IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

Data Deduplication and Load Balancing Techniques on Cloud Systems

Prof M.S. Pokale[1], Surabhi Dhok[2], Vaishnavi Kasbe[3], Gauri Joshi[4], Noopur Shinde[5]

Professor, Computer, PVG'S COET, Pune, India[1]

Student, Computer, PVG'S COET, Pune, India[2,3,4,5]

Abstract: Cloud storage systems are able to provide low-cost and convenient network storage service for users, which makes them more and more popular. However, the storage pressure on cloud storage system caused by the explosive growth of data is growing by the day, especially a vast amount of data waste plenty of storage space. Data deduplication can effectively reduce the size of data by eliminating redundant data in storage systems. However, current researches on data deduplication, which mainly focus on the static scenes such as the backup and archive systems, are not suitable for cloud storage system due to the dynamic nature of data. In this paper, we propose the architecture of deduplication system for cloud storage environment and give the process of avoiding duplication at the file-level and chunk-level on the client side. In the storage nodes (Snodes), DelayDedupe, a delayed target deduplication scheme based on the chunk-level deduplication and the access frequency of chunks, are proposed to reduce the response time. Combined with replica management, this method determines whether new duplicated chunks for data modification are hot and removes the hot duplicated chucks when they aren"t hot. The experiment results demonstrate that the DelayDedupe mechanism can effectively reduce the response time and achieve the storage load of Snodes more balanced.

Keywords: Cloud storage, deduplication, DelayDedupe, replica, chunk, load balancing.

I. INTRODUCTION

Data deduplication is a specialized technique for data compression or splitting of data into chunks for eliminating duplicate copies of repeating data. Few synonymous terms are intelligent (data) compression and single-instance (data) storage. This technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent.

Load Balancing is a method of distributing workload across multiple computing resources such as cluster of computers . The goal of Load Balancing is to optimize the resource usage, avoid overload, maximize throughput and to minimize the response time. This was identified as a major concern in Cloud Computing to scale up the increasing demands[1].

Cloud computing enables on-demand network access to a shared pool of configurable computing resources such as servers, storage and applications. Cloud storage refers to the delivery of storage resources to the consumers over the Internet. During such expansion, storage nodes in the cloud storage need to be balanced in terms of load. In order to maintain the load across several storage nodes, the data needs to be migrated across the storage nodes.[3] This data migration consumes more network bandwidth. The key idea behind this Application is to develop a dynamic load balancing algorithm based on deduplication to balance the load across the storage nodes.

II. RELATED WORK

There are different types of deduplication techniques. Local deduplication[5] occurs at a single data source, whereas global deduplication is used to integrate different data sources . Therefore, the latter is better than the former, but the overhead of the latter is much larger than the former. Client-side deduplication[5] happens at the client side before data is uploaded, hence reducing the network bandwidth, but takes up a large number of computing resources at the source end. Compared with client-side deduplication, server-side deduplication [1]i.e target deduplication happens at the target end where data is stored hence eliminating redundancy between different data sources to ensure that only one copy is stored.
Inline deduplication can realize real-time data reduction when data is written to the storage system, such as DataDomain, but consumes much system resources. Offline deduplication requires enough free space for data storage before deduplication . This approach is suitable for the storage protocols like Direct-Attached Storage and Storage Area Network. File-level deduplication, chunk-level deduplication and byte-level deduplication can increase storage utilization by eliminating data redundancy within or across files. Whole-file chunking cannot eliminate data redundancy within files, so finer-grained chunking strategies like fixed size chunking, content-defined chunking, and TTTD chunking[3] are introduced. In fixed-size chunking algorithm, all files are portioned into blocks with a fixed size such as 4KB and 8KB. It is suitable for EXE, PDF, VMDK files due to its low

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

computational overhead. But, it is very sensitive to the data change.

