A Framework for Consistency Control of Replicated Data in Distributed Environment

May Mar Oo
University of Computer Studies, Mandalay
maymaroo@gmail.com

Abstract

Nowadays, network – oriented computing environments require high – performance, distributed database systems that can satisfy both the data consistency control of replica management and data storage requirements in distributed network. Consistency maintenance is a fundamental issue in many areas of computing systems, including database systems and distributed systems. Replicas must be kept consistent, they must be locatable, and their lifetime must be managed. These and other issues necessitate a high level system for replica management in database system. So, this paper presents a new framework for consistency control of replicated data in distributed databases environment by using the proposed voting method.

1. Introduction

A consistency model is essentially a contract between processes and the data store. It says that if processes agree to obey certain rules, the store promises to work correctly. Consistency is an issue for both replicated objects and transactions involving related updates to different objects. An important issue in distributed systems is the replication of data. To achieve high performance of operations on shared data, designers of parallel computers have paid much attention to different consistency models for distributed shared memory systems.

Consistency models for shared data are often hard to implement efficiently in large-scale distributed systems. Moreover, in many cases simpler models can be used, which are also often easier to implement. One specific class is formed by client-centric consistency models, which concentrate on consistency from the perspective of a single client. We need to consider how consistency is actually implemented. Two, more or less independent, issues play a role in keeping replicas consistent [2].

Replication in database systems are generally used to improve performance, increase the fault tolerance and introduce enhanced availability of the data into database systems. When several replicas of a database are present, they can divide the workload of incoming requests between each other.

If a distributed database system spans a large geographical area, placing several nodes in the parts of the area with the highest workload is a way to efficiently improve the performance of the system. Since the distance to the node from the user is decreased the time it take to access the system is also decreased which leads to a performance gain from the user's point of view. This makes it possible for one or several nodes to

fail without causing the entire system to become unavailable.

The introduction of more nodes into a system isn't always positive, with an increased number of nodes; the system consumes more bandwidth in general and uses more processing power on all nodes in the system when it tries to keep all data consistent. This can be solved by lighten the consistency demands on the system, by allowing the consistency process to not be atomic, e.g. ,all the database don't look the same all the time, this approach can't be used on all types of databases since some systems have strict demands on the consistency of the databases [1].

The first issue is the actual distribution of updates, which concerns placement of replicas and how updates are propagated between replicas.

The second issue is how replicas are kept consistent. In most cases, applications require a strong form of consistency. Informally, this means that updates are to be propagated more or less immediately between replicas.

2. Related Works

D.D. Vecchio and S.H. Son [4] presented an adapted version of the traditional quorum-consensus approach that is better suited to the P2P domain. In addition, they explored the many tradeoffs introduced by that flexible update scheme, especially in the areas of data freshness, data availability, access latency and redundancy. S.Grottke et al, [8] described the principle layout of a typical distributed virtual environment or wireless sensor networks, and the most important classes of consistency algorithms used in their systems. They proposed several measures for the level of inconsistency, and the various cost factors such as network load, memory footprint, CPU load and in the case of WNSs, energy consumption .D.S.Dhaliwal et al, [5] presented the general discussion of memory consistency model like Strict Consistency, Sequential Consistency, Weak Consistency and etc. the problems that are associated with each protocol is discussed and other related things are explored.

3. Transparent Management of Distributed and Replicated Data

Transparency refers to separation of the higher-level semantics of a system from lower-level implementation issues. A transparent system hides the implementation details from users. The advantage of a fully transparent DBMS is the high level of support that it provides for the development of complex application. A system supports data replication if a given stored relvar or, more generally, a given fragment of a given stored relvar can be represented by many distinct copies or replicas, stored at many distinct sites. Fully transparent access means that the users can still pose the query as specified in local, without paying attention to the fragmentation, location or replication of data and let the system worry about resolving these issues [10].

For performance, reliability and availability reasons, it is usually desirable to be able to distribute data in a replicated fashion across the machines on network. Such replication helps performance since diverse and conflicting user requirements can be more easily accommodate. Assuming that data is replicated, the issue related to transparency that needs to be addressed is whether the users should be aware of the existence of copies or whether the system should handle the management of copies and the user

should act as if there is a single copy of the data. On the other hand, doing so inevitably results in the loss of some flexibility. It is not the system that decides whether or not to have copies and how many copies to have, but the user application. Any change in these decisions because of various considerations definitely affects the user application and therefore reduces data independence considerably. Given these considerations, it is desirable that replication transparency be provided as a standard feature of DBMS. It is important that replication transparency refers only to the existence of replicas, not to their actual location.

