A Dynamic Replication for Periodic Transactions in Weather Forecast Data Distribution by ORDER-RS

Ei Ei Khaing, Daw Khaing
University of Computer Studies, Yangon
maubineiei@gmail.com, khaingaugust@gmail.com

Abstract

Many real-time applications need data services in distributed environments. Replication can help distributed real-time database systems meet the stringent time requirements of application transactions. In this system, we present a dynamic replication control algorithm, On-demand Real-time Decentralized Replication with Replica Sharing (ORDER-RS), designed for distributed real- time database systems. In the propose system, the weather forecast data replicas are dynamically created upon the requests by the incoming transactions and their update frequencies are determined by the data freshness requirements of these transactions.

Keywords: *Replication, ORDER-RS, real-time database system*

1. Introduction

Many real-time applications need to share data that are distributed among multiple sites. In a distributed real-time database system, remote data access consists of multi-hop network operations and takes substantially more time than the local data accesses. This potentially leads to large number of transaction deadline misses. Another problem is that due to the long remote data access time, by the time a transaction gets all the data it needs, some of its data items may have already become stale.

Replication is an effective method to solve mentioned problems. By replicating temporal data items, instead of initiating remote data access requests, transactions that need to read remote data can now access the locally available replicas. This helps transactions meet their time and data freshness requirements.

The ORDER-RS algorithm is designed to work in a environment where all data types and relations in the system are known a priori and transactions are short-term periodic transactions. When a transaction arrives, it declares its period, data needs, execution time and deadline. There are multiple ways to handle the replication control. Different replication schemes are better suited for different data workloads and database specifications. Several methods of replication control in medium-scale or large-scale distributed real-time database systems and present a replication algorithm called On-demand Real-time Decentralized Replication with replica sharing (ORDER- RS). With this information, the algorithm decides where and how often the replicas are updated.

In term of scalability, this algorithm can be enhanced to a replication algorithm called On-demand Real-time Decentralized Replication with Replica Sharing (ORDER-RS). With this algorithm, large distributed systems are divided into small groups called cliques based on the network topology. The replicas within one clique are shared by the clique members.

2. Related Work

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 [1].

"Update Propagation on Lazy and Eager Replication With Group and Master Replication Databases", this system implement Lazy and Eager Replication on two server architecture, Group Architecture and Maser Architecture, with Sales Order Processing System. Master Architecture has three database that one is Master Office site and another two is Branch Offices. Product Price is replicate at each site and Update that price is done by

Master's authorizes persons but Branch office cannot change or update that price. Group Architecture also has three database same as Maser. That architecture can not only Master site but also Branch site change the Product Price.

3. Background Theory

Distributed System: The purpose of the distributed system is to coordinate the use of shared resources or provide communication services to the users.

Typical properties of distributed systems include the following:

- The system has to tolerate failures in individual computers.
- The structure of the system (network topology, network latency, number of computers) is not known in advance, the system may consist of different kinds of computers and network links, and the system may change during the execution of a distributed program.
- Each computer has only a limited, incomplete view of the system. Each computer may know only one part of the input.

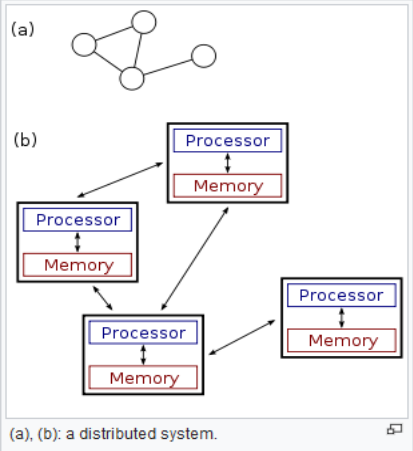

Figure (a) is a schematic view of a typical distributed system; as usual, the system is represented as a network topology in which each node is a computer and each line connecting the nodes is a communication link.

