Content Synchronization Architecture for dissemination of Media Contents between Institutional Hubs and Student Tablets

M.Tech. Stage I Report

Submitted in partial fulfilment of the requirements for the degree of

Master of Technology

by

Gyanaranjan Shial
Roll No : 113050075

under the guidance of

Prof. D. B. Phatak

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

October 2012
Acknowledgement

I am thankful to Prof. D. B. Phatak for providing me direction and insight in this work. I would like to express my deep gratitude to Prof. D.B.Phatak, who has always been making things simple to understand. Without his deep insight into this domain and his valuable time for this project, it would not have been possible for me to move ahead properly. He has been remarkable in his attempt to keep me motivated in this project and has always tried to improve me with proper feedback.

Gyanaranjan Shial
Roll No 113050075
Department : CSE
MTech
Abstract

Technology has become an integral part of our lives. Education with technology is a powerful combination which can revolutionise our society. And with the Aakash tablet slated to be in the hands of every Indian student, new avenues has opened. To provide better solution for transferring file from tablet devices to institutional hubs and vice-versa file synchronisation is the only solution in these days. Transferring file as a whole will take more time and also will cause overhead and will cause traffic in the network when there is small changes between remote device files. And the amount of data stored on the remote device will increases drastically after such type of file synchronisation if the contents are audio, video and image files. To solve this problem this project has found some solutions, when there is some few changes between the files of the two devices, in our case it is all about synchronising files between student tablets and institutional hubs. The solution is subsequently transferring updates and reconstructing files in the remote device to avoid all the above type of problems. This problem is previously tackled by using Rsync. But it is not more scalable in terms of network size and bandwidth usage. The main problem is to synchronise a of large set of media files between institutional hubs, also between institutional hubs and student tablets. The ultimate goal of this project is to save bandwidth and time during the these content transfer.
Contents

1 Introduction                                      1
  1.1 Content Synchronisation in tablet device scenario 1
    1.1.1 Push and Pull Based architecture : 1
    1.1.2 Benefits 3

2 Literature Review 4
  2.1 File synchronisation as a mean of transmitting updated contents to the destination host 4
    2.1.1 Some Application which support Android as the platform:[1] 4
    2.1.2 Some applications that works in android tablet devices: [1] 4
    2.1.3 Rsync 4
    2.1.4 One survey regarding Rsync : Automate the backup of files on remote machines to a centralized server using rsync 5
    2.1.5 Process to test rsync over ssh (with password): 6
    2.1.6 Experimental Result: 7
    2.1.7 Unison 7
    2.1.8 Syncany 8
    2.1.9 Dropbox 8
    2.1.10 HadoopRsync 9
    2.1.11 Design of hadoop cluster 10

3 Problem Formulation 12

4 Techniques of file synchronisation : 13
  4.1 Checksum matching : 13
  4.2 Improved technique over Rsync[5] : 14

5 File synchronisations protocols 15
  5.0.1 HotSync 15
  5.0.2 Intellisync 15
  5.0.3 SyncML 16

6 Solutions for minimizing file synchronization between two host : 17
  6.1 De-duplication: 17
  6.2 chunking or splitting[5]: 17
7 Properties of device synchronisation:

7.1 Scalability: ................................................................. 18
7.2 Cost-Effectiveness: ..................................................... 18
7.3 Data Transmission Load: .............................................. 18
7.4 Computation: .............................................................. 18
7.5 Network Size: .............................................................. 18
7.6 Robustness: ................................................................. 19

8 Observation and Analysis ................................................. 20

8.1 Implementation .......................................................... 20
8.2 Future Work: ............................................................... 20

9 Conclusion .................................................................. 22
# List of Figures

1.1 Shows Push based contents dissemination scenario ........................................ 2
1.2 Shows Pull based contents dissemination Scenario ........................................ 3
2.1 Different phases of file synchronisation between target and source device .......... 5
2.2 Shows comparisons of Rsync with and without compression ............................... 7
2.3 Physical architecture Hadoop. Source[13] ....................................................... 10
5.1 Intellisync Anywhere server on a college network ........................................... 16
List of Tables

2.1 Comparisons of different synchronisation applications ................. 10
2.2 Comparisons of different synchronisation applications ................ 11
Chapter 1

Introduction

The power of technology needs to be leveraged to its true potential in the education sector. For this the Aakash tablet has reached some solution by implementing it in quiz conduct scenario. Transferring the web content over a limited bandwidth is the main challenge for implementing tablet device in student quiz conduct scenario.

