High-PerformanceDistributed Storage Service Based On De-duplicationAnd File Compression

Dr. SRIKANTH THODETI
Professor , Dept: C.S.E.
Jayamukhi Institute of Technological Sciences, Narsampet, Warangal,Telangana, India

Abstract: Cloud storage supplies a rate-strong andexcellent decision to share the resources ofcloud customers. However, the storage capacity inside thecloud is increasing and has grown to be a rising developmentin the area of understanding storage. Because the customersload the information into the cloud there is also theprospects of inflicting the predominant challenge just like the redundancy of documents leading to wastage of house forstoring. To unravel this difficulty Key-value storecompete for the main position and exhibited optimisticoutcome. This paper presents Big FileCloud-based Storage approach (BFCSS) with its modulesand architecture that goals to control lots of the problems in a CloudStorage system for storing tremendous files, which is founded on the key valuestore. Here we are proposing less-problematic, fixed metadata design,which allows for rapid as well as totally-concurrent, disbursed fileinput/Output, and simple file and knowledge de-duplication systemfor static knowledge. For storing massive files of close terabytes of data, this method can be utilized to build the Cloud-situated Storageprocedure

Keywords-Big File, Cloud Storage System, DistributedStorage System, Key value

I. INTRODUCTION

Storage pools are the adequate space assembled fromvarious physical resources in a very distributedsurroundings. In that cloud storage provides the space tostay the info for abundant users. For instance, capabilityfor every user provided by the cloud infrastructure canbe in gigabytes and terabytes. The recognition of theapplications has become enormous since cloud isemployed by the users for his or her lifestyle right fromuploads, downloads of knowledge to exchange of data inany social networking sites, Zing me. So, the info dropsin a very large repository are going to be large in cloud,increasing the holding ability of the system. Problemsand difficulties are two-faced by the system to supplysmart quality of services to the individuals. Thus theseissues apply to the term.The essential technique to tackle the difficulty is tocipher the data and afterward transfer constant.

Miserably, providing an efficient and secured datasharing theme is kind of difficult. On uploading files tobig data by many varieties of users generally thedocuments could be similar and will cause to knowledgeredundancy that is again a problem of waste within thespace for storing. There are many different issueswhereas designing an efficient storage engine similar tomassive file process, de-duplication, distributed and highmeasurability.

Key value stores have various benefits for storinginformation in data-intensive operation. In recent years,key value stores have a really very good growth in eachfield. They have low latency with less time interval andhigh quantifiability with small and medium key valuecombine size. Current key value stores aren't designedfor directly storing big-values, or big file in our case. Wetend to executed many experiments within which weplace whole file-data to key value store, the systemfailed to have sensible performance as usual for severalreasons: first of all, the latency of put/get operation forbig-values is high, thus it affects alternative paralleloperations of key value store service and multipleconcurrent accesses to totally different value. And, whenthe worth is massive, then there's no area to cacheobjects in memory for quick access. Finally, it's tough toscale-out system once number of users and informationincrease. This research is enforced to resolve thoseissues once storing big-values or big-file victimizationkey value stores. It has and gets several benefits of keyvalue store in information management to analysisknown as cloud-storage system referred to as Big FileCloud Storage (BFCS)

II. RELATED WORK

Presently in information technology anefficient cloud storage system can be developed, byanalyzing the design of already existing systems andevaluating their impact of design choices on performance.Nowadays, personal cloud storage services are gainingimportance. It is believed that cloud storage generate hugeamount of Internet traffic as there is increase in thenumber of providers to enter the market and an increasingoffer of a cheap storage space. The understanding ofarchitecture, performance of such systems and

theirworkload is essential in order to design efficient cloudstorage systems and predict their impact on the network.Here, we will present a characterization of Dropbox, which is a leading solution in personal cloud storage inour datasets. Dropbox is the most widely used cloudstorage system which accounts for a volume equivalent toone third of the youtube traffic at campus networks. If weanalyze the usage of Dropbox on the internet, it shows thatthere is an increasing interest on cloud-based storagesystems. Major companies like Google, Apple, Microsoftare offering this service. Drop box service performance ishighly impacted by distance between the clients and datacenters, which are located in the U.S. In the client protocolthe usage of per-chunk acknowledgements combined withsmall chunk sizes will deeply limit the effectivethroughput of the service. We have identified two possibleimprovements to the protocol in this paper: a) the usage ofa chunk building scheme. b) introduction of delayedacknowledgements.

