Data Replication in Aircraft Components Database System using Distributed Database System

Nann Thin Thin Nwe
University of Computer Studies, Yangon
nannthin88@gmail.com

Abstract

Data replication is a key technology in distributed systems that enable higher availability and performance. Data replication consists of maintaining multiple copies of data, called replicas, on separate computers. Optimistic replication algorithms allow replica contents to become stale but in a controlled way. Therefore, replication becomes far more efficient and available than traditional replication algorithms that keep all the replicas consistent, especially when the network and computers are unreliable. This system presents optimistic replication for distributed aircraft parts database by using Push based approach. This system is not only improving the data availability but also reduce the network latency at remote site by replication process.

1. Introduction

Data replication consists of maintaining multiple copies of data, called replicas, on separate computers. It is an important enabling technology for distributed services. Replication improves availability by allowing access to the data even when some of the replicas are unavailable. It also improves performance through reduced latency, by letting users access nearby replicas and avoiding remote network access, and through increased throughput, by letting multiple computers serve the data simultaneously. In an interactive distributed client-server system, each client views, updates, and/or tracks changes made to a set of objects.

One widely used approach dealing with server overload is the creation of replicas of the existing object and the server. Conventional data management systems are pull-based: queries or transactions are executed only when they are explicitly requested by a user or an application program; i.e., transfer of data from servers to clients is initiated by an explicit client pull. However in a push-based system, updates are sent to clients not when they request it. In an environment where multiple users are interacting within a virtual world, an update to the spatial location of an object has to be transmitted to all the clients. In such a system, any update to any object has to be sent to all other clients interested in this particular object.

The organization of this paper is as follows: Section 2 presents the related work of the system. Theory background used in this system is described in section 3 and section 4 illustrates about Push-based algorithm. Section 5 discusses the proposed system and Section 6 is the implementation of the system. In section 7, conclusion of the system is presented.

2. Related Work

In the distributed systems, it is advantageous to place content as close to users as possible in order to remove sources of network delay. There have been several papers about data replication in traditional database systems. In the pull-based approach, pulling every time will incur large amount of network traffic and vastly increase server load [5]. In addition, the client response time includes an additional round-trip delay even when the local copy is the same as the server's copy. Replication can be quite effective at reducing network bandwidth consumption as well as server load. The value of replication is greatly reduced, however, if replicas are not updated when the original data change. Data consistency mechanisms ensure that replicated copies of data are eventually updated to reflect changes to the original data [7].

The optimal consistency protocol is precise expiration. In precise expiration, servers set the expiration time of each object to be the next modification time. Consistency protocols other than precise expiration use two mechanisms to meet consistency guarantees. First is client initiated approach, pulling (polling) and the other is server initiated approach, pushing. In the polling approach, client associates a time to live (TTL) or an expiration time with each replicated object [8]. This TTL can be regarded as a per-object lease to read the object; in particular, it places an upper bound on the time that each object may be cached before the client revalidates the cached version. To revalidate an object whose expiration time has passed, a client sends a Get-if-modified-since request to the server, and the server replies with “304 not modified” if the cached version is still valid or with “200 OK” and the new version if the object has changed. The HTTP polling protocol has several limitations. Each object is associated with an individual TTL. After a set of TTLs expire, each object has to be revalidated individually with the server to renew its TTL, thereby increasing server load and read latency [9].

3. Theory Background

3.1. Distributed Database

A distributed database is a database that is under the control of a central database management system (DBMS) in which storage devices are not all attached to a common CPU. It may be stored in multiple computers located in the same physical location, or may be dispersed over a network of interconnected computers. Collections of data (e.g., in a database) can be distributed across multiple physical locations. A distributed database can reside on network servers on the Internet, on corporate intranets or extranets, or on other company networks. Replication and distribution of databases improve database performance at end-user worksites. [2]

Replication involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. The replication process can be very complex and time consuming depending on the size and number of the distributive databases. This process can also require a lot of time and computer resources.

3.2. Replication

Replication is the process of sharing information so as to ensure consistency between redundant resources, to improve reliability, fault-tolerance, or accessibility. It could be data replication if the same data is stored on multiple storage devices, or computation replication if the same computing task is executed many times. The access to a replicated entity is typically uniform with access to a single, non-replicated entity. The replication itself should be transparent to an external user.