File synchronisation is a fundamental operation in distributed system which can solve the above problem. For maintaining large content across different institutional hubs and between student tablets, file needs to be synchronised in an efficient manner. This problem of file synchronisation is called file synchronisation problem. There are many method of file synchronisation between devices. File synchronisation solution are slow sync, fast sync, NFS, Rsync, cloudsync, hadoop-cloud-sync. There are many cloud based solution available to solve this problem. Applications that are available for file synchronisations are such as dropbox, sugarsync, ubuntu one, syncany, Unison etc. The solutions now providing limited amount of space, for dropbox it is 2GB and others are proving 5GB space for free. This type of solution cant overcome the problems for file synchronisation between tablet devices and servers(institutional hubs). This report considers file synchronisation as a mean to minimize the bandwidth usage and fast file transfer. For this we considered fast sink as the best method of file transfer which Rsync uses. Here in case of first sync the difference of the files will be transferred to the target host. But in case of slow sink file will be transferred totally even if the there is similarities between the two files.

1.1 Content Synchronisation in tablet device scenario

Tablet device has no internal hard drive to store data. For this some cloud based solution is best for synchronising contents between devices. For avoiding this storage constraint institutional hub is the best solution. And the architecture for implementing the tablet device is by introducing one or more institutional hubs across the the institution. So that the contents can be stored for quiz conduct and other applications scenario.

Synchronisation of contents is necessary when tablet user want to synchronise the study materials, quiz questions, synchronising audio and video files with the central servers.

1.1.1 Push and Pull Based architecture :

For content dissemination push or pull based architecture is followed. Here, When the professor puts quiz material on the quiz portal then it will be pushed to the registered student’s tablet. In some scenario when student wishes to view previous questions then he can pull it from the quiz
portal at any time. Figure 1.1 shows push based content synchronisation and figure 1.2 shows pull based content synchronisation.

Figure 1.1: Shows Push based contents dissemination scenario

- publisher: Professor
- Channel: Examiner
- Event Subscriber: Student Tablet
In push based content synchronisation, Professor will update the quiz material and stores it in the repository. Then the examiner will open the portal which will disseminate the content to the registered student tablets.

In pull type content dissemination method, the events that takes place are as follows. The Students request to the channel for already existing files. And channel get back with the existing files.

![Pull based contents dissemination Scenario](image)

Figure 1.2: Shows Pull based contents dissemination Scenario

1.1.2 Benefits

- Students can access the contents i.e quiz materials and any PDF files, web page easily by using their own tablet devices.

- In quiz conduct scenario when the professor wants to add some other content to the already existing file, then no need to transfer the whole file, rather to transmit only the file difference to the tablet devices and also simultaneously reflect the changes to the repository.
Chapter 2

Literature Review

Related work:

2.1   File synchronisation as a mean of transmitting updated contents to the destination host

2.1.1 Some Application which support Android as the platform:[1]
Dropbox, Google Drive, Box.com, ownCloud, Syncplicity, Yandex.Disk, Twindocs, JustCloud

2.1.2 Some applications that works in android tablet devices: [1]
Google Drive, Minus,FrostWire, BitTorrent, Cloudfogger, Sharefile, Zoolz, Alfresco in the Cloud, LifeCloud 247, FileGator, RES HyperDrive.

2.1.3 Rsync
Rsync is an open source software application, originally written for Unix systems, but now also running on Windows, Mac and linux platforms. It first checks whether any data has changed by looking at the file size and modification date. Rsync can cater for inserted or added data, removed data as well as shifted data, with a minimum transfer overhead. In this way Rsync provides better solution than other method of file synchronisation. All that need to do for file synchronisation is to transfer the file from one location to another with a minimum cost and in less time by exploiting the significant similarity between the two files located at different location.

