting method with the help of timestamp of last successful updation on data values.

V. PROPOSED ALGORITHM

The proposed algorithm assumes an environment within which database is replicated at different sites and these copies of a database are accessible from each site (see Fig 2). Users interact with the Distributed DBMS with the help of transactions. Application Processes (Aps) will initiate the query and updation in database. The database copy at each site is accessible only through a database managing process (DBMP) which resides at that site. Each access to the database is completed by DBMP acting on behalf of the initiating AP. Each site contains table carrying information about the timestamp of the last transaction that wrote into it.

Execution of transaction takes place in two phases. First phase is the one in which the transaction executes at the central site. For every execution a unique timestamp is recorded in the central site.

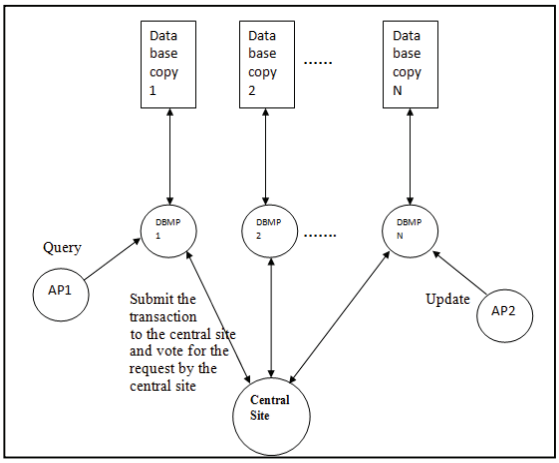

Fig. 2 Each database copy is directly accessible only through its local database managing process (DBMP)

In the second phase, central site sent the list of updated data items at the other replicated sites. This is done to maintain the consistency of data in distributed database system.

Here, in the proposed algorithm no node sends valuable information during message passing. That is, the metadata is sent instead of valuable information for message passing mechanism to reduce bandwidth load which is one of the algorithm's features.

A. Working of Algorithm

In this proposed algorithm, there is a central site that monitors all the connected sites. The algorithm assumes a complete redundant database, in which all the logical data items are stored at every site.

Central site holds all the key information about the data and the transactions (Fig. 3). All the transactions that come at any site are forwarded to the central site which then executes the request and broadcasts the request to the connected sites. Any home site can initiate any transaction but all the transactions are forwarded to the central site so each transaction have a unique timestamp value, which is used to decide which commitment request will be forwarded to other sites, in case of pending request and new request.

Connected sites now vote for the commitment of the transaction in terms of YES or REJ.

Site can vote yes only if it is free and all the data required by the transaction can be locked by it. It votes reject if the data required by the requested transaction is not free and the site is locally using the data.

Data about which site voted reject and which site voted yes is maintained at the central node in the database table table_ready.

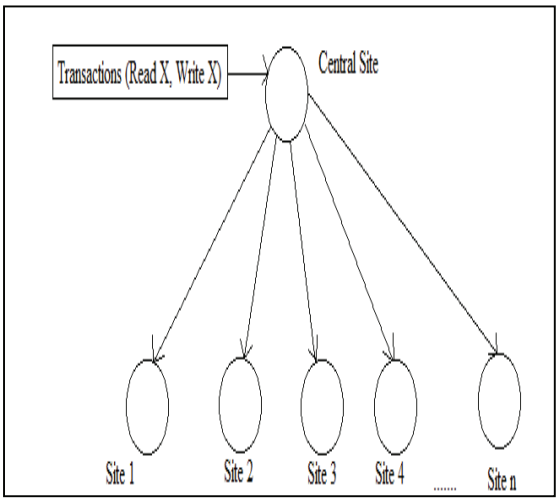

Fig. 3 DDBMS showing central site and all the connected sites and all the transactions arriving at the central site.

Each site maintains a database structure carrying the following information (Table 1):-

TABLE I. DATASTRUCTURE AT EACH SITE

Transaction ID	Timestamp	Data Item	Data Value	Status

Where:-
Transaction ID: This column stores the transaction ids of different transactions.
Timestamp: It stores the timestamp at which the data item was updated.
Data Item: Stores the name of the data item for which request is made.
Data Value: Stores the value which was updated at a given timestamp.
Status: It stores the status if the site accepted or rejected the request.

This database structure helps in maintaining a consistent database system. As when a mjority is claimed, only those sites that voted YES are updated and all the remaining that voted reject have the old data only. In this case this algorithm sents an update request to only those sites that voted REJ again after certain interval of time so that all the sites have a consistent data.

Request is sent untill a new request on the same data item arrives or untill all the sites are updated.

In case if a new request arrives before the commitment of the old request on all the sites then the algorithm compares the timestamp and the request with the old transaction ID is dropped.

B. Steps Involved

Step 1: Transaction Tj for updating a data item is forwarded to the monitoring node M. The monitoring node, M then executes the request and broadcasts a message (Metadata) to the other n-1 nodes asking them to check status of data Q1, …, Qn. Nodes should check for acceptance of the requested data Qi. With this message the algorithm is initiated.

