International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN: 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)
doi : https://doi.org/10.32628/IJSRSET229425

Big Data Backup Deduplication: A Survey

Hashem Bedr Jehlol*1, Loay E. George2
*1Iraqi Commission for Computers and Informatics, Informatics Institute of Postgraduate Studies, Baghdad-Iraq
2University of Information Technology and Communication (UoITC), Baghdad-Iraq

Article Info
Volume 9, Issue 4

Page Number : 174-190

Publication Issue :
July-August-2022

Article History
Accepted : 05 July 2022
Published: 22 July 2022

ABSTRACT
The massive explosion in the field of data such as images, video, audio, and text has caused significant problems in data storage and retrieval. Companies and organizations spend a lot of money to store and manage data. Therefore, there is an urgent need for efficient technologies to deal with this massive amount of data. One of the essential techniques to eliminate redundant data is data deduplication and data reduction. The best technique used for this purpose is data deduplication. Data deduplication decreases bandwidth, hard disc drive utilization, and backup costs by removing redundant data. This paper focuses on studying the literature of several research papers related to data deduplication for various techniques that several researchers have proposed. It summarized multiple concepts and techniques related to deduplication and methods used to improve storage. The data deduplication processes were examined in detail, including data chunking, hashing, indexing, and writing. Also, this study discussed the most critical problems faced by the data deduplication algorithm.

Keywords: Data Deduplication, Data Reduction, Redundant Data, Data Chunking, Hashing.

I. INTRODUCTION

The expansion of data that accompanied the information revolution is massive, and many organizations and people are already facing real problems in dealing with this vast volume of data and how to secure and store this data [1]. The International Data Corporation (IDC) defines global information as data produced, captured, or transcribed via globally distributed digital resources. Global data will expand from 33 zettabytes (ZB) in 2018 to about 175 zettabytes (ZB) by 2025, as shown by IDC forecasts [1]. Many large companies like Google, IBM, Microsoft, Intel, and Motorola found, through a study conducted on the existing global data, that almost three-quarters of the current digital data are duplicate data [2]. Therefore, there is an urgent need for organizations, IT companies, and industries to store a massive amount of their data securely and to be able to work on it and retrieve it quickly. Since this data is vast, one of the most challenging tasks in big data is the process of backup and maintenance, as these operations are costly and considered among the challenges in this field [3]. Therefore, there is a significant challenge in storing and managing such vast amounts of digital data [4]. As a result, the deduplication technique, which prevents storing duplicate data on hard disks, is among the most excellent solutions to these challenges [5]. Data deduplication technology has become the dominant technology that reduces the space required for backup data and primary file systems [6].

The data deduplication system has four major stages: chunking, fingerprinting, indexing, and data writing. The early phase considered a bottleneck in removing redundant data is chunking, in which vast amounts of incoming data are divided into small parts or chunks [7]. Chunk-level deduplication may be accomplished in two ways: fixed-size chunking (FSC) and variable size chunking (VSC) [8]. In FSC, the entire file contents divide into chunks of equal size. The FSC has low efficiency of deduplication and suffers from the boundary shifting problem, whereas VSC eliminates the issue of boundary shifting and divides the file into chunks; it does not have to be of equal sizes. The VSC method requires more computation and time but provides a more significant percentage of deduplication [9]. Fingerprinting is the second stage of the data deduplication system; each chunk is allocated a unique value. Hash functions such as Secure Hashing Algorithm SHA-1 or Message-Digest Algorithm MD5 are widely used in this stage of the data deduplication system, which generates a digest for each chunk called a fingerprint. The third stage is indexing when the previously stored fingerprint values are compared with the fingerprint values of the new chunks to obtain the duplicate chunks. Indexing relies on chunking fingerprints to find duplicate chunks that can be identified and then remove them. If indeed the fingerprints of the two chunks match, they are considered identical. Writing is the fourth stage of data deduplication, storing a unique copy of the data on the hard disk. The unique chunks that do not have identical fingerprints are considered non-duplicate and stored in a hash table [10].

This paper examines different classifications to remove duplicate data, including granularity-based (file-level deduplication or chunk-level deduplication), time-based (before or after data is stored on disk), and based storage location deduplication, side-based deduplication, and Implementation based duplicate data. It also provides a review of the development of data deduplication technology, the pros and cons of each algorithm, the technical methods used, and identifies the problems and challenges facing storage systems based on data deduplication technique. Several recent studies and survey contributions in the field of deduplication have provided new algorithms and methods for improving deduplication. In addition, some studies focus on a specific aspect of deduplication, such as chunking or indexing systems and cloud storage.

The remaining sections of the survey were set up as follows. The key benefits and downsides of data deduplication are outlined in Section 2. Some of the recent studies that investigated various deduplication methods are included in Section 3. Section 4 goes into great depth about several deduplication methods. Section 5 goes into great detail on the primary steps in data deduplication. Finally, the summary of the survey is presented in Section 6.