Figure (b) shows the same distributed system in more detail: each computer has its own local memory, and information can be exchanged only by passing messages from one node to another by using the available communication links.

3.1. Replications Algorithms

Replication algorithms can be divided into full replication algorithms and partial replication algorithms. Full replication is a replication strategy that replicates all data to all sites in the distributed system. The benefit of full replication is that all data are available to read locally. However, full replication may not be efficient in middle-scale or large-scale distributed systems in which tens or hundreds of databases maintain thousands of data items.

In those systems, the cost of maintaining replicas for all data, especially sensor data, is too high. Instead, partial replication is better suited for middle-scale or large-scale systems. With partial replication, only a portion of data items in the database are replicated and if a data item is replicated, it does not need to be replicated to all sites.

Replication algorithms can also be divided into static replication algorithms and dynamic replication algorithms. When a replication is static, the location and the number of the replicas are fixed. When a replication algorithm is dynamic, the location and the number of the replicas can change dynamically according to the transaction data needs and system conditions. Different replication schemes are better suited for different data workloads and database specifications.

3.2. Transaction Model and Scheduling

The transactions in the system are divided into two types, system update transactions and user transactions. System update transactions include temporal data (sensor data) update transactions and replica update transactions. User transactions are queries or updates from applications. Incoming application transactions get scheduled after temporal data update and replica update transactions. Transactions are represented as a sequence of operations on the data objects. Operations of one transaction are executed in sequential fashion. One operation cannot be executed unless all previous operations are finished. Once all operations of one transaction are finished, the transaction enters validation stage. At the validation stage, the system

checks the freshness of the accessed data. If the accessed data items are not fresh anymore, the transaction is restarted. After the validation stage, the transaction enters the commit stage which writes log and commits.

3.3. Dynamic Replication

First, in large-scale distributed systems, it might be very costly or impossible for every site in the system to maintain detailed information about all data items in the system. Second, sometimes it may not be efficient to fetch all fresh data items directly from their primary site. Instead, the site can get fresh copies from some existing active replicas that are closer, i.e., the replicas that could be reached with less transmission delays.

An example is given in Fig. 1. In the figure, two real-time databases DB1 and DB2 are connected by high speed LAN. DB1 has an active replica for a remote temporal data item.

The active replica is updated periodically using the fresh data values from the primary site. Now, DB2 admits a new transaction which needs the same remote data. As, it is not efficient for DB2 to fetch the data value from remote site again since there is already an active replica in DB1, which can be easily reached. Fetching fresh data directly from the primary site unnecessarily increases the workload of the network and the primary site.

Instead, it is more desirable to fetch the data from DB1. If the update frequency of the active replica at DB1 can satisfy the requirements of the new transaction in DB2, the system only needs to replicate data from active replica 1 to active replica 2. If the new transaction demands higher update frequency, the system can now stop replicating data from primary copy to active replica 1. Instead, it replicates data directly from the primary copy to replica 2 (at higher update frequency) and then replicates the data from replica 2 to replica 1.

