Update Propagation Protocol Using Tree of Replicated Data Items in Partially Replicated Databases

Misook Bae and Buhyun Hwang
Dept. of computer science, Chonnam national university, Korea
e-mail : msbae@chonnam.chonnam.ac.kr
Dept. of computer science, Chonnam national university, Korea
e-mail : bhhwang@chonnam.chonnam.ac.kr

Abstract : The replication of data is used to increase its availability, improve the performance of a system, and advance the fault-tolerance of a system. In this paper, it is required for the information about the location of a primary site of the replicas of each data item. The replicas of each data item are hierarchically organized to a tree based on the fact that the root is the primary replica in partially replicated databases. It eliminates useless propagation since the propagation can be done to only sites having replicas following the hierarchy of data. And our algorithm schedules transactions so that the execution order of updates at each primary site is identical at all sites by using timestamp. Using our algorithm, the consistent data are supplied and the performance of read-only transactions can be improved by using tree structure of replicas of each data item.

1. Introduction

Replication of data is used to increase its availability and reliability of a system, and to balance the server load in distributed database system. Replicated database systems are classified into fully replication and partially replication according to the degree of replication for the data. The previous works have nearly treated only fully replication[1,2,3,4,5,6]. But, the system performance degrades according to the increase of unnecessary messages when the fully replication mechanism is applied to the partially replication of data as it is.

The important issues in the replicated databases are the maintenance of consistency of the replicated data and the propagation of updates. Most of systems use 1SR(one-copy serializability) as the correctness criteria of the execution of transactions.

The methods to propagate updates to replicas can be classified into eager propagation and lazy propagation. Eager propagation keeps all replicas exactly synchronized at all sites. But it decreases update performance and increases transaction response times. Lazy propagation asynchronously propagates updates to other sites after committing a transaction. It gets a fast response time since it does not need to wait for the propagation to all nodes, but may causes the inconsistency of replicas due to the difference of the update propagation time

This paper proposes the replicated data management method in partially replicated database. As referred previously, if the fully replication is used for partially replication as it is, it would cause the unnecessary messages and thus the propagation would take a long time. To prevent these problems from being caused, we will organize replicated data items hierarchically to form a tree for each data item. And, we propose a mechanism that does not cause unnecessary messages and propagates updates fast. Our algorithm uses a lazy propagation and resolves non-serializable execution of transactions that can occur when updates are propagated.

The remainder of this paper is organized as follows. Section 2 presents the related works that control replicas in replicated databases. Section 3 presents the problems that can be caused in the previous works. Section 4 proposes our algorithm and its system model. Section 5 concludes the paper.

2. Related works

This chapter presents previous works proposed for methods controlling replicas in replicated databases.

In [7,8,9,10], it is shown that the replication consistency could be violated due to lazy propagation in lazy master replication databases, and it is proposed for the immediate-propagation strategy to maintain the replication consistency They indicated the low performance in 2PC and proposed the methods to improve the degree of freshness using relaxation of mutual consistency. The degree of freshness reflects the recentness of data. But, for this the reliable multicasting protocol must be supported.

The replicated databases must guarantee 1SR to maintain the replication consistency. 1SR(One Copy Serializability) means that the effects of transactions performed by various clients on replicated data items should be the same as they were performed one at a time on a single data item. So when data are replicated at several sites, an update propagation method is necessary to maintain the replication consistency. [11,9,12,13] proposed update propagation methods that can be classified into the eager update propagation and lazy update propagation.

According to the propagation of updates, there are a group method and a master method. In the group method, updates may be issued from any copies. The group method has more chances for update conflicts than the master method that all updates are propagated from primary copy of the data to its replicas.

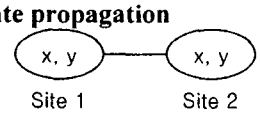
[13] referred to the problem that can occur to maintain strong consistency in large scale replicated databases. It proposed an asynchronous update propagation method based on hierarchical structure in order to decrease the overhead needed for all sites to have an identical state in large scale replicated databases. According to geographical

locations and logical criteria, the sites are grouped into clusters. And it relaxed the criteria of consistency according to the hierarchy of clusters. It significantly reduces the transaction abort rate and improves user-level transaction response time. In addition, it referred to the need of weakly consistency for highly available services which gives the clients reasonable response time, even if some results do not conform to sequential consistency[11,13].

3. Problems

This section presents update propagation problems that can occur in the replicated databases.