Content-defined chunking is a typical variable size chunking algorithm combining Rabin's key and sliding window to determine the chunking boundaries by the content of data, and the similarity of files can thus be detected. found when the window size is bytes and the expected chunk size is 8KB, content-defined chunking can provide better performance of deduplication. [3]However, it results in high system overhead and in extreme cases , the size of data chunk is too big or too small. Therefore, the improved content-defined chunking algorithm is used to effectively control the distribution of block size by setting the minimum and maximum chunk size .

After all files are divided into small blocks, the keys of all blocks are needed to be calculated as their identifiers by using MD5 or SHA-1 and the identifiers are used for key lookup. That is, two keys are same if and only if corresponding data in files is the same. Moreover, although cryptographic hash functions take a message of arbitrary length as input, they produce a specific bit length key as output, 128-bit for MD5 and 160-bit for SHA-1, respectively. The overall performance of SHA-1 is better than MD5, but the overhead of CPU is higher and the rate of operation is relatively slower.

The disk bottleneck problem constrains the performance of deduplication systems due to the querying of an index over all the existing keys. In general, as the memory space is very limited, only a fraction of index is stored in memory, and the remaining majority of the index is stored on disk. Extreme Binning exploits file similarity instead of locality to put files with the same representative chunk ID (that is, the minimum chunk ID of every file by Border's theorem) into the same bin (each bin is stored on the disk). Each representative chunk ID is kept in the primary index that resides in RAM and contains a pointer to its bin. Although Extreme Binning is a scalable, parallel deduplication technique, it isn't suitable for the workload mainly composed of small files. To this end, similar index uses similar hash as the feature of a file. In addition to these software methods above, demonstrates that performance acceleration can be achieved by a pipelined dual-level keying based on multi-core CPUs from a hardware perspective.

Additionally, data attributes are taken to improve the deduplication. Tan et al. proposed the SAM source deduplication for cloud backup system, which decreases the range of key comparison to increase the speed of deduplication process by exploiting file semantics in[4]cluding the locality, size, type, and timestamp of file. Tan et al. proposed a deduplication performance booster called CABdedupe, which captures and records the casual relationships among chronological versions of datasets to remove the unmodified data during the deduplication. Fu et al. proposed the AA-Dedupe scheme for cloud backup services, which chooses different chunking strategies and hash functions for different applications to improve deduplication efficiency. For the deduplication in cloud storage system, recent findings mostly focus on the security of cloud storage. Shen proposed a secure deduplication with proxy encryption and version control. Stanek et al. presented a novel threshold cryptosystem based on data popularity which permits a more fine-grained trade-off between storage efficiency and security. That is, it guarantees semantic security for unpopular data and provides weaker security and better storage for popular data. The dynamic nature of data is mentioned in The Redundancy Manager, calculates an optimal number of copies for files based on the number of reference and level of Quality of Service after identifying the duplication. However, new duplication for data modification by users isn't considered in storage nodes.

III. SYSTEMMODULE

This section gives the architecture of system with deduplication and how to avoid the duplication of data.

A .System Architecture

System architecture is the conceptual model that defines the structure behavior and more views of a system . An architectural description is a formal description and representation of a system organized in a way that supports reasoning about the structures of the system. It can comprise system components, the externally visible properties of those components , the relationships between them . It can provide a plan from which products can be procured, and systems developed that will work together to implement the overall system . There have been efforts to formalize language to describe system architecture ;[3] collectively these are called architecture description languages.

The system consists of different modules like Client, MS, Secondary MS (SMS) and Snode. Client is used to represent the location of the original data to be uploaded and Snode represents the location of the new data to be stored after deduplication Users can perform the following operations :- request for uploading the file, file access, file modification, download and deletion through Client . The metadata of files is stored in MS and the actual data is stored in Snode. With the help of metadata information, we can find the location of data in Snode and determine whether the data from Client is duplicated. MS acts as the manager of the system . If MS malfunctions, the system breaks down. To avoid single-point failure, SMS is responsible for synchronizing the backup of metadata images and operation logs.