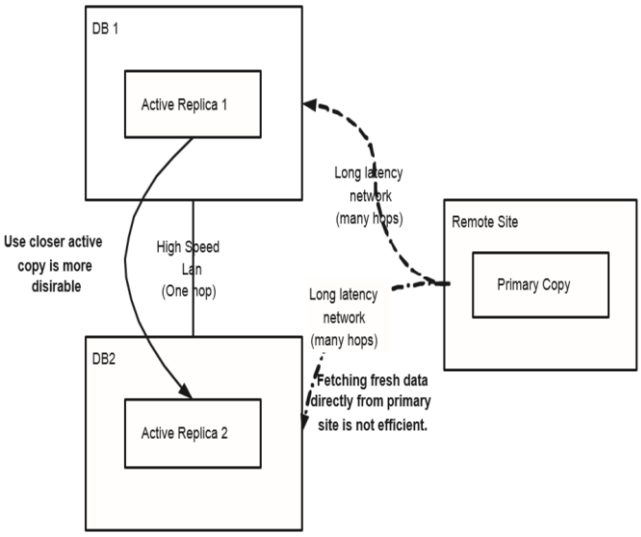

Figure 2. Fetching Fresh Data from Closer Active Replicas

4. ORDER_RS (On-demand Real-time Decentralized Replication with replica sharing)

Our algorithm can be enhanced to a replication algorithm called On-demand Real-time Decentralized Replication with Replica Sharing (ORDER-RS). The ORDER-RS algorithms is partial replication for middle scale or large sale distributed system. The algorithm is designed to work in an environment where all data types and relations in the system are known a priori and transactions are short-term periodic transactions. The algorithm decides where and how often the replicas are updated. When a transaction arrives, it declares its period, data needs, execution time and deadline. There are multiple ways to handle the replication control. With this algorithm, large distributed systems are divided into small groups called cliques based on the network topology. The replicas within one clique are shared by the clique members.

To use the idea described above, we extend our algorithm with a mechanism that shares active replicas among those sites that are close (in transmission latency) to each other.

An example is shown in Figure 3. In the figure, there is a medium-scale distributed real-time database system which consists of 24 real-time database servers. The real-time database servers are connected by 8 transmission stations. The system are divided into 6 cliques based on the network topology. While the cliques are connected by multiple-hop wireless

networks. Assume that the clique members from the same clique know the existence of each other.

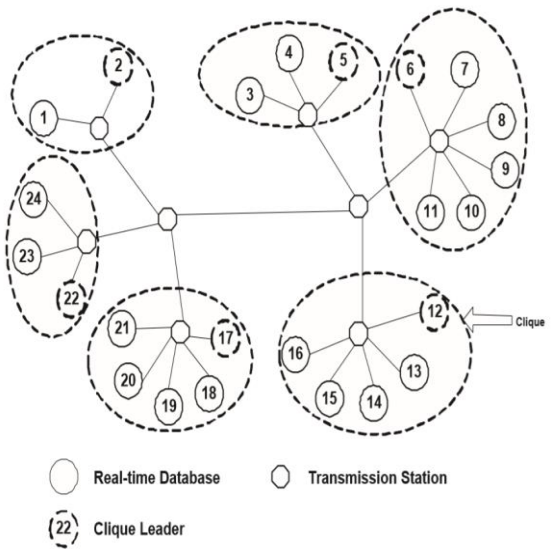

Figure 3. Replica Sharing in Large Distributed Systems

4.1. Objective of the Proposed System

The objectives of the proposed system are:

- To implement a data reliable System. (Consistency and data freshness)
- To reduce the data response time to remote site.
- To understand how consistency is important in database systems with multiple users.
- To implement a system in which all objects remain in a consistent state when they are accessed by multiple transactions.

4.2. ORDER-RS in Dynamic Replication Algorithm

Let PS = Primary Server;
CL = Clique Leader;
CM = Clique Member;
Active replica is updated periodically using the fresh data values from the primary site.

BEGIN
Accept the request from the CM;
CL ← CM's Request;
CL checks the requested data is satisfy in its local or not.

If (CL's has satisfy data)
{
CL checks the updated frequency.
If (the update frequency of local CL can satisfy the requirements of the CM's request)
{
The system only replicate data from local CL to requested local CM.
}
Else If (the requested CM's demands is higher than the update frequency of current active replica)
{
The system checks other CL has satisfy update frequency.
If (the update frequency of other CL can satisfy the requirements of the request)
{
The system only replicate data from other CL to the requested CM.
}
End If
}
Else
{
The system directly replicate data from primary copy to requested CM.
}
End If
END

4.3. Station Diagram of the Weather Forecast System

The proposed weather forecasting system has three weather forecast stations (Yangon station, Mandalay station and Naypyitaw station) as shown in figure 4. Each station is connected (Peer-to-Peer) to the remaining stations for the distributed data sharing. Every thirty minutes, each station uploads the updated weather forecast data to support the consistent weather information for other station.

