Replica Synchronization in Distributed File System using Asynchronous Replication

Mr. B. Muruganantham, Mr.Tushar Kumar Pandey
Assistant Professor(Sr.G), Department of Computer Science and Engineering, SRM University, India
M.Tech Scholar, Department of Computer Science and Engineering, SRM University, India

ABSTRACT *The availability of resources in a distributed environment is an important factor to be considered when designing a distributed system. Replication is a well-known technique to achieve fault tolerance in distributed systems, thereby enhancing availability. But some efficient techniques are required to keep all the replicas consistent i.e every replica should have the same copy of the data. When a user changes or does any modification, the similar modification has to be done on every other replica for the purpose of achieving consistency. Proposed method uses write through mechanism i.e. whenever the client make any modification on the file, that modification will be immediately transmitted to the server, so the server can also perform similar modification on the server copy of the file and then asynchronous replication is followed i.e whenever any modification is done, first the primary copy is updated and new version number is added with the dirty bit marked as 0. This shows that the data is a valid data. A threshold is set between subsequent modification and once the threshold is reached, then the updates will be propagated to other replicas also. It ensures data consistency from the viewpoint of client.*

Keywords *- File Replication, Update Propagation.*

1 Introduction

DFS is implemented on a cooperating set of server computer connected by a communication network, which together create the illusion of a single, logical system for the purpose of creation, deletion, and random accessing of data. There are many importance of distributed file system like it improves scalability, improves reliability and supports inherent distribution. In DFS, there are one or more servers that store the data and are accessed over the network. Challenges faced by distributed file system are partial failures and concurrency. Partial failures include network congestion and node failures. Concurrency problem arise because multiple nodes execute in parallel so challenge lies in keeping the data consistent. Some Other problems also needs to be addressed like if a client is accessing a file and some other client requested the same file from the same server at the same time. i.e. if a client say client A is modifying a file , but the modified data has not yet propagated to the server and at the same time client B puts a request and downloads the same file from the same server into its local cache. Here client B will have an inconsistent copy of the file and this problem can be avoided by using a write through mechanism i.e. the modification done by a client is immediately propagated to the server, so that the primary copy can also be updated. This mechanism introduces another problem known as cache consistency problem that can be solved by cache invalidation scheme. For increasing availability, redundancy is done which is called replication.

1.1 Replication

It is the process of creating and managing duplicate versions of a database. Replication not only copies a database but also synchronizes a set of replicas so that changes made to one replica are reflected in all the others. The beauty of replication is that it enables many users to work with their own local copy of a database but have the database updated as if they were working on a single, centralized database. For database applications where users are geographically widely distributed, replication is often the most efficient method of database access. A replicated file is a file that has multiple copies, with each file on a separate file server. A problem associated with replication is that whenever a file is modified or updated, then that modification has to be propagated to all other copies also. There are basically two types of replication: active replication and passive replication. Replication is used to increase availability and to reduce the access time, but it increases overhead like cost, time and to keep all the replicas consistent. There is no need to replicate every copy of the data each time when any modification is done, because if a data is changing now, then it has a higher probability that it will change in future also. So, certain threshold should be set, based upon which the other copies should be updated.

The rest of the paper is organized as follows. The next section discusses a brief literature survey of existing theories and work done so far. Section 3 discuss about the proposed approach. Finally, section 4 concludes the work followed by references.

2 Related work

Many distributed file systems support different replication policies for reliability and replica synchronization mechanisms.

An adaptive replica synchronization[1.] mechanism among storage servers (SSs) without the interference from the metadata server (MDS) in a distributed file system is a mechanism that employs a chunk list data structure, which holds the information about the relevant chunk replicas and is stored on the associated SSs corresponding to the replicas. In most of the conventional distributed file systems, the data replication is controlled between the MDS and is transparent to the SSs, which will place extra pressure onto the MDS. PARTE, a prototype parallel file[2.] system with active/standby configured metadata servers (MDSs). PARTE replicates and distributes a part of files metadata to the corresponding metadata stripes on the storage servers (OSTs) with a per-file granularity; meanwhile the client file system (client) keeps certain sent metadata requests. If the active MDS has crashed for some reason, these client backup requests will be replayed by the standby MDS to restore the lost metadata. Jajodia [3.] addressed data replication and claimed that the adaptive data replication algorithm aims at decreasing the bandwidth utilization and latency by moving data closer to clients. He considered that adaptive replication algorithms change the replication scheme of an object to reflect the read-write patterns and eventually converge towards the optimal scheme. Random replication[4.] is widely used in data center storage systems to prevent data loss. However, random replication is almost guaranteed to lose data in the common scenario of simultaneous node failures due to cluster-wide power outages. Due to the high fixed cost of each incident of data loss, many data center operators prefer to minimize the frequency of such events at the expense of losing more data in each event. Copyset Replication, a novel general purpose replication technique that significantly reduces the frequency of data loss events. Such systems require that each node’s data be scattered across several nodes for parallel data recovery and access. Clark et al. [5.] replicates objects both on insertion and retrieval on the path from the initiator to the target, mainly for anonymity and availability purposes. Google File System[6.], a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

