ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 1, Issue 8, December 2016

Online Big File Storage Mechanism

[1] Ms.Noopur Adarsh Purwar [2] Prof.Nilima Nikam [3] Prof.Vaishali Londhe
[1][2][3]Yadavrao Tasgaonkar Institute and technology Bhivpuri, Karjat

Abstract:-- **Online data storage refers to the practice of storing electronic data with a third party service accessed via the Internet. It is an alternative to traditional local storage (such as disk or tape drives) and portable storage (such as optical media or flash drives). It can also be called "hosted storage," "Internet storage". One of the biggest benefits of online storage is the ability to access data from anywhere. As the number of devices the average person uses continues to grow, syncing or transferring data among devices has become more important. Not only does it help transfer data between devices, online data storage also provides the ability to share files among different users. This paper presents the different techniques which help in making online storage an easier way of storing data. Data deduplication is one of important techniques for eliminating duplicate copies of repeating data, and has been widely used in online storage to reduce the amount of storage space and save bandwidth. To protect the confidentiality of sensitive data while supporting deduplication,we have also proposed an encryption technique which helps in encrypting the data before storing it. Our approach to deduplication ,encryption and compression in online storage aims at reduction in storage space and efficient use.**

Index Terms:— **BFS Big File Storage, Compression, Deduplication, Encryption**

I. INTRODUCTION

Any and every computer user knows that backing up data is absolutely vital to maintaining the integrity of their files and folders. Users who do not back up data appropriately are highly susceptible to losing access to important information due to hardware failures, file corruption, viruses, disasters, accidental deletions, and even theft. When this happens, many people are often devastated by the loss of important and unrecoverable data. Despite having this knowledge, many computer users still fail to backup their data regularly, or they simply rely on intermittent backups to USB and thumb drives. Today, there are more backup solutions than ever before, including online backup services that allow you to store your data, so there is little reason to forego routinely backing up important data. This paper focuses on the infrastructure services dealing with storage and network usage. Deduplication and compression, which are described in detail later, are some of the important data optimization services that the online storage system offers [9].

We have proposed an architecture wherein there is a chunk storage mechanism which leverages parallel computing capabilities of the underlying hardware. That make it easier to store data and scale-out the storage system. We have developed a deduplication mechanism to avoid storage of duplicate files by differentiating them on the basis of their content rather than their name.We also propose usage of a compression algorithm to efficiently store chunks of a file thus managing and utilizing storage space efficiently. A storage based on the proposed architecture helps in saving a lot of storage space. Data deduplication is a specialized technique for eliminating duplicate copies of repeating data in storage.[2] Deduplication techniques are widely employed to backup data and minimize network and storage overhead by detecting and eliminating redundancy among data.[4]

1.2 Contributions in this research

In this paper ,we propose an online storage mechanism which helps in storing big file data securely and efficiently. We propose the following:

∑ A light-weight meta-data design for a file. Every file has nearly the same size of meta-data. BFS has O(1) space complexity of meta-data of a file, while size of meta-data of a file in Dropbox , HDFS has space complexity of O(n) where n is size of original file[1].

∑ Propose a chunk storage mechanism which leverages parallel computing capabilities of the underlined hardware. This allows faster storage and processing of data as compared traditional sequential processing of data.

∑ A deduplication mechanism across users of the system. This is based on the actual content of the file rather than the name used for the file.

∑ Propose usage of a compression algorithm to compress chunks of a file.

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 1, Issue 8, December 2016

∑ An encryption algorithm which helps in encrypting the data being stored. This keeps the data secure and confidential.

1.3 Organization

This paper is organized as follows. In Section 2, we present the system model of big file storage system. The development is presented in Section 3. The results are presented in section 4,.Finally, we draw our conclusions in section 5.

II. SYSTEM ARCHITECTURE

The system architecture of the system is as shown in figure 1.

Shown Below is the diagram of Online Big File Storage Mechanism(BFS).A user uploads files on the server through its web interface.As soon as the file /files are uploaded,the file goes through a series of mechanism like deduplication,compression ,encryption and metadata design.The file spilts into chunks and a meta data is created for that particular file.The working of the modules is explained with the help of diagrams in the below sections.

