2014 Fourth International Conference on Communication Systems and Network Technologies

Comparative Study of Concurrency Control Techniques in Distributed Databases

Anand Mhatre
Department of Computer Engineering
Ramrao Adik Institute of Technology, Nerul
Navi Mumbai, India
anand9304@gmail.com

Rajashree Shedge
Department of Computer Engineering
Ramrao Adik Institute of Technology, Nerul
Navi Mumbai, India
rajashree.shedge@rait.ac.in

Abstract— **In today's world many of researches have been done on distributed databases. The main issue in distributed databases is to maintain consistency in databases. To maintain consistency in database, correctness criteria must be met. Many of the concurrency control methods are presented earlier, but they have problems about delay, performance, waiting time and number of message exchanges while maintaining correctness. Our paper presents comparison of the recent concurrency control methods considering the above mentioned parameters.**

Keywords- Distributed Database; Concurrency control;

I. INTRODUCTION

In distributed databases, there are multiple sites involved on which the data is stored. Hence concurrency control is necessary to ensure reliability and consistency of transactions on these databases [1]. Database is inconsistent when transactions are in deadlock. Therefore concurrency control is needed to maintain database in consistent state. For concurrency control serializability is the most important criterion. The serializability ensures the correctness of the database by converting conflict equivalent schedule to a serial schedule [1].

The basic concurrency control methods in distributed system are: two phase locking (2PL), where transaction obtain lock on data item when they read and convert this lock to write when they need to update it. In wound wait (WW) rather than waiting for the information from all sites deadlocks are prevented by use of timestamps. The third method is Basic Timestamp Ordering. Like WW it employ transaction startup timestamp but use it differently. Distributed certification is operated by exchanging certification information. In all above methods there is some point of deadlock situation formed between the executions of operations.

Hence some advance concurrency control methods are proposed like Speculative locking [2], Validation Queue [3], Stamp based [4]. In validation queue transactions are validated on client side and also on server side. In Stamp based, validation is done by matching stamp values. In speculative locking, validation is based on status of preceding transaction. But these methods vary in terms of performance, delay, waiting time and number of message exchanges.

Rest of the paper is organized as follows section II describes related work; Section III shows promises of distributed databases; section IV describes distributed concurrency control algorithms; section V gives comparative analysis; finally conclusion is done in section VI.

II. RELATED WORK

There are many techniques proposed for controlling the concurrent transactions in distributed databases apart from the basic techniques.

In [2] Mohit Goyal, T. Ragunathan and P. Krishna Reddy proposed the Speculative locking protocols for distributed environment. Fahren Bukhari and Santosh Shrivastava [3] proposed ROCC (Read Commit Order Concurrency Control) scheme. There are two validation queues, one CVQ (Cache validation queue) which is located at the client side and maintained by local cache manager and another is SVQ (Server validation queue) located at server and maintained by scheduler component. In [4], priority protocol is explained by implementing a timestamp where another value of flag is added and this value doesn't change unless transaction update has been successfully committed. Atul Adya, Robert Gruber Barbara, Liskov Umesh Maheshwari [5] describes an efficient optimistic concurrency control scheme for use in distributed database systems in which objects are cached and manipulated at client machines while persistent storage and transactional support are provided by servers. The scheme provides both serializability and external consistency for committed transactions; it uses loosely synchronized clocks to achieve global serialization. In [6] T. Ragunathan, P. Krishna Reddy have given semantics-based high performance asynchronous speculation based protocol to improve parallelism among Update transactions and read only transactions. In [7] the approach reduces waiting time for read only transactions and improves its performance. Speculation based locking along with synchronous approach is suggested in [8]. In [9] T. Ragunathan, P. Krishna Reddy, and Mohit Goyal propose semantics- based high performance asynchronous speculation based protocol to improve parallelism among Update transactions and read only transactions. In [10] general concurrency control algorithms in the distributed environment is proposed. These includes the locking algorithms, time stamp algorithm and optimistic algorithm. Arun Kumar Yadav and Ajay Agarwal also proposed the transaction processing in distributed environment. Kamal Solaiman, Graham Morgan [11] suggested optimistic algorithm for transactions resides on resource constraint system.

III. Promises of DDBS

There are many of the advantages of DDBS. These are basic for achieving concurrency in databases. All of these can be viewed as promises of DDBS [1]. These are:

A. *Transparent Management of relational and distributed data .*

Transparency ranges from higher system semantics to lower implementation issues. Its advantage is to provide higher level support in the development of complex applications.