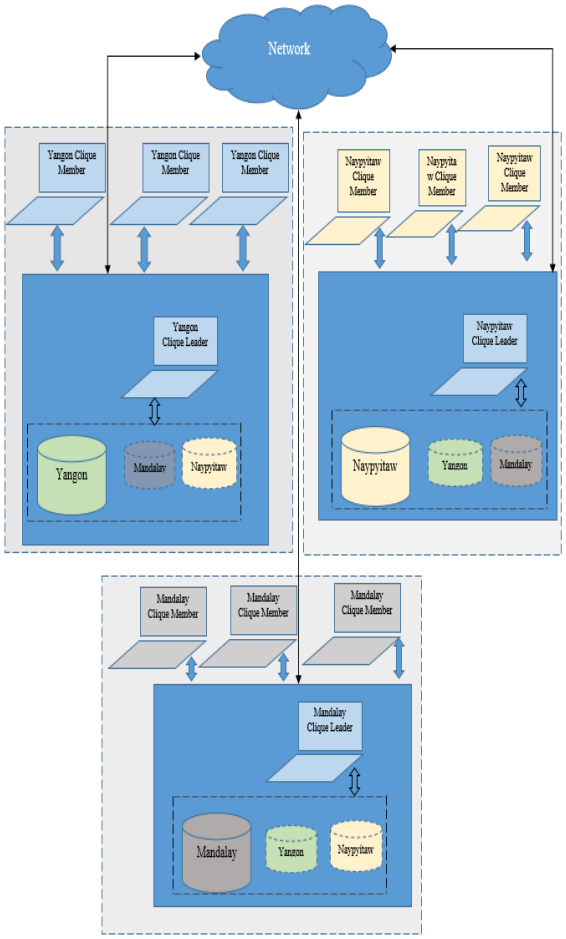

Figure 4. Station Diagram of Proposed System

4.4. Database Design for the Stations

The database design of each weather forecast stations are as shown in figure 5. Each station has four data tables (Station table, WeatherInfo table, Mile table and User table). Station table stores station information: StationID, StationName, Location, DBAddress, Status and DistanceMileID. WeatherInfo table contains detail information of weather forecasting. Mile table stores the distance miles between each station and this distance miles are used to determine to support the update data with lowest latency. User table stores the user information of each weather forecast station.