The above scenario arises in a number of file synchronisation process, like synchronisation of user’s files between different machines, remote backups, mirroring of large web and FTP sites, content distribution, and web search engines. Rsync solves this problem by creating file and block lists of both local and remote file system and compares them. Then transfers the changed blocks in a single batch per direction. File contents changes is detected by using a rolling checksum followed by a strong checksum only if a matching is found.
2.1.4 One survey regarding Rsync: Automate the backup of files on remote machines to a centralized server using rsync

rsync is a command line utility that is used to synchronize files between two computers over a network to synchronize files between two file systems. It was written as a replacement for rcp but with many new features. For example it uses an algorithm that will only transfer files that have been modified. SSH will be used to authenticate between the machines and to encrypt the network traffic.

For content synchronisation I have tried some experiments on Rsync. Here all that required for this experiment are using two computer devices. Aim of experiment to synchronise the contents between two computers.

Pre-requisites:

- Two computers
- Both computers need to have rsync and ssh installed.

Commands used for Rsync content synchronisation

- rsync -avz --progress --log-file=the.log -e ssh --delete shial@10.129.26.86:/home/shial/nf/ /home/gyana/Desktop/

Flags used for rsync in this experiment:
- rsync -avz --progress --log-file=the.log -e ssh --delete shial@10.129.26.86:/home/shial/nf/ /home/gyana/Desktop/
• -a or –archive

-a / –archive is a quick way of saying you want recursion and want to preserve almost all attributes of the source files (with -H being a notable omission).

Note that -a does not preserve hard links, because finding multiply-linked files is expensive. You must separately specify -H.

• -v or –verbose

This option increases the amount of information you are given during the transfer (rsync works silently without it). A single -v will give you information about what files are being transferred and a brief summary at the end. Two -v options (-vv) will give you information on what files are being skipped and slightly more information at the end. More than two -v options should only be used if you have insomnia.

• -z or –compress

With this option, rsync compresses the file data as it is sent to the destination machine, which reduces the amount of data being transmitted – something that is useful over a slow connection.

Note that this option typically achieves better compression ratios than can be achieved by using a compressing remote shell or a compressing transport because it takes advantage of the implicit information in the matching data blocks that are not explicitly sent over the connection.

However, it’s worth noting that some file types will not be compressed during transfer. The default list of file extensions that will not be compressed is: gz zip z rpm deb iso bz2 tbz tgz 7z mp3 mp4 mov avi ogg jpg jpeg.

• -e ssh: Specify the remote shell as ssh

• –numeric-ids: Tells rsync to not map user and group id numbers local user and group names

• –delete: Makes server copy an exact copy of the source by removing any files that have been removed on the remote machine

• shial@10.129.26.86:/home/shial/nf/: The remote machine name, then the directory to be backed up

• /home/gyana/Desktop/: The directory to place the backup

2.1.5 Process to test rsync over ssh (with password):

When I started Rsync over ssh it asked for remote desktop password. To avoid this problem in case of automatic backup from any pc to server can be solved by the following steps.

• step1: key generation command : ssh-keygen when it ask for password press enter. After executing this command it will generate two files in .ssh directory in your PC.

• step2: ssh-copy-id copies public key to remote host command : ssh-copy-id -i ~/.ssh/id_rsa.pub shial@10.129.26.86.

The above will ask for the password for the first time and will copy the public file(rsa.pub) to the destination to the appropriate folder. And next time for any ssh operation, it will not ask for password. And file synchronisation will take place smoothly.
2.1.6 Experimental Result:

![Content synchronisation using Rsync over ssh with and without compression](image)

Figure 2.2: Shows comparisons of Rsync with and without compression

The experiment figures out that during Rsync file synchronisation the whole file don’t get transmitted to the destination rather the incremental change will be transmitted to the destination.

The second experiment shows the difference of file size after using file compression techniques. Here the orange colour shows total content size in the source device after synchronisation. Green colour shows the amount of data transmitted in with compression i.e file difference and deep blue colour shows same size content transmitted without using compression. With this experiment the result shows transmitted content size is nearly half after using compression with rsync.

**The key benefits of Rsync:**

- Improves offsite backup speed through bandwidth optimization.
- Reduces network data transfer by transferring only new data.
- Open standard protocol for maximum compatibility and flexibility in choosing a backup destination.

