www.ijiarec.com

ISSN:2348-2079

Volume-5 Issue-2

International Journal of Intellectual Advancements and Research in Engineering Computations

An storage optimization using deduplication technique

1. Dr. A.Tamilarasi Msc,MPhil,MTech,PhD.
2. Ms. N. Naveena
Department of Computer Applications-PG, Kongu Engineering College, Perundurai.
naveenactkec@gmail.com

Abstract-- Data deduplication technology is to optimize the storage space. In cloud data storage, the deduplication technology plays a major role in the virtual machine framework, data sharing network, and structured and unstructured data handling by social media and, also, disaster recovery. In the deduplication technology, data are broken down into multiple pieces called "chunks" and every chunk is identified with a unique hash identifier. These identifiers are used to compare the chunks with previously stored chunks and verified for duplication. In the proposed system the files can be broken into multiple chunks of variable sizes by breaking them up based on the content rather than on the fixed size of the files. This method resolves the fixed chunk size issue. When working on a fixed chunking algorithm, fixed boundaries are defined on the data based on chunk size which do not alter even when the data are changed. However, in the case of a variable-size algorithm different boundaries are defined, which are based on multiple parameters that can shift when the content is changed or deleted. Hence, only less-chunk boundaries need to be altered.

Keywords-Chunking algorithm. Data deduplication. Fixed and variable chunking method.

1. INTRODUCTION

With the enormous growth of digital data in the cloud storage and enterprise organization, backup and disaster recovery have become crucial to the data center environment. Based on the latest forecast by Cisco, data center traffic on a global scale will grow at a 23 % CAGR, while cloud data center traffic will grow at a faster rate (32 % CAGR) or 3.9-fold growth from 2013 to 2018 [1].

There are four major steps involved in data deduplication. The chunking method splits the incoming large data into small portions or 'chunks'. Using hash algorithm, a unique identifier value is assigned to each chunk. The new incoming chunks are compared with the existing stored chunk using the unique identifier. If the identifier matches, the redundant chunk will be removed and will be given the reference point; otherwise, new chunk will be stored. The key objective of the chunking algorithm is to divide the data object into small fragments. The data object may be a file, a data stream, or some other form of data.

Most important issues in the big data is storage. Data reduction is the transformation of alphanumerical information into a simplified and ordered form. The basic concept is the reduction of infinite amounts of data split into the meaningful parts. There are different types of data reduction techniques used in data mining [2] like feature selection, feature extraction, Dimensionality reduction, Compression, deduplication. Data reduction techniques such as compression and deduplication are plays main role in the big data storage. Compression is used to reduce the size of the file and deduplication is used for eliminating the repeated duplicate data.

Duplication is a data reduction technique, commonly used in disk-based backup systems, storage systems designed to reduce the use of storage capacity. It works in a different time period to fed duplicate files in different locations of variable size data blocks. Duplicate data blocks replaced with the indicator. Highly redundant data sets[4][5] (such as backup data) from the data de-duplication technology to benefit greatly; users can achieve 10 to 1 to 50 to 1 reduction ratio. Moreover, data deduplication technology can allow users to efficiently between the different sites, the

economy back up data replication. Compression through the compression algorithms to eliminate redundant data contained in a document to reduce file size, but duplication is distributed through the algorithm to eliminate the same file storage system or data block.

2. LITERATURE REVIEW

2.1 Fixed size chunking

Fixed-size chunking method splits files into equally sized chunks. The chunk boundaries are based on offsets like 4, 8, 16 kB, etc. [3, 4]. This method effectively solve issues of the file-level chunking method: If a huge file is altered in only a few bytes, only the changed chunks must be reindexed and moved to the backup location. However, this method creates more chunks for larger file which requires extra space to store the metadata and the time for lookup of metadata is more. As it splits the file into fixed size, byte shifting problem occurs for the altered file. If the bytes are inserted or deleted on the file, it changes all subsequent chunk position which results in duplicate index values. Hash collision is likely to happen on chunking method by creating same hash value for different chunks. This can be eliminated by using bit-by-bit comparison which is more accurate, but requires more time to compare the files.