The recent deployment of bundling mechanism hasimproved the system performance dramatically. We canexpect that the overall performance can be improved by thedeployment of other data-centers in different locations.Personal Cloud storage services are data-intensiveapplications creating huge amount of Internet traffic. Let usanalyze 5 of these services. Dropbox implements most of thechecked capabilities, and its sophisticated client clearlyboosts performance, although some protocol tweaks seempossible to reduce network overhead. However, duplicationof data, metadata complexity become major problems givingrise to wastage of storage space and increasing networkoverhead. In Cloud Drive, wastage of bandwidth has amagnitude which is higher than other services, lack of clientperformance results in bottlenecks. SkyDrive also showssome limitations in performance like network latency. In OneDrive we can see limitation of data duplication. Google Drivefollows a different approach resulting in a mixed picture: itenjoys the benefits of using Google's capillary infrastructureand private backbone, which reduce network latency and
speed up the system. However, protocols and client featureslimit performance, especially when multiple files areconsidered. The existing Cloud storage systems have tried toimplement distributed cloud storage and improve theirperformance. However, there are certain limitations 27 likemetadata complexity, data duplication which in turn lead toother issues discussed below. In our project implementation,we will look forward to overcome the issues faced by theexisting cloud storage systems.

III. PROPOSED WORK

A. Storage of the Chunks:

Chunk is the basic element in the defined cloud storagesystem. Chunks are generated from a file. When the end useruploads a file, it will be split into a number of chunks. Allchunks which are generated from a file have the same size (thelast chunk of a file may have an equal or smaller size)exceptthe last chunk. The ID generator will generate id for the first
chunk with auto-increment mechanism after that. Next chunkthat follows in the chunks set is to be assigned with an ID andthen gradually increase till the final chunk. FileInfo object iscreated and saved with information such as fileid, file size,first chunk id, total number of chunks and will be stored tothe database and the chunks will be stored in key value storageas identity id of chunk and value is data of particular chunk.Chunk storage is one of the most important module of definedcloud storage system.

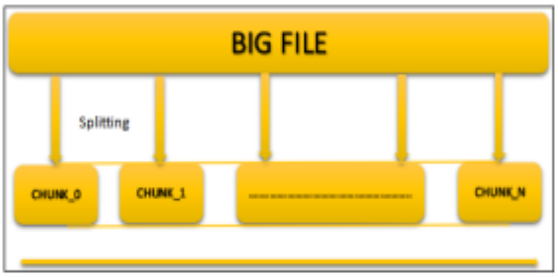

Fig. 1. Splitting a big file into number of chunks

B. Uploading and Deduplication Mechanism:

The Figure 2 describes an algorithm for uploading big fileto BFCSS. Deduplication of data can be defined in the cloudstorage BFCSS. There are many types and methods of datadeduplication [3] which can work both on client-side or serverside. We are using a simple method with SHA-2 hash functionto detect duplicate files in the system during the uploading offile. As the User uploads the file, SHA value of the content ofthe file will be generated and will be verified with the SHAValues already present, if already present a refrence to the filewill be generated and if SHA Value does not match the thenew file will uploaded on to the cloud.

