DATA DEDUPLICATION TECHNIQUE USING CONTAINER FOR STORING BIG DATA IN CLOUD

A.Thomas Mary Sheeba
Assistant Professor, Department of IT
Dr. Sivanthi Aditanar College of Engineering
Sheeba_it2005@yahoo.com

Abstract

The amount of data generated by the system is very large and the storage space required to store these data is very high. In this paper a container based data deduplication technique is proposed. The proposed technique chunks the data into fixed sized blocks. After chunking the data into fixed sized blocks, hash values are generated using MD5 algorithm. Then MapReduce technique is used to check whether duplicates are found or not. To check for duplicates the already generated hash values are stored in the container and is checked with the newly generated hash values. If duplicates are found then no need to store the data otherwise the datas are stored into the Cloud.

Keywords: Big Data; Chunking; Container; Deduplication.

I. INTRODUCTION

The International Data Corporation has predicted that the amount of data generated will reach 40 trillion gigabytes in 2020. The big challenging task today is how these huge amount of data are stored for future purpose. These stored data may contain duplicates. The large amount of data are processed within seconds in big companies. Data deduplication is a technique to find the redundant data. In the perspective of Deduplication architecture, there are two stratergies. They are 1)Target based Deduplication and 2)Source based Deduplication. In target based deduplication, the users are unaware of the deduplication technique. The users just upload the data and the server checks for duplication. In Source based deduplication, the users are aware of the deduplication technique and they generate the hash value and the same is send to the server. The server checks for redundancy using the hash value. The data may be in unstructured form without any format or media. This unstructured data may contain duplicate data used at multiple times so to identify duplicate data and create unstructured data into structured data format is a challenging task. To handle this kind of challenging task various authors provided different kind of mechanism like whole file chunking, content defined chunking, and fixed size chunking [1]. In whole file chunking, whole file is taken as chunk and produces hash values to find Duplicate data. Data may be duplicate within file if whole file chunking is used then duplication can be detected within files. And to produce hash values for whole file it may take more computation time. On the other hand, content defined chunking is based on variable size chunking. In this Content defined chunking file is divided into the blocks of the data and then hash values are produced from these blocks to detect duplication id the blocks. To find identical chunks or blocks in content defined chunking mechanism is very difficult task [2]. In Fixed size chunking mechanism, file is divided into fixed size chunks and then produces hashes to find fixed size duplicate chunks. In fixed size chunking there are fixed size chunks are created but when there is some changes in data then there may be a problem boundary shift problem [3][4].

DATA DEDUPLICATION STRATEGY

Data de-duplication technology to identify duplicate data, eliminate redundancy and reduce the need to transfer or store the data in the overall capacity [7][8]. Duplication to detect duplicate data elements, to judge a file, block or bit it and another file, block or bit the same.

Data de-duplication technology to use mathematics for each data element, "hash" algorithms to deal with, And get a unique code called a hash authentication number .. Each number is compiled into a list, this list is often referred to as hash index. At present mainly the file level, block-level and byte-level deletion strategy, they can be optimized for storage capacity.

A. File-level data deduplication strategy

File-level deduplication is often referred to as Single Instance Storage (SIS)[9], check the index back up or archive files need the attributes stored in the file with the comparison. If not the same file, it will store and update the index; Otherwise, the only deposit pointer to an existing file. Therefore, the same file saved only one instance, and then copy all the "stub" alternative, while the "stub" pointing to the original file.

B. Block-level data deduplication technology

Block-level data deduplication technology[1 0] [1 1] to data
stream divided into blocks, check the data block, and determine whether it met the same data before the block (usually on the implementation of the hash algorithm for each data block to form a digital signature or unique identifier) .. If the block is unique and was written to disk, its identifier is also stored in the index; Otherwise, the only deposit pointer to store the same data block's original location. This method pointer with a small-capacity alternative to the duplication of data blocks, rather than storing duplicate data blocks again, thus saving disk storage space. Hash algorithm used to judge duplicate data, may lead to conflict between the hash error. MD5, SHA-I hash algorithm, etc. are checked against the data blocks to form a unique code. Although there are potential conflicts and hash data corruption, but were less likely. To overcome these kinds of drawbacks a container based technique for data deduplication has been presented in this paper. The paper is organized in five sections. In section I introduction has been presented, in section II related work has been discussed, in section III proposed algorithms and system architecture has been presented and section IV covers results and analysis is presented.

II. RELATED WORK

Tang and Won [5] developed a prototype system that is content based file chunking which consist of two subsystems: one is CPU chunking subsystem and other is GPGPU subsystem. This system will decide which subsystem would use chunks.

Manogar and Abirami [6] analyzed different de-duplication techniques and compared these techniques and concluded that variable size data de-duplication is very efficient from other techniques.

Lin *et al.* [7] developed a data reorganize method that is ReDedup it works to address data fragmentation problem and reallocate files and places them on disk.