Figure 5. The Database Design

4.5. Implementation of the System

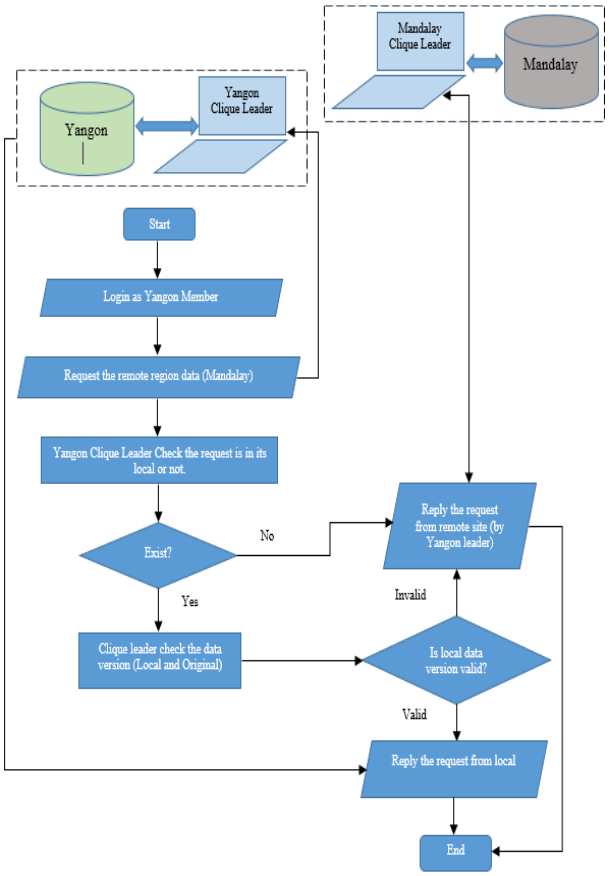

Figure 6. The Process Flow

The detail process flow is shown in figure 6. When the user enters the system as Yangon station member and then the user wants to get the remote station data (Mandalay station's data), Yangon station checks the user requested data in local station. If Yangon station has Mandalay data, the system checks the Mandalay data in Yangon station is valid or not. If Yangon station has updated Mandalay data, Yangon station returns the result to user. If not, the ORDER-RS system will get the user requested data from the nearest remote station and sends the result to user.

By the use of ORDER-RS, the system maintains the consistent data update between each station and will support the data update in lowest latency. For more clear explanation, the detail operations steps are also shown in the following sequence diagram (Figure 7).

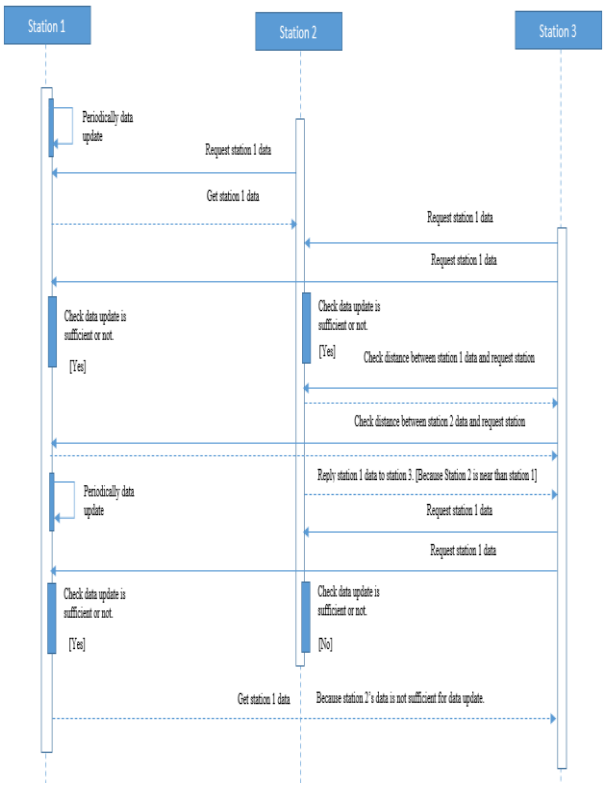

Figure 7. Sequence Diagram of the System

5. Conclusion

Weather is highly volatile and must provide the most precise data for respective city. A huge network of weather stations distributed throughout the world combine with the suitable replication algorithms. The propose on-demand weather forecast data sharing system is used for distribution, replication and data management by ORDER-RS algorithm.

REFERENCES

[1] Gray, J., Helland, P., O'Neil, P., Shasha, D.: The dangers of replication and a solution. In: Proc. of the ACM SIGMOD International Conference on Management of Data. Volume 25, 2 of ACM SIGMOD Record., New York, ACM Press 173–182

[2] Mya Thidar, "Update Propagation On Lazy and Eager Replication With Group and Master Replication Databases", August 2008, University of Computer Studies, Yangon

[3] Naval technology. (In: http://www.naval-technology.com/contractors/data management /index.html)

[4] Nann Thin Thin Nwe, "Data Replication in Aircraft Components Database System using Distributed Database System", December , PSC 2010, University of Computer Studies, Yangon.

[5] Son, S.: Replicated data management in distributed database systems. SIGMOD Record 17 62–69

[6] Wei, Y., Son, S., Stankovic, J., Kang, K.: Qos management in replicated real-time databases. In: 24th IEEE Real-Time Systems Symposium (RTSS).