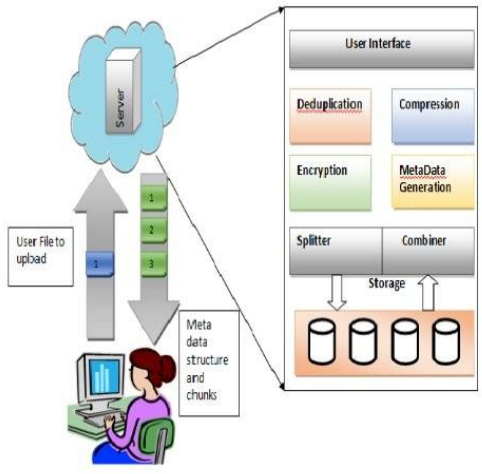

Figure 1: Online Big File Storage Mechanism

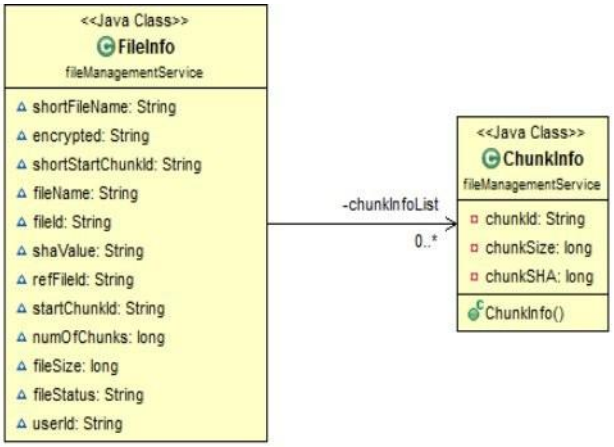

Figure 2: ChunkInfo based metadata design

Figure 3: Deduplication based on ref FileId

As seen in figure 1. a user/users select a file /files for uploading to the Big File storage system. The file goes through a set of processes before getting stored in the underlying storage. First, the metadata is generated for a file. Next the file is checked for duplicacy. It is then encrypted and split into chunks of predefined size.Each chunk is then encrypted. Finally each chunk is

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 1, Issue 8, December 2016

compressed and stored. This is the approach followed in the BFS system.

The figure 2 shows the metadata design of existing online storage systems available in the market. In this metadata design as we can see along with FileInfo, another datastructure ChunkInfo is created. FileInfo stores the metadata of a file and ChunkInfo stores metadata related to the chunks of a file.

The figure 3 shows improved metadata design of BFS wherein the need for ChunkInfo datastructure is eliminated resulting in improved storage.

III. DEVELOPMENT

The following figures depict a basic use case of file storage and retrieval performed using the Big File Storage system.

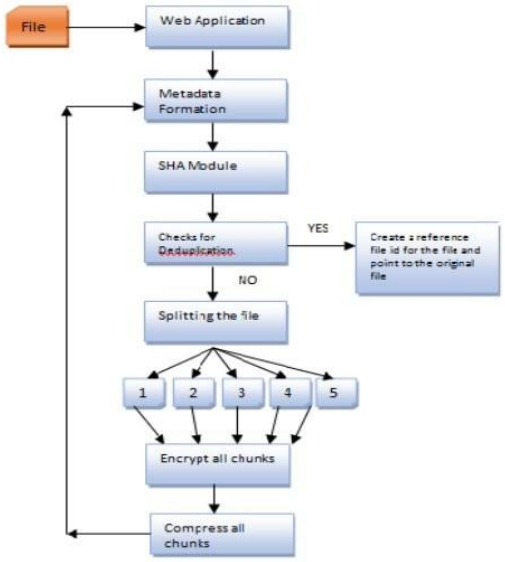

Figure 4:Upload Flow

Step 1: User uploads a file on the server.
Step 2: The BFS creates a metadata of the particular file.
Step 3: Based on the content of the file,the system creates an SHA value of the file based on its content.
Step 4: It then checks the database for files which have the same SHA value.If the content of the file is same,then BFS creates a reference FileId and points to the original file

Step 5: The file is split in different number of chunks where each chunk is of the same predefined size except the last chunk.
Step 6: The chunks are then encrypted using AES encryption standard.and compressed using Snappy Java Algorithm.