**Drawbacks:**

- Frequent updates of small files can not be resolved by rsync efficiently.
- It synchronises files between two hosts.
2.1.7 Unison

Another File synchroniser is Unison. It is similar to Rsync but it uses bidirectional synchronization algorithm. It follows two method for file synchronisation, one is to update detection and second reconciliation. In the first phase it will detect update based on file modification time and inode numbers. It marks a file dirty if one of the property change based on the previous synchronisation run. Unison also relies on rolling checksum to check part of the file has changed.

limitation of Rsync and Unison:

- In these two type of file synchronisation processing power is required in both end.
- Synchronizers such as rsync or Unison, for instance, completely lack any kind of version functionalities. Version control systems such as Git or Subversion on the other hand only support a few storage types and entirely lack encryption.

2.1.8 Syncany

For file synchronisation syncany uses some more technique which Rsync did not overcome. For this Syncany tries to use de-duplication techniques and chunking technique to reduce the bandwidth usage. By de-duplication technique, it tries to minimize the storage and synchronisation time. And by using multi chunk technique it tries to minimize the number of request arriving to the server. For this it tries to reduce the file size before uploading to the server end.

The advantages of de-duplication is, it enables version, minimizes disk usage on remote devices. The pitfall of this technique is it is very CPU intensive. The main problem identified by syncany is to find a suitable de-duplication algorithm. In this technique it tries to find out duplicate sequences of bytes (chunks) and only stores one copy of that sequence. If a chunk appears again in the same file (or in another file of the same file set), de-duplication only stores a reference to this chunk instead of its actual contents. Depending on the amount and type of input data, this technique can significantly reduce the total amount of required storage.

Here the de-duplication technique takes the advantages of chunking method, by which when the chunk size decreases to low size then the number of duplicate chunk increases. So the method used for chunking largely influences the de-duplication success. Using multi chunks, the de-duplication ratio can be further increased and the synchronization time has shown to be reduced significantly. It reduces the total synchronization time between Syncany clients.

2.1.9 Dropbox

Dropbox[7] is mostly used now a days than others cloud based file synchronisation and it operated by operated by Dropbox Inc. It has a variety of features among all cloud based synchronisation application. While Dropbox functions as a storage service, its focus is on synchronization and sharing. The main features are it stores and shares file among any number of users over the internet. Dropbox provides client software for Microsoft Windows, Apple Mac OS X, Linux, for Google Android, Apple iOS, and Blackberry OS, and web browsers.

- Faster file synchronisation capability.
- Stores file in folders and provides all the facility that a folder can do like drag and drop of files and folders. And then synchronises all these changes with all the registered devices and also with the associated email account.
- Files placed in this folder are also accessible through a web site and mobile phone applications.
- No one has access permission except the shared files and folders.
- supports multi-user version control, enabling several users to edit and re-post files without overwriting versions.

Dropbox uses an rsync-like chunking algorithm and only transfers binary diffs of updated files to save bandwidth and remote storage. Rsync and Unison are used for synchronising files between devices on demand. Due to the significant overhead required for the generation of file lists, they are not suitable. To overcome this problem Dropbox has solved this by using a watch listener in the operating system, which react when there is a change in the file list. Since every file update is registered by the software, generating file lists is easy. For storage purpose Dropbox uses Amazon S3.

It also provides a solution for file synchronisation known as LANSync, which allows users to synchronise the content between the computer devices within the LAN along with synchronising contents with the web server.

The main limitation of this application is that it is not so secure, one don’t know who is having the shared file because anyone can share the shared file to any others. So organisations may suffer from information leakage to outside.

2.1.10 HadoopRsync

Hadoop is an open source framework, it operates on thousands of nodes, petabytes of data and has been deployed on large number of users using simple programming model. HDFS[2] architecture is used to synchronize files in hadoopRsync. HDFS maintains and distribute multiple replicas of data blocks throughout a cluster to offer reliable and efficient computations.

HadoopRsync[13] uses the technology i.e mapreduce for file synchronisation to parallelise the file manipulation at multiple node. It has one namenode and several datanode. HadoopRsync consists of pair of algorithm to synchronize the files i.e hadoopRsync Upload and hadoopRsync Download. When a copy on users device has been changed then hadoopRsync Upload algorithm is used. And in other case when some changes need to be updated to users device then hadoopRsync Download is used with the Hadoop-end new copy. During update from the users device to the hadoop end is as follows.

