COMPARISON OF LATENCY TIME FOR TRANSACTIONAL REPLICATION IN P2P SYSTEM

Poe Ei Tun, The` The` Soe, Cynthia Myint, Nyein Nyein Ei, May Mar Oo, Lai Lai Yee, Aye Thida
University of Computer Studies, Mandalay
poeitun@gmail.com, myintmo08@gmail.com, cynthiamyint@gmail.com, nyeinnyeinei@gmail.com, maymaroo@gmaiil.com, lai2yee@gmail.com, ayethida.royal@gmail.com

ABSTRACT

The term replication in a distributed database refers to the operation of copying and maintaining database objects in more than one location. There are three types of replication in distributed system that are snapshot, transactional and merge. In this paper, we first propose transactional replication in Peer-to-Peer (P2P) systems and compute the latency time depends on file size when transferring data from one machine to another. Latency time means the time between initiating a request in the computer and receiving the answer. It is a measurement when replicating data. In this paper, we also compare the latency time for update, delete and insert of only transactional replication. According to the experimental results, the latency time might be greater as much as increasing the file size. But the average latencies of all operations are nearly the same state. Here, we ignore network connection. In the future, we intend to compare the latency time on three types of replication.

Key Words: replication, distributed database, P2P system, transactional replication, latency time.

1. INTRODUCTION

P2P approaches must address the two big challenges of maintaining data continuity in the face of nodes frequently joining and leaving, and keeping replicas consistent when updates to the data occur. Work must be done in order to take these ideas into the implementation phase. In this paper, we compute the latency time for transactional replication in P2P systems and then compare the latency time for update, delete and insert depends number of data records. Database replication is becoming more important role for database applications. Replicated data are becoming more and more of interest lately. The use of data replication has many advantages including the increased read availability and reliability but makes the data updating more complicated. This replication can be implemented between databases on the P2P systems [1].

Replication is a cost effective way to increase availability and used for both performance and fault tolerant purposes thereby introducing a constant tradeoff between consistency and efficiency. Replication is the most providing way for traveling salespeople and roaming disconnected users and enables mobile users with laptops to be updated with current database information when they connect and to upload data to a central server. Data is generated and then replicated.

The rest of this paper is organized as follows. Section 2 describes about the related work of this system. Section 3 introduces the three types of replication and about the details of transactional replication. Section 4 shows synchronization in this replication type. Section 5 evaluates the latency time. Section 6 presents the overview of our proposed system. Section 7 shows the experimental results of this system and the last section 8 concludes this paper.

2. RELATED WORK

Survey of data replication in P2P systems [1] presented overview of data replication. They focused on the optimistic approach that provides good properties for dynamic environments. They

also introduced P2P systems and the replication solutions they implemented. In particular, this paper showed that current P2P systems do not provide eventual consistency among replicas in the presence of updates apart from Atlas Peer–to–Peer Architecture (APPA) system, a P2P data management system have been built.

Transactional replication performance tuning and optimization [2] examined performance in transactional replications and based on the results of tests conducted using a variety of hardware configurations and replication environments. This paper provided recommendations in areas such as applying the initial snapshot, optimizing replication settings, and replication scalability. Some models of a distributed database management system with data replication [3] suggested using General Purpose System Simulation (GPSS) for simulation modeling.

Queuing systems models modeling the execution of two-phase locking (2PL) in Distributed Database (DDB) with Data replication are suggested: centralized 2PL, primary copy 2PL, distributed 2PL and voting 2PL. Results from the simulation of the distributed 2PL are presented. Open issues for effective dynamic replication in Wide-Area Network environments [4] issued such as node heterogeneity (in terms of processing capacity and available disk space for storing replicas), significant variations in bandwidth, lack of centralized control, lack of global knowledge, distributive ownership and scalability make replication in WAN environments significantly more challenging than in the case of traditional domains. Interestingly, they were fundamental issues which arise for replication in different types of WAN environments.

This paper specifically focused on replication in two representative WAN environments, namely P2P systems and GRIDs, and discussed open research issues concerning replication in these two environments as well as our perspectives on these issues.

3. REPLICATION TYPES

This section describes the basic concepts and three types of replication. Replication is one of the oldest and most important topics in the overall area of distributed systems. It is a process of copying/moving data between databases on the same or different servers.

3.1. Basic Concepts of Replication Types

Replication uses the following three servers [5].