Use Case A. Uploading a file
The processes running inside the proposed BIG FILE STORAGE system are split into following phases.

1) Metadata generation :- Metadata generation for a file is essential so as to be able to uniquely identify a file and its chunks.
2) Deduplication:- Using metadata generated a file being uploaded by a user is cross checked for duplicacy
3) Encryption (AES) and splitting of file: A new non duplicated file is split up into chunks based on the metadata generated and each chunk is encrypted.
4) Compression: Every chunk of the file generated in the previous step is compressed thus enhancing the storage capability of the storage system.

Phase 1: Metadata generation

Typically, in any data storage system such as Dropbox , the size of meta-data increases respectively with the size of original file. The metadata design in these systems contains a list of chunk metadata structure which stores information such as chunk id, chunk size, hash value of chunk and other chunk metadata. Size of this list is equal to the number of chunks which will be generated from the actual file. The overall size of this list grows and becomes a performance bottleneck for the overall system as the file sizes and the number of files being stored grows. As shown in the figure the metadata design in such systems is not optimized and it eats up the storage space and processing power of the system as the system matures. We propose a solution in which the size of metadata is independent of the number of chunks generated from a file. The solution just stores the id of first chunk, and the number of chunks which is generated by original file.

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 1, Issue 8, December 2016

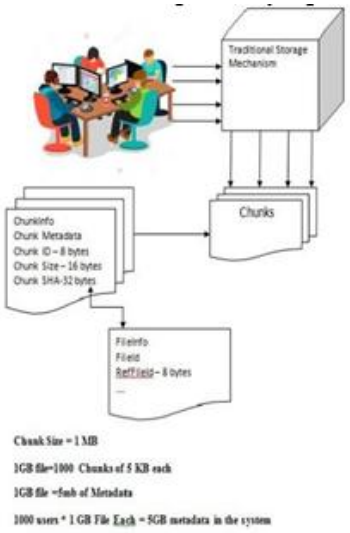

Figure 5: Metadata stored in file storage

The figure above shows how metadata is stored in the Big File Storage
Because the id of chunk is increasingly assigned from the first chunk, we can easily calculate the ith chunk id by the formula:

chunkid[i] = fileInfo.StartChunkID + i (1)

Meta-data is mainly described in FileInfo data structure which consists of following fields:

∑ File Name- the name of file;
∑ Encrypted: the file will be encrypted
∑ SHA value: 32 bytes - hash value by using sha-256 algorithm of file data;
∑ Start ChunkID : 8 bytes - the identification of the first chunk of file, the next chunk will have id as startChunkID+1 and so on;
∑ Number of Chunks :Chunk: 8 bytes - the number of chunks of the file;
∑ File ID: 8 bytes - unique identification of file in the whole file;
∑ Reference FileID: 8 bytes - id of file that have previous existed in System and have the same sha256 - we treat these files as one, refFileID is valid if it is greater than zero;
∑ File Size : 8 bytes - size of file in bytes;.

Phase 2: Deduplication:-

Data deduplication is a method of optimizing the storage by eliminating redundant data. Only one unique instance of the data is actually retained on storage media, such as disk or tape. Redundant data is replaced with a pointer or reference to the unique data copy. For example, a typical email system might contain 100 instances of the same one megabyte (MB) file attachment. If the email platform is backed up or archived, all 100 instances are saved, requiring 100 MB storage space. With data deduplication, only one instance of the attachment is actually stored; each subsequent instance is just referenced back to the one saved copy. In this example, a 100 MB storage demand could be reduced to only 1 MB.

