ISSN : 2494-9150 Vol-01, Issue 10, JAN 2016.

CONCURRENCY CONTROL IN DISTRIBUTED DATABASE SYSTEMS

[1]Prof. Vinayak Sinde, [2]Preeti A. Aware
[1]Asst. Professor, [2]ME Student, [1,2]SLRTCE, Mira Road, Mumbai, Maharashtra, India.
[1]vdshinde@gmail.com, [2]awarepreeti11@gmail.com

***Abstract* — Concurrency control in Distributed database system is analyzed in this paper. The need and improvement in distributed database system is of utmost importance in today's world. The difficulties mostly faced in distributed database system is Protecting the ACID property i.e. when concurrent transactions perform read and write atomicity, consistency, integrity and durability of the database should be preserved and Recovery method to be used when distributed database crashes.**

Keywords— Allocation, concurrency control, Distributed database, Distributed design, fragmentation, replication, transaction.

I. INTRODUCTION

Today's business needs secure, reliable and easily accessible information. One of the examples of distributed database is of Aadhar card database. If the data is located in the central database, it will be slow to retrieve the data. Also the central server could become bottleneck due to thousands of request. The distributed database system can be used in this case to enhance the retrieval and the query can be processed concurrently at different sites. Due to this fact there is an increasing demand for client/server applications and distributed database system. A distributed database can be viewed as a single logical database that is made up of several physical database from different computers that are connected through interconnection network. This logical database can also be called as a virtual database. The real database is physically stored at different locations. Data can be accessed by users at any location as if the data was stored at the user's own location. The access mechanism of distributed database is made transparent to the users by using a special software called distributed database management system. The main aim of distributed database management system software is to provide a mechanism by which the distributed database appears to be as centralized database to the users. This appearance of centralized database can be achieved with the help of several different kinds of transparencies like: Transaction transparency, performance transparency, location transparency, fragmentation transparency, schema change transparency, and local DBMS transparency. The most vital issue in distributed database system is concurrency control. Concurrency control is the method of synchronizing simultaneous access to a database in a multiuser database management system. There are different techniques that deliver concurrency control. Some of the techniques are: Time stamping, two phase locking and Multiversion time stamp.

II. MOTIVATION

The use of distributed database has been encourage due to several business situations

Data communications costs and reliability: In the distributed system when the data is geographically dispersed, application that required the data can be partitioned for processing at each site. The above method is economical as it reduces the communication cost. The second advantage is that it is cost effective to have smaller computing power that to have a large computing power of a single super computer.

Database recovery: Data can be replicated on several computers. If the primary copy is damaged, data can be still accessed due to replication and at the same time the primary copy can be restored. One of the normal forms of distributed

database recovery is replication of the data across several sites.

Data sharing: Every business decision requires sharing of data across different business unit. Data can be merged on demand.

Distribution and autonomy of business unit: In modern organizations departments and divisions are mostly geographically dispersed and cross national boundaries. Mostly each division has power to build its own information system. This divisions frequently want the native data on which they have control. Acquisition and business mergers produce this kind of environment.

Improved performance: Several sites can store the data retrieved from transactions. This helps in executing the transaction in parallel. The performance of the system is enhanced by using several resources in parallel.

Increased availability and reliability: The database becomes unavailable to the users if the centralized system fails. On the other hand in a distributed system if a component fails the database is still available to the users.

Faster response: The data needed by the users at a particular site can be fulfilled by the data stored at the site, and it depends upon the way the data is distributed. The query processing can be sped up as delays due to communication and central processing is minimized. The complex queries can be split up into simple sub queries and processed in parallel at different sites for faster response. [6]

III. DISTRIBUTED DATABASE DESIGN

Distributed database system are needed for applications that require distributed access and high availability in case of failures. The examples of system that require distributed database are automated system, financial institution and airline reservation system. The technique used for distributed system is same as that of centralized system in addition to few factors.

Data fragmentation: We need to define logical unit for data distribution and allocation in a distributed system. The database can be fragmented into logical units call fragment to be stored at separate sites. The tables are the simplest logical units.

Data fragmentation are of three types:

Horizontal fragmentation:

The subset of rows in a table form the horizontal fragment. The table is divided into fragment by selecting specific row and then distributing this fragment to several sites in distributed system. [1]

Consider a relation employee (Emp_id, Emp_name, Emp_designation, and Emp_salary)

Emp_id	Emp_name	Emp_designation	Emp_salary
001	Mr. Mark	clerk	10000
002	Ms.Jane	clerk	10000
003	MrWolworth	manager	30000
004	Mr. oxford	account	25000

Condition: Emp1: fragmented with Emp_salary<= 10000

Emp1

Emp_id	Emp_name	Emp_designation	Emp_salary
001	Mr. Mark	clerk	10000
002	Ms.Jane	clerk	10000

Condition: Emp2: fragmented with Emp_salary>10000

Emp2

Emp_id	Emp_name	Emp_designation	Emp_salary
003	MrWolworth	manager	30000
004	Mr. oxford	account	25000

Vertical fragmentation: In vertical fragmentation table is divided into attribute. Each vertical fragment should contain the primary attribute, so that the entire table can be built.