B. *Reliability through distributed transaction.*

DDBS improves reliability of data by the use of data replication and thereby reducing risk of single point of failure. In DDBS some of data may be unreachable in this case with proper care user are permitted to access data from other part of distributed database.

C. *Improved performance.*

Distributed databases first fragment conceptual database thereby enabling data stored in closed proximity to its point of use.
It has two advantages
1) Each site handles portion of database and contention for CPU and I/O services are not severe as for centralized databases.
2) Localization reduces remote access delay which involved in wide are network.

Most of databases are designed to gain full benefit from data localization.
This full benefit of reduced contention and communication network overhead is obtained from proper fragmentation and distribution of database.

D. *Easier system expansion.*

In distributed database it is much easier to accommodate increasing database size. Expansion can be handled by adding more processors and storage power to the network. One of the aspect of the system expansion is economic. It normally cost much less to put together smaller computers with equivalent power of single big machine.

IV. Distributed concurrency control Algorithms

A. *Distributed Speculative Locking (DSL): -*

There are two protocols 1) Distributed Synchronous Speculative Locking for Read Only Transactions. 2) Distributed Asynchronous Speculative Locking for Read Only Transactions.

1) *Distributed Synchronous Speculative Locking for ROT*

Execution phase:-

Read only transaction (ROT) request to update transaction (UT) for obtaining read lock on data object held by UT. If read request of ROT is in conflict with UT, ROT has to wait till UT produces updated value of data object. When UT produces updated values both original and updated values are sent as response to home site of ROT. Now ROT carries out speculative execution by accessing both original and updated values of data object. Two list are maintained dependent _set and dependent _list. Dependent _set contains identifiers of conflicting UT. And dependent _list contains identifiers of conflicting UT form which ROT access updated value of data object. Whenever ROT obtains all require locks on completion it enters into commit phase.

Commit phase:-

On complete execution of ROT to select appropriate speculative execution ROT communicates with UT's home site to know commit status of UT as specified in dependent _set. If status of UT is committed Speculative execution of ROT which has committed effect of UT is committed by considering dependent _list [2].

2) *Distributed Asynchronous Speculative Locking.*

Execution phase:-

ROT request to UT for obtaining read lock on data object held by UT. If read request of ROT is in conflict with UT, ROT doesn't wait UT to produces updated value of data object. It starts speculative execution by reading original value of data object. When UT produces updated value. Updated value is sent as response to home site of ROT. Now ROT carries out speculative execution by accessing that updated value of data object. Two list are maintained dependent _set and dependent _list. Dependent _set contains identifiers of conflicting UT. And dependent _list contains identifiers of conflicting UT form which ROT access updated value of data object. Whenever ROT obtains all require locks on completion, it enters into commit phase.

Commit phase:-

After complete execution of either one speculation of ROT. ROT communicates with UT's home site to know commit status of UT as specified in dependent _set. If status of UT is committed, Speculative execution of ROT which has committed effect of UT, is committed by considering dependent _list.

B. *Validation Queue Approach: -*

Here author proposes validation queue algorithm.
The validation occurs in both side, client side and server side. Client side is for validating local transaction and server side is for validation of update transaction. There are two queues used for both client and server. For client it is Client Validation Queue (CVQ) and for server it is Server Validation Queue (SVQ).

1) *Client Side Validation:-*

This algorithm is invoked by local cache manager when it validates the transaction. Its main objective is to prevent commit of incorrect execution of transaction. Client Side Validation uses CVQ for storing the execution order of the elements. In addition to Read, Commit and Validated elements, CVQ has Local Validated and Update Propagation elements. An Update Propagation element represents the execution of a remote update transaction. It contains set of read and write of update transaction. When the local manager receives an Update Propagation messages from server side update propagation elements are inserted in CVQ. Read and commit elements are inserted in CVQ when local cache manager receives read or commit request from respective local transaction [3].

Transaction succeeds local validation process if it is read only transaction and all its elements are merged to be validated elements. Otherwise it is update transaction and process is as follows:

1)Local validated elements have merged all elements of locally validated transaction .
2)It is checked that no any updates made by update propagation element in CVQ that conflicts with local validated elements. If no conflict is there local cache manager will submit commit request to server otherwise loca element is discarded and transaction aborts.
3) If commit message is received from server local validated element becomes validated element otherwise abort message is received and local element is discarded.

