A Mixture Distribution Based System Using Data Chuncking & Compression On Cloud

Omkar Bhat[1], Kunal Ahirrao[2], Ankush Kshirsagar[3], Aakash Ankush[4], Prof.Kiruthika Arunkumar[5]
Department of Computer Enigneering
[1, 2, 3, 4, 5] DYPIEMR,Akurdi,Pune.

Abstract-*Big sensing data is extensively used in both industry and scientific research applications where the data is generated with huge volume. Cloud computing provides a best platform for big sensing data processing and storage. It moves around four important areas of analytics and Big Data, namely (i) data management and supporting architectures; (ii) model development and scoring;(iii) visualisation and user interaction; and (iv) business models[7]. 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 redundant data waste alot of storage space. Data deduplication can effectively reduce the size of data by eliminating redundant data in storage systems. In cloud data storage, the deduplication technology plays a major role. In the deduplication technology, data are broken down into multiple pieces called "chunks". The Two Thresholds Two Divisors (TTTD) algorithm is used for chunking mechanism and is used for controlling the variations of the chunk-size.*

Keywords-cloud computing, data chunk, data compression, big sensing data, scalability.

I. INTRODUCTION

It is becoming a big necessity that we need to process big data from multiple sensing systems. Cloud storage systems are able to give low-cost, convenient and good network storage service for users, which makes them more popular. However, the storage pressure on cloud storage system caused by the huge growth of data is growing by the day, especially a vast amount of repetitive waste plenty of storage space. Data deduplication can operatively reduce the size of data by excluding repetitive data in storage systems. However, current researches on data deduplication, which mainly concentrate 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[4]. Deduplication applied in cloud storage systems can minimize the size of data and save the network bandwidth, the dynamicity of data in cloud storage systems are different from backup and archive systems, which brings different approaches for the study of data deduplication in cloud storage systems. Here, the dynamic characteristics of data are caused by dynamic sharing between multiple users. For example, the same data accessed by different users and the access frequency of different data at the same time is different, the access frequency of the same data changes overtime and duplicated data appears again in different (storage nodes)nodes for data modification by users[6].There are many different deduplication approaches depending on the range of deduplication (locally or globally), the position of deduplication (at the client or server side), the time of deduplication (inline or offline), and the granularity of deduplication (file-level or chunk-level). The process of deduplication mainly comprises four steps: (1) chunking; (2) calculating fingerprint; (3) fingerprint lookup (finding out the redundancy by fingerprint comparison); storing new data[6]. Chunking can break a file into small parts called chunks for detecting more redundancy. There are several typical chunking strategies of data deduplication [5], such as whole-file chunking, fixed- size chunking, content-defined chunking, and Two Thresholds Two Divisors(TTTD) chunking.

II. BACKGROUND

Some techniques have been proposed to process big data with traditional data processing tools such as database, traditional compression, machine learning, or parallel and distributed system. Those current popular techniques for big data processing on Cloud will be introduced and analyzed[1]. Nowadays, lots of big data sets or streams come from sensing systems which are widely deployed in almost every corner of our real world to assist our everyday life. In order to cope with that huge volume big sensing data, different techniques can have been developed, on-line or off-line, centralized or distributed. Naturally, the computational power of Cloud comes into the sight of scientist for big sensing data processing. With increasing number of cores on a chip, the demand of cache and main memory capacity is on rise. However, due to energy and bandwidth constraints, using large-size memory systems becomes infeasible and this has led to decreasing memory-to-core ratio in recent processors[2].