- step1: The Hadoop end send the checksums of all the file block stored on multiple node to the users device.
- step2: An encoding of the update is calculated by and send from the users device to the hadoop end.
- step3: The algorithm used on hadoop end use this encoding to convert it into update instructions.
- step4: Then the update is carried out accordingly based on the update script

Similarly for HadoopRsync Download steps are as follows.

- step1: The checksum from the users device is calculated and send to the hadoop end.
• step2: The hadoop end then distributes those checksum to all datanode where the difference between the new version and old version are calculated in parallel.

• step3: Then the difference are gathered together and and an update script is composed and sent to the users device.

• step4: Then the update is carried out accordingly based on the update script

Finally to speed up the process of file comparisons in hadoopRsync is more effective than others by using the parallelism feature of hadoop.

**HDFS**

For storing large amount of data across Hadoop clusters. The Hadoop cluster devides the file into blocks and it tipically replicates every blocks three time across different datanodes on more than one rack. The namenode maintains a filesystem namespace including file system tree and meta data for all files and folders in the tree. And the datanodes stores the file in terms of blocks. The architecture that HadoopRsync follows is a master-slave pattern: a namenode and many datanodes. The above figure describes the hadoop architecture.

![Figure 2.3: Physical architecture Hadoop. Source[13]](image)

2.1.11 Design of hadoop cluster

The physical architecture is quite simple, for the purpose of storage we need a hadoop cluster. The application server is responsible for all communication with the remote user client and takes charge of hadoop cluster by submitting map-reduce tasks and calling the HDFS interface.
Limitation of HadoopRsync

HadoopRsync doesn’t provide solution for failure of namenode of HadoopRsync.

Table 2.1: Comparisons of different synchronisation applications

<table>
<thead>
<tr>
<th>Technology</th>
<th>De-duplication</th>
<th>Compression</th>
<th>Versioning</th>
<th>client-side encryption</th>
<th>secure data transfer</th>
<th>multichunk</th>
<th>licence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rsync</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>open-source</td>
</tr>
<tr>
<td>syncany</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>open-source</td>
</tr>
<tr>
<td>Unison</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>open-source</td>
</tr>
<tr>
<td>Dropbox</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>proprietary</td>
</tr>
<tr>
<td>HadoopRsync</td>
<td>possible</td>
<td>possible</td>
<td>possible</td>
<td>possible</td>
<td>possible</td>
<td>possible</td>
<td>open-source</td>
</tr>
</tbody>
</table>
### Table 2.2: Comparisons of different synchronisation applications

<table>
<thead>
<tr>
<th>Applications</th>
<th>Techniques used</th>
<th>Features</th>
<th>limitations</th>
<th>update detection *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rsync</td>
<td>1. peer-to-peer. 2. checksum matching i.e rolling checksum and MD5 checksum. 3. file compression</td>
<td>1. Compressing file before transmission 2. Easy update detection using total file size or modification time</td>
<td>1. file synchronisation between more than two hosts is not available. 2. Frequent updates of small files cannot be resolved by rsync efficiently.</td>
<td>1. based on file modification time and file size. 2. rolling checksum 4. MD5 checksum.</td>
</tr>
<tr>
<td>syncany</td>
<td>1. Checksum matching. 2. de-duplication and chunking technique. 3. multi chunks algorithm</td>
<td>1. bandwidth optimization. 2. maintains versions. 3. minimises storage size using de-duplication</td>
<td>1. It synchronises files between two hosts.</td>
<td>1. More processing power required for maintaining de-duplication technology.</td>
</tr>
<tr>
<td>Unison</td>
<td>1. uses bidirectional synchronisation algorithm 2. file synchronisation method: update detection and second reconciliation 3. file difference based on file modification time and inode numbers</td>
<td>1. bidirectional synchronization</td>
<td>1. Processing power is required in both end. 2. Processing power is required in both end. 3. Not so secure. 4. lack of version functionality.</td>
<td>1. based on file modification time and inode numbers. 1. rolling checksum</td>
</tr>
<tr>
<td>Dropbox</td>
<td>1. client-server architecture. 2. chunking algorithm. 2. Introduced watch listener in the operating system. 3. LANSync</td>
<td>1. any number of users. 2. synchronising files between devices on demand. 3. LANSync. 4. Maintains versions</td>
<td>1. information leakage to outside because of insecure file sharing. 2. chunking algorithm and only transfers binary diffs.</td>
<td></td>
</tr>
<tr>
<td>HadoopRsync</td>
<td>1. Client-server architecture. 2. map-reduce functionality. 3. checksum matching. 4. hadoopRsync Upload and hadoopRsync Download</td>
<td>1. Storage of big data. 2. faster file comparison operation using map and reduce operation provided by hadoop. 3. fault tolerance 4. More number of device can participate single file synchronisation.</td>
<td>1. No solution for namenode failure. 1. checksum matching using parallelism features of hadoop.</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 3