3.3. Consistency

Consistency can be defined as maintaining the same state of data in all replicas. It is the most critical portion in the replication systems. There are various approaches for maintaining consistency. Consistency algorithms mainly based on who does the replication and when the replication is performed. And there is two main groups of consistency algorithms, pessimistic and optimistic. Pessimistic algorithm ensures the strong consistency, but the implementation costs very expensive and can fail because of locking. Optimistic algorithm is a little bit weak in consistency but it is cost effective and guarantee the success of transactions. In the optimistic approach, it can be further classified into –

- Client Initiated Approach
- Server Initiated Approach

3.4. Replication Algorithms

Replication algorithm is at the core of any replicated service, responsible for reading and updating replicas, or physical copies of an object. An important design issue in replication is how replicas are presented to users. Traditional pessimistic replication algorithms offer single-copy semantics, that is, they give users an illusion of having only a single, highly available copy of an object by keeping the replicas identical all the time. They are called “pessimistic” algorithms, because they prohibit accesses to a replica unless the replica contents are provably up to date. Although these algorithms are essential in a class of applications, such as banking, that must give correct answers all the time at all cost, they have one major drawback: stringent hardware requirements. [1]

Optimistic replication algorithms allow data presented to users to become stale but in a controlled way. A key feature that separates optimistic replication algorithms from pessimistic counterparts is the way updates to objects are handled: whereas pessimistic algorithms update all the replicas at once and possibly block read requests from users during the update application, optimistic algorithms propagate updates in the background and allow any replica to be read directly most of the time. This feature makes optimistic algorithms more available and more efficient using unreliable network media and inexpensive computers.

3.5. Optimistic Replication

Optimistic replication is a group of techniques for sharing data efficiently in distributed database system. The key feature that separates optimistic replication algorithms from their pessimistic counterparts is their approach to concurrency control. Pessimistic algorithms synchronously coordinate replicas during accesses and block the other users during an update. In contrast, optimistic algorithms let data be read or written without a priori synchronization, based on the "optimistic" assumption that problems will occur only rarely, if at all. Updates are propagated in the background, and occasional conflicts are fixed after they happen.

4. Push based Algorithm

Whenever a master copy (for example, of data A) gets updated at replica 1, an invalidation message M is constructed and flooded into the network to other replicas. Message M contains at least the following information: the origin server ID (e.g., IP address), the data record, and the new version number of that data record. A replica receives M will check if it caches data A, if yes, it compares the local version number of A with that in the message, and invalidates its local copy if the message contains a newer version number. It passes on message M to its neighbors no matter if it contains the specified file, just like the way it deals with query. Process sequence of push-based algorithm is shown in Figure 1.

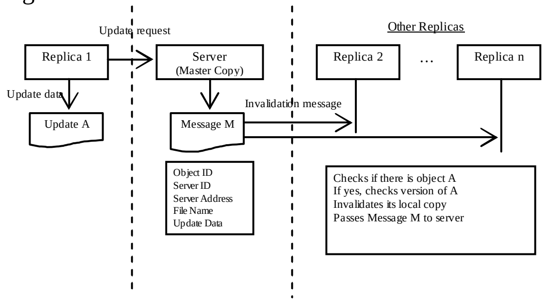

Figure 1: Process sequence of Push-based algorithm

5. Proposed System

This system presents the replication process to improve data availability in Aircraft Components Database System by implementing distributed database system in different physical location. It maintains the consistency of the distributed database by replication process. Push based approach of optimistic replication has been used to replicate data. The process of push based has been as follows: Whenever there has been an update in one replica, the update has been sent to the server. Server issues acknowledgement of the update to other replicas and wait for the replies. If all other replicas agree the update, the update has been committed (permanently saved), otherwise rollback (cancel the update).

Proposed system design is shown in Figure 2, where there is one Master database and one or more replicas in remote areas. When one of the objects is updated or added or deleted, the update information is sent to server and server sends update information to other replicas.

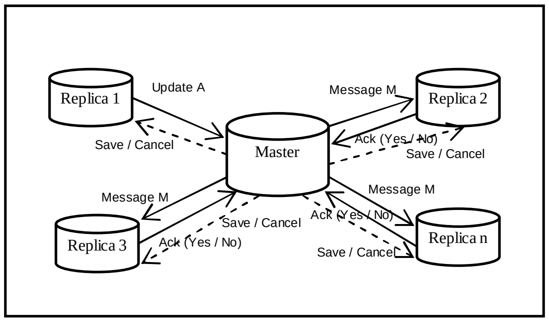

Figure 2: Proposed System Design

In this system, whenever, update request (update, insert, delete) for an object, following process will be performed. Replica (where update is performed) updates local database temporarily. It then sends the update information to Server (Master). Server sends this information to other replica for approval. Other replicas send back "OK" if they agree to update, or "Cancel" if they do not agree. If all other replicas send "OK", Server sends "COMMIT" to all replicas to permanently write in the database and if at least one of them sends "Cancel", the update transaction is Cancelled. This process is shown in following Figure 3.