[Example 1] an example of non-serializable execution due to lazy update propagation



Suppose that there are replicated two data items x and y in the site1 and site2 as upper figure. Suppose that there are two transactions T1 : w1(x) r1(y) and T2 : r2(x) w2(y), and the primary site of data item x is site1 and the primary site of data item y is site2. A write operation must be executed in primary site and a read operation can be executed in arbitrary site. T1 and T2 can be committed in site1 and site2, respectively. Site1 propagates the updates of data item x to site2, and site2 propagates the updates of data item y to site1. As a result, it is possible to generate a history H1: r1(y) w1(x) c1 w2(y) (T1 → T2) in the site1 and a history H2: r2(x) w2(y) c2 w1(x) (T2 → T1) in the site2. The union of H1 and H2 creates a cycle in their serialization graph.

A large scaled system uses a hierarchical structure because the hierarchical structure is better than the linear structure in the search time and scalability. There is a primary copy on the top-level node in the hierarchical structure and the update must be executed in the primary site. At this time, if an update transaction is submitted to the site not in the top-level, its update operations have to be submitted to the primary site, they are executed in the primary site, and the result of updates is propagated to the sites in the lower level than that of the primary site. If this structure has the deep level, it would take a long time to find a primary site and propagate the updates. If an update transaction is submitted to the site in the bottom-level(leaves), there are a lot of overheads to find the primary site in the top-level. It results in the inefficiency since even the sites that don't have the replicated data receive the update message.

4. Proposed algorithm

This chapter presents an algorithm of UP-RDT(Update propagate Protocol using Replicated Data Tree) that we propose and its example.

4.1 System model

UP-RDT is based on the system architecture like in Figure 1. The architecture is composed of a set of sites connected through a network. Each site consists of a global transaction manager (GTM) and a site manager. The data is replicated partially. GTM assigns a timestamp to each transaction. It becomes a coordinator for update transactions. Each status database (status-DB) maintains the location of a primary site that executes the update operation. Each site manager having a primary copy propagates the updates efficiently using a tree structure which maintains the location of the other replicas. The structure of sites can be reconfigured to propagate the updates efficiently using a replicated tree, too[13].

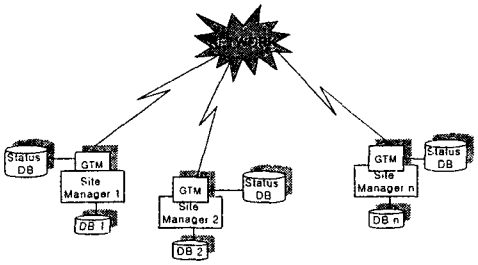


Figure 1. System architecture

4.2 Assumptions and data structures

Our algorithm has been developed under the following assumptions.

- Each data is partially replicated.
- A network is reliable. That is, the message is transmitted from a site to another site eventually.
- The multi-view for each replicated data is allowed. This means some data inconsistency can be caused since all replicas don't have the same values while propagating the updates.
- The master update method is used to update the replicated data. Only a primary site can execute updates and propagates them when the transaction commits.
- The clock of each site is synchronized to give unique timestamp to each transaction.

The status-DB and the replicated data tree structure are presented like in Figure 2 and 3. The status-DB maintains the location of primary replica for each data item. The primary site of each replica maintains the information about the location of the sites having the data using a tree, and propagates its updates efficiently. The example of tree structure in a primary site is like in Figure3. We treat the detailed contents of the configuration of tree in section 4.3.

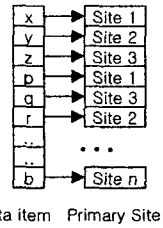


Figure 2. Example of status-DB

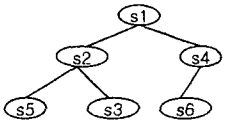


Figure 3. Example of tree structure for arbitrary data

4.3 UP-RDT algorithm

This section presents the transaction execution and the updates propagation method using the tree of sites having replicas in the partially replicated database.

• The organization of status-DB and the reconfiguration of sites

GTM collects the information about primary copy of the each data item to organize status-DB. The primary site of each data item has the most frequent access for the data. And the system can reconfigure the site architecture in the same of tree-structured architecture of replicated data to propagate the updates efficiently.

• The submission of a transaction and the allocation of timestamp

The user transaction is submitted to GTM of a site, and the site to which a transaction is submitted becomes a coordinator. The coordinator gives an unique timestamp to each transaction. The timestamp is composed of a local timestamp and a site number. The local timestamp indicates the clock time of the site that a transaction is submitted, and the site number is used to differentiate the same local timestamps for the uniqueness of timestamps. And the site manager can classify transactions into the update transactions and the read-only transactions.