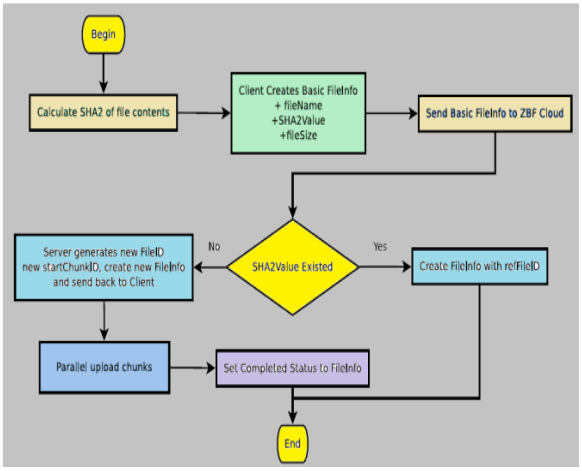

Fig. 2. Uploading Mechanism

C. Downloading Mechanism

Figure 3 describes an algorithm for downloading big file from BFCSS. Firstly, the end user selects the id offile that will be downloaded to the server.IfFileInform ofthe fileidexists, this information will be sent back to theclient. The client uses the FileInform information to schedulethe download process. Every downloaded chunk will besaved directly to its position in the respective file. Thedownload process is completed only when all chunks arefully downloaded successful.

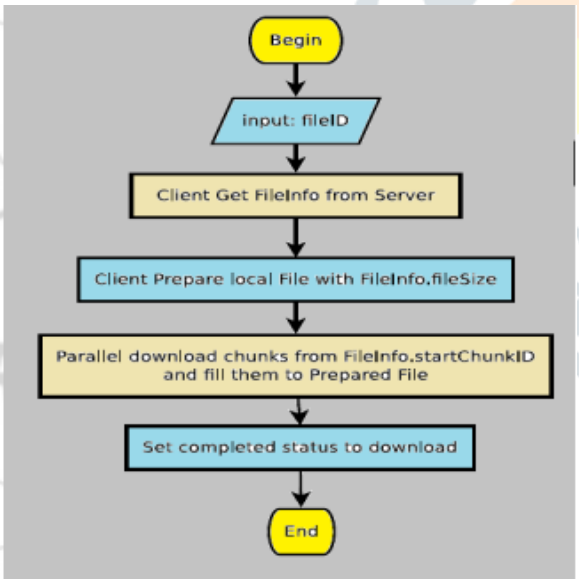

Fig. 3. Downloading Mechanism

D. Secure Data Transfer Protocol:

Data confidentiality is one of major requirements of cloudstorage system. To ensure quality of service, a light-weightand fast network protocol for transfer data is also required.In the proposed system , the data transferred between clientsand servers are securely encrypted using AES[9] algorithmusing key exchange.

E. Compression:

Data Compression has an important role in the proposedsystem. Data compression is a process by which a file istransformed to another file(compressed), so that the originalfile is fully recovered, without any loss of information in thefile, from the compressed file . This process is used to savethe storage space.

F. Implementation Design And Algorithms:

SHA-2 transforms an input message into the 256 bits message digest. According to Secure Hash SignatureStandard the input message whose length are shorter than 264 bits, becomes a 256-bit message digest after operating by 512 bits in groups. The algorithm isillustrated as follows :-

Step 1: Message Padding: Input message is appended with 1's and padded with 0's until the length becomeslength = 448 mod 512. Original length of the messageis then appended to 64-bit binary number. The paddedlength of the message is a multiple of 512 bits, whichwill help to decides how many '0' to be padded.

Step 2: Parsing: The message that is paddedis then parsed into N 512 - bits blocks :M(1),M(2),.....M(N).These M(i) message blocks are passedindividuallyto the message expander.

Step 3: Expansion of Message: Into 16 32-bit words, each 512 bit block is to be divided as follows : M(i) 0 ,M(i)1 ....M15 (i) , which are then expanded into 64 wordsunder the certain rule prescribed by SHA-2 standardlabeled W0, W1.....W63.

Step 4: Compression of Message: From Messageexpansion stage thwWt words are then given tothe SHA compression function or SHA Core. Thecompression function utilizes 8 working variableslabeled A,B,.....,H which re then initialized to predefinedvalues H(0)0 - H0(7) at the start of each call to the hashfunction.

