Dynamic Replication Management Mechanism Based on Accessing Frequency Detecting

May Sit Hman, Khaing
University of Computer Studies, Yangon
maysismhan@gmail.com, khaing@ucsy.edu.mm

Abstract

This paper describes replication management policies in distributed file system, and shows a decentralized dynamic replication management mechanism dependent on accessing frequency detecting named FDRM (Frequency Detecting Replication Management). In FDRM, in order to support better system performance and decrease network traffic, system nodes scan their local replicas to monitor replicas' access pattern, and makes decision independently to add, delete or migrate replicas. In addition, the scanning interval of a replica is variable according to the accessing frequency to that replica, which makes FDRM more sensitive to the change of system behaviors, so that one can receive better performance with less system overhead.

Key words: *FDRM, distributed, replica management, dynamic, frequency, adaptive*

1. Introduction

Replication of data is a necessary module of peer-to-peer systems and distributed file systems. There are two vital inspirations for data replication: enhancing system performance and rising data availability. Appending replicas for the files in the distributed file system grants the system to maintain the capability of accessing file data in cases of nodes failure and network partition, and thus improves file reliability and availability. In addition, if a file is replicated near the site where it is often obtained, communication costs could be decreased by reading the data near at hand, thus the response time will be decreased and system performance will be improved. For instance, FreeNet and Napster execute a replication strategy to produce replicas for a file based on its well-liked, thus enhance file's obtainable and accessing efficiency.

Distributed file system must supply a constant view for many replicas of files, which needs distinctive effort to maintain data consistency. With quick improvement of system scale, and rising complication of network environment, network costs and system expenses for data synchronization among file replicas are also improved greatly, which may damage the system performance. Under this condition, an optimal replication management mechanism for a large-scaled distributed file system is required.

This paper represents an effective distributed dynamic replica management mechanism based on accessing frequency detecting named FDRM. The purpose of FDRM is to enhance system performance by dynamically adding or removing file replicas to optimize the message traffic in the network. FDRM leads in a distributed manner. Every node does its own decision independently to add, delete or move its replicas, and also balances replica degree of a file to adjust the performance of both write and read. FDRM keeps eyes on a replica in a periodical approach, and the length of interval of each replica is changeable according to the access frequency during lifetime of replicas. By this means, FDRM handles replicas of a file individually, and attains better performance with low system overhead.

2. Related Work

In a distributed file system, there are two necessary components for replica management policy: replica degree and allocation of the replicas. Replica degree is the quantity of replicas of a given file, and allocation of the replicas resolves the nodes of the system in which replicas must be physically stored. Replica degree and the allocation of the replicas together decide the replication scheme for a file, i.e., a subset of the nodes in the system which hold a replica of that file.

For the most part, for a read-intensive file, a broadly distributed replication scheme is commanded in order to increase the quantity of local reads and to decrease the load on a center node. Then again, an update of an object is usually written to all replicas.

So, for a write-intensive file, a wide distributed scheme backs off each write, and increases its communication cost. Consequently, for this situation, a barely distributed replication scheme is commanded.

As per the minute at which decisions are taken, there are two important kinds of replication management policies: static replication policy and dynamic replication policy. A static replication policy determines the replica degree and the allocation of the replicas at the moment when the file enters the system, and this determination will not be changed during the lifetime of the file. This determination is normally based on probable access patterns, and has been widely studied. The static plan performs well if the access patterns are known previously, however this is not always possible, particularly in a large-scaled distributed system environment. A dynamic replication strategy takes its choice at the moment of file creation and this choice keeps changing during the lifetime of the file. For example, both the replica degree and the allocation of the replicas are probably going to change now and again. The difference of replication scheme depends on the changing pattern of file accesses as well as the load of system. By adjusting the replication degree and the allocation of the replicas, dynamic replication scheme can decrease the network traffic, and give better system performance.

Corresponding to the technique that dynamic replication policy uses to make decision for replica scheme, we can arrange dynamic strategy into two kinds: centralized solution and distributed solution. A centralized solution must have a global view on the access pattern of all the replicas of a file, and settles on its choice dependent on the information which is gathered in a central point. The advantage of centralized solution is simplicity. Be that as it may, a centralized solution is probably to become a bottleneck for a large distributed file system and it is less flexible. Rather, in a distributed solution, a node settles its decision of replica scheme independently based on its local information of this replica, and requirements not to know the information of replicas on other nodes. Distributed solutions have two benefits over centralized ones. They react to changes of read-write pattern in a timelier way, since they avoid the delay involved in the accumulation of statistics, calculation, and their overhead is lower because they eliminate the additional messages required in centralized cases.

