Big Data Implementation using Data De-Duplication Strategies and Its Recent Developments

Mr.Seemant Kumar Sharma (Research Scholar),Dr.Anil Pimpalapure (Dean), Dr.Shahnawaz Ansari (Director, Research Cell)
Department of Computer Science & Engineering
Eklavya University Sagar Road, Near Tool Plaza, Damoh (M.P)

Abstract

The amount of digital data, including text, images, audio, and video, has been growing steadily over the past year. By the end of 2022, according to a poll of projections from international data specialists, people will generate and use almost 94 zettabytes of data. Many issues with data storage and retrieval operations have been brought on by data in data centres and data backup. To store safe data, the majority of businesses and organisations invest a lot of money. Digital data is no longer efficient or distinctive. A portion of digital material is original, and another portion is copied. We will present a novel big data processing technology to address this issue.

In this post, we suggest setting up a novel method for deduplicating content from a sizable data set. The field delimiter is utilised to automatically choose the separators, and various dictionary indexing techniques will be employed to deduplicate the content of fields with little variation. We also accelerate the De-duplication of lengthy string arrays using a set of computationally cheap hashing methods. There will be testing of the quantity, kind, and length of fields. Before doing De-duplication, the best strategy to reduce data size will be determined using some sort of indexing and clustering methodology.

Keywords: Big Data; Hashing; Encoder; Indexing; Data De-duplication.

I. INTRODUCTION

The massive growth in information puts a significant strain on storage systems. Data loss is disastrous to the modern corporation and can completely shut it down, as evidenced by the 9/11 terrorist attacks and the loss of commercial data they caused. To ensure data availability and integrity, it is crucial to frequently backup data at a DR (Disaster Recovery) site

[1]. Images, music, video, email exchanges, scanned documents, and other types of data are included in business data. Every organisation obtains this information for administrative and judicial functions. The continuously expanding amount of data presents numerous difficulties for the current storage technologies. A larger storage medium must be used for a huge volume of data. As data increases, there is more evidence to support it

[2]. The price of storage media has come down, but the management of backup systems' disc count remains a major challenge. In fact, a significant amount of data in enterprise archives is redundant with a minor change to another piece of data. Have numerous methods been created to eliminate duplicate copies from data storage? Data De-duplication is currently becoming more and more well-liked among researchers. A specialised data compression technique called data De-duplication removes superfluous data, usually to increase storage efficiency. A single copy of the data block and a pointer to a unique copy of the data will be stored since redundant data is removed during the De-duplication process and not stored.

[3]. Due to the fact that only distinct data is kept, De-duplication is a technique to reduce the necessary storage capacity, as shown in Figure 1.

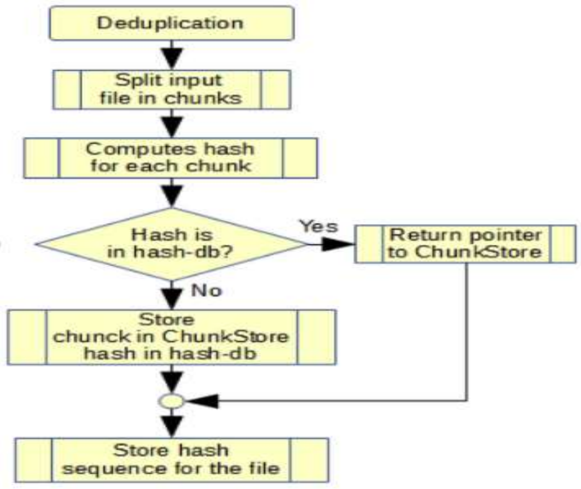

Deduplication
Split input file in chunks
Computes hash for each chunk
Hash is in hash-db?
Yes
Return pointer to ChunkStore
No
Store chunck in ChunkStore hash in hash-db
Store hash sequence for the file

Figure 1. De-Duplication process [4]