• The schedule of each operation

A read operation is executed in the submitted site, and an update operation should be executed in its primary site. If the submitted site has not the data item to read, the nearest site having it is selected since it has the least access overhead. GTM maintains GSG(global serialization graph) to guarantee the serializable execution of transactions. The transaction should not cause a cycle to execute the read or write operation when the edge is inserted in GSG. If there is no cycle in GSG, it has to be executed in the order of timestamps.

• The commit of a transaction

The coordinator of the transaction uses 2PC protocol to commit the transaction.

• The updates propagation

When the update transaction commits, the coordinator submit its update operations to the corresponding primary sites. After executing update operations, the primary site propagates its update results to its children. The timestamp of the committed transaction is assigned to the update operation to be propagated. The primary site of each data item maintains the tree structure that consists of sites of the replicas. The root of tree for a data item is a site having the most frequent access to it. To minimize propagation time, a tree structure is a balanced tree that has the same depth for all nodes.

Update propagation algorithm

1. When a transaction Ti is submitted a site Si;
   GTM1 assigns unique timestamp to Ti;
   If Ti is a read-only transaction, then
   For each read operation,
   submit it to the site that has the least overhead to execute the operation;
   else
   For each operation,
   If it is a read operation, then submit it to the site that has the least overhead to execute the operation;
   If it is a write operation, then submit it to the primary site for a data item to write;
2. When a transaction Ti is committed;
   If Ti is a read-only transaction, it is committed;
   If Ti is an update transaction, a commit protocol (2PC) is performed;
3. If the result of the commit protocol is commit,
   then T1 is committed;
   else T1 is aborted;

Commit Protocol

- Coordinator

1.a coordinator sends a vote request to primary sites(participants) which have data items to be updated;
2.if all participants voted to YES, send commit message to all participants;

- Participant

1.a participant votes YES if the execution of update operations guarantees serializability;
otherwise, it votes NO;
2.if a participant receives a commit message, it propagates the updates following down the tree of each data item;

[Example 2] Example of the execution of UP-RDT algorithm

Suppose that the partially replicated databases include four sites and a status-DB that maintains the location of primary site of each data item. Let the primary site of data items x, y, z, and a be S1, S2, S3, and S4, respectively.

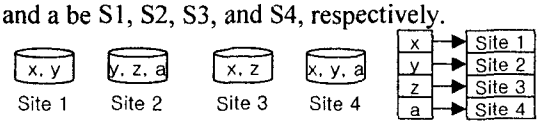


Figure 4. Partially replicated database and its status-DB

Suppose that following 5 transactions are submitted to this system.

T1 : r1(x) r1(y)
T2 : w2(x) r2(z)
T4 : r4(x) r4(a) r4(y)
T5 : w5(x) w5(z)
T3 : r3(x) w3(y) w3(z)

Suppose that the timestamp of the each transaction is given ts(T1)=10.1, ts(T2)=12.2, ts(T4)=15.4, ts(T5)=20.5, ts(T3)=25.3. The tree structures of the replicated data in each primary site are presented as in Figure 5. The root of the tree indicates the primary copy.

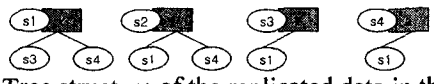
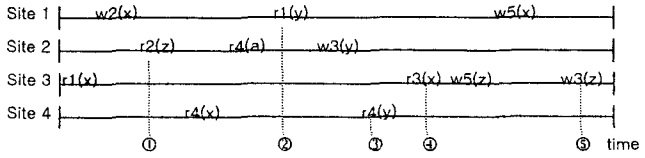


Figure 5. Tree structure of the replicated data in the primary site

The execution of transactions in each site is as follows.



A write operation must be first executed in the primary site of a data item and the update has to be propagated to child nodes of a tree of replicas. A read operation can be executed in arbitrary site._For that, GTM should maintain the information about primary copy of the each data item.

Transactions T2, T1, T4, and T5 can be committed at the time ① for T2, at the time ② for T1, at the time ③ for T4, and at the time ④ for T5, respectively. But T3 is aborted at the time ⑤ because w3(z) becomes too-late operation. When T2 commits at the time ①, the update of x is propagated to S3 and S4 through the replicated data tree. Every GTM maintains SGT as follows. We can know SGT guarantees the serializability of transactions.

The serializable order for x : T1→T2→ T4 → T5

Or T1→T2→T3→T5

The serializable order for y: T1 → T3 → T4

The serializable order for z: T2 → T5

The serializable order for a: T4 □

We prove UP-RDT algorithm guarantees serializability.

[Theorem 1] UP-RDT algorithm guarantees the serializability of transaction.

Proof. If UP-RDT algorithm does not guarantee serializability of transactions, SGT generated by UP-RDT would include a cycle. We will show that SGT generated by UP-RDT does not include the cycle.