2) *Server Side Validation:-*

Main tasks of the Server Validation Algorithm: to validate an update transaction at the server, to propagate the updates to the caches, and to maintain Cache elements. A Cache element contains the information about the objects stored at a cache. This algorithm uses SVQ in the same manner as the client uses CVQ. SVQ may contain Cache, Commit, or Validated elements. Each cache has its own version number.

When fetch request is received, at server side it is treated as from cache transaction. When server receives fetch request it creates commit element of cache transaction and put into the SVQ. Server has to validate that cache transaction. If validation is successful fetch operation is sent to the object manager. Otherwise it is removed from SVQ.

Whenever the server receives a commit request of a transaction, it compares sequence number carried by commit request message and sequence number recorded on its cache transaction at the server if it is not match the commit request is sent back to its originated cache manager for verification; if it is match the server creates two elements. These are a Read element and a Commit element. The Read element contains the list of object identifiers that have been read by the update transaction. The Read element will not be executed; it is needed for the validation purposes only. Commit element contains the list of object identifiers that the update transaction wants to update. The Read element is inserted into SVQ at the position right after the position of the Cache element of the associated cache transaction in SVQ.

If the validation succeeds, the server sends a commit message as acknowledgement to the originating cache manager, executes the updates of the transaction, and refreshes other caches by sending Update Propagation messages, with new sequence numbers(cache version no.) [3]. If the validation is failed, the server removes the commit element from SVQ and sends an abort message to the originated cache manager.

C. Stamp Based Approach: -

This algorithm is based on stamp value. Stamp is assigned to each data object. When client side transaction send request to server for data object, Server assign stamp value to that data object and send response back to the client transaction. At time of validation whenever client side transaction send its commit request to server, on server side server validate value of data object by comparing stamp value in commit request with stored in its database. If match is found success flag will be generated and sent to the client side else it indicates that value of accessed data objects are modified by another transaction and all data objects are return to the client. It inform client to restart the transaction. If match found stamp value of data object is stored in server database and it is incremented by one every time a successful update is held on a data object [4]. Basically user receive data and stamp value and wait for few cycle before the commit process. DBMS will be able to determine if data received from server is update meanwhile execution of the procedure or not. This stamp value is initially set to zero and incremented when successful update is held on record.

V. COMPARATIVE ANALYSIS

TABLE 1. COMPARATIVE ANALYSIS OF CONCURRENCY CONTROL METHODS

Technologies → **Parameters ↓**	**DSL**		**VQA**	**SBA**
	DSSLR	**DASLR**		
Waiting time	More	Less as compare to DSSLR	Very less	Little waiting
Validation	Based on commit status of UT	Based on commit status of UT	Based on conditions .	Based on matching of stamp flag.
Performance	Less than DASLR	Better than DSSLR	Moderate	Faster
Delay	More	Slight less than DSSLR	More	Less
No. of message exchanges	Message exchanges 4	Message exchanges 5	Message exchanges 9	Message exchanges 2

TABLE 1 shows comparison of the techniques with following parameters.

1) Waiting time: - Time required for the transaction to access data object when it is already accessed by other transaction.

- In DSSLR, ROT has to wait for UT until it produces updated value to access data items. Because ROT operates synchronously that is ROT has Commit dependency on UT. In DASLR there is less waiting for ROT because there is no dependency of ROT on UT.

- In validation queue approach, there is very less waiting of transactions because ROT are validated on client side and UT are validated on server side and waiting is among ROT and UT.

- In stamp based approach, there is little waiting because they don't wait on each other and simply access object from server but waiting is at the commit time of the transaction and it is less than validation approach.

2) Validation: - Checking for Consistency of database.

- In DSSLR and DASLR, validation of ROT is based on commit status of UT.

- In VQ validation is based on two conditions.
  1) If there is no any element of other transaction Tj in between read and commit element of transaction Ti, then transaction Ti is successfully committed.

  2) First read element of transaction Ti is in conflict with validated element Tj during the execution of Ti but commit element of Ti is not in conflict with validated element of Tj.

- In stamp based approach, validation is based on matching of stamp flag. If only stamps are match transaction is validated.

3) Performance: - Unit of execution in less amount of time.

- DSSLR performs better than DASLR but till the number of active transactions in system =60 but its overall performance is less than DASLR. Because of less waiting than DSSLR and data contention DASLR performs better.

- VQ algorithm outperforms basic snapshot isolation schemes. The novel feature is that load for validating transactions at commit time is divided into client side and server side there by reducing load on server side. Thus improving scalability and performance.

- In stamp based approach, performance is faster because committed or aborted transaction is based on stamp values. Faster performance requires faster execution. And execution is faster because communication between client and server is less.