Wang *et al.* [8] explained about clustering architecture with several storage nodes for data de-duplication. In this architecture, there was a removal data redundancy at file-level and chunk-level and examine for duplicate chunks in all nodes at the same time.

Yu-xuan *et al.* [9] developed a cluster de-duplication system AR-Dedup to reach high data de-duplication rate and low communication overhead and to maintain load balancing. In this system an application-aware method is also used in the de-duplication. In AR-Dedup there were routing was used in the cluster de-duplication.

III. PROPOSED WORK

This section presents the proposed architecture with simple steps.

1. Algorithm for Chunking

- The data is given as input
- The size of the chunk is initialized
- Bytes from the data are extracted
- Output the bytes
- Chunks are created from the input data based on the

above procedure.

2. **MD5 Algorithm**

- Give the input data
- The input data are divided into blocks
- Eight rounds are used to process the blocks
- MD5 digest is produced after performing eight rounds.

C. Description of proposed work

In the proposed algorithm the data is divided into different chunks. To perform this task

we applied fixed size chunking algorithm. In fixed chunking algorithm initialize the number of chunks and size of chunks is to be generated for example size of 32 MB. It indicates file is divided into various chunks of size 32MB. These chunks are used to find duplicate content. After creating chunks using fixed size chunking apply MD5 algorithm to generate hash values of these chunks. These hash values are secret values so that data in chunks cannot be accessed by any other person that may violates security of system. Now these hash values are pushed into the clod for storage. Now initialize different containers which are used to store hash value. Hash values are stored in corresponding containers. Now run the MapReduce programming model to distinguish duplicates hashes of the files. When duplicate hashes are detected then remove the duplicate files from the data and store only unique data into cloud. When new data is stored into cloud then firstly use the fixed size chunking algorithm to create chunks. Then generate hashes from the chunks. And then transfer these chunks for the verification in cloud. Now apply MapReduce model to detect the hashes are duplicate or not. If hashes are detected as duplicate then do not store the data in cloud otherwise stores into containers. This will removes duplicated data and reduces storage capacity that was increased due to duplicate content. Fig 1 shows the architecture of proposed system.

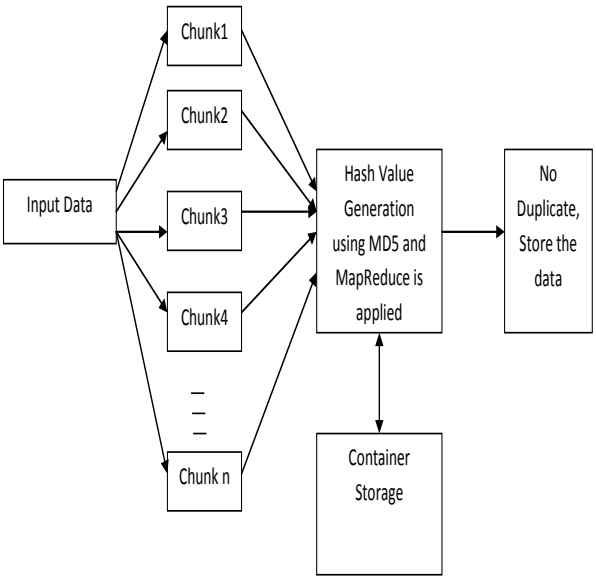

Fig. 1. Proposed work Architecture

IV. RESULTS AND ANALYSIS

The Proposed work is implemented on a computer with configuration Intel i3 CPU with 4.00 GB memory on 32bit OS in Windows. To implement the proposed work the dataset is collected from online resources like e-books and then by using Matlab fixed size chunking algorithm and MD5 hashing algorithm is implemented. The duplicate hash values are detected using the Map Reduce model. If duplicates are found, then the datas are not stored in the cloud and if duplicates are not found then the datas are stored in the cloud. Then the results are compared with the existing techniques.

A. Dataset Used

To implement the proposed work the dataset used is collected from online resources like e-books. Different versions of the same book is taken and compared with the existing technique.

B. Performance metrics

The following metrics are analysed for efficient deduplication.

- Data size after deduplication
  This is the size of data reduced after deduplication
- Time to chunk the data
  This is the time taken to produce chunks
- Time to generate hash value
  This is the time taken to generate hash value using the proposed algorithm
- Deduplication ratio
  This is the ratio of deduplication which is nothing but the data after deduplication/ data before deduplication.
  Ratio of Deduplication=(Size)a/(Size)b
  Where (Size)a=Data after Deduplication
  (Size)b=Data before Deduplication

Technique	Data before deduplication	Data after deduplication (GB)	Deduplication Ratio
Fixed	3.8 GB	2.7	0.71
Container		2.1	0.55
Fixed	2.4 GB	1.6	0.66
Container		1.1	0.45
Fixed	1.2 GB	0.75	0.62
Container		0.7	0.58

Table 1: Container Based Deduplication Technique Results

The above table shows the deduplication ratio of data of varying sizes. The proposed method is compared with the existing methods.