(1) Publisher
(2) Distributor
(3) Subscriber

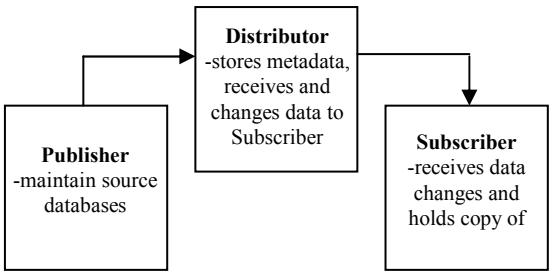

Figure 1. Replication process

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 sites holding a copy of the object might be unavailable (because of a site or network failure) at the time of the update. The 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. In a sense, in fact, data is less available under this strategy than it would be in the nonreplicated case.

3.2. Three Types of Replication

There are three types of replication in distributed database as follows:

(1) Snapshot Replication
(2) Transactional Replication
(3) Merge Replication

3.2.1 Snapshot Replication

Snapshot replication [6] is the simplest form of replication and it simply takes a "snapshot" of the data on one server and moves that data to another server (or another database on the same server). It distributes data exactly as it appears at a specific moment in time and does not monitor for updates to the data. It is the best used method of replicating data that change infrequently or where the most up-to-date values are not requirement.

When synchronization occurs, the entire snapshot is generated and sent to the subscribers. After the initial synchronization snapshot, replication can refresh data in published tables periodically—based on the schedule you specify. Although snapshot replication is the easiest type to set up and maintain, it requires copying all data each time a table is refreshed.

3.2.2 Transactional Replication

Transactional replication [6] involves copying data from the publisher to the subscriber(s) once and then delivering transactions to the subscriber(s) as they occur on the publisher. It is also known as dynamic replication and typically used in server-to-server environments. The application requires low latency between the time changes are made at the publisher and the changes arrive at the subscriber. An initial snapshot of data is applied to subscriber, and when data modifications are made at the publisher, the individual transactions are captured and propagated to subscriber. By default, subscribers to transactional publication should be treated as read only, because changes are not propagated back to the publisher.

3.2.3 Merge Replication

Merge replication [6] is the process of distributing data from publisher to subscriber, allowing the publisher and subscriber to make update data while they are connected or disconnected, and then merging the updates between sites when they are connected. It is typically used in server-to-client environments and appropriate where multiple subscribers might update the same data at various times and propagate those changes to the publisher and to other subscribers that need to receive data, make changes offline, and later synchronize changes with the publisher and other subscribers, each subscriber requires a different partition of data, conflicts might occur and, when they do, the users need the ability to detect and resolve them.

4. SYNCHRONIZATION

Synchronization refers to the propagation of data changes between subscriber and publisher. How the data is synchronized dependent on the type of replication used. In cased of snapshot replication, a snapshot file is replicated at the subscriber. In case of transactional replication all data modification through Insert/Update and Delete are distributed between publisher and subscriber. In case of merge replication, data modifications at various servers are merged. Conflicts, if any, are deleted and resolved. In synchronous replication, sending and receiving processes synchronize at every message.

Whenever a received is issued the process blocks until a message arrives. The available a synchronous replication has the greatest potential excessive latency during primary or secondary I/O will tolerate directly into longer I/O elapse time for application. The synchronous update propagation approaches apply the changes to all replicas within the context of transaction.

Synchronous replication has been around for a long time in the high-end, mainframe-class storage devices and more recently in the higher-end open-systems storage devices. Synchronous replication is where every write from the application is sent to the local disk system, which sends it to the remote storage system. The write is not acknowledged that the write is complete. In other words, the application has to wait for the written data to be written to the local and remote storage before it can continue processing [7].

5. LATENCY TIME

This section describes latency time in replication. In a computer system, latency is often used to mean any delay or waiting that increases real or perceived response time beyond the response time desired. Latency is the delay that occurs after a send operation is executed before data starts to arrive at the destination computer [8]. Specific

contributors to computer latency include mismatches in data speed between the microprocessor and input/output devices and inadequate data buffers. Within a computer, latency can be removed or "hidden" by such techniques as perfecting (anticipating the need for data input requests) and multithreading, or using parallelism across multiple execution threads.

The latency assumption seems to be that data should be transmitted instantly between one point and another (that is, with no delay at all). The contributors to network latency include:

(1) Propagation: This is simply the time it takes for a packet to travel between one place and another at the speed of light.

(2) Transmission: The medium itself (whether optical fiber, wireless, or some other) introduces some delay. The size of the packet introduces delay in a round trip since a larger packet will take longer to receive and return than a short one.

