Directory Synchronization in Distributed Environment

Qiang Li, Ligu Zhu
College of Computer Science
Communication University of China
Beijing, China
cucliqiang@gmail.com

Wenqian Shang, Saifeng Zeng
College of Computer Science
Communication University of China
Beijing, China
shangwenqian@163.com

***Abstract*—Directory synchronization in distributed environment means making the directories specified in multiple computers under different network to achieve the consistent state. When implement the directory synchronization, there are several changes including concurrent writing from multi-nodes, nodes join and leave frequently and the efficient data transmission. In order to satisfy the high availability, optimistic algorithm is adopted in building distributed system in WAN. Vector Clock algorithm is commonly used in optimistic algorithm to detect confliction, but it`s too complex and lack of flexibility for the join and leave of nodes. An improved algorithm has been announced to resolve the problems for master-slave synchronization mode in the paper. CDC algorithm has been combined into directory synchronization to eliminate redundancy within files. Finally, some rules of conflict resolution have been applied in the paper.**

Keywords-Directory Synchronization, Distributed Environment, Vector Clock, CDC algorithm

I. INTRODUCTION

With the increase of user owned computers or intelligent terminals, more and more users want their data can be synchronization between them. These computers are connected through heterogeneous network, and they are frequently to join and leave the network. In order to satisfy the high availability, eventual consistency is often adopted to synchronized multi nodes in WAN environment. Eventual consistency let data be accessed without a priori synchronization. Updates are not synchronized immediately but propagated in the background, and occasional conflicts are fixed after they happened. The semantic consistency of the user operations is ensured by tentative scheduling and conflict detection[1].

Vector clock algorithm is commonly used to maintain the replications and discover the conflicts. For vector clock algorithm, too many information preserved to manage replications, conflict detection is complex and lack of flexibility for the dynamic addition and deletion of nodes, it is often used in back-end server cluster, for example Dynamo[2]. And it`s not applicable to the system which is composed by the user computers. In this paper, after analysis the mechanism which is used to discover conflict in SVN, an improved algorithm has been approved. For the low bandwidth characteristic of WAN, CDC algorithm[3] has been applied to eliminate redundancy within files.

In the second section, the key issues and the main solutions of data synchronization will be described, at the same time the shortage of Vector Clock algorithm will be deeply exposed. The architecture of system will be expressed in the third section, and the improved Vector Clock algorithm will be also introduced. In part four, the implement of system, duplication chunk elimination in files and the rules of conflict resolve will be discussed. Finally, summary the work we do.

II. RELATED WORK

When synchronize the data in distributed environment, the topology between nodes, changes file discovery, replications management, updates transmission, conflict detection, and conflict disposal are the key issues.

There are three main topologies: Master-Slave, Client-Server and Peer-to-Peer[4]. In Master-Slave topology, all updates must be sent to master firstly, after the master apply the updates, the updates will be resent to the slaves which only take charge of read request. While in Client-Server topology, the updates will be applied on the clients directly, but synchronization between clients is forbidden. The changed data must be uploaded to server, and the other client retrieved the changed data from server. The last one p2p topology, each node is equal to received and sent changed data, in order to archive the consistency a lot of information should be exchanged between nodes.

Two main ways are used to discover changed files. One way is through comparing the state of current file system with last synchronization state; Another is to track the user`s all operations to explore the file changes. In this paper the two methods are combined. A monitoring programmer using the system API has always running to track the user operations. But when the program has been closed, and some files have been modified after that, then the comparing method is used to discover the changes.

Vector Clock (also called Version Vectors) is a method for tracking updates in a distributed system. The version vector is an array of length equal to the number of replicas. Each replica Rj maintains its own vector independently, and tracks in position i the number of updates generated by replica Ri that are known at Rj. In other words, each element is a monotonically increasing counter, and each counter i tracks the total number of known updates generated by replica Ri. Version vectors are compared in a pairwise fashion, matching elements of one vector with the

Published by Atlantis Press, Paris, France.

corresponding element from another and comparing two matched elements. Matching is done by replica identifier so that both replicas compare elements for the same replica Ri. The comparison function(comparing version vectors V1 and V2) has three possible output states:

V1 dominates V2: Each element of V1 is greater than or equal to its corresponding element of V2. When comparing vectors of different lengths, the unmatched elements of V2 must be zero; the unmatched elements of V1 are irrelevant.

V2 dominates V1: The above case with the roles of V1 and V2 reversed.