Problem Formulation

We are living in the data age. Large amount of data of the order of Terabytes, Exabytes and Zettabytes are of contents are transferred from one device to another. The bad news is that the network bandwidth provided by internet providers is limited. It will take long time to transfer all files among different computer system. The immediate solution is to synchronise the files in a better way in the clusters of computer system which will consume less bandwidth and will give faster file synchronisation operation.

The main problem of file synchronisation is to transfer media files in a bandwidth efficient manner. Apart from this file need to be transferred in small time, so that client should not wait for file transfer to happen. There is a very wide research on this topic in Rsync[7]. For this we need to develop some synchronisation technique between institutional hubs, so that files can be transferred from any remote location to the institutional hubs in less time and in bandwidth efficient manner. And simultaneously the files need to be transferred to the student tablets with all the required privilege. After all the aim of the project is to transfer the updated contents to the student tablets and corresponding institutional hubs and deliver content like quiz materials, video lectures, class notes, application software along with collecting student response with less effort and in less time.
Chapter 4

Techniques of file synchronisation:

A common method is to send just the differences between A and B down the link and then use this list of differences to re-construct the file. The problem of file synchronisation is easier if all the updates are logged in the log file that can be transmitted to the remote device. But problem here is that not all devices can maintain log file for all files. And it will become critical to maintain separate log file if the number of files are very large. To check which file is modified in a particular time one can check the modification time of the file. Based on the modification the recent modified file is taken as the latest file to be transferred to the remote location. But the problem is not exactly the same, we have to consider the modified content in a particular document i.e. if it is a content like google spreadsheet where the updated content should be transferred rather than the whole file. Some other techniques are proposed in Rsync are as follows.

4.1 Checksum matching:

In this technique the difference between the two files are calculated and delta is transferred to the other devices. In Rsync two checksums are required to compare the files between two remote devices. The first one is a week checksum and second one is strong checksum. The week checksum is a rolling checksum which is fast to calculate from the previously received checksum. The strong checksum is a MD4 checksum and it will cost more than calculating a week checksum. So the protocol here first checks the matching in the week checksum and when it finds any matching, then it will use the strong checksum for confirmation.

When a match is found, the source machine will send the file offset, previous match and followed by the index of block of target machine that matched. If no match is found at particular offset, then the offset is updated to next position of the file and search proceeds. If match is found then search proceeds further at the end of the match block. This technique saves considerable amount of computation for matching file when the two files are nearly identical.

The above process will take more time for checking if there are a number of files. The above problem can be solved by using pipe-lining the processes for considerable latency advantages.
4.2 Improved technique over Rsync[5] :

In Rsync the client sends hashes to the server, where in file synchronisation framework[5] the hashes are transmitted from server to client. File synchronisation framework tries to find out the difference of file using Map Construction[5] and delta Compression techniques.

Map Construction

Here according to map construction technique a map is maintained in server side. Suppose the server has a file $f_{\text{new}}$ and the client has file Fold, then the server will send the hash to the client and the client will compare the hashes with its own map file. For example server S has a file which contains abcxyzbbb and the client C contains a file whose contents are pppxyzaaa, then C can create a map of the $m_{\text{new}}$ that looks as follows: ???xyz???, and transmit it to S. Then C now can send 010 to server, based on that server will send the missing part to C. Here 1 means block exits and 0 means block not exist.

