Data Update Propagation on Synchronous Replication with Master and Group Replicated Databases

Yin Nyein Pann , Nu Yin Kyaw
Computer University, Pyay
nyein.nyeinnyein3@gmail.com;nuyinkyaw@gmail.com

Abstract

All over the world, as IT technology is increasingly, important services are also becoming more convenient. As a result of it; most of the enterprises are using database system. A distributed database system manages and controls the entire databases as a single collection of data. Many distributed database application need to replicate data to improve data and availability and query response time. [8] Most of database products use asynchronous replication, which is very efficient but can compromise consistency. As an alternative, synchronous replication guarantees consistency. [7] In this paper, we use Two-Phase Commit Protocol (2PC) to ensure that both master and slaves install the updates and analyze existing synchronous replication based on server architecture: master and group replicated databases implemented by sale processing system.

Keywords: *synchronous replication, master replicated database, group replicated database, update propagation*

1. Introduction

In the distributed systems community, software based replication is seen as a cost effective way to increase availability. In the database community, however, replication is used for both performance and fault tolerance purposes thereby introducing a constant trade-off between consistency and efficiency. In fact, many commercial and research databases are based on the asynchronous replication model (also called lazy update model) where changes introduced by a transaction are propagated to other sites only after the transaction has committed. This results in minimal overhead but inconsistencies among the copies may arise. [7]This characteristic trade-off does not necessarily imply that consistency is not important in databases.

It is well known to users and designers that the inconsistencies created by lazy replication techniques can be very difficult to solve. It is also well known that such inconsistencies can be eliminated by using synchronous replication models (also called eager replication, i.e., a transaction synchronizes with all copies before it commits).Unfortunately, it is by no means trivial to design efficient eager replication protocols. In practice, given the serious limitations of traditional data replication techniques (deadlocks, overload, lack of scalability, unrealistic assumptions), many database designers do not regard eager replication as a feasible option. [7]

The rest of the paper is structured as follows. Section 2 describes related works. Section 3 introduces background theory. Section 4 describes the architecture of the system. Section 5 describes the process flow of the system. Section 6 demonstrates conclusion of the paper and section 7 describes the related references.

2. Related Works

In Kemme and Alonso demonstrate the practically of synchronous database replication in a local area network environment, where network partitions cannot occur.[10] In Yair Amir and Jonathan identify the performance bottleneck of the existing synchronous replication schemes as residing in the update synchronization algorithm. The authors compare the performance of several such synchronization algorithms and highlight the large performance gap between various methods. They design a synchronous replication scheme that uses an enhanced synchronization algorithm and demonstrate its practicality by building a prototype that replicates a PostgreSQL database system. [10]

Replication supports a variety of applications that often have different requirements. For example, sales force automation, field service, retail and other mass deployment applications require data to be synchronized between central database systems and a large number of small, remote sites which are often disconnected from central databases. In may tar hla myint demonstrates the use of replication theory in banking system. The author describes the nature of replication theory. In

this system, the user checks in his account in bank by using ATM machines and then ATM machine searches the nearest replica by using discovery services. [11]

3. Background Theory

3.1 Replication

Replication is the process of copying and maintaining database objects in multiple databases that make up a distributed database system. Basically, the main idea behind replication is that an update operation is performed first locally and then propagated to other remote replica databases belonging to a single whole system (which becomes therefore distributed). Theoretically, supposing there are no delays due to synchronization and no update transactions conflicts, this system would assure the end user a fast and consistent way of information retrieval with high availability (due to alternate data access).[3] The motivation for replication are

- Increased availability
- Increased performance
- Fault tolerance
- Consistency
- Replication Transparency

By storing the data at more than one site, if a data site fails, a system can operate using replicated data, thus increasing availability and fault tolerance. At the same time, as the data is stored at multiple sites, the request can find the data close to the site where the request originated, thus increasing the performance of the system. [5] Consistency is the ability of different replica databases to give the same results at the same request done by the clients, at an exact time and regarding the same logical set of data. [1]

A common requirement when data are replicated is for replication transparency. That is, clients should not normally have to be aware that multiple physical copies of data exist. As far as clients are concerned, data are organized as individual logical objects and they identify only one item in each case when they request an operation to be performed. [1]

3.2 Replicated Database

A replicated database system is composed of many copies of database distributed across different sites. Each database, being called a replica, can work individually to accept client's request. The database replicas work cooperatively as a global database system to provide database services to clients at all sites .[4]The main challenge of database replication is to keep the data copies consistent in the presence of updates. If a client updates a data copy, the update has to be propagated to other copies. If clients connected to different replicas submit updates on the same data items, such updates have to be coordinated to guarantee that the data remains consistent. [4]