(3) Router and other processing: Each gateway node takes time to examine and possibly change the header in a packet.

(4) Other computer and storage delays: Within networks at each end of the journey, a packet may be subject to storage and hard disk access delays at intermediate devices such as switches and bridges.

In our proposed system, we can compute the latency time by the following equation:

$$LT = R_{bt} - S_{at} \quad (1)$$

where,

LT = Latency time between different machines

R_{bt} = Receiving time at destination point

S_{at} = Sending time at source point

As mention, we compute and compare the latency time for the three types of operations on transactional replication.

6. OVERVIEW OF PROPOSED SYSTEM

In this section, we present the overview of the proposed system. Before describing the proposed system, there are some assumptions. This system is only for homogeneous databases. Different types of representative environments for replication are LAN, P2P systems, Grids and so on. Among them, this proposed system uses P2P system. P2P system provides a decentralized approach to solve classical problems in replication and caching. They offer many advantages over traditional centralized approaches: e.g., organic scaling, no costly infrastructure, and fault tolerance. P2P systems need to deal with data location, data integration, data querying, and data consistency issues. Replication in P2P system is a key effectiveness of distributed system in that can provide enhanced performance, high availability and fault tolerance. It can provide all transactions such as insert, update and delete with data synchronization. Synchronization refers to the propagation of data changes between publisher and subscriber. Therefore, synchronization is the crucial role for replication. There are three types of replication such as snapshot, transactional and merge.

Among them, this system proposed the transactional replication. Transactional replication allows data modifications to be propagated incrementally between different locations in a distributed environment. It is the most efficient and flexible synchronization model in terms of response time and customizability. We choose the transactional replication because it can transfer only the changed data (instead of sending the completed data every time) to the subscriber with minimal latency time than others (such as snapshot and merge replication). In this transactional replication, data alteration and custom processing can be easily implemented using custom replication stored procedures, while transferring the data.

Figure 2 shows overall process of transactional replication including the computing and comparing of latency time for data records. The publisher can update three transactions and access the replication request. At that time, the calculation of latency time is performed. Finally, the updated replication is established and propagated to the subscriber. This

system can support the data integrity and consistency in distributed system because of data synchronization.

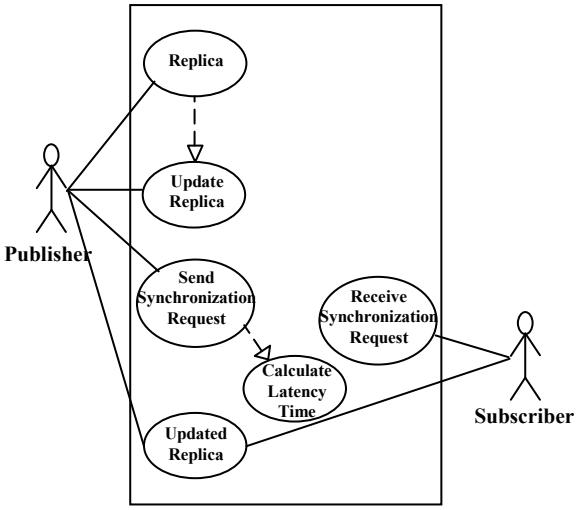

Figure 2. Use case diagram of the system

7. EXPERIMENTAL RESULTS

In this section, we present the latency time of insert, update and delete operations in transactional replication and compare these results.

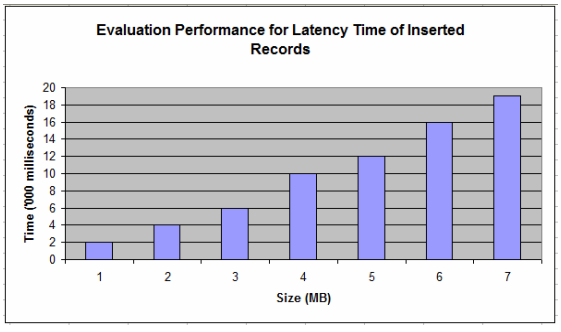

Figure 3. Latency time for inserted records

Figure 3 shows the latency time for inserted records. The horizontal axis represented file size in megabytes. The vertical axis represented latency time in milliseconds when transferring data from one machine to another. Latency time is absolutely depending on the file size. The bigger the file sizes, the higher the latency time.