V1 and V2 conflict: V1 does not dominate V2 and V2 does not dominate V1.

Because the length of the version vector and the space consumed both grow with the number of replicas, the algorithm is lack of flexibility for dynamic joining of nodes. Ratner suggest remove the nodes which loss the distinguish ability, and dynamically save the hot node information[5].

The common ways in changes transmission including: log-based update, push-pull updates and anti-entropy updates which is used in Bayou[6]. The update log has been adopted to define the global order among nodes, while push-pull mode been used in the updates propagation. Facing the conflict disposal challenge, rule-based processing and user intervention are the main methods. In this paper, at the expense of the consistency, the no intervention conflict solution has been proposed.

III. DESIGN AND ALGORITHM

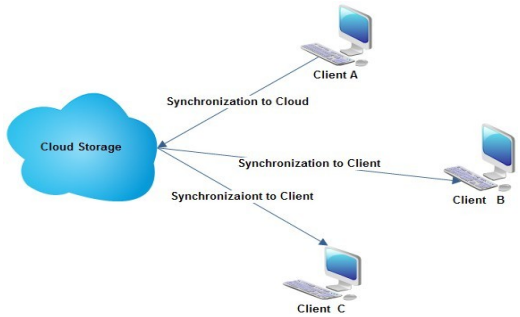

Figure 1. The architecture of System

The wan environment is very complex. Nodes which need to be synchronized are connected through heterogeneous network, and they are frequently to join and leave the network. The Client-Server topology has been adopted, servers which deploy in the cloud storage used to preserve, apply and transfer the copy of changed files. All client nodes are synchronized with the server node to indirectly implement directory synchronization with other nodes, as shown in figure 1.

Push-pull mode is adopted in changed data transmission. By running an agent on the user machine, the directory changes are captured and pushed actively to the server. When the server received the updates, all of them will be preserved. Then determined the order and discovered the conflicts among the updates finally apply the changes and recorded in log. Agent periodically pulls changes from the server, and then takes the initiative to apply to user directory.

The core of system is the improved Vector Clock algorithm which follows four goals. First, preserve the feature which modification can be launched on any node. Second, simplify the detection of conflict. Third, reduce the replication management information. Finally, unrestricted the dynamic node joins.

Above all, discusses the logic of conflict discovery in CVS and SVN. The central node in the SVN is responsible for the allocation of global version for each file. When edit a file, user should download a copy of the file firstly. After editing, upload the file, then the central node will reallocate a global version for the file. The above logic has three features: the version of file on each node is unified managed by the central; Updates take effect in the central node before synchronizes; when the central node applies updates, take the first come first service strategy.

After combine SVN and Vector Clock algorithms, in the improved algorithm, a file is represented as File[Ci,cv,lv].

Ci - the site which update the file

cv – the current version of file

lv – the version which the master node allocated

The initial state: File[x,0,0] both cv and lv are set to zero.

When file changed including delete file, File[x,cv,lv] -> File[x,cv=cv+1,lv], the cv plus 1.

Following, the process of conflict detection and the change mechanism of both version numbers will be explained, during sending updates, applying updates both in master and client nodes.

Client Ci submits updates:

Ci.cv = Ci.cv + 1

Submit File[Ci,cv,lv]

The server applies updates. In the following figure, X means the last update client.

```
1  if  X == Ci
2  {
3      if  X.cv < Ci.cv
4          {   Apply Update;
5              X.cv = Ci.cv;  X.lv = X.cv; X = Ci; }
6      else if  X.cv == Ci.cv
7          {  Ci send a repeat request;  }
8      else     //X.cv > Ci.cv
9          { it's impossible }
10 }
11 else  //X != Ci
12 {
13     if  X.lv == Ci.lv
14         {   Apply Update;
15             X.cv = Ci.cv; X.lv = X.cv; X = Ci; }
16     else if X.lv > Ci.lv
17         { Ci update request is conflict; }
18     else
19         { it's impossible }
20 }
```

Figure 2. Detect conflict on server side

Line 1: For the synchronized file, the last update comes from Ci client.

Line 3-5: Apply the update request and change the corresponding file in server-side.

Line 12-20: For the synchronized file, the last update didn`t come from Ci.

Line 13-15: When the client node and the server have same lv for the file, accept the update request.

Line 16-17: When the client Ci missed the changes from other clients and modified its file, then this update request is conflict.