Delta Compression

Then the server will create a $F_{\text{ref}}$, consisting of all part of $F_{\text{new}}$ that are unknown to the client and based on that it will create a $F_{\text{target}}$ and tries to compress the file by referring to $F_{\text{ref}}$, then send it to the client. The client uses its map to recreate the file and merge it to create the new file $F_{\text{new}}$. 


Chapter 5

File synchronisations protocols

Content synchronisation protocol is used for identifying changes, resolve possible conflicts, and propagate updates to the various synchronizing devices. Moreover, Aakash tablets have limited CPU, memory and power resources and, thus, are unable to quickly process or transfer large amount of data. In a networked setting, scalable and efficient data synchronization protocols are essential for data intensive applications such as e-mail editors, spreadsheets, and even collaborative work environments. Synchronising content with just one desktop machine to is easy, but here we give more importance to synchronising contents over multiple devices i.e multi-device synchronization. The motive of designing protocol is for addressing the scalability issues inherent to synchronizing data over large, heterogeneous, tether less networks.

5.0.1 HotSync

- Slow sync: In this technique, when there is a change in the new file and the old file between the two devices, the new file will be transmitted to the target devices and synchronisation is finished afterwards. This method of file synchronisation will consume more bandwidth as well as time. Slow sync will give more latency time and more bandwidth usage when the size of content is very high. It may stop synchronising in some point of time due to network bottleneck.

- Fast Sync: In this technique, the difference between the two files will be transmitted if the files are similar in some part. This is useful when more users are editing a single file and the change in the all the documents are transmitting to the central device, here it is a institutional hub. And by this technique the bandwidth usage in comparatively low than slow sync. Many more synchronisation applications use this technique for file synchronisation with some extra modification to it. This needs a previous synchronisation with same device means synchronising contents with last synchronisation. First sync doest give efficient solution when a number number of devices want to change the content to same database, for this we need some efficient protocol for data synchronisation among devices.

5.0.2 Intellisync

It maintains a client server architecture where all devices don’t use peer to peer connection, rather devices who had an update information need to synchronise with the Intellisync Anywhere server in the network. And Intellisync server will synchronise with the microsoft exchange server, not periodically but within a certain time duration. This synchronisation between Intellisync and Microsoft
exchange server is controllable. The problem with this architecture is client-server architecture where server is namely a central point of failure and communication, and a lack of scalability with increasing numbers of clients. The central point of failre can be mitigated by introducing several servers, but without introducing extra work of synchronising the servers. The performance of Intellisync is very similar to fast sync.

Figure 5.1: Intellisync Anywhere server on a college network

5.0.3 SyncML

SyncML is open industry standard for file synchronisation. In this type of file synchronisation protocol every device will maintain some status flag for each record. During modification of the file will toggle the status flag with respect to every other device in the network. It different from fast sync of Hot-sync protocol by maintaining a set of status flag across a set of devices. Modifying a set of file in a device will toggle not only the status flag of that device but also the status flags of other devices. When two devices finishes synchronisation then the status flag of these two devices will be cleared. Maintaining a set of status flag, which is a substantial amount of memory for a portable device. The amount of memory needed by a network of n devices each with r records would be roughly n*r units per device.
Chapter 6

Solutions for minimizing file synchronization between two host:

Synchronising contents over two devices is extremely important because its efficiency has a multiplicative effect on the overall efficiency of synchronizing the network.

By reducing the file size, the amount of requests can be reduced significantly and thereby speed up the upload (and download) of a certain file set. Though the size of the server is limited, so by introducing a concept of reduction on the number of contents in the remote device can also solve the problem by eliminating number of request to the server. De-duplication is a valid technology for end-user applications if controlled properly.

6.1 De-duplication:

By using fast sync the difference between the files will be sent to the server or any device. In some cases the chunks of the files may be similar because of similar file format or similar operating systems used to create the file. In that case before uploading the bunch of chunks one need to reduce the size. So one technique is reducing the number of similar block to a single block and let the other refers to this block, it can reduce the size before uploading the files. The similarity between the file chunks depends on the size of the chunks. The less the size of the chunks, the more the similar chunks we have. This is used by Syncany[6]. So experiment can be conducted for finding the size of chunk size for getting maximum similarity between the files.

