International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 06 | June-2016 www.irjet.net p-ISSN: 2395-0072

A Dynamic Deduplication Approach for Big Data Storage

Harshita Sharma[1], Shobha[2]

[1]*P.G student, Dept. Of Computer Science and Engineering, UIET KUK, India*

[2]*P.G student, Dept. Of Computer Science and Engineering, UIET KUK, India*

Abstract- *As data is increasing every day, so it is very challenging task to manage storage devices for this explosive growth of digital data. Data reduction has become very crucial problem. Deduplication approach plays a vital role to remove redundancy in large scale cluster computing storage. As a result, deduplication provides better storage utilization by eliminating redundant copies of data and saving only one copy of data in storage devices. In this paper we propose a dynamic deduplication approach with higher efficiency, in which the data files are distributed across multiple tightly coupled computers shared by multiple users. As an effective data elimination approach it exploits data redundancy. Data deduplication first divides large data objects into smaller parts called chunks and represent them by their unique hash values using MD5 or SHA1 to identify duplicate data. The experimental results with real big data show that represented Deduplication approach improves DER (data elimination ratio) and gains storage space.*

Keywords- Deduplication, Whole file chunking, Content based chunking, Fixed size chunking, Deduplication ratio, Deduplication gain

1. INTRODUCTION

Enormous increase in amount of digital data has motivated the need to upgrade the data storage devices. The data produced by social networking sites, servers, personall computers, IT companies includes different forms of data such as images, videos, text files, PDFs etc. Big Data[1] in 2010 is defined as data stream which cannot be wield and processed by general computer systems. Deduplication is a well known technique that mainly focuses to save storage space by removing redundant copies of data. The process of deduplication involves partitionaing of input data into blocks of fixed or variable size. Then provide each chunk with a unique hash value calculated by MD5 or SHA1. A lookup process is then followed to check out the redundant chunk by comparing the hash values with already stored hash values. After comparison, if the chunk is not found, the chunk index is updated with the new data, else reference is created pointing the existing data. Deduplication can be performed on three levels named as whole file level, block level or byte level. In file level deduplication, whole file is considered as a single chunk and the hash value for whole file is generated. Block level deduplication performs more fined deduplication by dividing each file into blocks, removing data redundancy at block level. Byte level deduplication compares data chunks byte-by-byte and performs checks for duplicate chunks more accurately. Being a powerful data reduction approach deduplication divides large data sets into small chunks. In this paper, a dynamic deduplication approach has been used where in which Content Defined Chunking (CDC) is used to create variable size chunks which defines the breakpoints with the use of Basic Sliding Window (BSW). Frequency Based Chunking (FBC) identifies the frequently obtained chunks by following a two phase process [4], one is chunk frequency estimation using an estimation algorithm and second is two stage chunking in which coarse grain and then fine grain chunks are obtained using CDC. Furthermore, hash based algorithms are applied on these fine grain chunks and unique hash values are obtained for every chunk.

The objective of our studies is to improve the deduplication ratio (DER), deduplication gain, storage efficiency of input data streams that are being passed. This paper includes a theoretical analysis in a cluster based environment [5].

The remainder of this paper is organised as follows: The related work of existing deduplication techniques is provided in section II. In section III, We represent our approach to improve the performance of data de-duplication, performance evaluation and results are presented in section IV. Finally section V concludes this paper.

2. RELATED WORK

Data deduplication is a lossless technique that has become very convenient in large scale storage systems for space optimization. The de-duplication process has been classified as inline process and post-process. The inline process performs the data deduplication before it is being written to the storage device unlike post process which involves the identification of duplicate data after writing data to the storage device. The input data streams can be divided into parts called chunks either in fixed size or variable size. There are various data deduplication

algorithms including content defined chunking, static chunking, delta encoding and whole file chunking. Static chunking is robust and efficient but it has a limitation of "boundary shift problem". Variable size chunking provides the solution by using a sliding window. Venti [6], a well known static chunking scheme uses 160bit SHA1 hash value to address the block's content as its identifier. So, the blocks with identical addresses are easily identified and stored once. CDC [7] approach of data de-duplication achieve high deduplication ratio, but it is too much time consuming as compared with the other approaches. CDC prevents the boundary shift problem of the static chunking approach by partitioning the input data stream according to the content of the data but not to the local boundary. A CDC based network file system LBFS [8] eliminates redundant data in low bandwidth networks. It is the first file system that partitions data stream into variable sized chunks by finding proper cut point for each chunk. Chunking algorithms like TTTD [9], TTTD-S provide much more improved results with variable sized chunks.