Client node applies updates. Cm means some site submitted the update

```
1  if Ci == Cm
2  {
3      if Ci.cv < Cm.cv
4          { it`s impossible; }
5      else if Ci.cv >= Cm.cv
6          { Ci.lv = Cm.cv }
7  }
8  else    // Ci != Cm
9  {
10     if Ci.lv <= Cm.lv
11     {
12         if Ci.cv != Ci.lv
13         {  Cm update request is conflict; }
14         else
15         {   Apply update;
16             Ci.cv = Cm.cv; Ci.lv = Ci.cv; }
17     }
18     else
19     {
20         it`s impossible;
21     }
22 }
```

Figure 3. Detect conflict on client node

Line 1-7: Ci receives the update requests which were submitted from its own, which means the master node apply the request from Ci node.

Line 9-23: Received requests from other nodes.

Line 10-17: When the Ci.lv equals to the Cm.lv, the last version of the file are equal both in Ci and Cm. When the Ci.lv is less than Cm.lv, it means Ci miss some update requests. In both cases, if Ci didn`t modify the file before receiving the update, then it accept the update request. If Ci modified the file then it`s conflict.

When the improved Vector Clock algorithm has been adopted, it`s not necessary to preserve version vectors for each object on every node. The relation File (site, current version, last version) is only be preserved. When detect conflict, just compare twice between lv and cv. The new node can join the network anytime, before synchronize the object just initialize it.

IV. SYSTEM IMPLEMENT

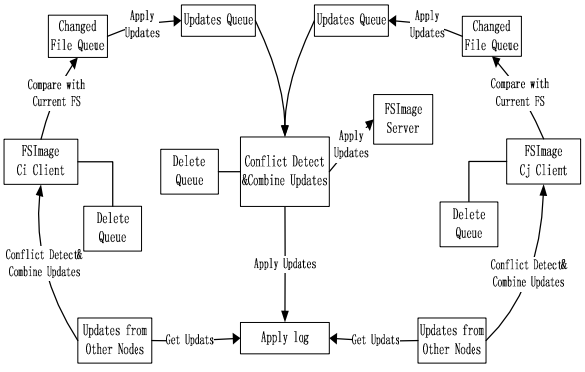

Figure 4. Synchronization between Server and Clients

Directory synchronization in distributed environment, both the directory structure and file contents needed to be synchronized. The directory structure has been extracted from file system and saved to a separate file called FSImage. The changed files can be explored through monitoring file system or compared with the FSImage, all the changed files will be put into changed file queue at local computer. Server maintains a request queue for each client, all changed file will be sent to the queue. The server constantly applies the update requests, modifies the FSImage on the server side, and appends the update log. Agents periodically pull the update log from server, download the changes file contents, and the merge the updates in the local. Continuously repeate the cycle, the directory on multiple nodes will be synchronized eventually. Whether on the server or client side, when combine changes, detects the conflict.

A. Eliminate duplicate data within files

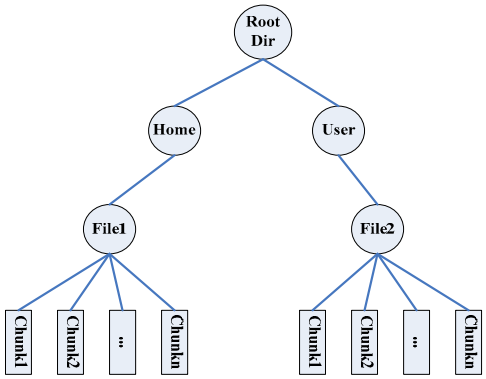

Figure 5. File-Chunks

In this paper, file is no longer regarded as an indivisible whole, but composed of many chunks, which is marked by its content hash. Comparing the file`s last modify time to determine whether the file changed, the use the CDC algorithm to segment the file. The CDC algorithm can identify the changed data and limited the impact of changes on the adjacent chunks. For the files under the same path, more duplicate can be detected between the different versions. The files changes have been translated to the chunks add, delete and update operations. Doubly linked list is used to hold chunks, then all the operations have been converted to the nodes add, delete and update.

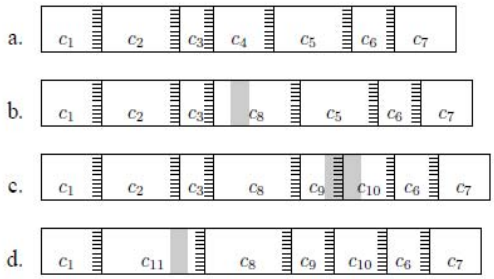

Figure 6. the changes between different file version

Figure6 has shown the four versions of some file after three revisions. Through CDC algorithm comparison, the changed chunks between adjacent versions can be found.