Compression can help in storing the data in smaller amount of physical memory, thus, giving the impression of a large size memory to the application or end-user. Cache compression (CC) can reduce costly off-chip accesses and memory compression can reduce page faults which trigger

disk accesses. Compression also helps in reducing the memory bandwidth requirement, since multiple consecutive compressed blocks can be fetched at the cost of accessing a single uncompressed block [10][11][12]. Huffman coding works by analyzing the data to determine the probability of different elements. Afterwards, the most probable elements are coded with fewer number of bits than those which are least probable. Thus, Huffman coding uses a variable-length coding scheme. LZ compression algorithm works by replacing repeated occurrences of data elements with references to a single copy of that element existing earlier in the uncompressed data. Several variants of it have been proposed in the literature, which are used by various techniques[2].

2.1 Huffman Coding:

Huffman Encoding Algorithms use the probability distribution of the alphabet of the source to develop the code words for symbols. The frequency distribution of all the characters of the source is calculated in order to calculate the probability distribution. According to the probabilities, the code words are assigned. Shorter code words for higher probabilities and longer code words for smaller probabilities are assigned. For this task a binary tree is created using the symbols as leaves according to their probabilities and paths of those are taken as the code words. Two families of Huffman Encoding have been proposed: Static Huffman Algorithms and Adaptive Huffman Algorithms. Static Huffman Algorithms calculate the frequencies first and then generate a common tree for both the compression and decompression processes. Details of this tree should be saved or transferred with the compressed file. The Adaptive Huffman algorithms develop the tree while calculating the frequencies and there will be two trees in both the processes. In this approach, a tree is generated with the flag symbol in the beginning and is updated as the next symbol is read[9].

2.2 TTTD:

The TTTD algorithm was proposed by HP laboratory at Palo Alto, California. This algorithm use same idea as the BSW algorithm does. In addition, the TTTD algorithm uses four parameters, the maximum threshold, the minimum threshold, the main divisor, and the second divisor, to avoid the problems of the BSW algorithm. The maximum and minimum thresholds are used to eliminate very large-sized and very small-sized chunks in order to control the variations of chunk-size. The main divisor plays the same role as the BSW algorithm and can be used to make the chunk-size close to our expected chunk-size. In usual, the value of the second divisor is half of the main divisor. Due to its higher probability, second divisor assists algorithm to determine a backup breakpoint for chunks in case the algorithm cannot find any breakpoint by main divisor[5].

III. MODULE STRUCTURE

our framework is divided into six modules.

1) User Registration
2) Compression
3) Decompression
4) Cloud Connectivity
5) Authentication
6) Performance Analyzer

3.1. SYSTEM ARCHITECTURE:

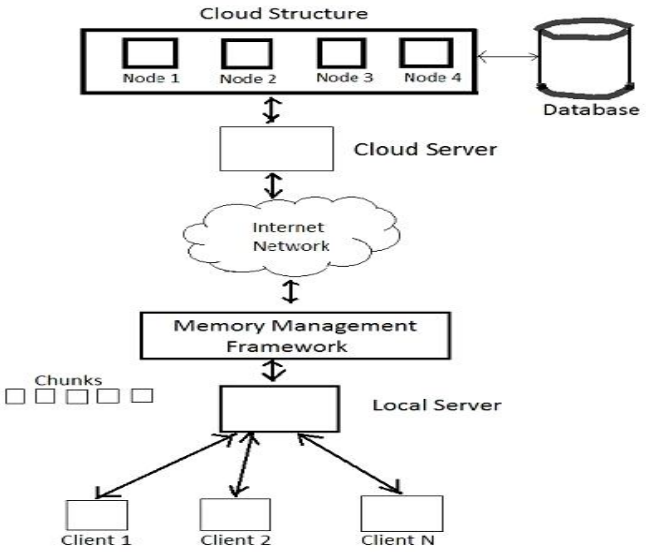

fig 3.1 – System Architecture

Our system will be working on processing big data on cloud by using compression technique working Huffman algorithm, by overcoming some drawbacks of Map Reduce used in existing system. The data which is to be stored on cloud after compression will be available in its original size on local server. This data on local server on time of processing will be divided into chunks so that the data will be processed parallel and faster. To divide big data into chunks we will be using TTTD algorithm. The compression will be applied on every single chunk on cloud server and we will be showing that till which level the big data is being compressed and what the compression level bar will show that till what extent the file size has been compressed while storing on cloud. One of the best advantage of out proposed system will be that compression will be applied on any type of file such as audio, video or text present in user's system by applying Huffman algorithm , while in existing system the compression is applied only on text data using Map Reduce Algorithm.