Step 5: The algorithm is implemented by 64 - cycle iterative computation each block. The eight working variables are labeled A, B, C,D,E,F,G,H, which are updating the value during the 64-cycle.

Step 6: An intermediate hash value H(i) is calculated, after 64 iterations of the compression function.The SHA-256 compression function , is repeated andprocessed with another 512-bit block from the messagepadder. After all the data

blocks have been processed,final 256-bit output H(N) is calculated as follows:-

HN = HN0 ; H1N; H2N::::::::::H7N

IV. CONCLUSION

Every file inthe system has a same size of meta-data regardless of file-size.Every big-file stored in BFCSS is split into multiple fixed-sizechunks (may except the last chunk of file). The chunks of a filehave a contiguous ID range, thus it is easy to distribute dataand scale-out storage system, especially when using MYSQL.This research also brings the advantages of key value store intobig-file data store which is not default supported for big-value.The data deduplication method of BFCSS uses SHA-2 hashfunction and a key value store to fast detect data-duplicationon server side.

REFERENCES

[1] ThanhTrung Nguyen, Tin Khac Vu, Ha Noi, Minh Hieu Nguyen, VietNam, BFC: High-Performance Distributed Big-File Cloud Storage BasedOn Key value Store, June 1-3 2015, IEEE SNPD 2015, 978-1- 4799-8676-7/15(Base Paper).

[2] M.H.Nguyen and T.T.Nguyen, Design Sequential Chunk identity withLightweight Metadata for Big File Cloud Storage, IJCSNS InternationalJournal of Computer Science and Network Security, VOL.15 No.9,September 2015.

[3] Jin Li, Xinyi Huang, Xiaofeng Chen, Shaohua Tang and YangXiang, Mohammad Mehedi Hassan, AbdulhameedAlelaiwi, SecureDistributed Deduplication Systems with Improved Reliability, 2015,10.1109/TC.2015.2401017, IEEE Transactions on Computers.

[4] D. Borthakur. Hdfs architecture guide.HADOOP APACHE PROJECThttp://hadoop.apache. org/common/docs/current/hdfs design. pdf, 2008.

[5] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach,M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: Adistributed storage system for structured data. ACM Transactions onComputer Systems (TOCS), 26(2):4, 2008.

[6] L. Chappell and G. Combs. Wireshark network analysis: the officialWireshark certified network analyst study guide. Protocol AnalysisInstitute, Chappell University, 2010.

[7] I. Drago, E. Bocchi, M. Mellia, H. Slatman, and A. Pras. Benchmarkingpersonal cloud storage. In Proceedings of the 2013 conference onInternet measurement conference, pages 205–212. ACM, 2013.

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

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

[10] S. Ghemawat and J. Dean.Leveldb is a fast key-value storage librarywritten at google that provides an ordered mapping from string keys tostring values. https://github.com/google/leveldb. Accessed November2, 2014.

[11] 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.

[12] Y. Gu and R. L. Grossman. Udt: Udp-based data transfer for high-speedwide area networks. Computer Networks, 51(7):1777–1799, 2007.

[13] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed. Zookeeper: wait-freecoordination for internet-scale systems. In Proceedings of the 2010USENIX conference on USENIX annual technical conference, volume 8,pages 11–11, 2010.

[14] P. Jin, P. Yang, and L. Yue.Optimizing b+-tree for hybrid storagesystems.Distributed and Parallel Databases, pages 1–27, 2014.

[15] D. Karger, A. Sherman, A. Berkheimer, B. Bogstad, R. Dhanidina,K. Iwamoto, B. Kim, L. Matkins, and Y. Yerushalmi. Web cachingwith consistent hashing. Computer Networks, 31(11):1203–1213, 1999.

BIODATA

Author

Dr. SRIKANTH THODETI received his Master of Computer Applications (M.C.A.) from Kakatiya University Warangal, M.Tech.(C.S.E.) from J.N.T.U. Hyderabad and Ph.D. (Computer Science) from Dravidian University, A.P.. He Published 2 ISBN books & 16 papers in various reputed International Journals and National / International Conferences.