II. DEDUPLICATION

The data deduplication technique is one of the most important techniques used to remove redundancy data [1]. This technique helps companies provision much money by reducing the cost of bandwidth and storage. It is helpful in cloud services because it reduces the need for additional storage devices [2]. Data deduplication is a technique to resolve storage problems. Four main steps are included in removing data duplication: data chunking, fingerprinting, indexing, and writing [3]. "Fig. 1" illustrates the general view of the four main stages of data deduplication There are various advantages of Data Deduplication, such as:

1. Improved efficiency of the network.
2. The space required for storage is Low.
3. The cost of storage is reduced.
4. Storage efficiency is increased.

Reduced upload bandwidth [4]. On the other side, there are several disadvantages to the deduplication technique, which are listed as follows [5]:

1. The duplicate data removal method requires some additional resources
2. Hash function inconsistencies can cause data to lose accuracy and consistency.
3. Security and Privacy.
4. Reducing data duplicates affect storage system availability.

III. Related Work

Table I shows several papers that dealt with deduplication in recent years and reviews the most critical techniques used and the results obtained. In addition, shows the data sets that each researcher relied on to obtain the results and the limitations of each paper.

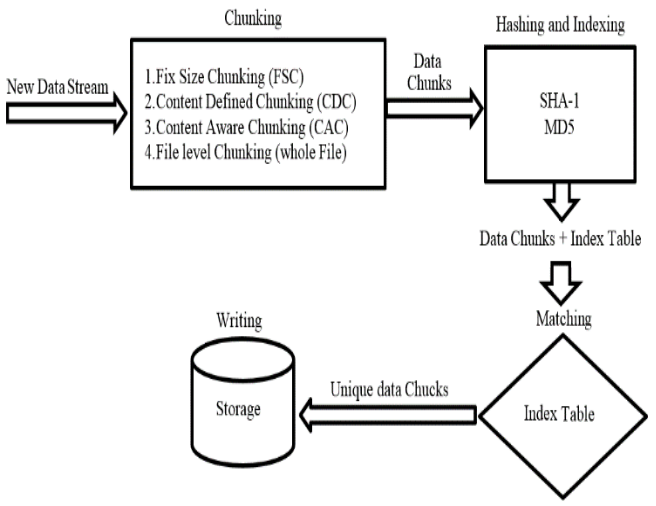
Chunking
New Data Stream
1.Fix Size Chunking (FSC)
2.Content Defined Chunking (CDC)
3.Content Aware Chunking (CAC)
4.File level Chunking (whole File)
Data Chunks
Hashing and Indexing
SHA-1
MD5
Data Chunks + Index Table
Matching
Index Table
Unique data Chucks
Writing
Storage

Figure. 1. General view of data deduplication stages [6].

TABLE I. The list of techniques, datasets, achievements, and limitations of recently released publications on data deduplication.

Paper	Technique Used	Dataset	Achievement	Limitation
2021 [12]	● Effective mathematical bounded linear hashing ● The hierarchal fingerprint lookup strategy	● Linux Kernel 10.9 GB ● SQLite 6.44 GB ● Oracle RMAN Backup 18.7 GB	● Decreases the hashing time ● Reduces hash index table by 50%. ● Minimize hash comparison time by up to 78%.	● The size of a hash index table grows greatly ● Using a fixed number of hashes (five hashes)
2021 [13]	● Matching based on forwarding/end feature vectors ● Uses dynamic adjustment of mask bits	● Glibc, GCC, and MySQL 56 GB ● Redis 111GB ● SYN 108GB	● Achieve a 222.3% deduplication ratio compared to Rapid CDC. ● Chunking speed was 11.4x faster than Rapid CDC. ● Productivity is higher by 111.4% than Rapid CDC	● New fingerprints improve processing speed. ● The deduplication ratio is slightly improved.
2021 [14]	● A collection of repeating patterns is utilized to detect breakpoints. ● Three-level lightweight hash function.	● (Linux 3.9, Linux 4.14.157, and Linux 5.8.12) 2.32 GB	● Faster than BSW by 15 times ● Ten times quicker than TTTD ● Five times faster than MD5 and SHA1	● It does not use a dynamic set of divisors.
2020 [15]	● Use of five main techniques ● Quick-rolling hashing based on gears ● Simplify and enhance the Gear's hashing rule ● Skip sub-minimum cut-off points	● TAR 56 GB ● LNX 178 GB ● WEB 237 GB ● VMA 138 GB ● VMB 1.9 TB ● RDB 1.1 TB ● SYN 2.1 TB	● Chunking speed is 3 to 12 faster than CDC approaches. ● Improve system throughput	● The same redundant data removal rate as the CDC.