Synchronising large replicated collections of file over slow network can be done by some more techniques that is described in Improved file synchronization techniques[10].

6.2 chunking or splitting[5]:

In this technique file is divided into parts, if similar part is found then the splitting is stopped there. Otherwise recursive splitting is carried on that unmatched blocks by which communication cost will be minimised. Choosing the exact size of block when to stop splitting is a related proposal over rsync.
Chapter 7

Properties of device synchronisation:

7.1 scalability:
All the cloud based solutions are more scalable than other methods of file synchronisation. Slow-sync is less scalable than other methods of file synchronisation. Scalability is also a major advantage of cloud storage, because it appears to the users that storage capacity is unbounded.

7.2 Cost-Effectiveness:
HadoopRsync will cost more in the initial stage as expected. But it is more effective afterwards. Cloud storage needs low maintenance cost, so more cost-effective than other.

7.3 Data Transmission Load:
It is the amount of data transmitted between any two devices. It is an important metric because it directly affects the time needed for data transmission. This metric will affect directly to the willingness for using the particular synchronisation application.

7.4 Computation:
In slow-sync no more computation is required during file synchronisation. It is less important if the files are synchronized between different institutional hubs in our case. And is more important when file is synchronised between student tablets and institutional hubs because the processing power in student tablet (Aakash tablet) is comparatively low than a dual-core or core-to-duo processor.

7.5 Network Size:
Among all the techniques of file synchronisation HadoopRsync is more scalable in network size than other methods of file synchronisation. Slow sync network is least scalable in network size.
7.6 Robustness:

In this case HadoopRsync failure of the namenode can’t be recovered easily. In this case HadoopRsync is less robust than other methods of file synchronisation. In other cloud based methods where file is synchronised based on client-server architecture, the failure of server device is less robust than simple file synchronisation methods.
Chapter 8
Observation and Analysis

Rsync has the limitation of synchronising the contents with any two devices. Rsync doesn’t consider
de-duplication and compression technique. Because of limited bandwidth between institutional hubs
and student tablets, it can be implemented between them where during online class quiz conduct
scenario the professor wants to change some contains to the already sent question files to the student
tablets.

And other cloud based methods of file synchronisation are though have reached to some extend
of user’s need during synchronising contents. Considering all the techniques of file synchronisation
hadoopRsync is most suitable because of it scalability in terms network size and robustness in terms
of datanode failure. This project can be implemented in tablet device where the hadoopRsync
architecture can be used in the scenario : the remote node wants to transfer some updated portion
of the file to the institutional hub.

8.1 Implementation

For implementing hadoopRsync in Aakash tablet we need to establish a hadoop architecture inside
the campus where we need to store the data for all file operation. And for this we need to use
some low cost computer for making them datanode. And one special computer will high processing
power to make it a namenode and approximately 6 tablet devices for conducting the experiment in
hadoop cluster. And all need to be able to connect to the internet connection.

8.2 Future Work:

There is also a clear interest in file transfer in between student tablet devices, institutional hub and
in between institutional hubs. This can be solved by implementing hadoopRsync in hadoop cluster.
And file synchronisation between tablet devices can be solved by using rsync over the institutional
hub.

• Building an application for disseminating contents between student’s tablet and teacher’s PC.
• Implementing Hadoop cluster in tablet architecture.
• Managing group between registered students tablets for peer-to-peer content sharing.
• Experiment can be conducted for finding the size of chunk size for getting maximum similarity
  between the files.
• Providing secure content transfer protocols.
Chapter 9

Conclusion

Synchronising the contents over the wide area network can be solved by all the above type of file synchronisation techniques. But based on various features of different synchronisation techniques we have concluded that hadoopRsync is more robust, scalable in network size and faster operation. So using this technology we can connect to a number of student’s tablet at a time and delivering updated contents within a fraction of time to all tablet devices. With the help of this synchronization system, people can store much more useful documents, images and video/audio files to the network-based storage, without agonizing by the problem of keeping synchronization of content between different devices.
Bibliography