Consider a relation Emp (Emp_id, Emp_name, Emp_designation, Emp_salary)

Emp1

Emp_id	Emp_name	Emp_designation
001	Mr. Mark	clerk
002	Ms.Jane	clerk
003	MrWolworth	manager
004	Mr. oxford	account

Emp2

Emp_id	Emp_salary
001	10000
002	10000
003	30000
004	25000

Hybrid fragmentation: The combination of both horizontal and vertical fragmentation is called hybrid fragmentation. With the help of join and union operator the original table can be reconstructed.

Data replication: To achieve fault tolerance the data is replicated at two or more sites. Several sites can contain the copy of each fragment.[1]

IV. FUNDAMENTAL OF TRANSACTION PROCESSING AND CONCURRENCY CONTROL

Transaction: A series of operations performed on database is called transaction. The vital concern regarding transaction management is if the database is consistent before the initiation of transaction, then it has to be consistent after the completion of transaction.

Properties of transaction: The consistency and reliability of database can be achieved using ACID properties. The Properties are Atomicity, Consistency, Isolation, and Durability.

Atomicity: In this transaction is considered as a unit of operation. It directs that the operations related to a transaction should be complete or should not take place.

Consistency: The correctness of a transaction is its consistency

Isolation: At all times the transactions should see consistent database. When one transaction is modifying the database other transactions should not be able to read or modify the database.

Durability: This feature guarantees that when a transaction commits its outcomes are permanently stored in the database. Hence after a commit of transaction, even if the system crashes, the outcomes after commit are not modified.

Concurrency control: In distributed database system many users are concurrently accessing the database. The mechanism of coordinating concurrent access to a distributed database is called concurrency control. The user is under the impression that he is working on a dedicated database. The central objective for attaining this goal is to prevent one database update interfere with another database update.

V. DISTRIBUTED CONCURRENCY CONTROL ALGORITHM

In this paper, we study some of the distributed concurrency control algorithm. We review some of the prominent features of the four algorithms. First we will explain the concept of distributed transaction.

Distributed transaction: In this the transaction executes on multiple processes on different machines. Distributed transaction processing system are intended for heterogeneous systems having transaction-aware resource managers. Distributed transaction execution requires coordination between the local resource managers and the global transaction management system. The two basic element of a distributed transaction are the transaction processing monitor and the resource manager.

Acid properties must also be observed by distributed transactions However it is difficult to maintain these properties in distributed transaction because here any

process can fail. In case of such failure each process should undo the operation it had done on behalf of the transaction.

By using the following two features distributed transaction processing system can preserved the acid properties.

1. Recoverable process: These processes log the activity and in case of failure they can be restored to their earlier state.

2. Commit protocol. In this several processes coordinate for aborting or committing of a transaction.

The 2 phase locking protocol is the most common commit protocol.

Distributed Two Phase Locking (2 PL): To guarantee serializability of data there are different concurrency control methods. Locking is one of the methods. There are several locking methods. In distributed database system 2 phase locking is the basic concurrency control protocols. It has the method 'write all and read any'. Read locks are set by the transaction that requires to read the data. The read locks are converted to write locks when the data requires to be updated. For read lock it is sufficient to lock any one copy of the data item but for write lock it has to be set on all the copies of the data item. As the transaction executes write locks are obtained. A write request is blocked until all the copies of the previous write have successfully updated. Untill the transaction have successfully committed or aborted all the locks are withheld by the transaction.[2]

The 2 PL locking determines when the transactions can obtain or release the locks. The transactions are forced to acquire and release the locks in 2 steps by the 2 PL protocol.

Growing phase: Transactions can obtain locks but cannot release any locks.

Shrinking phase: Transaction cannot obtain any new locks they can only release the locks. At first the transaction enters the Growing phase where it request for the desired locks and then it enters the Shrinking phase where all the locks obtained by it are released and new locks are not obtained. In 2PL transactions should acquire all the needed locks before moving into unlock state. The 2PL ensures serializibility of data but it does not guarantee that deadlock will not happen. When a transaction blocks, local deadlock is checked and resolved by restarting the transaction with the most initial startup time among the ones tangled in the deadlock cycle.' Snoop 'process is used to detect global dead lock. It requests from all sites 'wait –for' information to check and resolve any global deadlocks.