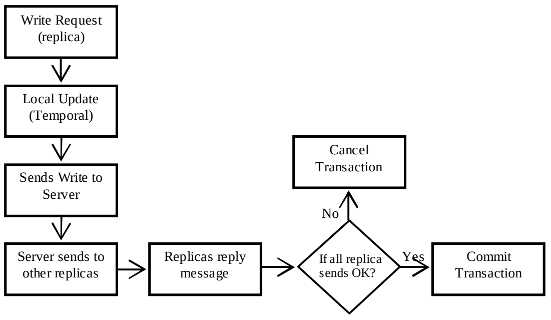

Figure 3: Process Flow of the System

Sequence diagram for updating information in one replica is shown in Figure 4.

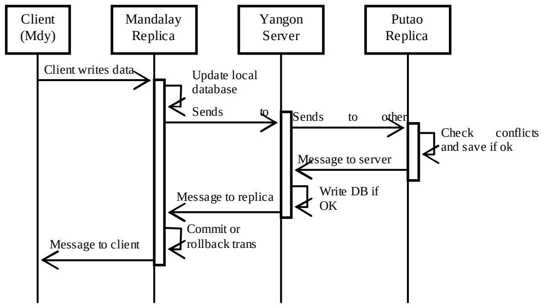

Figure 4: Sequence Diagram for Updating data

6. System Implementation

This system is implemented using Java programming language and jdk 1.6 is used to implement the system. In this system, there is one server component and three client components are implemented. Yangon server is used as server database, and three other components (Putao, Mandalay and Myitkyinar) are used as client machines. Even though aircraft database seems small, there may be large volume of data in Parts. Therefore it is necessary to replicate aircraft data for fast retrieval. This system is implemented as simulation process and tested in the local area network system.

Each database has two types of database, database that stores actual data and other database that stores metadata (data about data). There will be one or more clients in each site (both server and replicas) and client access data at each site. Client can read or write data in each site and all clients in all sites will see the same data (in the consistent state). The implementation of distributed database structure is shown in Figure 5.

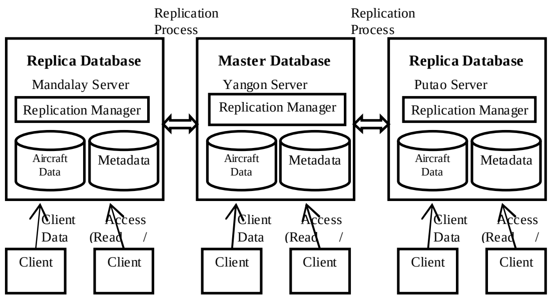

Figure 5: Implementation of Distributed Database

Table 1 shows the processing times for local and remote (from server) data retrievals. According to experimental results, processing time for requesting data from local is far faster than requesting from server. Therefore, replication assures the fast access of data from local copy.

Table 1: Retrieval Times for different data sizes

No.	No. of records	Local (sec)	Server (sec)
1.	100	1.2	2.7
2	200	2.5	3.7
3	300	3.1	4.5
4	400	3.7	5.7

7. Conclusion

This system presents the database replication process in distributed database. Replication improves the data availability and fault-tolerance. It also reduces the network latency. This system is implemented by using optimistic replication which is used to avoid dead-lock problem that mostly found in pessimistic replications and improve the system performance. It also reduces the network costs. Push based algorithm of optimistic replication is the server initiated process, whenever there is an update, server replicates it to all other replicas in the optimistic way. According to the above facts, optimistic replication is an appealing technique; it improves networking flexibility and scalability.

References

[1] S. Acharya, M. J. Franklin and S. B. Zdonik, “Balancing Push and Pull for Data Broadcast”, Procs. of the ACM SIGMOD, May 1997.

[2] C. Bell, and R. Nishtala, "Firehose: An Algorithm for Distributed Page Registration on Clusters of SMPs", CS262B Final Project Computer Science Division, University of California, Berkeley, May 2004.

[3] V. Cate, “Global File System”. Proceedings of the 1992 USENIX File System Workshop, May 1992, pages 1-12.

[4] S. E. Chu, Kim, N. Jae and D. W. Kang, "RMI Object Consistency Maintenance Techniques at Distributed Computing".

[5] P. Deolasee, A. Katkar, K. Ramamritham and P. Shenoy,“Adaptive Push-Pull: Disseminating Dynamic Web Data: Complete Report, code, and traces”, http://www.cse.iitb.ernet.in/˜krithi/dynamic.html.

[6] V. Duvvuri, P. Shenoy and R. Tewari, “Adaptive Leases: A Strong Consistency Mechanism for the World Wide Web”, InfoCom, March 2000.

[7] C. Gray, and D. Cheriton, “Leases: An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency”, Proceedings of the Twelfth ACM Symposium on Operating System Principles, 1989, pages 202-210.

[8] T. M. Kroeger, D. D. E. Long, and J. C. Mogul. Exploring the Bounds of Web Latency Reduction from Caching and Prefetching. Proc. of USITS'97, 13-22, 1997.

[9] J. Lan, "Cache Consistency Techniques for Peer-to-Peer File Sharing Networks", MASTERS PROJECT REPORT, June 26, 2002.