Delta Encoding [10] stores only the differences between sequential data. Backup systems adopt this technique to reduce the cost in amount of storing differing versions of same piece of information. A two phase algorithm FBC uses the frequency of chunks to eliminate the redundant data. At first phase, the frequency is obtained by using a statistical chunk frequency estimation algorithm. Then at second phase, coarse grained chunks are obtained and then further fine grained chunks using CDC. Fine grained chunk uses the frequency of chunks obtained by a statistical chunk frequency estimation algorithm. Then, a two stage procedure is followed to obtain fine grained chunks. At first level, CDC is employed to obtain the coarse-grained chunks. Then fine grained chunks are obtained by using the unique hash values produced by using hash function at second level. MD5 or SHA1 are some of the algorithms used to produce the unique identifiers called hash value of the chunks. SHA1 is more secure but not faster than MD5 [11]. Hash function MD5 is the backbone of Byte index chunking algorithm [12]. The process of algorithm involves elimination of identical data between different files stored in different machines very rapidly. The algorithm transfers only unique chunks from source to destination.

Improved de-duplication throughput during de-duplication processes is also an important concern. Extreme binning, a scalable and stateless routing algorithm [13] works on file similarity that maintain the throughput and allow maximum storage by reducing the redundant data. De-Frag [14] effectively improves the de-duplication throughput and de-duplication efficiency. With the explosive growth of data, a de-duplication system of single-node is not capable to satisfy with its performance in data protection and storage. Thus, globally based de-duplication system improves the lacking performance of single-node de-duplication system. AR-Dedupe [15], a cluster de-duplication system achieves high data de-duplication rate with low communication overhead through routing. It maintains the load balancing with the increasing files and de-duplication server nodes.

3. PROPOSED APPROACH

In this section, we will describe the architecture and its key features followed by the detailed discussion of its design and working.

3.1 Architecture Overview

The architecture is designed to improve the Data elimination ratio and storage space in a globally based storage system.

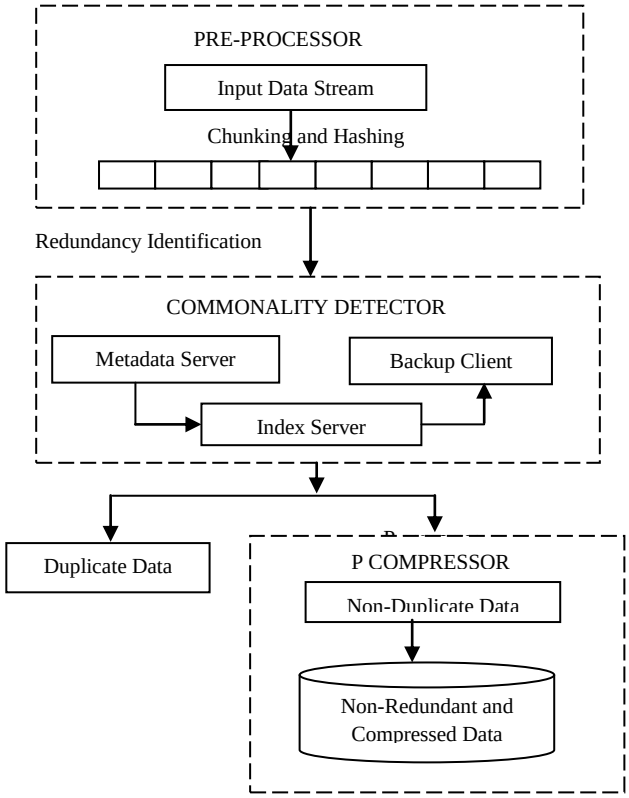

Fig-1: Architecture of Global Based Data Deduplication.

As shown in figure 1, the architecture consists of three functional modules named as: Pre-processor module, Commonality Detector module, and Pcompressor module, in addition with the three key features namely as Metadata Server, Index Server, Backup Client which are defined below:

1. Pre-processor: The pre-processor performs the very first step of a de-duplication process by dividing the

input data into chunks. It uses rabin based CDC for the formation of chunks and generation of their hash values.

