Peer-to-Peer Replication

Matthieu Weber

September 13, 2002

Contents

1 Introduction

Research on the W.W.W. about peer-to-peer replication have given two kind of results:

- replication of databases in a peer-to-peer way,
- data replication in an unstructured peer-to-peer network.

The two approaches, if leading to the same results (i.e. data is replicated on different nodes of a network), are rather different. Database replication aims to replicate data along with its metadata (e.g. a phone number and the fact that this piece of data is a phone number and it belongs to Mr Jones) across a well-known set of database in a controlled environment, whereas peer-to-peer network allow the users to share (i.e. allow other users to copy, therefore replicate) files without much checking about metadata, with whoever wants to copy the files. The consequence is that data is replicated, but the user might get the recipe of pea soup instead of the phone number of Mr Jones, due to inconsistency in metadata.

2 Database Replication

The fundamental issue in data replication is maintaining data integrity in across multiple database images. Two primary solutions exist, synchronous and asynchronous replication[1].

2.1 Synchronous Replication

Synchronous replication requires that every image of the database be written at once. Data integrity is maintained in traditional database solutions with a two-phase commit within a transaction that accesses every image of the database element to be written, and processes only when every image is available for update. In traditional database solutions, synchronous replication eliminates the distinction between master/slave and peer-to-peer configurations.

However, traditional synchronous solutions are vulnerable to system failure. If one image of the database is unavailable due to server failure, the transaction is prevented from completing.

While synchronous replication assures data integrity, it does it so at the expense of availability. System availability is greatly reduced with synchronous replication if the links between the database images are fragile. This vulnerability to network failures also prevents synchronously replicated database servers from being geographically distributed, leaving them vulnerable to location specific disasters, like earthquakes.

2.2 Asynchronous Replication

Asynchronous data replication provides two benefits, improved read performance and continuous availability. The failure of a particular server does not

generally affect the operation of he other servers, but does force more work to be done in the application to maintain data integrity.

When changes are made, a single image of the database is updated, and then the changes are propagated to the remaining images. The two popular mechanisms for propagating changes to the other databases images are embedded triggers and passing change logs. Triggers are implemented inside the database and add to the overhead associated with performing transactions in the database. Passing change logs also increases network overhead during the replication of the update, but is less intrusive on the local database.

Asynchronous replication also allows database images to be isolated geographically, since each site has its own version of the data. This fact is of growing importance, since more and more e-commerce companies grow overseas, and need to provide access points (i.e. the company's web page) in different countries, thus allowing efficient and uninterrupted access to the company's database, which can be greatly improved with the help of peer-to-peer database replication[12].

2.3 Conflict Avoidance, Detection and Resolution

Conflict resolution is required when two images if the same database are updated at the same time, without knowledge of the other image.

The simplest solution for this type of conflict is to have the application avoid such conflict by ensuring that each data element belongs to one database image or another. This forces a master/slave relationship between database images.

Once the conflict exists, traditional databases provide mechanisms to detect and resolve them. Conflict resolution algorithms assign priorities to the conflicting updates based on update requester status (master or slave), timestamps, or some type of application dependent algorithm.

3 Peer-to-Peer Systems

Peer-to-peer networks are born because of the need to share files among users, which didn't have the possibility to publish their files in the traditional client/server way. Typically, peer-to-peer network users have huge amounts of files, that cannot be fit on public shared servers (like for example free web hosting services). The idea of peer-to-peer computing is that each user becomes at the same time server and client, thus having an equal role than other users.

This paradigm can be extended to whatever interaction where all the protagonists have the same rank. There is no notion anymore of master/slave or client/server.

3.1 Peer-to-Peer File Sharing Systems

The main problem in peer-to-peer file sharing network is the advertisement of shared files. It has been solved first with the help of a central index (Napster[9]). This structure has proved to be easy to tear down, since each node of the network depends on the index, which is a potential point of failure. Other index-based solutions exist, using multiple indexes loosely replicated (eDonkey[10], FastTrack[11]). Another solution has been to build a completely distributed network. This kind of peer-to-peer network might be either strongly structured (Chord[6], CAN[5], Past[7], Tapestry[8]) loosely structured (Freenet[3]) or unstructured (Gnutella[4]).

Unstructured peer-to-peer network don't rely on any central index for searching. The nodes instead send search queries and ask who owns the file one is looking for. Structured network additionally rely on their implicit structure to make the search faster.

