ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2016

A Survey on Compaction Technique for Big Data Processing on Cloud

Jaai Arankalle, Saee Joshi, Rutuja Limaye, Shubangi Bagade

B.E. Student, Dept. of Computer Engineering, Siddhant College of Engineering, Pune, Maharashtra, India

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) visualization 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. RELATED WORK

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

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2016

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].

III. COMPRESSION ALGORITHM

A. RUN LENGTH ENCODING ALGORITHM :

Run Length Encoding or simply RLE is the simplest of the data compression algorithms. The consecutive sequences of symbols are identified as runs and the others are identified as non runs in this algorithm. This algorithm deals with some sort of redundancy. It checks whether there are any repeating symbols or not, and is based on those redundancies and their lengths. Consecutive recurrent symbols are identified as runs and all the other sequences are considered as non-runs. For an example, the text "ABABBBBC" is considered as a source to compress, then the first 3 letters are considered as a non-run with length 3, and the next 4 letters are considered as a run with length 4 since there is a repetition of symbol B. The major task of this algorithm is to identify the runs of the source file, and to record the symbol and the length of each run.[9]

B. 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].

C. THE SHANNON FANO ALGORITHM :

This is another variant of Static Huffman Coding algorithm. The only difference is in the creation of the code word. All the other processes are equivalent to the above mentioned Huffman Encoding Algorithm [8][9].

D. ARITHMETIC ENCODING :

In this method, a code word is not used to represent symbol of the text. Instead it uses fraction to represent the entire source message. The occurrence probabilities and the cumulative probabilities of a set of symbols in the source message are taken into account. The cumulative probability range is used in both compression and decompression processes. In the encoding process, the cumulative probabilities are calculated and the range is created in the beginning. While reading the source character by character, the corresponding range of the character within the cumulative probability range is selected. Then the selected range is divided into sub parts according to the probabilities of the alphabet. Then the next character is read and the corresponding sub range is selected. In this way, characters are read repeatedly until the end of the message is encountered. Finally a number should be taken from the final sub range as the output of the encoding process. This will be a fraction in that sub range. Therefore, the entire source message can be represented using a fraction. To decode the encoded message, the number of characters of the source message and the probability/frequency distribution are needed [8][9].

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2016

E. THE LEMPEL ZEV WELCH ALGORITHM :

Dictionary based compression algorithms are based on a dictionary instead of a statistical model. A dictionary is a set of possible words of a language, and is stored in a table like structure and used the indexes of entries to represent larger and repeating dictionary words. The Lempel-Zev Welch algorithm or simply LZW algorithm is one of such algorithms. In this method, a dictionary is used to store and index the previously seen string patterns. In the compression process, those index values are used instead of repeating string patterns. The dictionary is created dynamically in the compression process and no need to transfer it with the encoded message for decompressing. In the decompression process, the same dictionary is created dynamically. Therefore, this algorithm is an adaptive compression algorithm[9].

F. MAP REDUCE:

Map-Reduce is a programming model and an associated implementation for processing and generating large number of data sets with the parallel and distributed algorithm on a cluster. A Map-Reduce program is composed of a Map() procedure (method) that performs filtering and sorting. The model is inspired by the map and reduce functions commonly used in functional programming, although their purpose in the Map-Reduce framework is not the same as in their original forms. The key contributions of the Map Reduce framework are not the actual map and reduce functions, but the scalability and fault-tolerance achieved for a variety of applications by optimizing the execution engine once. As such, a single-threaded implementation of Map Reduce will usually not be faster than a traditional (non-Map Reduce) implementation; any gains are usually only seen with multi threaded implementations. The use of this model is beneficial only when the optimized distributed shuffle operation (which reduces network communication cost) and fault tolerance features of the Map Reduce framework come into play.

G. *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].

IV. SYSTEM ARCHITECTURE

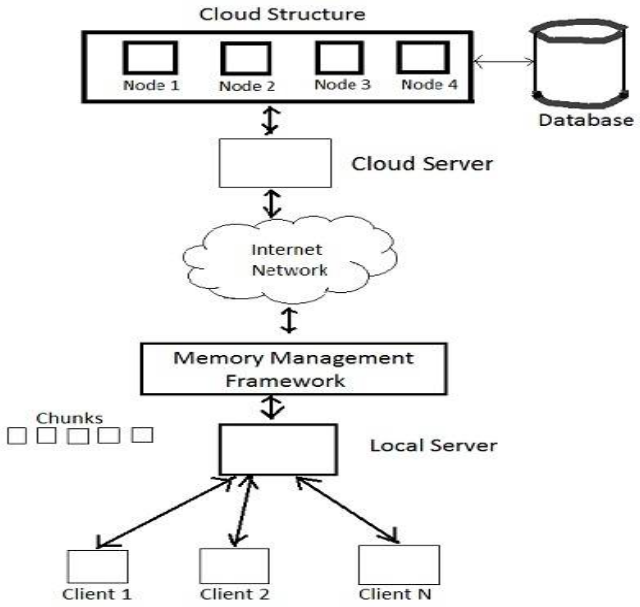

Fig 1: System Architecture

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2016

Our proposed 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.