4) Delay: - Time required to commit transaction.

- In DSSLR, there are more number of speculative executions of ROT. After completion of both speculative executions to select appropriate execution one round of communication is performed with home site of UT and ROT is committed by retaining one execution if it contains effect of commit status of UT based on commit/abort replies. So delay is more. In DASLR here are also more number of speculative executions of ROT. After one of them is completed one round of communication is performed with home site of UT and ROT is committed by retaining one execution if it contains effect of commit status of UT based on commit/abort replies. So delay is lesser than DSSLR.

- In validation queue, delay for ROT is less than UT. Because for commit of UT check is done on both client and server. So for validation of update transaction delay is more than ROT.

- In stamp based approach, client side transactions are committed if their stamp values are matched with stamp values in server side database. Once this is done transaction is ready to commit. So delay is less than other approaches.

5) Number of message exchanges: - It is number of messages executed for transaction to commit

- In DSSLR, there are less number of messages exchanged (i.e. 3) between transactions as compared to DASLR (i.e. 4) because when conflict between UT and ROT transactions occurred ROT doesn't wait on UT and proceed with its execution. So in DSSLR ROT access both original and updated values in one request message where as in DASLR ROT first received original value and after production of updated value it is received in another request message therefore one request message is more in DASLR.

- In VQ algorithm, total number of message exchanges are 9.On client side they are 3 and on server side they are 6.

- In stamp based approach, first client side transaction send request message for object on server side and server assign stamp to it and grant the request. This is one message. After performing its execution, client side transaction send commit message to server. If stamp value match, server can send committed or aborted message to client side successfully. So total number of messages are 2.

VI. Conclusion

Our paper has focused on recent concurrency control techniques like distributed synchronous speculative locking, distributed asynchronous speculative locking, validation queue approach and stamp based approach. Out of them stamp based approach for validating transaction at server side is seem to be best in terms of waiting time, delay, performance, number of message exchanges. Also this method reduces restart time and enhances efficiency.

References

[1] M.T. Ozsu and P. Valduriez," Principles of Distributed Database Systems", Third Edition, DOI 10.1007/978-1-4419-8834-8_11, © Springer Science Business Media, LLC 2011.

[2] Mohit Goyal, T. Ragunathan and P. Krishna Reddy, "Extending Speculation based Protocol for Processing Read Only Transactions in Distributed Database System" 12th IEEE International Conference on High Performance Computing and Communications (HPCC), vol., no.pp.527,532, 1-3 Sept. 2010.

[3] Fahren Bukhari and Santosh Shrivastava, "An Efficient Distributed Concurrency control Scheme for Transactional System with client side caching", Proceedings of the 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems, pp.1074,1081, June 2012.

[4] Obaidah A. Rawashdeh, Hiba A. Muhareb and Nedhal A. Al-sayid, "An optimistic approach in Distributed Database Concurrency Control", 2013 5th International Conference on Computer Science and Information Technology (CSIT), pp.71-75, March 2013.

[5] Atul Adya Robert Gruber Barbara Liskov Umesh Maheshwari, "Efficient Optimistic Concurrency Control Using Loosely Synchronized Clocks",Proceedings of the ACM SIGMOD International Conference on Management of Data, San Jose, CA, May 1995.

[6] T. Ragunathan and P. Krishna Reddy, "Semantics- Based High Performance Asynchronous Speculative Locking Protocol for Improving the Performance of Read- only Transactions" 2008.

[7] T. Ragunathan, "Extending Speculation for improving Performance of Read only Transactions", 2008.

[8] T. Ragunathan and P. Krishna Reddy, "Exploiting Semantics and Speculation for Improving the Performance of Read Only Transactions ", International Conference on Management of Data COMAD 2008.

[9] T. Ragunathan, P. Krishna Reddy, and Mohit Goyal, "Semantics-Based Asynchronous Speculative Locking Protocol for Improving the Performance of Read- only Transactions", SpringSim' 10, Apr 12-15, 2010, Orlando.

[10] Arun Kumar Yadav and Ajay Agarwal, "An Approach for Concurrency Control in Distributed Database System", International Journal of Computer Science and Communication vol.1, No1. January-June 2010, pp.137-141.

[11] Kamal Solaiman, Graham Morgan, "Later Validation/Earlier write: Concurrency Control for Resource Constrained System with Real time properties" IEEE 30th Symposium on Reliable Distributed Systems, pp.9- 12, Oct. 2011.