4. Replicated data management strategies

Replicated data management strategies in the traditional distributed environment and broadly categorized the replica control methods into three groups, namely, primary copy method, quorum consensus method and available copies method. The primary copy method is suitable for a database where most accesses made to the data which originate from the node having the primary copy, thus local "read" and "write" requests are satisfied immediately by the primary copy. The primary copy becomes a bottleneck in a large system with frequent updates as all "write" requests from transactions are directed to it. In replication management methods, on the other hand, the objective is to provide a high degree of concurrency and thus faster average response time without violating data consistency.

In contrast, the primary copy method requires access to only one replica. With the quorum consensus method, "read" and "write" operations can succeed only when a sufficiently large number of replicas is available. One major advantage of quorum consensus method is that communication link failures that partition the network require no special attention.

In the basic available copies method, updates are applied to replicas at nodes that are operational and failed nodes are ignored. "Read" operation can use any available replica but this could lead to inconsistencies among the replicas. The correct available replicas scheme operates are "read" operations can be directed to any node holding the latest value of the data and "write" operations will succeed only if at least one replica records the update.

The lack of a flexible consistency management solution hinders P2P implementation of applications involving updates, such as read-write file sharing, directory services, online auctions and wide area collaboration. While a number of applications have been suggested for peer-to-peer networks, perhaps chief among the obstacles to effective P2P databases are the problems of data placement and data coordination [3]. The goal of P2P data placement is to distribute data and computation among peers so that queries can be executed with the smallest possible cost in terms of response time, bandwidth usage or some other criteria [9]. Data placement choices and tradeoffs include: the level of shared (global) knowledge and autonomy peers have to make decisions, how dynamic the network is in terms of peer membership, the level of data replication and granularity, freshness and consistency requirements, etc. In contrast, the data coordination problem involves managing data dependencies and semantic mappings between peers [6]. Here the challenge is reconciling numerous individual data schemas to associate related data stored under different names and formats. In this system, we use the transactional replication method.

5. Replicas Update Consistency

In this system, replicas usually adhere to a certain consistency scheme. Modifications need to be propagated to all the replicas of a file before the write lock can be released in consistent read/write replicas. This is true for a synchronous replication model where all replicas have the same values and updates need to lock all existing replicas. More relaxed replication schemes that use asynchronous update mechanisms allow certain replicas to be out of synchronization for a certain amount of time. In the asynchronous replication model, write locks, only need to be applied to certain replicas but has the consequence that not all replicas are always up-to-date. Versioning is a combination of read-only and read/write replicas. Once a file is written, a new version of the file is created. Version information is stored in a meta-data catalogue such that either a particular version or the latest version may be retrieved.

The Consistency service performs two major tasks are it keeps the set of replicas of a file consistent, which is of particular importance if updates of replicas are allowed and it keeps the information stored in the replica location and metadata indices consistent with the physical files. Inconsistent replicas, i.e. replicas that are not exact copies, may be due to unauthorized modification of a file and malicious or spurious modifications. Transactions must be scheduled to meet the timing constraints and to ensure data consistency.

The main consistency problem is to keep replicas consistent; we need to ensure that all conflicting operations (read-write conflict that a read operation and a write operation act concurrently and write-write conflict those two concurrent write operations) are done in the same order everywhere. Guaranteeing global ordering on conflicting operations may be a costly operation, downgrading scalability and solution is weaken consistency requirements so that hopefully global synchronization can be avoided.

In Gifford's scheme [7], to read a file of which N replicas exit, a client needs to assemble a read quorum, an arbitrary collection of any N_R servers, or more. Similarly, to modify a file, a write quorum of at least N_W servers is required. This scheme emphasis a single file is replicated across the nodes. The values of N_R and N_W are subject to the following two constraints:

$$N_R + N_W > N \quad (1)$$

$$N_W > N/2 \quad (2)$$

The first constraint is used to prevent read-write conflicts, whereas the second prevents write-write conflicts.

5.1 Framework of the proposed system

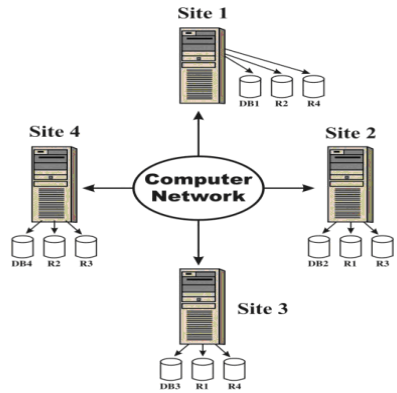

Figure 1. Overall Architecture of the System

According to the Figure 1, there are multiple servers in the distributed environment. In one server, there are a single own database and the other replicated databases. That other replicated

databases are replicated from the other servers in the nearest location to share the data and to get the reliability and availability in the distributed environment.