II. **DATA DE-DUPLICATION METHODOLOGY**

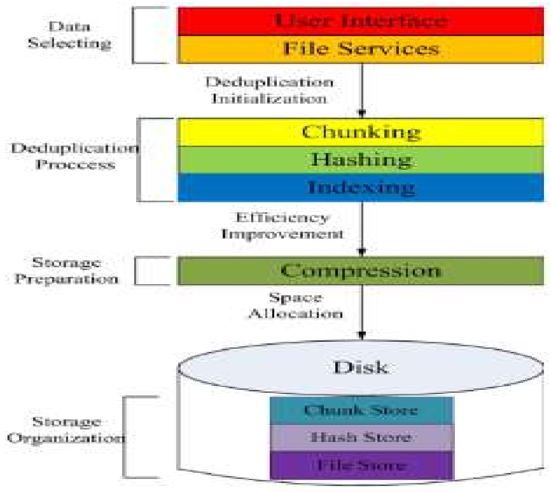
A cryptographic hash of the incoming data is computed as part of the De-duplication approach to identify duplicate data. A message of any length can be represented by a hash, which has a fixed length. It is significantly simpler to compare two fixed-size hashes than it is to laboriously compare two huge data blocks or records. The De-duplication server generates a hash signature for each data block and searches the database's already-stored hash index for that signature. Instead of storing the complete data block, the server makes a reference to the existing entry if there is already a record for this signature in the database records. If no match is discovered, the server saves this data file to disc and adds an entry for its hash signature to the hash index database. Otherwise, if no match is discovered.

III. **TYPES OF DATA DE-DUPLICATION**

There are numerous ways to categorise data De-duplication, however the following four are the most common:

- Point of application

- Time of application,
- Granularity
- Algorithm



1) Figure 2: De-Duplication Classification

1. Point of Application De-duplication

Data De-duplication can happen at either the source or the target, depending on the situation.

A) Source-De-duplication (Client Side): This sort of approach is referred to as source-De-duplication if it does De-duplication on the client side where the data is created prior to being delivered to the storage server for backup. The decrease in bandwidth utilisation is this method's key advantage. We can cut network traffic, save a lot of bandwidth, and put little strain on backup systems. This technique was utilised by the authors of [5] to develop the LBFS (low-bandwidth network file system). Cloud file system wrappers that employ this method to cut down on network bandwidth include Open Dedup [6] and S3QL [7].

B) Target De-duplication (Server Side): This technique is referred to as server side or target De-duplication if it is carried out at the backup server end after data transmission. Server-side De-duplication can solve this issue by processing data on the server, as client-side De-duplication processing can result in client-side performance restrictions for huge volumes of data.

lateral. This scheme's drawback is that it necessitates expensive, high-end server-side hardware, which can vary. For networks with limited bandwidth, this strategy is not the best plan. Two well-known systems that employ the targeted strategy are Venti [8] and ZFS [9].

2. Time Based De-Duplication: For time-based De-duplication algorithms, there are two basic approaches: offline and inline (online).

a) Offline De-duplication: De-duplication takes place after saving incoming storage data to disc. After examining the stored portions of the data, the storage De-duplication process removes duplicate portions and inserts a pointer pointing to the previously stored data if duplicate portions are found. No changes are done if no duplicates are discovered. Almost no CPU-intensive calculations are performed while writing data to disc using the offline method, which results in better write performance. Examples of software that employs this technique for storage De-duplication are IBM Store Tank [10], Netapp ASIS [11], and EMC Clara [12].

b) Inline De-duplication: Before writing to disc, this technique searches for redundancies in the file data from an incoming request. Up till de-duplication is finished, incoming data will stay in RAM. The data is sent to the storage disc via the IO channel if it is unique, and a new hash record is also created in the hash index database. Due to heavy CPU calculations occurring at IO time, this strategy lengthens the time it takes to write data to storage. Data archiving applications that use inline De-duplication include Venti [8], OpenDedup [6], and S3QL [7]. Both approaches rid the storage system of duplicate data and make more room for fresh data. Because of the decrease in storage needs, businesses or organisations can spend a long time without needing to buy new storage. Low storage requirements necessitate cheaper storage devices, which reduces costs.

3) De-duplication Based On Algorithms

The method that De-duplication utilises to look for redundancy is its primary justification. In essence, two algorithms are employed. Verify De-duplication:

- Hashing
- Delta coding

a) De-duplication Based On Delta Coding: In the past, file compression techniques like B-zip and 7-Zip used delta coding as their fundamental concept. When comparing two files for similarity in delta coding De-duplication, the encoded difference file is saved to the first file with a reference link from the first file rather than saving the second file in its entirety [13]. In situations when a very comparable file needs to be stored as backup storage, delta encoding has proven to be quite useful. A patch file or difference file is the term used to describe the encoded file produced by delta De-duplication.

b) De-duplication Based On Hashes It is a common De-duplication technique. This method's fundamental concept is to calculate the hash of the data blocks and compare it to the stored blocks already in existence to determine whether they are comparable. If the hashes match, the parsed data is already in storage; otherwise, the block must be put to disc because it is unique.