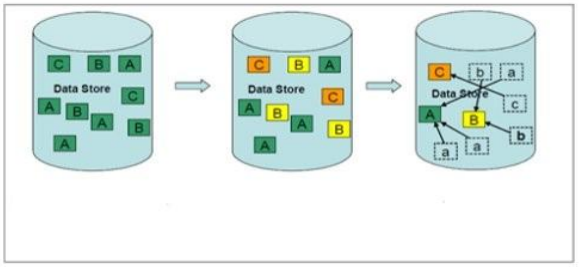

Figure 6: Data Deduplication process

In Figure 7,
1. Data store contains multiple instances of the same data
2. Deduplication process checks the content of the files to identify duplicates
3 .If the content of the file found is same, a pointer/reference is created to original file

Algorithm 1: Metadata generation and Deduplication check

Terms	***Description***
1) Createfileinfo() 2) Idgeneratorservice 3) Hashgeneratorutils. Hashfile() 4) Dbutils.retrievefil einfo() ***5)*** Dbutils.savefileinf o()	Creates basic metadata for the file such as filename, fileId, fileSize, No. of chunks that will be created from the file. Generates a sequence of ids for the fileId field and the start chunkId field in the FileInfo datastructure Generates an SHA256 hash value derived from the content of the file. It is used during the deduplication

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 1, Issue 8, December 2016

	process. Retrieves any existing metadata from the database using the hash value. If found then the file being uploaded is considered duplicate. This method persists the FileInfo metadata created in the database.

Data: file
Result: Metadata
1 begin
2 FileInfoService.saveFileInfoAndSplit (file)
/* The entry point of the entire flow. */
3 FileSplitter.createFileInfo()
/* Create a FileInfo object. FileInfo object is populated fileId, fileSize, the total number of chunks which the file will be split into.refFileId is populated later.
4 HashGeneratorUtils.hashFile (fileName);
/* Generate the hash value using SHA256 MessageDigest api. Hash value is generated for each incoming file upload request. */
5 retrieveFileInfo(hash)
/* check for exiting metadata based on the hash value generated above*/
6 if (existingMetadata) is found then
populate refFileId with fileId of the existing metadata
else
persist new metadata and
continue to split the file
7 end

Algorithm 2: Encryption and Chunk generation

Terms	Description
AES Encryptor Decryptor getAesEncryptCipher() chunkNoToChunkIdMap() getChunkSize() FileIO	Uses the 128 bit AES encryption algorithm to encrypt the chunks generated. Creates an AES encrypt mode cipher with the SHA value as the secret key Prepares a map of the chunkNo. To chunkId . It uses the FileInfo object's start chunkId and no. of chunks field to generate a map of the chunkNoToChunkId Predefined chunk size The FileIO api which is used to split the file in a number of chunks each of the predefined size except the last chunk.

Data: file
Result: chunks
1 begin
2 FileSplitter.split(FileInfo)
/* The entry point of the entire flow. */
3 AESEncryptorDecryptor. getAesEncryptCipher (SHA,ENCRYPT.MODE)
/* Creates an AES 128 bit encrypt cipher. The SHA value is used as the secret key in encrypt mode The encrypt cipher is used for encrypting the chunks generated from the file*/
4 Utilities.getChunkNoToChunkIdMap (getFileInfo())
/* Prepare a map of chunkNo. to chunkId . This map will be used in naming the chunks and saving them on the file system */
5 if(filesize <chunksize)
/*splitting of the file is not done and the file is saved as is/*
Else
/* split the file in the number of chunks. Encrypt all the chunks*/
6 end

Phase 3: Compression

The main advantages of compression are a reduction in storage hardware and the resulting cost savings. A compressed file requires less storage capacity than an uncompressed file, and the use of compression can lead to a significant decrease in expenses for disk and/or solid-state drives. A compressed file also requires less time for transfer, and it consumes less network bandwidth than an uncompressed file. We have proposed

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 1, Issue 8, December 2016

compression technique using Snappy algorithm which is helping in further reducing the size of the file and helps in saving storage space.

Algorithm 3: Chunk generation and compression

Terms	Description
Chunk Deflater Compress Chunk()	Uses the Snappy Java api to compress and decompress the chunks Uses Snappy Output Stream to compress the incoming chunk

Compression of the chunks is carried out simultaneously while the chunk is generated and encrypted .