Agent will only transmit the affected data to the server, on the basis of existing file a new file will be generated by the server. The chunk list must be maintained at both client and server, and the historic chunks should also be preserved by server. So the user can quickly obtain the latest contents of the file, and restore the needed historical versions.

B. Conflict resolution

All operations in File System have been divided into follow category: Create Dir, Create File, Del Dir (recursively delete subdirectories and files, but identified as a single operation), Del File, Modify File. Rename and Move operations are recognized as the combination of delete and create operations. Three major conflicts exist among the operations: Current write conflict, Update/Delete conflict, Update/Delete Dir conflict. Both create and modify operations are recognized as update request.

For different systems, the conflict resolution strategies will be different. We will try our best to preserve what the user has done. Despite conflict detection will be carried out on both server and client, but the conflict disposal will be applied only at client node.

1) Current write conflict

When current write conflict was discovered on server side, the first come request will be applied and the other requests will be put into conflict queue for the corresponding node. After the client node gets the conflict files, it will rename its local file and resend the requests.

If current write conflict was detected at client side, it will rename its local file directly.

2) Update/Delete Conflict

Whether Update/Delete Conflict was discovered on server or client side, the conflict needed to be disposal in two ways. One situation is applying Delete operation first and then applying the update operation. Another situation is opposite.

For the server, when taken the first situation, all create and modify requests will be put into the conflict queue for the corresponding node, the client node will rename its local file and resend update requests; when confront the second situation, the delete requests are ignored directly.

For the client node, when encounter the first situation, the node will subsequently carry out the update operation, downloading file from server directly; when confront the second situation, the delete requests will be also ignored.

3) Update/Delete Dir Conflict

Update/Delete Dir means when a node want to update a file, but it`s parent dir has been delete by another node. Update/Delete Dir conflict is the subset of Update/Delete conflict. So the conflict will also be deal with in two cases. First, Delete Dir operation is applied before Update operation. Second, it is just opposite.

For the server, if taken the first case, all create and modify operations will be put into the conflict queue for the corresponding node. The client node will resend Create Dir operations for the delete dir. But if the second case has been executed, the Delete Dir conflict won`t be recognized, and the file will be delete. So the delete queue play an important role for this situation, user can find the delete file from delete queue.

For the client node, when confront the first case, the parent dir will be created automatically for the created or modified file. For the second case, move the dir which needed be delete into a temp dir, and record it into delete queue. It will be useful, when user want to recover the data.

V. SUMMARY

In this paper, deeply analysis the Vector Clock which is commonly used in distributed data synchronized problem to manage replications and detect conflicts. An improved algorithm which combined SVN and Vector Clock has been proposed and achieved three goals: rapid conflict detection, reducing the information required to manage replications and supporting dynamic joining of nodes. Meanwhile, in order to meet the low-bandwidth data transmission in the WAN environment, the CDC algorithm is used to find the redundancy data for the same file of different versions. Finally, renaming the conflicted files and creating the deleted directory are employed as the rules of conflict resolution.

The work was supported by a grant from the National High Technology Research & Development Program of China No.2009AA01A405, and Foundation for 211 Project on Phase II of Communication University of China.

REFERENCES

[1] Yashushi Saito and Marc Shapiro, “Optimistic Replication”, ACM Computing Surveys, Vol. V, No.N,3 2005, Pages 1-44.

[2] Giuseppe DeCandia,Deniz Hastorun, “Dynamo: Amazon`s Highly Available Key-value Store”, SOSP`07 October 14-17, 2007.

[3] Athicha Muthitacharoen, Benjie Chen, David Mazieres, “A Low-bandwidth Network File System”,Proceeding of the eighteenth ACM symposium on Operating systems principles, October 21-24,2001,Banff, Alberta, Canada

[4] Zhou Jing, Research on Data Consistency Maintenance of Massive Data in Peer-to-Peer Distributed Storage Systems.

[5] David Howard Ratner,Roam: A Scalable Replication System for Mobile and Distributed Computing.

[6] Douglas B. Terry, Marvin M. Theimer, Karin Petersen, Alan J. Demers, Mike J.Spreitzer, and Carl H. Hauser. Managing update conflicts in Bayou, a weakly connected replicated storage system. In Proceedings of the 15th ACM Symposium on Operating Systems Principles(SOSP-15), Copper Mountain Resort, Colorado, December 1995.