The three following methods can be used for hash-based De-duplication:

Hash based de-duplication can be classified into three methods:

- File Level De-duplication
- Block Level De-duplication
- Byte Level De-duplication

A) File-Level De-duplication : A file's full contents are used as a record in file-level De-duplication. When a file is reached at the storage server for backup, the server computes and compares the incoming file's hash signature with the files that have already been saved. The server stores a reference to the file if the hash value matches, else it stores the complete file and adds an entry for the file's hash signature to the hash table (Ref. Figure 3).

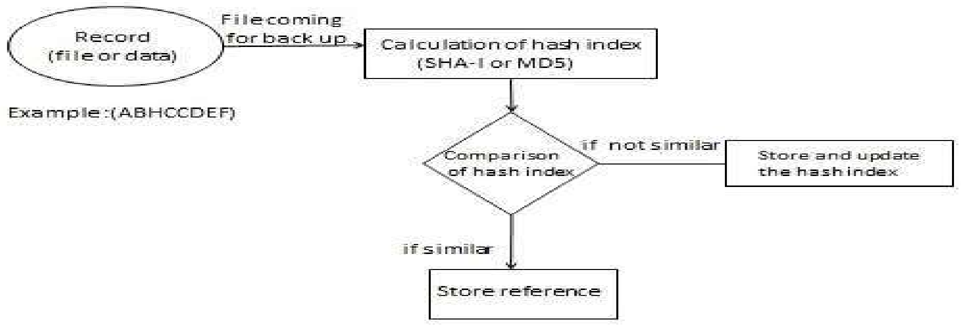


Figure **3: File Level De-Duplication**

B) Block Level De-duplication: This approach divides the data stream to be saved into data blocks (which may be fixed or variable), calculates the hash signature of each block, and compares it to the database of hash indexes for the stored data block. and using the previously saved data block, the server looks for a comparable block. The server writes a data block to disc and stores its hash value in the hash index database if the block's hash is unique; otherwise, it stores just a pointer that will lead to an existing block address, as shown in Figure 4. When compared to storing a whole block of data, this location pointer significantly minimises the amount of storage needed. Block-level hashing has a lower granularity than file-level De-duplication, which improves the De-duplication ratio.

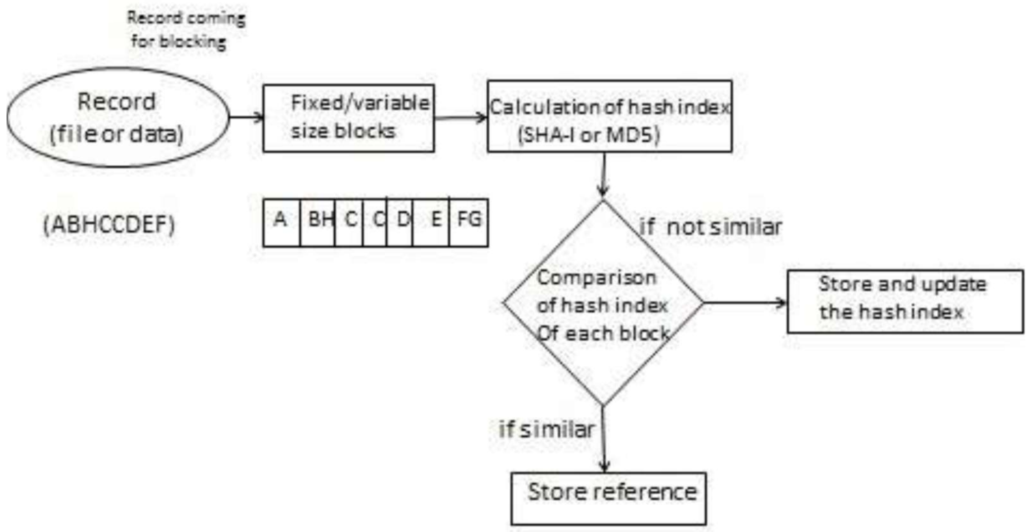

Record coming for blocking
Record (file or data)
(ABHCCDEF)
Fixed/variable size blocks
A BH C C D E FG
Calculation of hash index (SHA-I or MD5)
Comparison of hash index Of each block
if not similar
Store and update the hash index
if similar
Store reference

Figure 4: Block Level De-Duplication

Because a hash will be generated for each block of data, block-level De-duplication has the drawback of requiring a large hash table, which also implies that duplicate checking and backups will take more time to complete. In comparison to file-based De-duplication, performance will be slower.