3. 2. Architectural Techniques Use For Compression :

3.2.1 Huffman Coding:

Huffman Encoding Algorithms use the probability distribution of the alphabet of the source to develop the code words for symbols. The frequency distribution of all the characters of the source is calculated in order to calculate the probability distribution. According to the probabilities, the code words are assigned. Shorter code words for higher probabilities and longer code words for smaller probabilities are assigned. For this task a binary tree is created using the symbols as leaves according to their probabilities and paths of those are taken as the code words. Two families of Huffman Encoding have been proposed: Static Huffman Algorithms and Adaptive Huffman Algorithms. Static Huffman Algorithms calculate the frequencies first and then generate a common tree for both the compression and decompression processes. Details of this tree should be saved or transferred with the compressed file. The Adaptive Huffman algorithms develop the tree while calculating the frequencies and there will be two trees in both the processes. In this approach, a tree is generated with the flag symbol in the beginning and is updated as the next symbol is read[9].

IV. CHUNKING MECHANISM

There are two approaches in partitioning a file into chunks: fixed size chunking and variable size chunking.

4.1 Fixed Size Chunking:

In fixed size chunking, a file is partitioned into fixed size units, e.g., 8 KByte blocks. It is simple, fast, and computationally very cheap. A number of proceeding works have adopted fixed-size chunking for backup applica-tions [42] and for large-scale file systems [18]. However, when a small amount of content is inserted to or deleted from the original file, the fixed size chunking may generate a set of chunks that are entirely different from the original ones even though most of the file contents remain intact.

4.2 Variable Size Chunking:

Variable size chunking partitions a file based on the content of the file, not the offset. Variable size chunking is relatively robust against the insertion/deletion of the file. The Basic Sliding Window (BSW) algorithm [19] is widely used in variable size chunking.

4.3Concept of TTTD:

The TTTD algorithm was proposed by HP laboratory at Palo Alto, California. This algorithm use same idea as the BSW algorithm does. In addition, the TTTD algorithm uses four parameters, the maximum threshold, the minimum threshold, the main divisor, and the second divisor, to avoid the problems of the BSW algorithm. The maximum and minimum thresholds are used to eliminate very large-sized and very small-sized chunks in order to control the variations of chunk-size. The main divisor plays the same role as the BSW algorithm and can be used to make the chunk-size close to our expected chunk-size. In usual, the value of the second divisor is half of the main divisor. Due to its higher probability, second divisor assists algorithm to determine a backup breakpoint for chunks in case the algorithm cannot find any breakpoint by main divisor[5].

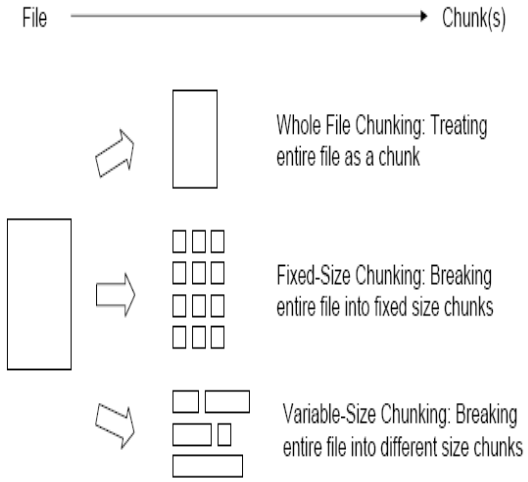

Fig-TTTD Workflow

V. FILE SHARING MECHANISM

In the file sharing network, each peer is responsible for attempting to maximize its own download rate and obtain new pieces quickly. In BitTorrent, peers achieve this by downloading from whoever they can and deciding which peers to upload to via the TFT strategy. Our system differs from BitTorrent in which peers can only make a request to others for downloading. Meanwhile they should contribute to others according to the choking algorithm, so that they can upgrade their contribution coefficient to help them get new pieces. We propose the file sharing mechanism which includes the choking algorithm for peers to share file and the peer selection algorithm to decide which peers to unchoke.