2. Commonality Detector: In this module, there are three key features namely Index Server, Metadata Server and Backup Client.

- *Metadata Server:* The hash values of chunks generated by pre-processor are stored at metadata server. The stored values are further being forwarded to the index server for comparison of hash index.
- *Index Server:* The hash values of input data are then compared with the already stored hash values in the index at index server for duplicate detection. If similar data is found then it is redundant. If not, the index will be updated. The Frequency and Commonality detector detects the most frequent chunk occurring which is to be stored at backup client for further use if needed.
- *Backup Client:* The backup Client stores the most frequently occurring chunks. The tiger hash is employed to generate hash values because of its fast processing. On the basis of generated hash values the redundant and non-redundant data is separated.

3. Pcompressor: the Pcompressor employ Pcompress[16] utility at the redundant data. The Pcompress utility provides much meaningful and useful data after compression.

4. PERFORMANCE EVALUATION

In this section, we evaluate the data reduction in terms of data elimination ratio and deduplication gain. We compare the existing file size with the size after employing our approach. In order to refer to a real scenario, we use a real data set.

4.1 Experimental Setup

We have designed a global system architecture with, a server node with operating system Windows 8 running on a Intel(R) Core i3 processor at 2.53GHZ, with a 4GB RAM; two Backup nodes namely C1 and C2, C1 of Intel(R) Core i3 processor at 2.30GHZ with a 4GB RAM; C2 of Intel(R) Core i3 processor at 2.30GHZ with a 2GB RAM.

4.2 Experimental Results

Our experiment was performed on three different real datasets. We have improved DER and deduplication gain effectively by using our proposed approach as shown in fig. 2 and fig. 3 respectively. Table1 shows how much de-dupe gain and ratio was achieved on different datasets [17].

Table1: Deduplication gain and ratio of three different datasets.

Data Set	Original Size	Reduced Size	De-dup Gain	DER
Dataset 1	13 GB	4 GB	3.25	0.692
Dataset 2	14 GB	4.5 GB	3.11	0.678
Dataset 3	15 GB	5 GB	3.00	0.666

A. Deduplication Elimination ratio(DER): $\frac{Byte\ In}{Byte\ Out}$

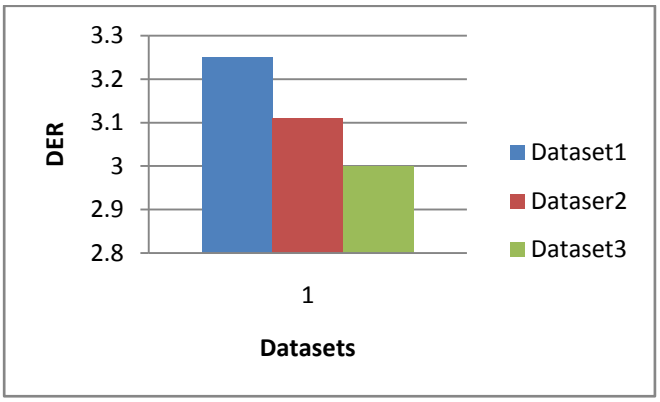

Fig-2: Deduplication Elimination ratio on different datasets.

B. De-dup Gain: $1 - \frac{Byte\ In}{Byte\ Out}$

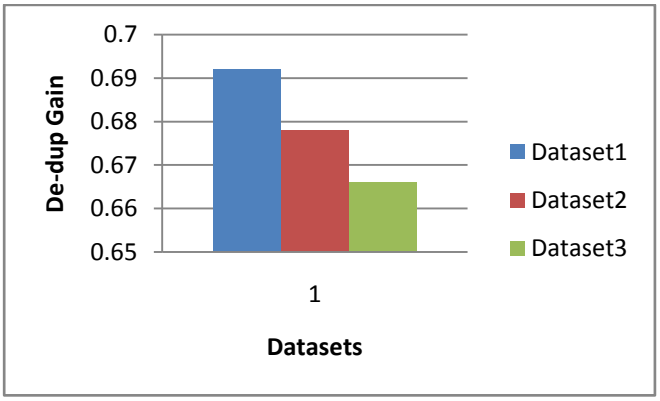

Fig-3: Deduplication gain on different datasets.