In this paper, we propose the new voting method for different replicated database files. In it, we add the additional constraint.

$N_W = 1/M$ (3)

It is to prevent the write-write conflict for among the multiple database file replicas. N_W=write quorum for different replicated database files and M= number of servers.

This proposed voting method provides sequential consistency for read and write operations. In this proposed scheme, the multiple files are replicated across the nodes. Client can read any replica from their appropriate group. If client want to write a file, it assembles the write quorum as shown in equation number (3). So, this method can handle the multiple heterogeneous replicated database files to protect the write- write conflict across the network.

In this system, we design to place the replicas to half of the total servers. In that point, we choose the nearest server to place the replicas according to the distance. We will measure the level of consistency accurately by evaluating the query response time according to proposed method.

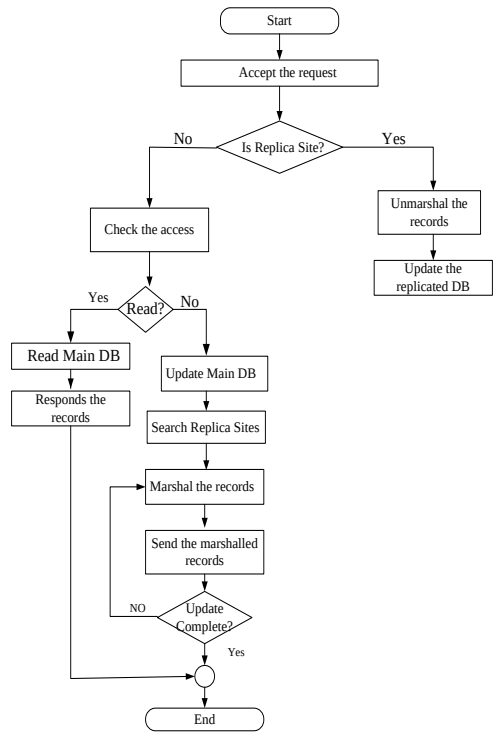

Figure 2. Data Flow Diagram of the System

According to the Figure 2, firstly, in this system the server accepts the request from the client. Then it checks the requested site which is the replica site or not. If it is a replica site, the system unmarshals the records and updates the replicated database. If it is not a replica site, then the system checks the read operation or write operation to the main database. If it is a read operation to the main database, it read the main database and responds the records of the database. If it is a write operation to the main database, it updates the main database and searches the replica sites to update. If the concerned replica sites are found, it marshals the records. Then it sends to those sites and updates

the records into those sites. At that point, if it does not complete the update operation, it returns to the marshal the records process state and restarts it to that state.

6. Conclusion

This proposed system can provide the consistency with the optimal latency time for different replicated data files among the multiple databases in distributed environment. In addition, this system can also provide the data availability, reliability and will intend to reduce the high probability of stale data access to zero probability of stale data access among the multiple heterogeneous replicated database files as the future work.

References

[1]A. S. Tanenbaum and M.V.Steen, Distributed System, Pearson Education, Inc., Upper Saddle River, New Jersey,2007.

[2]A. Zaslavsky and et.al, "Database Replica Management Strategies in Multidatabase Systems with Mobile Hosts Department of Computer Technology", Monash University, Melbourne, Australia.,1995.

[3]D. Barkai, "Technologies for Sharing and Collaborating on the Net" In Proceedings, First International Conference on Peer-to-Peer Computing, August 2001, pp. 13-28.

[4]D.D. Vecchio, and S.H. Son, "Flexible Update Management in Peer-to-Peer Database Systems, "Database Engineering Conference", Department of Computer Science, University of Virgina, 2005.

[5]D.S.Dhaliwal,P.S.Sandhu and S.N.Panda, "Consistency Model and Synchronization Primitives in SDSMS",World Academy of Science, Engineering and Technology, 2009.

[6]F. Giunchiglia and I. Zaihrayeu, "Making Peer Databases Interact",A Vision for an Architecture Supporting Data Coordination, Conference on Information Agents, Madrid, September 2002.

[7]Gifford , et.al, "Weighted Voting for Replicated Data", Proc. Seventh Symp, Operating System Principles, ACM, 1979.

[8]S.Grottke,A.Kopke,J.Sablatnig,J.Chen,R.Seiler,A. Wolisz,"Technische Universitat Berlin Telecommunication Networks Group",2007.

[9]S. Gribble, et al., "What Can Peer-to-Peer Do for Databases, and Vice Versa? ", 4th Int'l Workshop on the Web and Databases (2001).

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