Every time new data is uploaded, system first performs local deduplication and uploads the metadata information to MS.[1] Then it finally uploads the new no-duplicated data to Snode. There are four modules on the Client side: File Preprocess Module, Local Deduplication, Metadata Manager, and File Transfer Module. File Preprocess Module is responsible for calculating the keys of files by Hashing algorithm. Local Deduplication discards

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

duplicated data successively at the file-level as well as chunk-level deduplication. Metadata Manager maintains the keys of files and chunks that had been uploaded to Snode in order to prevent the duplicated data from being uploaded repeatedly. File Transfer Module is used to transfer the metadata to MS which is processed by local deduplication .[3]

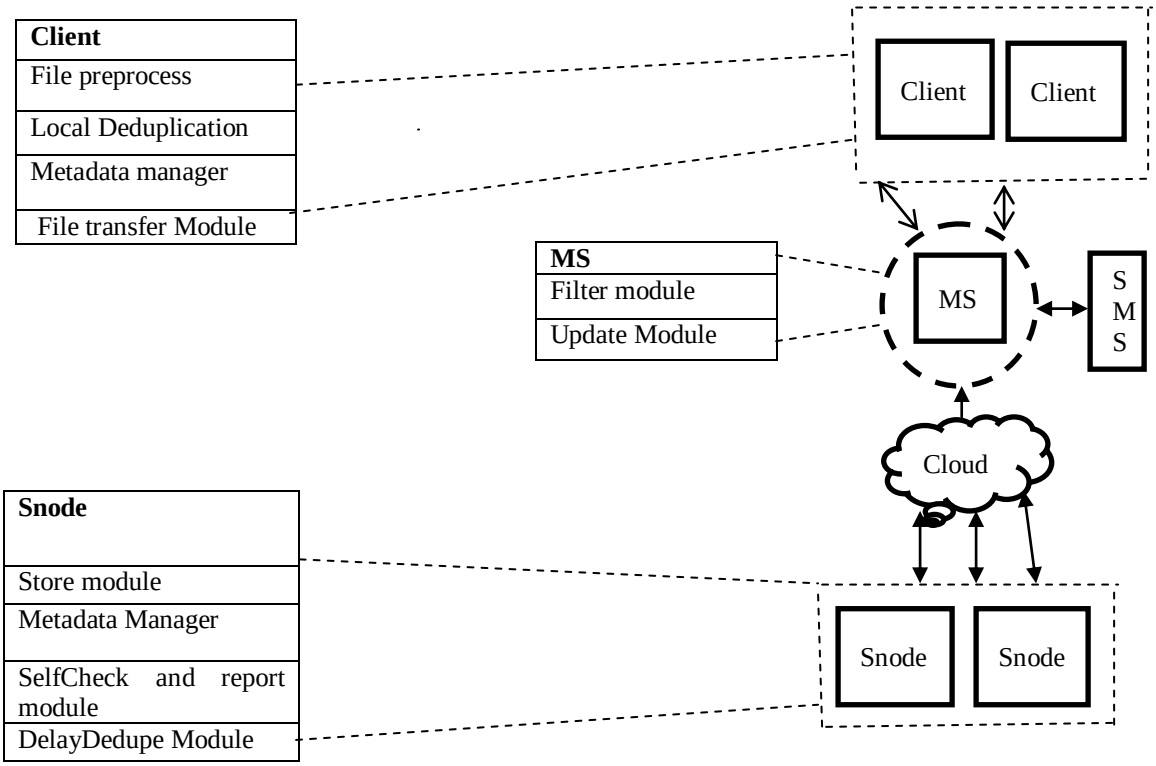

Fig . System Architecture

There are two modules in MS:- Filter Module and Update Module. Filter Module is responsible for filtering duplicated data from different clients. Update Module is used to update the metadata index in MS according to the modified metadata information from Snode.

Snode has four modules : Store Module,
Metadata Manager, Self-check & Report Module, and DelayDedupe Module. Store Module is used to store the actual data . Metadata Manager maintains the metadata information including the key and reference of the chunk stored in Snode. Self-check & Report Module detects the duplicated data for data modification by different users and reports the modified metadata information to MS. DelayDedupe Module determines whether the duplicated chunk is hot. To realize the function of each module in this system, Client, MS, and Snode need to maintain various data structure and tables.