3 Proposed work

Replica synchronization is the process of keeping every replica consistent. In a distributed environment, whenever a data is modified by any client, in order to maintain replica consistency the changes have to be propagated to all other replicas. So if any other client puts a read request, it should get a valid copy of the data. This method uses write through mechanism but has a small difference that whenever the client make any changes on the file, that changes will be first transmitted to the primary server, so that the server can be synchronized and then asynchronous replication is followed. All write operations will be carried out on a primary copy and then later based on certain threshold, other replicas will be updated. Replica consistency will be maintained from the viewpoint of client i.e. though all the replicas will not be consistent, then also client will always get valid data. Replication is reduced by using asynchronous replication which increases the performance.

Each operation is either a read operation or a write operation, where write means writing a value and read means reading the data. For write operation, every time they are carried on a single primary copy which is later replicated to all other replicas. Read operations are done based on the version and dirty bit. When a read request comes, the system checks for the replica with the highest version number and also checks its dirty bit. A value one in the dirty field represents that the data is dirty and a value zero represents that the data is valid and is the latest written value. Read operations are carried out based on the version and dirty bit. Upon receiving a read request the system checks for the replica with the highest version number and also checks its dirty bit. A value one in the dirty field represents that the data is dirty and a value zero represents that the data is valid and is the latest written value.

The goal is to perform replica synchronization efficiently. Asynchronous replication helps to do so because this mechanism does not replicate the data immediately after the write operation, rather than it will wait for a threshold and based on that it will make all the other replicas consistent.

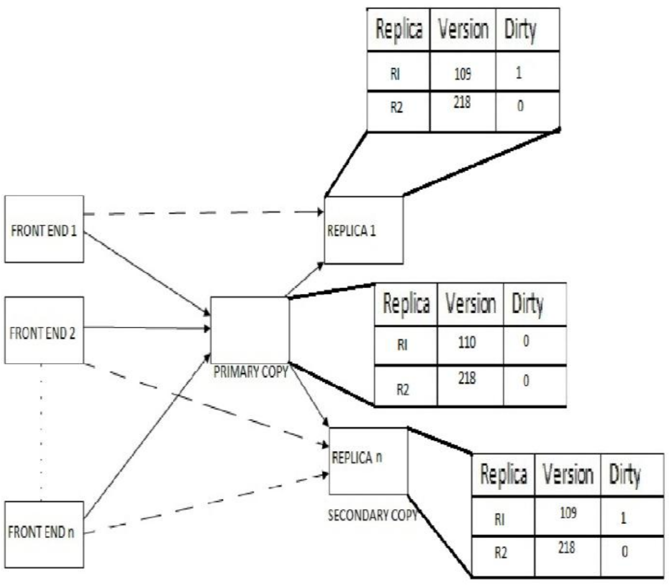

Figure. 1. Replica Synchronization

Replica synchronization is done by using asynchronous replication i.e whenever any modification is done, first the primary copy is updated and new version number is added with the dirty bit marked as 0. This shows that the data is a valid data. A threshold is set between subsequent modification and once the threshold is reached, then the updates will be propagated to other replicas also. Replication list contains the information about all replicas and is stored on the SSs. After getting updateList request, information in the list is updated. It can ensure the replica consistency from the viewpoint of clients, although certain replicas have not been updated. List helps to maintain replica consistency from the viewpoint of clients.

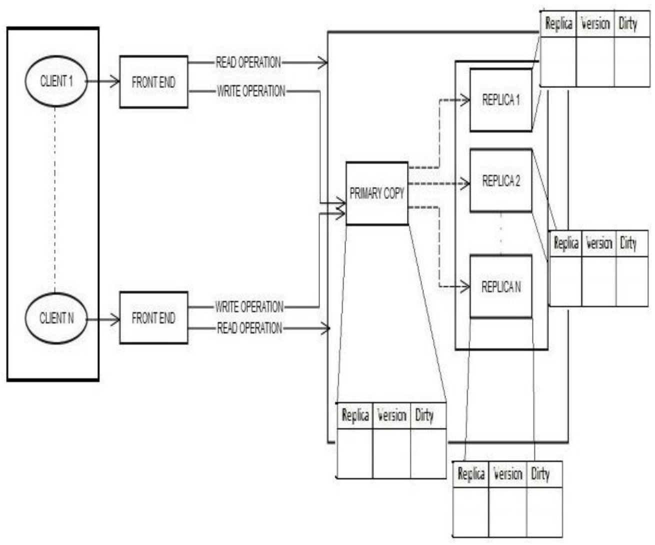

Figure. 2. System Architecture

Every modification is done on the primary server and then is replicated to other copies. By using the versioning concept and dirty bit, consistency is maintained because during every read operation, the data is read from the replica where the value of the dirty bit is 0 and version is the maximum.