2.2 File level chunking

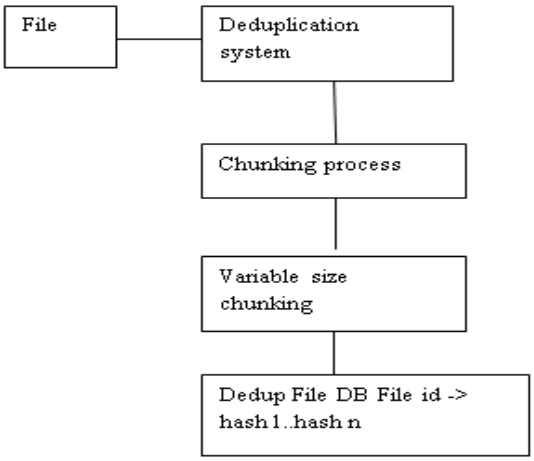
File-level chunking or whole file chunking considers an entire file as a chunk, rather than breaking files into multiple chunks. In this method, only one index is created for the complete file and the same is compared with the already stored whole file indexes. As it creates one index for the whole file, this approach stores less number of index values, which in turn saves space and helps store more index values compared to other approaches. It avoids maximum metadata lookup overhead and CPU usage.

3. PROPOSED ALGORITHM

Variable size chunking relies on a pattern of data to chunk the data streams. The advantage of this kind of chunking is that even if the same data stream to be deduplicated has a new block of data the rest of data match except the chunk having the new block. Hence this method gives a better deduplication result.

1. Variable size chunking

The files can be broken into multiple chunks of variable sizes by breaking them up based on the content rather than on the fixed size of the files. This method resolves the fixed chunk size issue. When working on a fixed chunking algorithm, fixed boundaries are defined on the data based on chunk size which do not alter even when the data are changed. However, in the case of a variable-size algorithm different boundaries are defined, which are based on multiple parameters that can shift when the content is changed or deleted. Hence, only less-chunk boundaries need to be altered. The parameter having the highest effect on the performance is the fingerprinting algorithm. One of the widely used algorithms for variable-size chunking is Rabin's algorithm [7] to create the chunk boundaries [8-10]. Basically, the variable-size chunking algorithm uses more CPU resources [5, 6]. Based on the characteristics of the file such as content, size, image, color, etc., we can apply variable-size chunking and fixed-size chunking.



Step By Procedure:

Step-1. Read file.
Step-2. Apply deduplication on that file.
Step-3. Then split the file as variable sized at block level.
Step-4. After splitting the file use hash technique to create the unique identifier for each block.
Step-5. Entire index search for matches if match found remove it.
Step-6 . Save unmatched data on to the disk.
Step-7. Before storing on the disk reduce the data size by using compression algorithm.
Step-8. After compression save the file to the disk.

4. SIMULATION RESULTS

Comparison of file size before deduplication and after deduplication is done in this experiment. Before optimization process (i.e. compression and deduplication) file size is more than the after optimization process. Fig3 explains about the comparison of storage space reduced after deduplication.

BEFORE AND AFTER DUPLICATION OUTPUT:

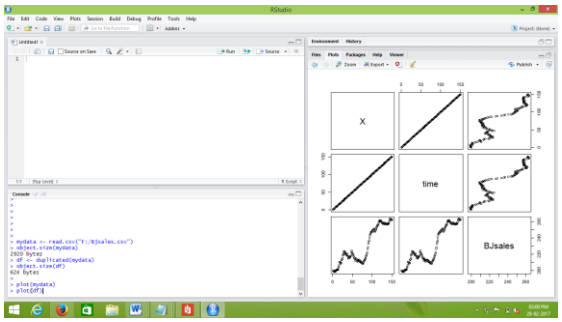


Fig 1 .Before deduplication

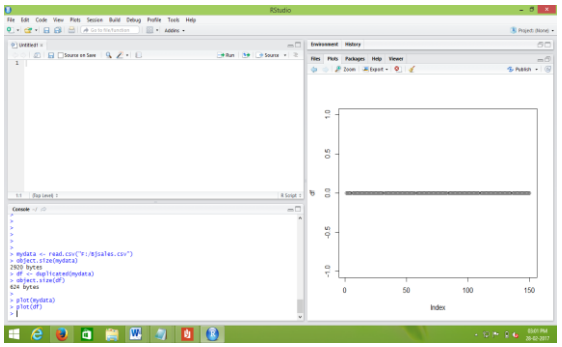


Fig 2. After deduplication

BEFORE AND AFTER COMPRESSION OUTPUT:

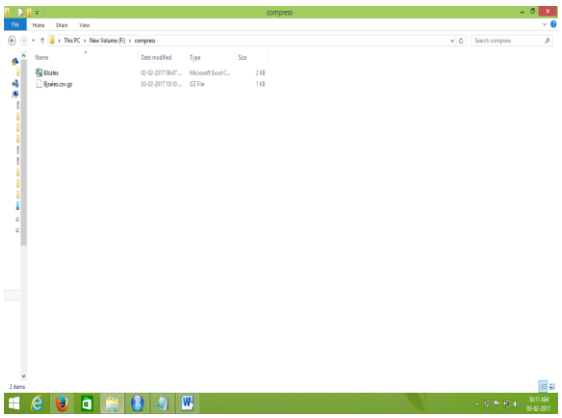


Fig.3. Compression

V. CONCLUSIONS

In the data deduplication process, the first and the most important step is the granular division and subdivision of data. 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. The problem with the fixed-size chunking method is that it fails to detect redundant data if some bytes are altered. The variable-size chunking methods overcome this fixed file size chunking problem by creating boundaries.

REFERENCES

1. Cisco Global Cloud Index: Forecast and methodology (2015)whitepaper.http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-indexgci/ Cloud_Index_White_Paper. html. Visited last on 02 Apr 2015.

2. Gayathri R,Malathi A,"A Novel Deduplication Technique Using an Evolutionary Approach", International Journal of Advanced Research in Computer and Communication Engineering vol.2,Issue 11,2013.Hadoop, http://hadoop.apache.org/, 2013.

3. Kubiatowicz J et al (2000), Ocean store: an architecture for global store persistent storage. In: Proceedings of the 9th international conference on architectural support for programming languages and operating systems.

4. Quinlan S, Dorwards S (2002), Venti: a new approach to archival storage. In: Proceedings of USENIX conference on file and storage technologies.

5. Won Y, Kim R, Ban J, Hur J, Oh S, Lee J (2008), Prun: eliminating information redundancy for large scale data backup system. In: Proceedings IEEE international conference computational sciences and its applications (ICCSA'08)

6. Won Y, Ban J, Min J, Hur J, Oh S, Lee J (2008), Efficient index lookup for de-duplication backup system. In: Proceedings of IEEE international symposium modeling, analysis and simulation of computers and telecommunication systems (MASCOTS'08), pp 1–3, Sept 2008

7. Rabin M (1981), Finger printing by random polynomials. Center for Research in Computing Technology, Aiken Computation Laboratory, University.

8. Lillibridge M, Eshghi K, Bhagwat D, Deolalikar V, Trezise G, Camble P (2009), Sparse indexing: large scale, inline deduplication using sampling and locality. In: Proceedings of the 7th USENIX conference on file and storage technologies (FAST'09), San Francisco, CA, USA, Feb 2009, pp 111–124

9. Muthitacharoen A, Chen B, Mazi`eres D (2001), A low-bandwidth network file system. SIGOPS Oper Syst Rev 35(5):174–187 16. Zhu B, Li K, Patterson H (2008) Avoiding the disk bottleneck in the data domain deduplication file

system. In: FAST'08: Proceedings of the 6th USENIX conference on file and storage technologies, Berkeley, CA, USA, pp 1–14.