1)Client: Client's Metadata Table often refered as CMT is a table used to store the keys of files and chunks at file-level and chunk-level deduplication , hence avoiding uploading of the data again. CMT is present on every client Each record in CMT is stored as:
(filefp, chunk fp, chunk fp, ・・・, chunk fp) where file fpis to the key of a file, and
chunkfpis the key of the chunk. Each
chunkfpmust be kept by the sequence of the file segmentation.

2) MS: MS is the metadatserver . It is responsible for storing the chunk metadata in Snode as well as to avoid duplication globally. Chunks are stored across the different, which makes it difficult to access the file. Also, there is FRT (File Reconstruction Table). Using this , Client can obtain the chunks of respective files stored at different snodes, quickly to reconstruct Metadata Table (MMT) and File Reconstruction Table (FRT).

a) MMT: Each record consists of:
(chunkfp,Snodeip, reference count, user id)
wherechunk fpis the key of a chunk
user id is the user identifier sharing the chunk. Both of them are unique globally. Snodeipstores the ip address of Snode where chunk is stored .
reference count represents the reference count of the chunk. It is used to store the number of files which share the respective chunk . If reference count is equal to 0 , then that chunk can be removed physically from the Snode
b) FRT: It is used to reconstruct a file requested by the

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

user quickly. Once the keys for files requested by user are received , Meta Server firstly obtains Snodeipof their chunks with the help of FRT and MMT. Each record consists of :
(filefp, chunk fp, chunk fp, ・・・, chunk fp) where file_ipis key of the file
Chunk_fpis key of the chunk

3) Snode: There exists a metadata table for each Snode used to store the address of the chunks. Incase a chunk resding in Snode is modified , key of that particular chunk is modified and compared with the local Snode's Metadata Table (SMT) to check for duplication of data .The main purpose of SMT is to maintain the mapping between address and the key of chunk . Each record in SMT consists of :
(chunkfp, offset, reference count, access num)
Where ,chunk fp is the key of a chunk stored on local disk, and offset gives the physical address of the chunk. Here, reference count is the same as the one in MMT. accessnummaintains the amount of times particular chunk was accessed.

B. Detection and elimination of duplicated data
1) Avoid duplication at each client:
Usually, in different types of applications the amount of data shared is negligible. But, there's a high probability of duplicated data in different files having same type and size [1]. For achieving better efficiency of deduplication, our system performs file-level as well as chunk-level deduplication at each client. Steps are as follows:

a) Collect the attributes of file to be uploaded such as file size, file type. Then classify the files according to the file type in ascending order of file size.
b) Calculate th key for each file using SHA-1 algorithm.
c) The key generated is compared with the keys stored in CMT. If the keys are same , then goto step g)
d) Now, again classify the files according to files greater than a particular size and less than that size. For the files greater than that size , use chunking algorithm to divide the file into blocks of a particular size(say 64kb)
e) For the chunks generated in previous step , calculate the key for each of the chunks.
f) Again, compare the chunk keys with chunk keys stored in CMT.
g) Remove the duplicated data. Store the metadata information.

2) Avoiding data duplication at MS:
As stated earlier, MS is used to store the metadata of all data stored in Snodes. It can be thought of a bridge between client and Snode. MS basically avoids the duplicated data in Snode as well as client by comparing the keys tored in MMT.

Steps are as follows:
a) First step is to receive the metadata from clients and then read the keys of file in some sequence.
b) memory, disk and buffer is checked for key index . If the same key is found then send "found" message to client. If the key is found then update the reference count else send "Not found to client "
c) Store the keys and the data not found in Buffer

3) Avoid duplication of data at each Snode:
In the above mentioned steps the duplicated data is removed from the client globally as well as locally .the chunk can be acceseed directly by the user using the snode_ip and the data can be modified in snode itself This may result in new duplication.