Step 2: Upon receiving the message (Metadata), node i check whether transaction j can lock its requested data or not. If node i locks data copy Qj by its local lock manager, then it will send a message for transaction j to the monitoring node saying YES it can lock the data requested by the transaction Tj.

Step 3: In this step, monitoring node does this: Let Mj be the message monitoring node has received from node i in step 2. So, this message is a vote for transaction j. Therefore, monitoring node counts received votes for transaction j. If number of votes for each transaction is at least n/2+1, transaction j can be run on these sites that voted yes.

Step 4: In case where the sites send a reject vote but still majority is claimed, then transaction is not committed at those sites, and the updation request is again sent to those sites only and steps1 to 4 are repeated for those sites only, that voted reject.

Step 5: In case when a new request arrives while the old one is still not committed at all the sites, the old request is rejected and step 1 to step 5 are started with the new transaction on the same data value based on the time stamp values.

VI. CONCLUSION

Distributed database is a collection of databases that are stored at different sites connected through computer network. Local autonomy, improved reliability, better data availability, increased performance and lower communications costs because of local access are the vsrious characteristics of distributed databases. Therefore, concurrency control in distributed environment is a critical task due to highly dynamic environment. In recent years, several concurrency control protocols have been proposed for distributed systems and prominent among them are locking protocols (2PL), timestamp protocol and majority protocol The dynamic nature of such systems makes it highly susceptible to various inconsistency problems. Dynamic infrastructure and decentralized administration make such systems vulnerable to inconsistency and complexity. In this paper, the optimized majority consensus algorithm for concurrency control in distributed environment is studied. Although many solutions have been proposed but still these solutions are not perfect in term of effectiveness and efficiency. So there is a requirement of protocol which can provide consistency to distributed databases. In this paper the limitations of the majority protocol are analysed and solution to the problem has been proposed. Some extra parameters along with Timestamp values are added to the algorithm in order to remove the problems of Majority protocol. The newly proposed algorithm will help in better consistency of distributed databases.

REFERENCES

[1]. M.Tamer Ozsu, Patrick Valduriez, "Distributed Data Management: Unsolved Problems and New Issues".
[2]. Coronel Rob, "Introduction To Database Concepts", 2011
[3]. Parul Tomar, Megha, "An overview of distributed databases," International Journal of Information and Computation Technology, Volume 4, Number 2, pp. 207-214, Feb 2014
[4]. George Coulouris, Jean Dollimore, Tim Kindberg, "Distributed Systems Concepts and Design" 3rd edition, *Addison-Wesley*.
[5]. Elmasri and Navathe, "Fundamentals of Database Systems".
[6]. M.T. Özsu and P. Valduriez, "Principles Of Distributed Database System" *Englewood Cliffs, NJ: Prentice-Hall*, 1991.
[7]. Bhargava, B., "Concurrency Control in Database Systems", *IEEE transactions on knowledge and data engineering*, 1999.
[8]. http://www.cs.ucl.ac.uk/staff/W.Emmerich/lectures/DS97-98/dsee3.pdf.
[9]. http://www.moreprocess.com/database/sql/advantages-and-disadvantages-of-distributed-database.
[10]. Nitesh Kumar, Saurabh Bilgaiyan, Santwana Sagnika, "An Overview of Transparency in Homogeneous Distributed Database System".
[11]. Parul Tomar, Mohit Bhardwaj , "A Review on Deadlock Detection in Distributed Database", Advances in Computer Science and Information Technology", vol 2, no 8, pp. 63-65, April-June 2015
[12]. "Components of Distributed database system," May 24, 2004.
[13]. Philip A. Bernstein and Nathan Goodman, "Multiversion Concurrency Control-Theory and Algorithms", Harvard University.
[14]. Breitbart, Y., Korth, H. F., Silberschatz, A., "An optimistic concurrency control protocol for replicated databases. Fundamental Problems in Computing", Part II, 327-351, 2009.
[15]. http://en.wikipedia.org/wiki/Distributed_Database.
[16]. Agrawal, R., Carey, M. J., Livny, M., "Concurrency Control Performance Modeling: Alternatives and implications", ACM Transactions on Database Systems, 12(4), 1987.
[17]. Bhargava, B., "Concurrency Control in Database Systems", IEEE transactions on knowledge and data engineering, 1999.
[18]. Bhargava, B., Van Nostrand Reignhold, "Concurrency Control and Reliability in Distributed Database System," Software Eng. Handbook, 331-358,1983.
[19]. Robert H. Thomas, " A Majority Consensus Approach to Concurrency Control for Multiple Copy databases" Bolt Beranek and Newman*, Inc.*
[20]. S.M. Almasi Mousavi, H.R. Naji And R. Ebrahimi Atani , "Optimization Of Majority Protocol For Controlling Transactions Concurrency In Distributed Databases By Multi-Agent Systems" *, IJAOR Winter 2013*