VI. EXPERIMENTAL RESULT

6.1 Dataset 1:

Sr.no	Document Type	Extension
1	Word document	.doc
2	Word document	.doc
3	Word document	.doc

Table-6.1: Dataset 1

6.2 Dataset 2:

Sr.no	Document Type	Extension
1	Image document	.jpg
2	Image document	.jpg
3	Image document	.jpg

Table-6.2: Dataset 2

6.3 Dataset 3:

Sr.no	Document Type	Extension
1	Pdf document	.pdf
2	Pdf document	.pdf
3	Pdf document	.pdf

Table-6.3: Dataset 3

6.4 Performance Evaluation:

Data Set	Original Data Size	Compressed data size
DS1	153 KB	85 KB
DS2	7110 KB	6053 KB
DS3	1813 KB	1418 KB

Table-6.4: Performance Evaluation

VII. CONCLUSION

On the basis of our experiments we conclude that using compression and decompression technique we can efficiently utilize the available cloud space. Storing the compressed data over cloud manages the cloud space efficiently and also while the data is to be accessed the data is decompressed and can be used.This helps in maintaining the exact quality of data.

ACKNOWLEDGMENT

It gives me an immense pleasure and satisfaction in submitting this paper. In this endeavor of preparing this paper many people gave a helping hand, I would thank them all. I heartily pay gratitude to my guide Prof.Kiruthika Arunkumar and Prof. Nareshkumar R.M/ Prof. Ishwar Kalbandi, Project Coordinator of Academic Research who gave me this opportunity to work upon this topic. I am also grateful to Prof. P.P. Shevatekar, Head of Computer Engineering Department, DYPIEMR for her indispensable support and suggestions.

REFERENCES

[1] Chi Yang, Jinjun Chen, “A Scalable Data Chunk Similarity based Compression Approach for Efficient Big Sensing Data Processing Cloud”, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, February 2016.

[2] Sparsh Mittal, Member, IEEE and Jeffrey S. Vetter,Senior Member, IEEE ”A Survey Of Architectural Approaches for Data Compression in Cache and Main Memory Systems” IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 27, NO. 5, MAY 2016.

[3] Santoshi Tsuchiya, Yoshinori Sakamoto, Yuichi Tsuchimoto, Vivian Lee, “Big Data Processing on Cloud Environment”, FUJITSU Science and Technology Journal, 48(2):159-168, 2012.

[4] Venish and K. Siva Sankar, “Study of chunking algorithm in Data Deduplication”, Proceedings of the International Conference on Soft Computing Systems, Advances in Intelligent Systems and Computing 398, DOI 10.1007/978-81-322-2674-1_2.

[5] Chang, BingChun, "A Running Time Improvement for Two Thresholds Two Divisors Algorithm" (2009).Master's Projects. Paper 42.

[6] K. Tanaka and A. Matsuda, “Static energy reduction in cache memories using data compression,” in Proc. IEEE TENCON, 2006,pp. 1–4.

[7] S. Roy, R. Kumar, and M. Prvulovic, “Improving system performance with compressed memory,” in Proc. Int. Parallel Distrib. Process. Symp., 2001, pp. 7–13

[8] J.-S. Lee, W.-K. Hong, and S.-D. Kim, “Design and evaluation of a selective compressed memory system,” in Proc. Int. Conf. Comput. Des., 1999, pp. 184–191.M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.