Steps to avoid the newly generated duplication:
a) Request : The modification of chunk is received by Snode i. Let the modified chunk be A . A is copied to memory .
b) Modify : A is modified by the 'i' Snode in memory. Let this modified chunk be called as B then , the key of B is calculated by using SHA-1 .
c) Check :Now , Snode i compares B's key with the keys it has in its SMT . If not, go to step e) else the existing chunk that is same as B is denoted as B'.
d) Deduplicate: The pointer pointing to address of B' rather than B itself is stored .
e) Store: Store B and updation of the SMT is done.
f) Check: The updated data is sent to MS by Snode i, where MS checks for duplicated data in Snode (j_= i)
If found, then go to step h).
g) Duplicate: B is a new chunk and MS needs to create replicas for it in other snodes.
h) DelayDedupe: It uses or applies the DelayDedupe strategy for B.

IV. DELAY DEDUPLICATION

Often, users complain about the delay system takes to give access of data user wants. Also because of the system downtime or may be in some situation like hardware or software damage system may collapse and because of which data availability gets affected. So, the cloud storage system should be developed such that it should handle such system downtime and provide users the data file he wants in lesser time. Thus, for this purpose this system is using the concept of replica creation and management of these replicas across the systems. Replica management is the way to increase data availability.

In this initially we are maintaining three replicas of each chunks. These three replicas are in different snodes. The number of replicas [2] for each chunk are created and managed afterwords by the replica manager according the access of that chunk by different users. Hence, by creating these replicas the system bottlenecks will get reduced and system will become more efficient. Also by using replica management the replicas are created only for those chunks which are accessed frequently by the clients.
Creating replicas for the chunks which are rarely accessed by clients will increase cost of system and also those

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

replicas will occupy more space in the memory which will lead to memory management issues for the chunks that actually requires space in memory because of there frequent access.

This is basically done by calculating frequency access of each chunk and comparing it with the standard ideal value for the access of chunk we have set for the system. Different replicas of the same data should be equally available to each client and they should be updated properly each time.

A) HOT DATA: In the big data system the data that is accessed frequently is called as hot data.[2] The frequency access of the data is used to determine whether the data is hot or not. The data which is not hot is termed as cold data which is not accessed frequently or is rarely accessed.

FreqAccess>α
FreqAccess means the average access frequency of the particular chunk and α represents ideal value for the access defined for the system.

How to calculate FrequencyAccess:

$$\text{FreqAccess} = \sum_{J \in z} \frac{Aj\,(tp+1) - Aj\,(tp)}{tp+1 - tp}$$

, i, j ∈ Z, i = j, tp+1 >tp

Above is the formulae to calculate frequency access for the given chunk.

There are n Snodes. In Snode i, chunk A is modified by a user (denoted as B) at tp+1, and chunk B is found in Snode j as described in the first case. Therefore, chunk B in Snode i is duplicated. Z is the set of the id of Snode where chunk B is found. In Snode j, Aj (tp) denotes the access num of chunk B at tp and Aj (tp+1) at tp+1, respectively.

B) DelayDedupe: Delay deduplication is the strategy in which we are temporarily delaying the deduplication of the particular chunk if it is a hot duplicated chunk in some storage node.

If FreqAccess of chunk B is bigger than α, B is a hot chunk in some storage node. At the same time B is also present in some other storage node so here, in that case instead of eliminating the chunk B we delay the elimination of chunk B as it is a hot duplicated chunk. In this way we are maintaining the availability of the data and reducing the pressure on the system by creating more replicas of chunk B and store them on server nodes according to the space available using load balancing algorithm.

To illustrate the method specific steps are given as follows:

1. MS decides whether the chunk B is hot chunk or not by the formula specified previously:

2. Calculate S at tp+1 by equation (7). S means the average remaining storage space of n Snodes, Sm(tp+1) denotes the rest storage space of Snode m at tp+1.

$$S = \sum_{m=1}^{n} \frac{Sm(tp+1)}{n}$$

3. Compare S with Sk(tp+1) of Snode k where non-hot chunk B is found (k ∈ Z), and remove chunk B in Snode k (the optimal Snode) where Sk(tp+1) is the smallest. Update metadata in MS.