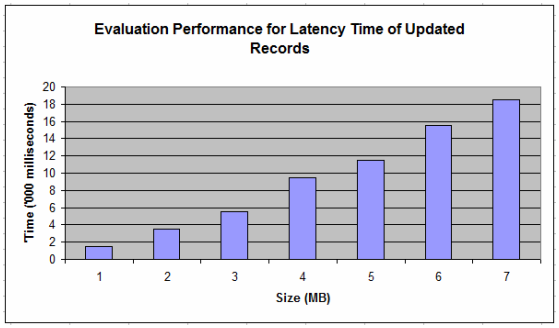

Figure 4. Latency time for updated records

The evaluation performance of updated records depending to their file size and the latency time is described in Figure 4. According to the above experimental results, the latency time is gradually increased when 1MB of file size is reached 1,500 milliseconds to 7 MB of file size is reached 18,500 milliseconds. The more the file size, the more the latency time is.

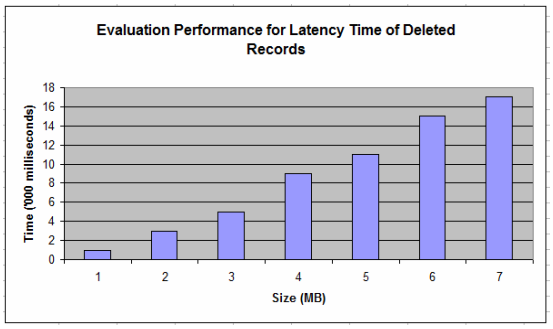

Figure 5. Latency time for deleted records

Similarly, the horizontal axis represented measurement of file size in megabytes as shown in Figure 5. The vertical axis represented latency time when transferring data from one machine to another. Latency time is absolutely depending on the file size. The bigger the file sizes, the higher the latency time.

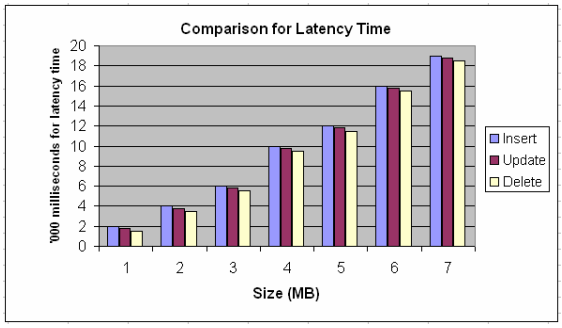

Figure 6. Comparison of latency time between insert, update and delete operations

Figure 6 expresses the overall comparison for three operations (such as insert, update and delete) of evaluation performance on latency time. As mentioned above, the latency time of three operations are slightly difference. When transferring file size of 1MB, Insert operation takes 2,000 milliseconds, Update operation takes 1,500 milliseconds and Delete operation takes 1,000 milliseconds. Although we can see the more file size, the more latency time, it is not clearly difference latencies among them. Therefore, we can show the overall average time is nearly the same according to our experimental results.

8. CONCLUSION

In this paper, we present the transactional replication with latency time. The proposed system computes latency time of update, delete and insert operations depend on the file sizes. The bigger the file sizes, the higher the latency time. According to the experimental results the average file sizes of three transactions are nearly the same. There is a little difference in latency when transferring data from one machine to another. So, this proposed system can transfer data between publisher and subscriber with minimal latency. The proposed system is suitable for distributed system when distributing data in a replicated fashion across the machine on the network with optimal results. It also improves the performance, availability and reliability for distributed system. It can supplement disaster-recovery plans by duplicating the data from a local database server to remote database server. Furthermore, we should also consider concurrency control (such as locking, time stamping) when replicating data to multiple database servers. In the future, we intend to compare the performance evaluation for three types of replication.

REFERENCES

[1] V. Martins, E. Pacitti and P. Valduriez, *Survey of data replication in P2P systems*, 2006.

[2] B. Newman, X. Schildwachter and G. Yvkoff, *Transactional replication performance tuning and optimization*, 2001.

[3] S. Vasileva, P. Milev and B. Stoyanov, *Some models of a distributed database management system with data replication*.

[4] A. Mondal and M. Kitsuregawa, *Open issues for effective dynamic replication in Wide-Area Network environments*.

[5] M. Ahamad, M. Ammar and S. Cheung, *Replicated data management in distributed systems: In reading in distributed computing systems*, Eds. T. Casavant & M. Singhal, IEEE CS Press, 1994.

[6] http://en.wikipedia.org/wiki/Distributed_data-base

[7] D.M. Sutton, *How to implement synchronization on a replicated database*, Content Master Ltd.

[8] G. Coulouris, J. Dollimore and T. Kindberg, *Distributed systems concepts and design*, Third Edition.