3. Basic Model of FDRM

In this paper, we inscribes on distributed dynamic replication policy. We have some suspicions: FDRM works in the read-one-write-all context, which implies a file read includes only one replica and the write update is proliferated to all replicas; the read request is fulfilled locally or by the nearest replica; the read-write pattern during a period is unsurprising dependent on the pattern in the promptly preceding time period.

3.1. Rules for Replica Adding and Deleting

We expect that there are n replicas of a given file f in the system, $node_i$ (i=1, 2, …, n) signifies n nodes which store these replicas, α means the system cost to fulfill a read request at the closest replica, and β indicates the cost for system to update a replica. Working in a distributed way, FDRM utilizes local information of a replica to determine replication scheme. During a time interval t, a replica of f on $node_i$ keeps some counters to record requests to f both locally and remotely: r_{in} is the quantity of local read requests, $r_{out,j}$ is the quantity of read requests from $node_j$, w_{in} is the quantity of local write requests, w_{out} is the quantity of update requests that $node_i$ receives.

By the third suspicion above, we can recommend that the access pattern in next interval t' is the same as the pattern during current period. So the precondition for $node_i$ to add a new replica to another $node_j$ is that the overall expenses after adding is less than the expenses before adding:

$$r_{out,j}\alpha + nw_{out}\beta + (n\text{-}1)w_{in}\beta > (n+1)w_{out}\beta + nw_{in}\beta \quad (1)$$

The precondition for $node_i$ to delete its replica of f is the overall expense after deleting is less than the expenses before deleting:

$$nw_{out}\beta+(n\text{-}1)w_{in}\beta > r_{in}\alpha + (n\text{-}1)w_{out}\beta+ (n\text{-}1)w_{in}\beta \quad (2)$$

In a actual system, we can suggest $\alpha \approx \beta$, so Eqs. (1) and (2) can be clarified as

$$r_{out,j} > w_{out} + w_{in} \quad (3)$$

$$r_{in} < w_{out} \quad (4)$$

3.2. Interval of Periodicity

After a period of time *t*, a node will make its decision of adding or deleting a replica. But as we mentioned above, the length of *t* is important to system's performance. In order to solve this problem, the intervals of periodicity in FDRM are variable and sensitive to the frequency of file access. The interval will be short when access is not frequent and the interval will become longer when frequency of file accessing increases. By this means, different parts of the replication scheme may have different scan frequencies. (left)

We expect that there is a succession of interval period $t_1, t_2, \ldots, t_n$ for a replica, A_i is the aggregate number of access during the period. If the current interval of periodicity is t_n, the next interval t_{n+1} is

$$t_{n+1} = t_n \frac{An-1\ tn}{Antn-1} \quad (5)$$

3.3. Implementation of FDRM

Each $node_i$ that possesses a replica of f keeps up three counters r_{in} , w_{in} and w_{out} for *f*. $node_i$ also maintains a record *outside_read_list* of information of all outside reading during current period, incorporating counters $r_{out,j}$ recording the quantity of read requests *f* received from $node_j$. At the point when $node_j$ reads the replica on $node_i$, *outside_read_list of* $node_i$ will be filtered to find out whether the replica has been accessed before. In the event that there is a record of previous operation from $node_j$, $node_i$ increases corresponding $r_{out,j}$, otherwise, $node_i$ adds a new record to *outside_read_list*, and sets $r_{out,j}$ to 1.

So as to change the interval of periodicity *t*, $node_i$ keeps a record for the replica of *f*. In this record, we have current interval length t_n, previous interval length t_{n-1}, and the number A_{n-1} of all access that replica had gotten during last period. At that point we can use Eq. (5) to decide the length of next interval t_{n+1}.

There are some exceptional instances of interval of periodicity we should worry about. In the event that there is no access during this period, then no matter how many accesses the replica had during last period, the length of next period ought to be stretched out to double length of current interval. On the off chance that there is no access during the last period, then no matter how many accesses during current period, the length of next period ought to be reduced to half length of current interval. To avoid intervals being too short or too long, we set a lower and maximum limit for it.

$$MIN\_PERIOD <= t <= MAX\_PERIOD \quad (6)$$

At the end of each period of replica, $node_i$ chooses what activity will be taken to the replica. Finally, $node_i$ sets next interval of periodicity for the replica of file *f* by Eq. (5), and resets every of the counters of the replica.

4. Proposed System

In this system, we will use three distributed databases for implementing of the system and many registered users at each site. Figure 1. shows the system overview.