Suppose that SGT generated by UP-RDT includes the cycle such as T1→T2→ ... Tn→T1. For the transaction Ti and Tj, the edge Ti→Tj in SGT means ts(Ti) < ts(Tj) since conflict operations are executed in order of timestamps. If SGT include the cycle, it means the edge T1→ ...→ Ti indicates ts(T1) < ts(Ti) and the edge Ti→... →T1 indicates ts(Ti) < ts(T1). It results in ts(T1) < ts(Ti) < ts(T1). It is contradictory to the assumption that timestamp is unique and conflict operations are executed in order of timestamps. So UP-RDT algorithm schedules the transactions so that SGT does not include the cycle, and guarantees the serializable execution of transactions.

5.Conclusion

Replication of data is used to increase its availability and improve the performance of a system. The replicated data control mechanisms of the previous works are based on the fully replicated databases. They have the shortcoming of the possibility of the nonserializable execution of transactions and of the long propagation time due to unnecessary message overheads since the messages can pass through sites not having replicas.

The recent replicated database applications have a tendency that their databases are largely scaled. The efficient replicated data control mechanisms in large scaled systems need the correctness criteria to guarantee the correct execution and the efficient update propagation method. And a large scaled system may have a lot of loads to propagate update results when a update transaction commits.

This paper proposed the update propagation algorithm applicable to partially replicated databases. The algorithm guarantees the serializability of transactions using timestamp and reduces the updates propagation time using a tree of replicated data. It uses 1-SR as correctness criteria of the execution of transactions, and can fast transmit update operations to primary sites using the status-DB that maintains locations of primary sites of data items. It reduced the updates propagation time by maintaining the balanced tree of replicated data in the primary site of each data item.

Acknowledgement

This work was supported by Brian Korea 21 Progect.

6. References

[1] Mesaac Makpangou, Guillaume Pierre, Christian Khoury and Neilze Dorta, "Replicated Directory Service for Weakly Consistent Distributed Caches", IEEE, p92~p100, 1999.

[2] P. Chundi, D. Rosenkrantz, S. Ravi, "Deferred Updates and Data Placement in Distributed Data- Bases", 12th International Conference on Data Engineering, IEEE, 1996.

[3] R.Jimenez, M.Patino-Martinez, G.Alonso, S.Arevalo, "Reducing The Latency of Non-Blocking Commitment Using Optimism and Replication", 2001.

[4]Yuri Breitbart, Raghavan Komondoor, Rajeev Rastogi, S. Seshadri, Avi Silberschatz, "Update Propagation Algorithms for Replicated Database systems", Technical Report BL0112370-981028TM, Bell Labs,Oct. 1998.

[5] Yuri Breitbart, Raghavan Komondoor, Rajeev Rastogi, S. Seshadri, Avi Silberschatz, "Update Propagation Protocols for Replicated Databases", In Procs. of ACM SIGMOD International Conf. on Management of Data, Philadelphia, 1999.

[6] Yasushi Saito and Henry Levy, "Optimistic replication for Internet data services", 14th International Conference on Distributed Computing (DISC), Oct 2000, Toledo, Spain.

[7] E.Pacitti, E.Simon, "Update Propagation Strategies to Improve Freshness in Lazy Master Replicated Databases", VLDB Journal, p305-p318, 2000 .8

[8] Esther Pacitti, Pascale Minet, Eric Simon, "Fast Algorithms for Maintaining Replica Consistency in Lazy Master Replica Databases", Proceedings of The 25th VLDB conference, Edinburgh, Scotland, p126~p137, 1999.

[9] Jim Gary, Pat Helland, Patrick O'Neil, Dennis Shasha, "The Danger of Replication and a solution", In Procs. of ACM SIGMOD International Conf. on Management of Data, Montreal, Canada, 1996.

[10]Y. Breitbart, H. Korth, "Replication and consistency: Being lazy helps sometimes", In Procs. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Tucson, Arizona, 1997.

[11] Coulouris, Dollimore and Kindberg, "Distributed Systems: Concepts and Design", Edition 3, Addison-Wesley 2001.

[12] Todd Anderson, Yuri Breitbart, Henry F. Korth and Avishai Wool, "Replication, consistency and practicality: Are these mutually exclusive?", In Procs. of the ACM SIGMOD International Conf. on Management of Data, Seattle, WA, 1998.

[13] Xiangning Liu, Bdelsalam Heal, Eimin Du, "Multiview Access Protocols for Large Scale Replication", ACM Transaction on Database Systems, Vol.23 , No.2, p158~198, June 1998.