Data: Encrypted chunk

Result: chunks

1 begin

2 chunkDeflater.compress (incomingChunk,outgoingChunk)

/*start compressing the chunk generated */

3 SnappyOutputStream.write (SHA,ENCRYPT.MODE)

/* SnappyOutput stream uses a buffer size of [64*1024] and writes a compressed chunk on the storage*/

4 end

Use Case B. Downloading a file

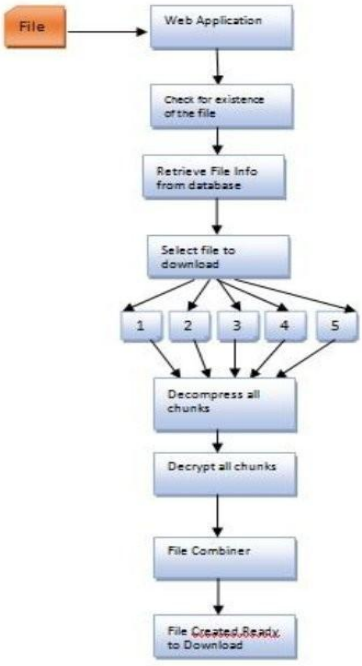

Figure 7:Download Flow

Step 1: User logs into the account and selects which file to download

Step 2: The server retrives metadata of the file from the database

Step 3: The decompression engine first decompresses all the chunks of the file using Snappy Java Algorithm.

Step 4: After decompression ,the chunks are decrypted by the decryption engine

Step 5: Then all the chunks of the file are combined by the file combiner engine

Step 6: The file is thus recreated and is downloaded at the users end.

Terms	Description

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 1, Issue 8, December 2016

getFileInfo And Combine () retrieve File Info() chunk Deflater decompr ess Chunk() get Aes Decrypt Cipher() File Combiner.join()	The entry point of File Info Service download flow Retrieve FileInfo metadata based on the filename Decompress the chunks of the file Decrypt the chunks of the file.Uses SnappyInputStream to decompress the chunk Join the chunks to get the original file

Algorithm 4: Chunk Decompression, Decryption and file generation

Data: Compressed and Encrypted chunks
Result: Decompressed, decrypted and chunks
1 begin
2 DbUtils.retrieveFileInfo
/* Retrieve FileInfo metadata from the database for the file to be downloaded*/
3 FileCombiner.getNumberParts
/*Prepare a map of the chunkNoTochunkId which will be used to look up the chunk */
4 ChunkDeflater.decompressChunk
/* Use the SnappyInput stream api to decompress*/
5 getAesDecryptCipher.decrypt
/*Use the AES 128 bit decryption process to decrypt the chunk.*/
6 Join the chunks using the FileIO api
7 end

IV. RESULTS

A.Metadata Design

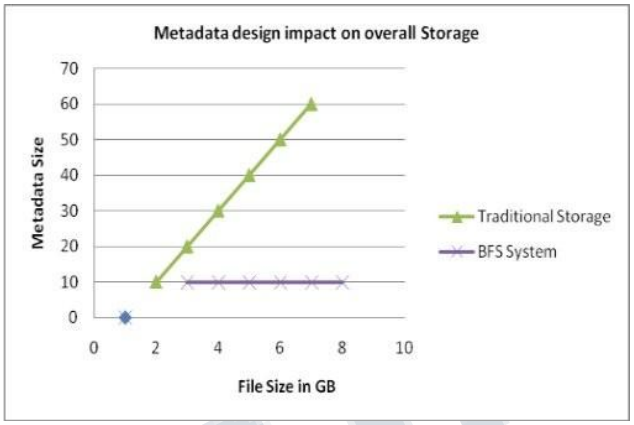

Figure 8:Metadata Size comparison with traditional storage

The above figure shows the impact of the metadata design on storage.The x-axis shows the file size stored and the y axis shows the metadata size.The traditional storage shows the considerable increase in metadata size with respect to file size.While the BFS shows that the metadata size does not increase with respect to file size.