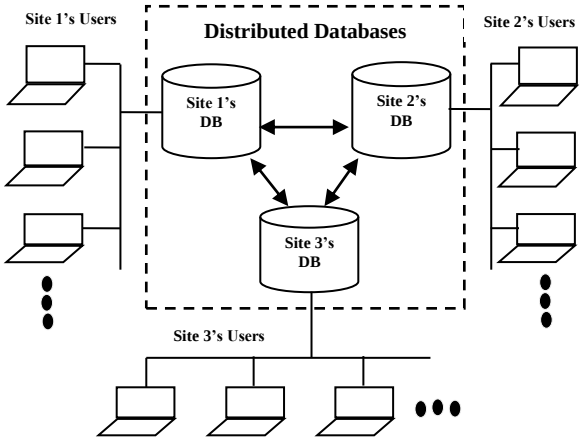

Figure 1. System Overview

4.1 Objective of the Proposed System

The objectives of our thesis are as follows:

- To reduce the work load on single server
- To allows various sites to work autonomously and merge updates into a single, uniform result (To provide a uniform view for multiple replicas of files)
- To distribute database load to different database servers rather than having a single mainframe server
- To build this system as a reliable system

4.3 FDRM Algorithm

Rules for Replica Adding and Deleting

BEGIN

Let :
t = time interval;
r_{in} = the number of local read requests;
$r_{out, j}$ = the number of read requests from $node_j$;
w_{in} = the number of local write requests;
w_{out} =the number of update requests that $node_i$ receives;

[The system has $node_i$ to add a new replica to another $node_j$]

Calculate for before adding replica (BA) ➔$r_{out, j}$
Calculate for after adding replica (AA) ➔$w_{out} + w_{in}$

If (BA>AA) {
Allow to add new replica $node_i$ to $node_j$;
}
Else {
Denied to add new replica $node_i$ to $node_j$;
}
End If

[The system has $node_i$ to delete its replica of f (file)]

Calculate for before deleting replica (BD) ➔ w_{out}
Calculate for after deleting replica (AD) ➔r_{in}

If (AD<BD) {
Allow node $_i$ to delete its replica of f;
}
Else {
Denied node $_i$ to delete its replica of f;
}
End If

END

4.2 Implementation of the System

There are two types of users in this system: admin and normal user. Admin can create account for normal user. And admin can view all users' data (name, owner) and update these data. Admin can create document and load existing document for sites. Admin can view each site's own document and all documents. Admin can view all users' transaction history. Figure 2. describes the system flow of administrator.

All the documents' counters are recorded by normal users' activities. These activities are reading and writing documents. These activities are stored in transaction history table. Figure 3. shows the system flow of normal user.

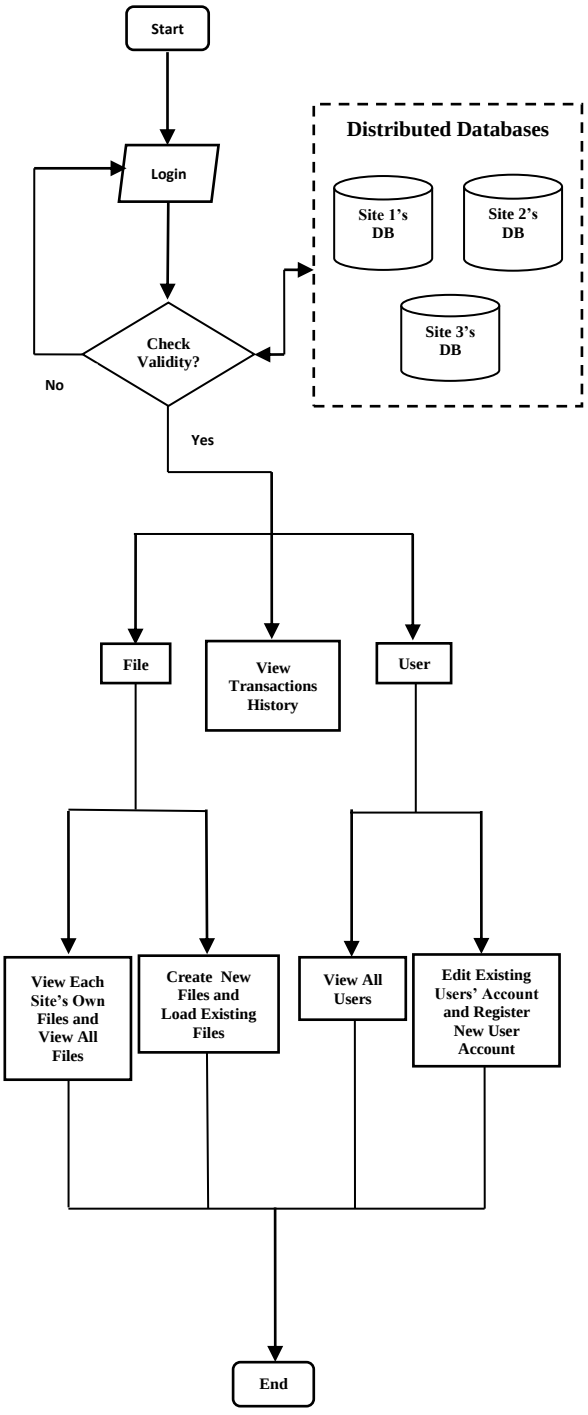

Figure 2. System Flow (Admin)