Wound Wait (WW): The distributed wound- wait is the second algorithm. It follows the same method as 2PL protocol. The deadlock problem is differently handled in this algorithm. Rather than using 'wait for' information as in 2PL for checking local and global deadlock, timestamps are used to prevent deadlocks in this algorithm. According to their initial startup time each transaction is numbered. The younger transactions are prevented from making the older transaction wait. If the lock is requested by an older transaction and this request requires the older transaction to wait for younger transaction, the younger transaction is wounded. It is restarted lest it is in the second phase of its commit protocol. The possibility of deadlocks can be eliminated when younger transaction waits for an older transaction. [2]

	T1 is allowed to
t(T1) < t (T2)	Abort and rolled back
t (T1) > t (T2)	wait

Example wound-wait algorithm

t (T1)<t (T2) : If the transaction that is requesting for the lock on the data item t (T1) is older than the transaction that currently holds the lock t (T2), then the requesting transaction has to be aborted or rolled back.

t (T1) > t (T2):- If the transaction that is requesting for the lock t (T1) is younger than the transaction t (t2) that is currently holding the lock on the data item, then the requesting transaction has to wait.

Basic timestamp ordering: To identify a transaction a unique timestamp is created by the DBMS. According to the order in which the transaction are submitted to the system, a timestamp value is assigned to it. The start time of a transaction is its start time. Basic timestamp ordering algorithm is the third algorithm. Based on their timestamps the transactions are being ordered. The participation of the transaction can be serializable and the equivalent of a serial

transaction is to schedule the transaction according to their timestamp values. This is referred as timestamp ordering.

It uses transaction start up time like the wound-wait but rather in a different way. Time stamp is associated with recently associated data items by the BTO. Instead of lock approach concurrent data access is performed in the order of timestamp values of the transactions. When a transaction tries to perform out of order access, it is restarted. A read request is permitted for a data item if the timestamp of the requestor is greater than the item's write request. A write request is permitted if the requestor's timestamp is greater than the read timestamp of the data item. When the timestamp of the requestor is less than the write timestamp, then the update is ignored.[2] In case of replicated data ' write all ' and 'read any' method is used, so that write request should be sent to all copies whereas read request can be sent to any one copy. This algorithm can be integrated with 2 phase commit as follows. Until commit time writes keep their updates in private workplace.

Distributed Optimistic: The fourth algorithm is timestamp based, distributed, optimistic concurrency control algorithm. In this certification algorithm are exchanged during the commit protocol. A write timestamp and a read timestamp is maintain for each data item. Reading and updating of data items by transaction is done freely. Updates are stored in local workplace until commit time. To read the data item, the transaction must remember version identifier (i.e. write timestamp) related with the data item when it was read. The transaction is assigned a global time stamp when all of its cohorts have completed the work and reported the work to the master. Each cohort is sent this timestamp in the 'prepare to commit' message, this certifies all writes and reads as follows. [2]

The request for read is certified when

1. The current version of the item and the item that was read is same.
2. None of the write with new timestamp has already been certified.

The request for write is certified when

1. None of the later read have been certified and consequently certified.
2. None of the later reads have already been locally committed.[2]

Oracle's Distributed Database

A distributed database is a set of databases stored on multiple computers that typically appears to applications as a single database. Consequently, an application can simultaneously access and modify the data in several databases in a network. Each Oracle database in the system is controlled by its local Oracle server but cooperates to maintain the consistency of the global distributed database.

VI. CONCLLUSION

In this paper, we have discussed that compared to centralized database, distributed database has more advantages. Also we have described concurrency control algorithms in distributed database like distributed 2PL, wound wait, basic timestamp ordering, and distributed optimistic. ACID properties of database is of utmost importance and it has to be maintained while concurrently accessing the database.

REFRENCES

[1] Gupta V.K., Sheetlani Jitendra, Gupta Dhiraj and Shukla Brahma Datta, *Concurrency control and Security issues in Distributed Database system*, Vol. 1(2), 70-73, August (2012)

[2] Arun Kumar Yadav& Ajay Agarwal, an *Approach for Concurrency Control in Distributed Database System*, Vol. 1, No. 1, pp. 137-141, January-June (2010).

[3] Navathe Elmasri, *Database Concepts*, Pearson Education, V edition (2008).

[4] Fundamentals of DBMS, Lakhanpal Publisher, III edition (2008).

[5] Swati Gupta, Kuntal Saroha, Bhawna, *Fundamental Research of Distributed Database*, IJCSMS International Journal of Computer Science and Management Studies, Vol. 11, Issue 02, Aug 2011.

6] Distributed Database http://wps.pearsoned.co.uk/wps/media/objects/10977/11240737/Web%20chapters/Chapter%2012_WEB.pdf