4. Don't remove hot chunk B and synchronize with Snode j at
tp+1, then go to step a) at tp+2 (tp+2 > tp+1).

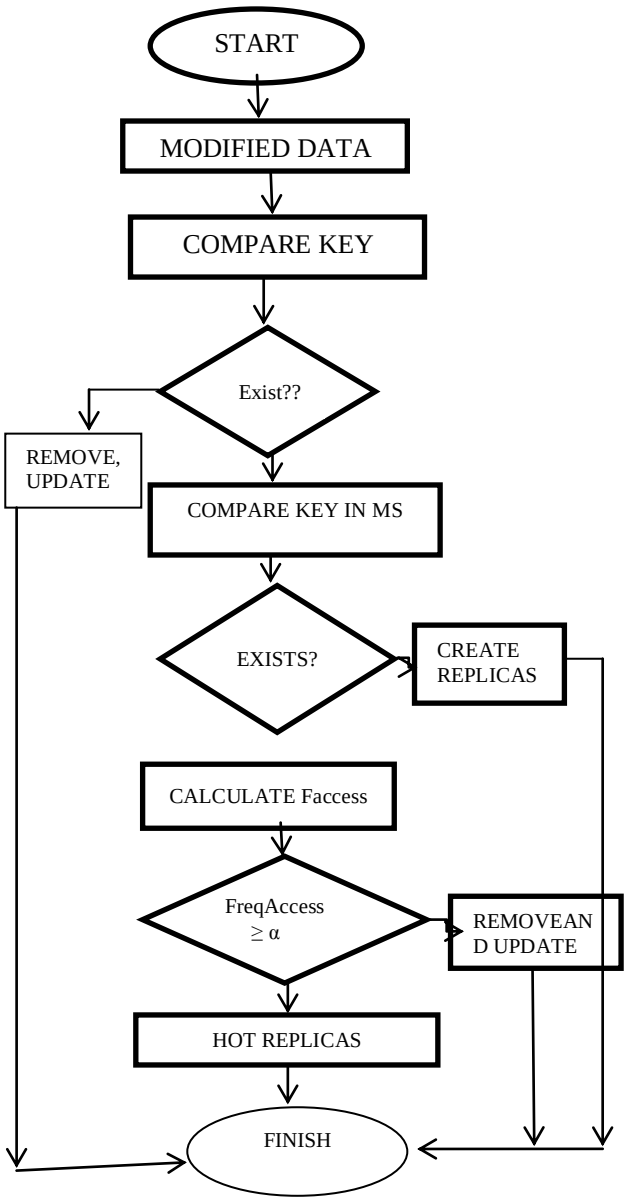

Fig 2. Flowchart of process

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

V .CONCLUSION

In this paper we proposed the architecture of deduplication system for file server and cloud server storage environment and give the process of avoiding deduplication in each stage. Also the data is stored at different servers using load balancing . From this it is concluded that user will be able to avoid deduplication of data over sever and will store the data according to the loads

ACKNOWLEDGMENT

We would like to thank the reviewers for their detailed comments, suggestions and constant support throughout the reviewing process that helped us to significantly improve the quality of paper.

REFERENCES

[1] XiaolongXu and Quntu , "Data Deduplication Mechanism For Cloud Storage Systems," in Nanjing University Of Posts And Telecomunication Nanjing, China, 2015
[2] P. Xie, "Survey on deduplication techniques for storage systems," Computer. Sci., vol. 41, no. 1, pp. 22-30, Jan. 2014.
[3] B. Cai, F. Zhang, and C. Wang, "Research on chunking algorithms of data de-duplication," in Proc. 2012 Int. Conf. Commun., Electron. and Automation Eng., Berlin, 2013, pp. 1019-1025.
[4] Y. Fu, N. Xiao, and F. Liu, "Research and development on key techniques of data deduplication," J. Comput. and Res., vol. 49, no. 1, pp. 12-20, Jan. 2012.
[5] T. Zeng,"Research and implementation of data deduplication technology," M.S. thesis, Dept. Comput. Architecture, Huazhong Univ. Sci. and Tech., Wuhan, Hunan, 2011.