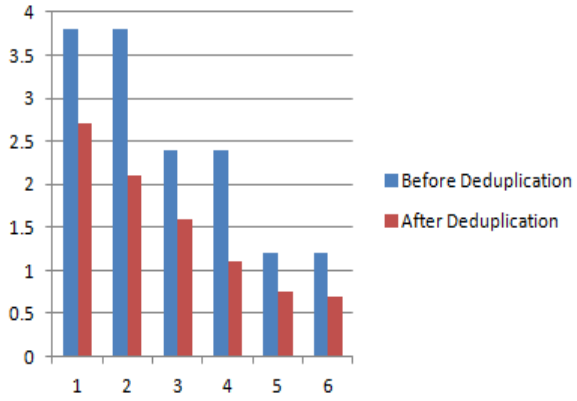

Fig 2: Data before deduplication Vs Data after Deduplication.

V. CONCLUSION

In big data storage data is too large to store in physical storage devices. So this is an big challenging task. To solve this problem we store the data in the cloud using deduplication technique. This paper presents a container based technique to avoid the redundant copies of data. In proposed technique different containers are used to store data and when same data is accessed by map reduce i.e. already stored in container then that data will be discarded so this technique definitely increases efficiency of big data storage. Results shows that in proposed mechanism deduplication ratio is high, data size reduction is high hash time and chunk time is low as compare to existing fixed size chunking technique. In future we will continue working on it and refine results with low computation time also we propose new mechanism in which all modules are combined like chunking, deduplication and hashing that can find more duplicate content and remove them in proper manner with less time duration.

REFERENCES

[1] Qinlu He, Zhanhuai Li and Xiao Zhang, "Data Deduplication Techniques", 2010 International Conference on Future formation Technology and Management Engineering, IEEE 2010, pp. 430-433.

[2] Won, Lim and Min, "MUCH: Multithreaded Content-Based File Chunking". IEEE Transactions on Computers, IEEE 2015, pp. 1-6.

[3] Wen Xia, Hong Jiang, Dan Feng and Lei Tian, "DARE: A Deduplication-Aware Resemblance Detection and Elimination Scheme for Data Reduction with Low Overheads", IEEE Transactions on Computers, IEEE 2015, pp.1-14.

[4] Yukun Zhou, Dan Feng, Wen Xia, Min Fu, Fangting Huang,

[5] Yucheng Zhang and Chunguang Li, "SecDep: A User-Aware

[6] Efficient Fine-Grained Secure Deduplication Scheme with

[7] Multi-Level Key Management", IEEE 2015, pp. 1-4.

[8] Zhi Tang and Youjip Won, "Multithread Content Based File Chunking System in CPU-GPGPU Heterogeneous Architecture", 2011 First International Conference on Data Compression, Communications and Processing, IEEE 2011, pp. 58-64.

[9] E. Manogar and S. Abirami, "A Study on Data Deduplication Techniques for Optimized Storage", 2014 Sixth International Conference on Advanced Computing(lCoAC), IEEE 2014, pp. 161-166.

[10] Bin Lin, Shanshan Li, Xiangke Liao and Jing Zhang, "ReDedup: Data Reallocation for Reading Performance Optimization in Deduplication System", 2013 International Conference on Advanced Cloud and Big Data, IEEE, pp.117-124.

[11] Guohua Wang, Yuelong Zhao, Xiaoling Xie, and Lin Liu, "Research on a clustering data de-duplication mechanism based on Bloom Filter", IEEE 2010, pp. 1-5.

[12] XING Yu-xuan, XIAO Nong, LIU Fang, SUN Zhen and HE Wan-hui, "AR-Dedupe:

[13] An Efficient Deduplication Approach for Cluster Deduplication System", J. Shanghai Jiaotong Univ. (Sci.), 2015, pp. 76-81.

[14] Kun Gao and Xuemin Mao, " Research on massive tile data management based on Hadoop", 2016 2nd International Conference on

Information Management (ICIM), IEEE 2016, pp. 16-20.

[15] Apache Hadoop, http://hadoop. apache.org, Acessed on 11-June-2016.

[16] Destor, https://github.com /fomy/ destor, Accessed on 15-June-2016.

[17] College Scorecard, https:// catalog.data.gov/dataset/college-scorecard, Accessed on 8-June-2016.

[18] ZCTA, https://catalog.data.gov /dataset/tiger-line-shapefile-2015-2010-nation-u-s-2010-census-5-digit-zip-code-tabulation-area-zcta5-na, Accessed on 8-June-2016.

[19] Lu, Jin and Du, "Frequency Based Chunking for Data De-Duplication", 2010 18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, IEEE 2010, pp. 1-5