4 Peer-to-peer Database Replication

In Objectivity/DB[1], peer-to-peer replication is achieved as follows: multiple database images are set up, and a a transaction is completed only when enough images are available to form a quorum.

4.1 Quorum Calculation

The number of database images required to complete a transaction is calculated implicitly at runtime. This is called the "quorum calculation". Simply put, the database images vote and the majority wins.

When a database image is accessed, and a lock is implicitly requested as part of that access, the application process running on the local computer contacts each lock server to determine which database images are available to vote. If a majority of the images of a database is available, called a "quorum", the access is permitted.

The images which are not available, for whatever reason, will be automatically resynchronized when they come back on line.

4.2 Non-Unitary Weighting

There are many replication scenarios in which some database images require special access or ownership of particular data, regardless of server or network failures elsewhere in the system. This is addressed by assigning voting weights to each database image.

A master/slave configuration can thus be achieved by granting the master more weight than the sum of the weights of the slaves. Thus the master must always be on line in order to have a quorum of available images.

5 Peer-to-Peer File Replication

Peer-to-peer file sharing systems don't provide, nowadays, any version control system nor fail-proof data identification system through the use of metadata. The consequence is that data replication in peer-to-peer networks is done at file level and is not very useful for a completely automated system: conflict resolution is entirely left to the users. Today's peer-to-peer file sharing systems allow only to replicate pieces of data, not to manage update, different versions nor concurrent write access to the data.

Whereas data in a database is well organized, distributed peer-to-peer file sharing systems are completely chaotic. File replication is usually done only when a user is requesting it, therefore creating a distinction between popular files (files which are often requested) and unwanted files. Because there is no index of all available data, searching is required before copying a file.

The performance of searching data depends greatly on how much that piece of data has been replicated in the network[2]. Various file sharing systems have different policies: in Gnutella[4], a file is replicated when the requester node gets a copy of it; in Freenet[3], a file is proactively replicated at node which have note explicitly requested the file, but which are involved in the retrieval process.

In[2] are compared three replication strategies:

- Uniform replication, for which a fixed number of copies is made for each file,
- Proportional replication, where a fixed number of copies is made after each request of a file,
- Square-Root replication, for which the number of copies made is proportional to the number of probes which have been made when searching for the file, once the file has been found.

The Square-Root replication is proved as the most efficient one. It is, however, the most difficult to implement.

6 Conclusion

Data replication in a database environment and in a file sharing system is rather different, although the final goal is to make a copy of some data. Whereas the constraints on database replication are rather strong (control over the number of nodes, their behavior, consistency of data regarding meta-data), they are rather loose in file sharing systems, which allow a greater freedom of behavior, at the cost of numerous organizational problems.

Given the little amount of information found on the Internet about peer-to-peer database replication, this technology seems not yet ready to be sold to customers.

References

[1] *Data Replication in Objectivity/DB — An Objectivity, Inc White Paper.* Objectivity, Inc. 2001.

[2] Qin Lv, Pei Cao, Edith Cohen, Kai Li and Scott Shenker. *Search and Replication in Unstructerd Peer-to-Peer Networks.*

[3] Ian Clarke. *A Distributed Decentralised Information Storage and Retrieval System.* Master's Thesis, Division of Informatics, University of Edinburgh, 1999.

[4] Open Source Community. *Gnutella.* In *http://gnutella.wego.com*, 2001.

[5] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp and Scott Shenker. *A scalable content addressable network.* In *Proceedings of SOSP'01*, 2001.

[6] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek and Hari Balakrishnan. *Chord: A scalable peer-to-peer lookup service for Internet applications.* In *Proceedings of SIGCOMM'2001*, August 2001.

[7] A. Rowstron and P. Druschel. *Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility.* In *Proceedings of SOSP'01*, 2001.

[8] Ben Y. Zhao, john Kubiatowicz and Anthony Joseph. *Tapestry: An infrastructure for fault-tolerant wide-area location and routing.* Technical report UCB/CSD-01-1141, University of California at Berkeley, Computer Science Department, 2001.

[9] In *http://www.napster.com.*

[10] In *http://www.edonkey2000.com.*

[11] In *http://www.fasttrack.com.*

[12] Michael C. Morrison *The Growing Importance of Peer-to-Peer Replication* In *DM Direct*, September 2001.