3.3 Update Propagation

The basic problem with data replication is that an update to any given logical object must be propagated to all stored copies of that object. A difficulty that arises immediately is that some site holding a copy of the object might be unavailable (because of a site or network failure) at the time of the update. This obvious strategy of propagating updates immediately to all copies is thus probably unacceptable, because it implies that the update-and therefore the transaction-will fail if any one of those copies is currently unavailable. [8]

3.4 Two-Phase Commit Protocol (2PC)

Two-phase commit (2PC) protocol is the most widely accepted commit protocol in distributed DBMS environment that helps in achieving replica synchronization. A 2PC protocol is defined as below: A coordinator is typically the site where the transaction is submitted or any other site which keeps all the global information regarding the distributed transaction. Participants are all other sites where the sub-transaction of the distributed transaction is executing. Following steps are followed in a 2PC:

The coordinator sends *vote-request* to all the participating sites. After receiving the request to vote the site responds by sending its vote, either *yes* or *no*. If the participant voted *yes*, it enters in *prepared* or *ready* state and waits for final decision from the coordinator. If the vote was *no,* the participant can abort its part of the transaction.

The coordinator collects all votes from the participants. If all votes including the coordinator's vote are *yes*, then the coordinator decides to *commit* and sends the message accordingly to all the sites. Even if, one of the votes is *no* the coordinator decides to abort the distributed transaction.

After receiving *commit* or *abort* decision from the coordinator, the participant commits or aborts accordingly from prepared state. The state diagram of 2PC is shown below:[5]

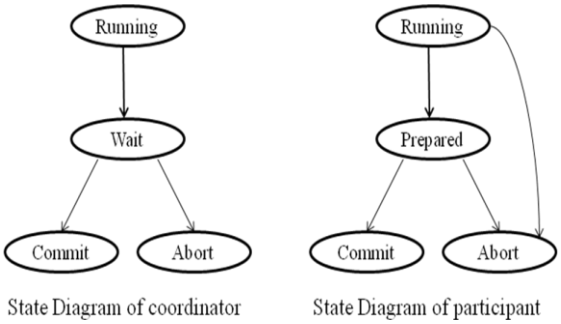

Figure 1.State Diagram of coordinator and participants in 2-phase commit

3.5 Replication Mechanisms

How to replicate data?
There are two basic parameters to select when designing a replication strategy: *where* and *when*. [2]

- Depending on *when* the updates are propagated:
  - Synchronous (eager)
  - Asynchronous (lazy)
- Depending on *where* the updates can take place:
  - Primary Copy (master)
  - Update Everywhere (group)

3.5.1 Synchronous replication

Using eager replication, updates are propagated to the replicas within the boundaries of a transaction, and hence, conflicts are detected before the transaction commits. [7] Eager replication keeps all replicas as part of one atomic transaction. Transaction processing in eager replication has been suited in several protocols such as Two-Phase commit protocol (2PC) for serialization. Therefore, there are no serialization anomalies (inconsistencies). Furthermore, eager system does not need for reconciliation because of locking which detects potential anomalies and converts them to waits or deadlocks. [9]

3.5.2 Asynchronous replication

Lazy replication algorithms that asynchronously propagate replica updates to other nodes after the update transaction commits. [9] Some continuously connected systems use lazy replication to improve response time. Lazy replication also has shortcomings, the most serious being stale data versions. When two transactions read and write data concurrently, one transaction's updates should be serialized after the other. This avoids concurrency anomalies. Lazy replication has a more difficult task because some replica updates have already been committed when the serialization problem is first detected. There is no automatic way to reverse the committed replica update, rather a program or person must reconcile conflicting transaction.[9]

3.5.3 Primary Copy (Master)

Primary copy replication requires to have a specific site--the primary copy—associated with each data item. Any update to the data item must be first sent to the primary copy where it is processed. The primary copy then propagates the update (or its results) to all other sites.[7] With a primary copy approach, there is only one copy which can be updated (the master), all others (secondary copies) are updated reflecting the changes to the master.[2] This approach avoids concurrent updates on the same object at two different nodes. Notice that transactions not running at the primary copy node cannot be sure whether their modifications are valid until they receive the propagated updates from the primary.