C) **Byte Level De-duplication**: The data to be stored is broken down into bytes, and a hash signature is calculated for each byte. These hash signatures are compared with the stored byte signature on the server, and appropriate actions are taken based on the match of the mismatched records, as shown in Figure 5. When compared to file- and block-level De-duplication, byte-level De-duplication has a higher De-duplication ratio, but it has a number of performance problems, including the following:
Finally, byte-level De-duplication will slow down speed. It can result in huge file fragmentation. The size of the hash table will be quite large.

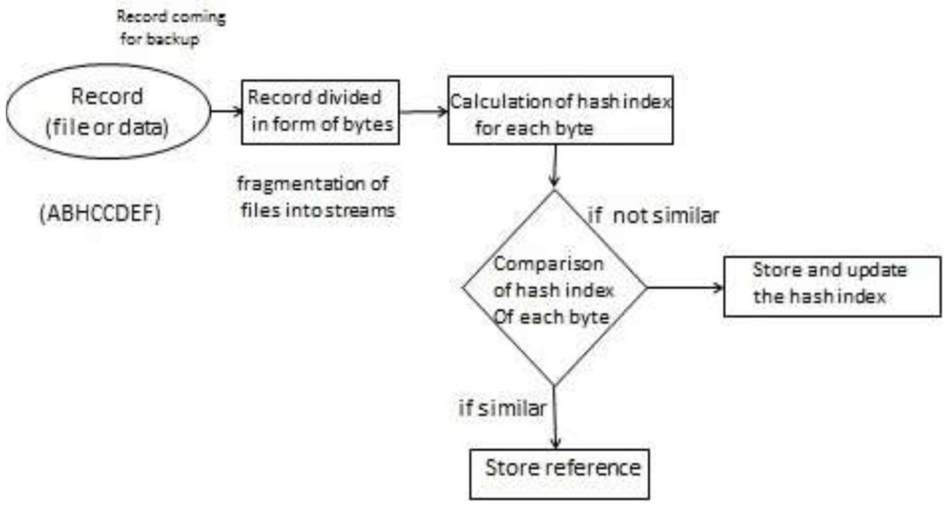

Record coming for backup
Record (file or data)
(ABHCCDEF)
Record divided in form of bytes
fragmentation of files into streams
Calculation of hash index for each byte
Comparison of hash index Of each byte
if not similar
Store and update the hash index
if similar
Store reference

Figure. 5: Byte Level De-Duplication

IV. MISCELLANEOUS PRODUCTS AND RESEARCH

De-duplication research is now concentrated on two areas. The primary factor is the effectiveness of data reduction, or the deletion of redundant data to the greatest extent possible, in order to reduce the impact of storage space limitations. In order to achieve optimal utilisation, more attention is placed on enhancing the De-duplication process' effectiveness. Hardware. File-level De-duplication is used by almost all traditional storage systems now in use [14]. Only a small percentage of the currently used architectures support source De-duplication and provide De-duplication in the user's file system [15]. Other alternative architectures that use a target De-duplication method feature single system De-duplication, which means that one system (Server) handles the server side. File system De-duplication causes a delay in data transfer to the backup server. all IO requests for data archiving and maintains track of the number of attached discs in data signatures.

The architectures VENTI [8], LBFS [15], SIS (single instance store) [16], and PASTICHEL [17] are a few that have been previously proposed. The file is divided into blocks using a fixed size partition mechanism by VENTI and SIS. Each file is divided into blocks of varying sizes using LBFS and PASTICHEL. Although the fixed-size partitioning method is straightforward and simple to use, it has a significant drawback in that any data blocks after the change point will be impacted and mistakenly identified as non-duplicated blocks.