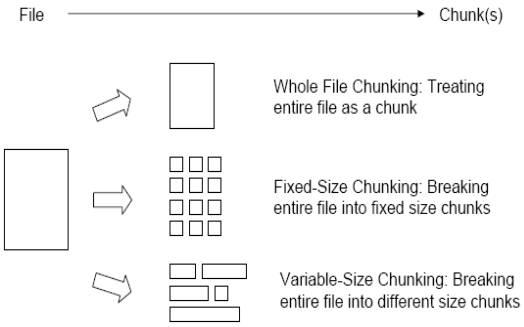

Fig 2: TTTD Workflow

A. CLOUD COMPUTING :

Cloud computing is a type of Internet-based computing that provides shared computer processing resources and data to computers and other devices on demand. It is a model for enabling ubiquitous, on-demand access to a shared pool of configurable computing resources (e.g., computer networks, servers, storage, applications and services), which can be rapidly provisioned and released with minimal management effort. Cloud computing and storage solutions provide users and enterprises with various capabilities to store and process their data in third-party data canters that may be located far from the user–ranging in distance from across a city to across the world. Cloud computing relies on sharing of resources to achieve coherence and economy of scale, similar to a utility (like the electricity grid) over an electricity network.

B. DISTRIBUTED STORAGE NETWORK :

Distributed Networking is a distributed computing network system, said to be "distributed" when the computer programming and the data to be worked on are spread out over more than one computer. Usually, this is implemented over a network. Prior to the emergence of low-cost desktop computer power, computing was generally centralized to one computer. Although such canters still exist, distribution networking applications and data operate more efficiently over a mix of desktop workstations, local area network servers, regional servers, Web servers, and other servers. One popular trend is client/server computing. This is the principle that a client computer can provide certain capabilities for a user and request others from other computers that provide services for the clients. Enterprises that have grown in scale over the years and those that are continuing to grow are finding it extremely challenging to manage their distributed network in the traditional client/server computing model. The recent developments in the field of cloud computing has opened up new possibilities. Cloud-based networking vendors have started to sprout offering solutions for enterprise distributed networking needs. Whether it turns out to revolutionize the distributed networking space or turns out to be another craze remains to be seen

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2016

V. CONCLUSION

In this paper, we proposed a novel scalable data compression based on similarity calculation among the partitioned data chunks with Cloud computing. For proper granularity, an effective and efficient chunking algorithm is a must. If the data is chunked accurately, it increases the throughput and the net deduplication performance. The file-level chunking method is efficient for small files deduplication, but not relevant for a big file environment or a backup environment. TTTD-S algorithm, not only successfully achieves the significant improvements in running time and average chunk-size, but also obtains the better controls on the variations of chunk-size by reducing the large-sized chunks.

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. Xiaolong Xu, Qun Tu, "Data deduplication mechanism for cloud storage systems", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING VOL:25 NO:7 SEPTEMBER, 2015
7. Marcos D. Assunção a, Rodrigo N. Calheiros b, Silvia Bianchi c, " Big data computing and clouds : Trends and future directions",Journal of parallel and distributed computing,79-80(2015) 3-15
8. Amit Jain, a Kamaljit I. Lakhtariab, Prateek Srivastava," A Comparative Study of Lossless Compression Algorithm on Text Data", Proc. of Int. Conf. on Advances in Computer Science, AETACS
9. S.R. KODITUWAKKU," COMPARISON OF LOSSLESS DATA COMPRESSION ALGORITHMS FOR TEXT DATA", S.R. Kodituwakku et. al. / Indian Journal of Computer Science and Engineering Vol 1 No 4 416-425.
10. K. Tanaka and A. Matsuda, "Static energy reduction in cache memories using data compression," in Proc. IEEE TENCON, 2006,pp. 1–4.
11. S. Roy, R. Kumar, and M. Prvulovic, "Improving system performance with compressed memory," in Proc. Int. Parallel Distrib. Process. Symp., 2001, pp. 7–13.
12. 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.