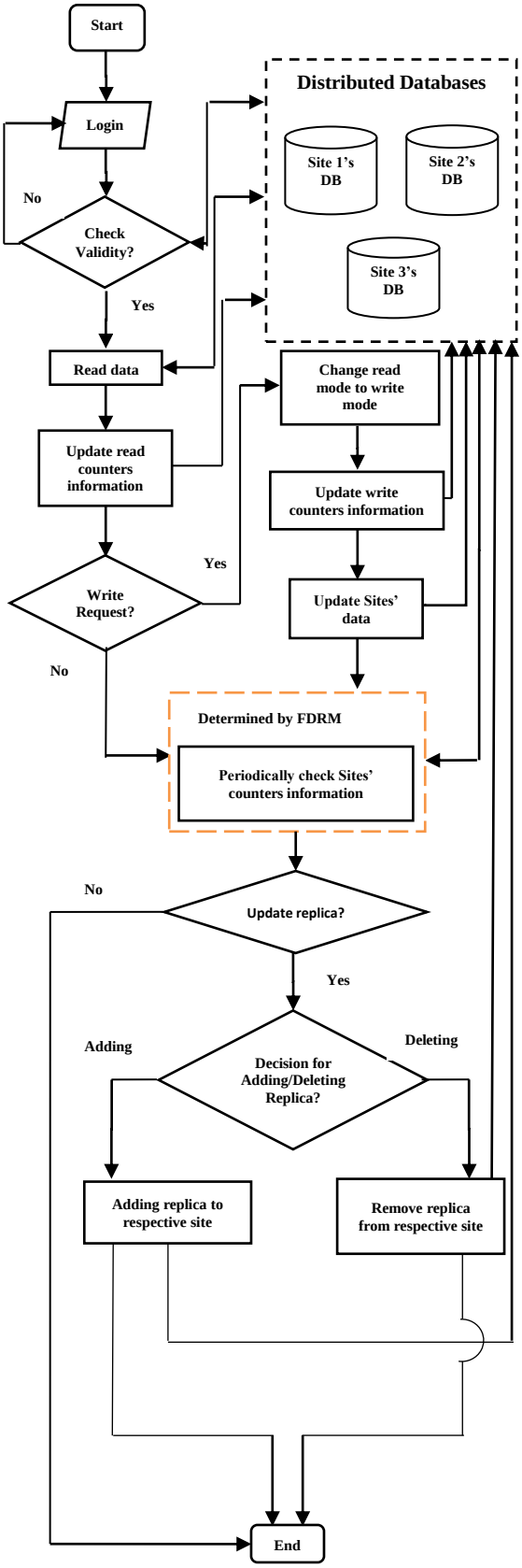

Figure 3. System Flow (Normal User)

5. Conclusions

In this paper, we have explained a decentralized dynamic replication management system depend on accessing frequency detecting named FDRM. In FDRM, system node checks local replicas to detect their access pattern, and settles decision independently to add, delete or migrate a replica. Also, the examining interval of a replica is variable according to the accessing frequency to that replica, which makes FDRM more sensitive to the change of system behavior.

References

[1] *ZHOU Xu, LU Xian-liang, HOU Meng-shu, WU Jin*, "Research on Distributed Dynamic Replication Management Policy", School of Computer Science and Engineering, University of Electronic Science and Technology of China, 2005.

[2] *Han H*, "Studies on Internet Oriented Distributed Massive File Storage", Dissertation of Peking University, 2002.

[3] *Ranganathan K, Foster I*, "Identifying Dynamic Replication Strategies for a High Performance Data Grid", International Workshop on Grid Computing. Denver: Springer-Verlag, 2001.

[4] *Dowdy D, Foster D*, "Comparetive models of the file assignment problem", Computing Surveys, 1982.

[5] *Bartal Y, Fiat A, Rabani Y*, "Competitive algorithms for distributed data management", In Proc. 24th ACM Symp. On Theory of Computing. NY: ACM Press, 1992.

[6] *Cabri G, Corradi A, Zambonelli F*, "Experience of Adaptive Replication in Distributed File Sytems", Proc. of EUROMICRO-22, Prague, 1996.

[7] *Acharya S, Zdonik S B*, "An Efficient Scheme for Dynamic Data Replication", Brown University CS-93- 43, 1993.

[8] *Wolfson O, Jajodia S, Huang Y*, "An adaptive data replication algorithm", ACM Transactions on Database Systems, 1997.

[9] *Ranganathan K, Iamnitchi A, Foster I*, "Improving Data Availability through Dynamic Model-Driven Replication in Large Peer-to-Peer Communities", Proceedings of the Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems. Berlin: IEEE Computer Society, 2002.