Another problem that might cause data corruption in the current architectures is hash collision, for example, when two independent data blocks produce the same hash signature and an individual block is mistakenly deleted. Even though SHA-1 is regarded as a negligible hashing method, techniques like LBFS [15] still employ it.

Numerous organisations are grappling with the idea of data De-duplication in this context. IBM, Symantec, and NetApp are just a few businesses. An essential part of the Data ONTAP operating system is NetApp De-duplication. The first to be widely utilised in various applications, including primary data, backup data, and archive data, is NetApp De-duplication. Additionally, Symantec offers backup appliances with three-step reduction methods. In the beginning, it offers data De-duplication at both the source and destination and lessens data De-duplication's complexity. Using De-duplication technology, the IBM TS7610 ProtecTIER De-duplication Appliance Express offers quick, dependable, and simple backups.

V. References

[1] H. Biggar, "Experiencing data de-duplication: Improving efficiency and reducing capacity requirements," The Enterprise Strategy Group, 2007.

[2] He, Qinlu, Zhanhuai Li, and Xiao Zhang. "Data deduplicat ion techniques." In 2010 Internat ional Conference on Future Informat ion Technology and Management Engineering,vol.1, pp. 430-433. IEEE, 2010

[3] J. Stanek, A. Sorniott i, E. Androulaki, and L. Kencl. A secure data deduplicat ion scheme for cloud storage. In Technical Report, 2013.

[4] Jiang, T., Chen, X., Wu, Q., Ma, J., Susilo, W. and Lou, W., 2016. Secure and efficient cloud data deduplicat ion with randomized tag. IEEE transactions on informat ion forensics and security, 12(3), pp.532-543.

[5] CWADN, http://www.computerweekly.com/

[6] Managing Meta Data for the Business: Reducing IT Redundancy, http://www.eiminstitute.org

[7] S. Quinlan and S. Dorward, "Awarded best paper! -venti: A new approach to archival data storage," in Proceedings of the 1st USENIX Conference on File and Storage Technologies, ser. FAST '02. Berkeley, CA, USA: USENIX Association, 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=1083323.1083333

[8] J. Bonwick and B. Moore, "Zfs: The last word in file systems," 2007.

[9] J. Menon, D. A. Pease, R. Rees, L. Duyanovich, and B. Hillsberg, "Ibm storage tanka heterogeneous scalable san file system," IBM Systems Journal, vol. 42, no. 2, pp. 250–267, 2003.

[10] C. Alvarez, "Netapp de-duplication for fas and v-series deployment and implementation guide," Technical ReportTR- 3505, 2011.

[11] EMC and E. E. Services, Information Storage and Management: Storing, Managing, and Protecting Digital Information. LibreDigital, 2010.

[12] K. Jin and E. L. Miller, "The effectiveness of de-duplication on virtual machine disk images," in Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference. ACM, 2009, p. 7.

[13] G. Wang, Y. Zhao, X. Xie, and L. Liu, "Research on a clustering data de-duplication mechanism based on bloom filter," in Multimedia Technology (ICMT), 2010 International Conference on. IEEE, 2010, pp. 1–5.

[14] A. Muthitacharoen, B. Chen, and D. Mazie`res, "A low- bandwidth network file system," SIGOPS Oper. Syst. Rev., vol. 35, no. 5, pp. 174–187, Oct. 2001. [Online]. Available: http://doi.acm.org/10.1145/502059.502052

[15] W. J. Bolosky, S. Corbin, D. Goebel, and J. R.Douceur, "Single instance storage in windows&#174; 2000," in Proceedings of the 4th Conference on USENIX Windows Systems Symposium - Volume 4, ser. WSS'00. Berkeley, CA, USA: USENIX Association, 2000, pp. 2–2. [Online]. Available:http://dl.acm.org/citation.cfm?id=1 67102.1267104.

[16] L. P. Cox, C. D. Murray, and B. D. Noble, “Pastiche: Making backup cheap and easy,” in Proceedings of the 5th Symposium on Operating Systems Design and implementation Copyright Restrictions Prevent ACM from Being Able to Make the PDFs for This Conference Available for Downloading, ser. OSDI ’02. New York, NY, USA: ACM, 2002, pp. 285–298. [Online]. Available: http://doi.acm.org/10.1145/1060289.1060316