3.5.4 Update Everywhere (Group)

Update everywhere scheme allows updates to a data item to be performed anywhere in the system. In this scheme updates can concurrently arrive at two different copies of the same data item (which cannot happen with primary copy). It requires mechanisms to resolve conflicts between concurrent updates to correct inconsistencies. Update everywhere introduces no performance bottlenecks. However, this technique requires that instead of one site doing the work (primary copy) all sites doing the same work.[7] Once a replica accepts a write operation, this write is performed locally and propagated to all other replicas. This update anywhere replication model permits maximum availability since applications can continue to operate even if some replicas are unavailable due to machine failures or network partitions.[6]

3.5.5 Synchronous Master Scheme

An update transaction is first done at master node and then propagated to secondary copies before master transaction commits. In this paper, we use Two-Phase Commit Protocol (2PC) to ensure that both master and slaves install the updates. In this case, master site initiates 2PC protocol and master site is a coordinator site and the slaves are participant ends. The master site waits for the slave sites until the transaction ends. So, it causes long response times. There are no serialization anomalies and there are also no conflicts between servers because there are one server executing update transactions.[2]

3.5.6 Synchronous Group Scheme

Synchronous group scheme allows any site to update the data and then propagated to others before the update transaction commits. The update transaction is locally executed at coordinator site and then propagated to participant sites. If there is two update transactions occur at the same time, conflict can detected before the update transaction commits. So there is no conflict and Lost Update problem because Eager Group scheme uses 2PC protocol.

4. Architecture of the System

For Master architecture, three databases are located at each site and one is Main Store and two sites are Store 1 and Store 2. Product table is replicated at each site. At Main Store, an authorize person can change the product data. But other Store cannot change product data and can access local query. This architecture is illustrated in Figure 2.

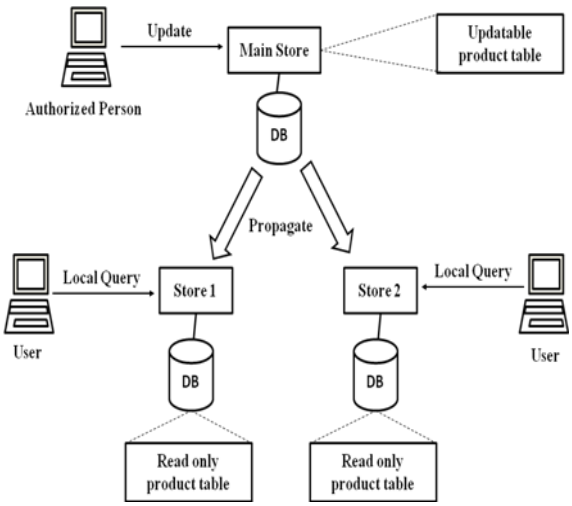

Figure 2. Update propagation with master replicated databases system

For Group architecture, three databases are located at each site and Salesperson table is replicated at each site. In this architecture, every site has an authorized person and can change salesperson data and propagate the updated data to other replica sites. This architecture is illustrated in figure 3.

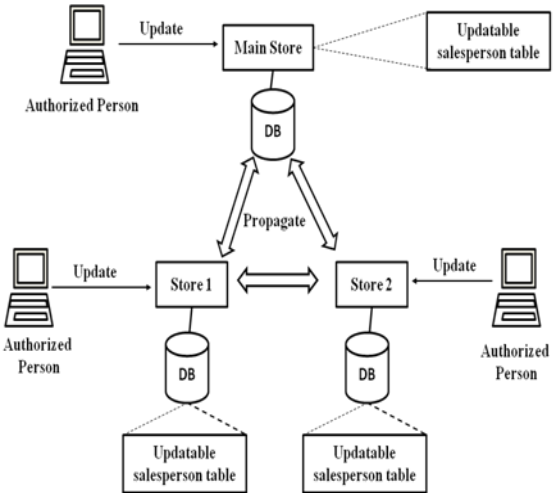

Figure 3. Update propagation with group replicated databases system

5. Process Flow of the system

In Master View, product table is replicated at each site by using synchronous master replicated theory. In Master view, authorized person in main store can update product table. The users in store1 and 2 can read only the product table updated by authorized person in main store. An authorized person can do update, insert and delete transactions for update propagation. When an authorized person updates the product table, an update transaction is first done at main store and then propagated the updated data to the store 1 and store2 before update transaction commits. In this system, we use Two-Phase Commit Protocol (2PC) to ensure that both master and slaves install the updates. The flow of the system is shown in figure 4.