5. CONCLUSION

In this paper, we present global approach with improved efficiency. We proposed an efficient system to improve the overall DER and storage space gain. The key idea is to optimize the storage space utilization by employing

Pcompress utility to the redundant chunks. To verify the feasibility and effectiveness of the proposed approach we have designed a cluster based system and perform the evaluation experiment. The experiment result shows that the DER and storage space gain has significantly improved by our proposed approach.

6. REFERENCES

[1] Min Chen, Shiwen Mao and Yunhao Liu," Big Data: A Survey", Springer Science, Springer, pp. 171–209, **2014**.

[2] Jaehong Min, Daeyoung Yoon, and Youjip Won, "Efficient De-duplication Techniques for Modern Backup Operation", IEEE Transactions on Computers, IEEE, vol. 60, no. 6, pp. 824-840, **2011**.

[3] Dutch T. Meyer and William J. Bolosky, "A Study of Practical De-duplication", ACM Transactions on Storage, vol. 7, no. 4, pp. 14.1-14.20, **2012**.

[4] Guanlin Lu, Yu Jin, and David H.C. Du,"Frequency Based Chunking for Data De-duplication", 18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, IEEE, pp. 287-296, **2010**.

[5] Xingyu Zhang and Jian Zhang, "Data De-duplication Cluster Based on Similarity Locality Approach" IEEE International Conference on and IEEE Cyber, Physical and Social Computing, IEEE, pp. 2168-2173, **2013**.

[6] Sean Quinlan and Sean Dorward,"Venti:A New Approach to Archival Storage",Conference on File and Storage Technologies, pp. 89–102,**2002**.

[7] Kave Eshghi and Hsiu Khuern Tang,"A Framework for Analyzing and Improving Content-Based Chunking Algorithms", Hewlett Packard Development Company and Intelligent Enterprise Technologies Laboratory, Feb **2005**.

[8] Athicha Muthitacharoen, Benjie Chen, and David Mazieres, "A Low-Bandwidth Network File System" , in the proceeding of the 18th ACM symposium on Operating
System principles(SOSP 01) Review, vol. 35, no. 5, pp. 174- 187,2001.

[9] Jyoti Malhotra and Jagdish Bakal,"A Survey and Comparative Study of Data De-duplication Techniques", IEEE International Conference on Pervasive Computing (ICPC), IEEE, pp. 1-5, **2015**.

[10] Miklos Ajtai, Randal Burns, Ronald Fagin, Darrell D.E. Long, and Larry Stockmeyer, "Compactly Encoding Unstructured Input with Differential Compression", Journal of Association for Computing Machinery, vol. 49, no. 3, pp. 318-367, **2003**.

[11] Piyush Gupta and Sandeep Kumar,"A Comparative Analysis of SHA and MD5 Algorithm" ,International Journal of Computer Science and Information Technologies, vol. 5, no. 3, pp. 4492-4495, **2014**.

[12] Ider Lkhagvasuren, Jung Min So, Jeong Gun Lee, Jim Kim and Young Woong Ko, "Design and Implementation of Storage System using Byte-index Chinking Schemes", International Journal of Security and Its Applications, vol. 8, no. 1, pp. 33-42, **2014**.

[13] D. Bhagwat, Kave Eshghi, D.D.E. Long and M. Lillbridge, "Extreme Binning: Scalable, Parallel De-duplication for Chunk-Based File Backup", Proceeding of 17th IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecomunation System (MASCOTS), IEEE, pp. 1-9, **2009**.

[14] Yujuan Tan, Zhichao Yan, Dan Feng, Xubin He, Qiang Zou and Lei Yang,"De-Frag: An Efficient Scheme to Improve De-duplication Performance via Reducing Data Placement De-linearization", Springer Cluster Computing, Springer, vol. 18, no. 1, pp. 79–92, **2015**.

[15] Yu-xuan Xing, Nong Xiao, and Fang Liu, Zhen Sun, Wan-hue He "AR-Dedupe: An Efficient De-duplication Approach for Cluster De-duplication System" , Journal of Shandhai Jiaotong University, vol. 15, no. 1, pp. 76-81, **2015**.

[16] Moinak Ghosh,"P Compress", https://github.com/moinakg/pcompress.

[17] www.freedb.com.