In the Fig. 2, the given architecture describes the read/write operations and asynchronous replication mechanism. Client performs its operations by using front end. For write operation, every time they are carried on a single primary copy which is later replicated to all other replicas. Read operations are done based on the version and dirty bit. When a read request comes, the system checks for the replica with the highest version number and also checks its dirty bit. A value one in the dirty field represents that the data is dirty and a value zero represents that the data is valid and is the latest written value. Hence, replica consistency is maintained from the viewpoint of client.

3.1 ALGORITHM

```
1: while Storage server is active do
2:     if Write request received then
3:         [version]=request id;
4:         conduct write operation on the
           primary copy
5:         update the fields in the list;
```

```
6:              set dirty =0 &&
                version<-request id;
7:              broadcast update list request
8:          **endif**
9:          **if Read request received then**
10:             check if dirty=0
11:             conduct read operation
12:         **else**
13:             select other replica
14:             goto step 10
15:         **if update List request received then**
16:             perform update and send back
                ACK
17:             set dirty=0;
18:             break;
19:             **endif**
20:**End while**
```

Whenever a write request comes, the value of request id is stored in the version number field and the write operation is carried out on the primary copy. Fields in the list are also updated like dirty bit is marked as zero to show that the given data is a valid data and is the result of the latest write operation. Dirty bit of all other replica are marked as one to indicate that the data is inconsistent. If a read request comes, it checks the dirty bit in the list, if the dirty bit is marked zero, then the read operation will be carried out on that replica, otherwise, it will continue to search a replica in which value of dirty bit is equal to zero. This technique achieves consistency from the viewpoint of client i.e. though the updated data is not copied to every replica, then also client will always get an updated copy on every request. Whenever any update list request comes, the entire lists are updated and acknowledgement is sent to the requestor.

3.2 TABLES USED

4 TABLE 1. FILESTATUS TABLE

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION
Username	VARCHAR	Stores username
Fileid	INT	Unique Id of file
Filename	VARCHAR	Name of file that is replicated
Filepath	VARCHAR	Location of file
Filesize	INT	Size of file in bytes
Date	Date	Date of recent modification
Ver	INT	Version number

TABLE 2. FILESDETAILS TABLE

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION
Fileid	INT	Unique Id of file
Requestcount	INT	Number of requests for a particular file
Replication Threshold	INT	Maximum number of requests for a particular file, after which file will be replicated on other servers.
Dirty	INT	Flag bit

The table 1 shown above comprises of various fields that stores various attributes of the file like filename, fileid, filesize, ver, etc. the table 2 comprises of detailed file attributes like requestcount, replication threshold, dirty bit, etc. Here we are maintaining three replicas. Changes at one will immediately reflect on other two replicas. When a node selects a file of its interest from number of files, it is copied into the primary copy first and any changes on existing file or document causes the same changes to be propagated to other two replicas also. Versioning concept is used here in order to identify the most recent copy of the replica. With every modification of the file replica, a notification of the update is sent to all other nodes that have the replica of the file that has been updated and with that the version number is automatically incremented. Every modification is done on primary server to reduce unnecessary write operation and asynchronous replication is done at a later point of time, when the threshold is reached. Upon failure of the primary server, one of the other replica server act as primary server to satisfy the read/write request. This technique ensures replica consistency from the viewpoint of client.

4 CONCLUSION

The paper presents a replica synchronization mechanism using asynchronous replication mechanism. Once the file has been replicated on 'n' replica servers, the issue is to maintain consistency among them. Here the replica is propagated based on the threshold, i.e. when the threshold is reached, all other replicas are made consistent. Every read request is given to the server with highest version number for the requested file and with dirty bit marked 0. This avoids unnecessary replication and also achieves consistency from the viewpoint of clients.

References

[1] Jianwei Liao, Li Li, Huaidong Chen, Xiaoyan Liu, "*Adaptive Replica Synchronization for Distributed File Systems*", IEEE System Journal, 2014.

[2] D Jianwei Liao, Yutaka Ishikawa, "*Partial Replication of Metadata to Achieve High Metadata Availability in Parallel File Systems*," in Proc. 41st ICPP, 2012.

[3] Wolfson, O., Jajodia, S. and Huang, Y. 1997. An adaptive data replication algorithm. ACM Transactions on Database Systems, 22(2):255–314.

[4] Cidon, Stephen Rumble, Ryan Stutsman, Sachin Katti, John Ousterhout and Mendel Rosenblum, "*Copysets: Reducing the frequency of data loss in cloud storage*," in Proc. USENIX Conf. ATC, 2013.

[5] Clarke, I., Sandberg, O., Wiley, B. and Hong, T. 2000. Freenet: A distributed anonymous information storage and retrieval system.

[6] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, "*The Google File System*," SOSP'03, October 1S9–22, 2003, Bolton Landing.