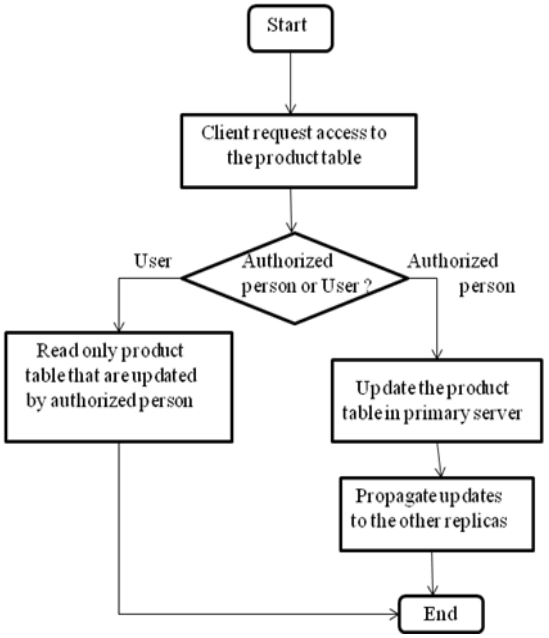

Figure 4. System flow diagram for master view

In Group view, salesperson table is replicated at each site by using synchronous group replicated theory. In Group view, every site has an authorized person and can change data in salesperson table and then propagated the updated data to other sites. The authorized person can do update, insert and delete transactions for update propagation. If the authorized person in Store 1 updates the salesperson table, authorized person in other stores must wait to update the salesperson table because of using 2PC algorithm. So, it resolves conflicts between concurrent updates to correct inconsistencies. Once an authorized person in Store 1 updates the salesperson table, the update transaction is performed locally and propagated the updated data to other stores before update transaction commits. The system flow of the group view is shown in figure 5.

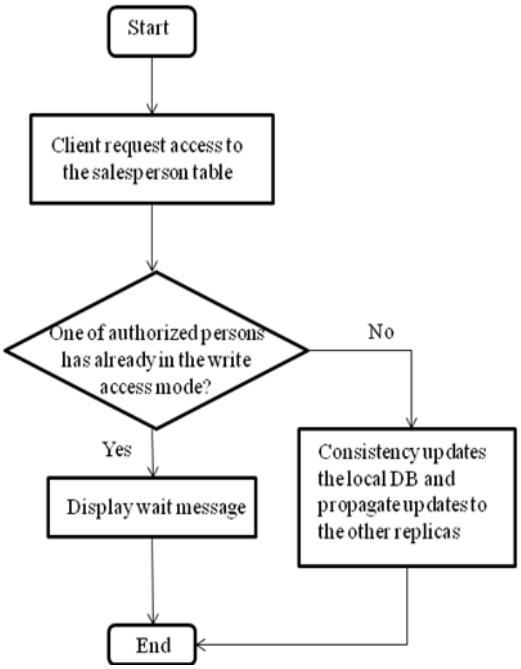

Figure 5. System flow diagram for group view

6. Conclusion

Many distributed database applications need to replicate data to improve data availability and query response time. The two-phase commit protocol guarantees mutual consistency of replicated data but does not provide good performance.[12] Replication of data can offer improved efficiency in a network such as high data availability, increased fault tolerance and load balancing.

In this paper, we discuss update propagation on synchronous replication with two server architecture: master and group. Update propagation with synchronous master scheme need to choose master copy at design time and any updates processes at that master copy. Update propagation with synchronous group scheme can avoid bottleneck problem. When client would like to access the data, the system will produce the required information at minimum time. Clients should not normally have to be aware that multiple physical copies of data exist.

7. References

[1] George Coulouris , Jean Dollimore , Tim Kindberg. " Distributed Systems Concepts and Design".
[2] Gustavo Alonso. IKS. ETH Zurich " Introduction to Database Replication".
[3] Gabriele Bartolini. " Replication in MySQL ".
[4] Yi Lin. " Database Replication in Wide Area Networks ".
[5] Sushant Goel , Rajkumar Buyya. " Data Replication Strategies in Wide Area Distributed Systems ".
[6] Douglas B. Terry , Karin Petersen , Mike J.Spreitzer and Marvin M. Theimer. " The Case for Non-transparent Replication: Examples from Bayou".
[7] Matthias Wiesmann, Fernando Pedone."Database Replication Techniques: a Three Parameter Classification".
[8] Khin Mar Win, Tin Htar Nwe, "Revenue and Expenditure based on Distributed Database in Replication".
[9]Jim Gray, Pat Helland, Patrick O'Neil. "The Dangers of Replication and a Solution".
[10] Yair Amir, Jonathan Stanton, Claudiu Danilov." On the Performance of Wide-Area Synchronous Database Replication".
[11] Maytarhlamyint." Implementation of consistent replication system".
[12] Esther Pacitti, Eric Simon." Update propagation strategies to improve freshness in lazy master replicated databases"