B. Deduplication

Table I
Space Saved In BFS System

File	Number of Files	File Size before Deduplication	Space saved After Deduplication
1.	10	500 MB	440 MB
2.	15	700 MB	550 MB
3.	25	1000 MB	815 MB

The above table shows the different size of files we have saved in BFS .As the content of the file is checked for deduplication and if the file already exists, the system just refers to the original file by creatibg a reference fileid.Thus,it helps in saving storage space.The first reading shows that 10 files were uploaded ,out of which one was a copy,so the file was referenced to the original file.In the second reading,15 files were uploaded,out of which 3 files were copy,so only the 12 files got saved ,copy files were referenced to the original one.More than 50% of the space is saved overall thus increasing scalability.

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 1, Issue 8, December 2016

C. Compression

Table II
File Size Before And After Compression

Type of document	Size of the File	File size after Compression
Text	100 MB	80 MB
Mp3	40 MB	33.5 MB
Video	700 MB	615 MB

As we can see from the above table, the first reading shows the text file uploaded. The file size of the text was 100 MB ,but after compression the file size got reduced to 80 MB. The second reading shows the audio file(mp3) which was of size 40 MB but got reduced to 33.5 MB after compression.Similarly, the video file of 700 Mb got reduced to 615MB after compression.Thus compression helps in reducing storage space and increasing scalability.

V. CONCLUSION

BFS optimizes an online storage system by redesigning a simple metadata system to create an efficient storage mechanism. Every file in the system has a constant size of metadata regardless of the file size. The data deduplication methodology of BFS uses SHA-256 based hash function to fast detect duplicate files being uploaded and thus reduces data redundancy. The chunks generated from the file are encrypted using AES-128 algorithm which improves the security aspect of the online storage. Finally the encrypted chunks are compressed using a fast compression algorithm Snappy Java which further reduces storage space on the server.

REFERENCES

[1]. Thanh Trung Nguyen, Tin Khac Vu, Minh Hieu Nguyen:"BFC:High Performance Distributed Big-File Cloud Storage Based on Key Value Store.",Information Technology Faculty Le Quy Don Technical University, Ha Noi, Viet Nam VNG Research, Viet Nam ,IEEE ,SNPD 2015, June 1-3 2015, Takamatsu, Japan

[2]. Jin Li,Yan Kit Li;Xiaofeng Chen;Patrick P.C Lee;WenjingLou"A Hybrid Approach for Secure Authorized Deduplication"IEEE Transactions on Parallel and Distributed Systems,Pages :1206-1216,Volume :26,Issue :5,2015

[3]. L. Chappell and G. Combs. Wireshark network analysis: the official Wireshark certified network analyst study guide. Protocol Analysis Institute, Chappell University, 2010.

[4]. Jin Li, Xiaofeng Chen, Xinyi Huang, Shaohua Tang and Yang Xiang Senior and Mohammad Mehedi Hassan and Abdulhameed Alelaiwi "Secure Distributed Deduplication Systems with Improved Reliability" IEEE Transactions on Computers, Volume: 64, Pages: 3569 – 3579, Issue: 12,2015

[5]. I. Drago, M. Mellia, M. M Munafo, A. Sperotto, R. Sadre, and A. Pras. Inside dropbox: understanding personal cloud storage services. In Proceedings of the 2012 ACM conference on Internet measurement conference, pages 481–494. ACM, 2012.

[6]. P. FIPS. 197: the official aes standard. Figure2: Working scheme with four LFSRs and their IV generation LFSR1 LFSR, 2, 2001.

[7]. S. Ghemawat and J. Dean. Leveldb is a fast key-value storage library written at Google that provides an ordered mapping from string keys to string values. https://github.com/google/leveldb. Accessed November 2, 2014.

[8]. S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In ACM SIGOPS Operating Systems Review, volume 37, pages 29–43. ACM, 2003.

[9]. Amrita Upadhyay; Pratibha R Balihalli; Shashibhushan Ivaturi; Shrisha Rao "Deduplication and compression techniques in cloud design" 2012 IEEE International Systems Conference SysCon 2012
Pages: 1 - 6,,2012