Paper	Technique Used	Dataset	Achievement	Limitation
2020 [16]	• Bytes Pair Frequency-based Chunking (BFBC) algorithm • The proposed triple hash function	• Linux Kernel 5.93 GB • SQLite 6.44 GB	• DER is better than other CDC algorithms • Three times faster than TTTD. • Ten times faster than the BSW algorithm. • Hashing is 5 times faster than SHA1 and MD5	• Efficiency is affected by content data set similarity. • Potential hashing collision increases with a large dataset. • Computational overhead increases when the size of the hash table increases.
2018 [17]	• New fingerprint function • A multi-level approach to hashing and matching • New indexing method for storing metadata.	• Versions of Emacs and 3DLDF (GNU 580 MB, GNU 1.27 GB)	• Improves the TTTD algorithm. • Reduce system resource usage	• Efficiency is affected by content data set similarity. • Potential hashing collision increases with a large dataset.
2017 [18]	• An asymmetric local range's maximum value	• Bench: 108 GB • Open-source: 169.5GB • VMDK: 1.9TB	• 2.3X increase in throughput. • Increases system speed by 50%. • Overcoming the problem of the boundaries-shifting	• Deduplication strategies cannot be used directly on security systems.
2016 [19]	• Bucket-based and Map Reduce under HDFS • Fixed-size chunks • MD5 algorithm module to generate hash • MapReduce model is applied	• Zip Code Tabulation Area (ZCTA) 2.6 GB and 1.7 GB	• Distinctive buckets used for hash storage • Reduce hashing time and chunk lookup. • High deduplication ratio. • Significantly reduces data volume.	• Uses fixed-size chunking to reduce duplicate data removal • Boundary problem • It uses md5 algorithm

IV. Types of Deduplication Technique

There are different ways to remove duplicate data saved in the data store. However, most companies use deduplication approaches to solve and reduce the duplication problem [7]. "Fig. 2" shows the different approaches used to remove duplicate data [2]:

- Based on Granularity
- Based on Time deduplication.
- Side-based deduplication.
- Implementation-based deduplication.

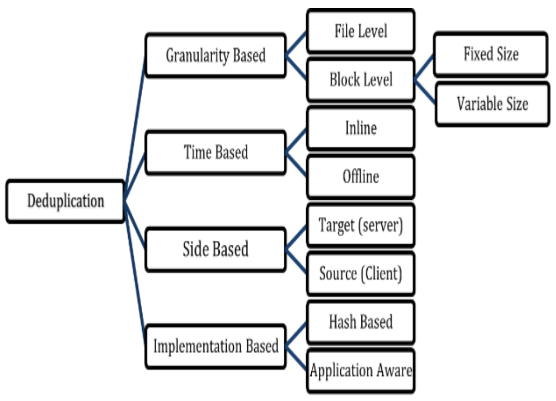

Figure. 2. Types of deduplication approaches

A. **Based on Granularity**

Depending on the first criterion, there are two classifications of deduplication, as follows [8]:

1) File-level deduplication: When using file-level deduplication, the entire file is handled as a single chunk rather than divided into many chunks [9]. In this technique, one hash value is constructed for the whole file, and the hash value for the new file is compared to the hash values of the stored files to find and eliminate duplicate files [10]. This method is not concerned with the internal contents of the file. For example, when two files are saved with the same internal content but different names, they are considered separate files. This approach is quick, easy, and requires little processing power. Single-instance storage is another name for this approach [8]. "Fig. 3" shows the deduplication technique with file-level deduplication.

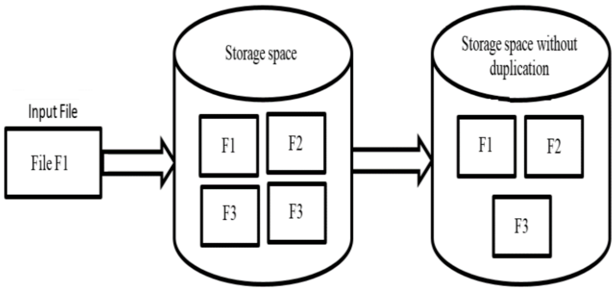

Figure. 3. File-level deduplication[11].

2) Chunk-level deduplication: In this approach, the file is divided into several small blocks, and each is called a chunk. In data deduplication, the search for duplicate chunks is within the file, and each chunk's unique copy is stored. Files can be divided into two ways to de-duplicated chunks [23]. Files can be divided into chunks of fixed length, i.e., the chunks with the same size, or into chunks of variable length, i.e., chunks with variable size [25]. Data deduplication using chunks level is far more efficient than deduplication of file-level [26]. The Content-Defined Chunking CDC algorithm breaks the data stream into chunks of varying sizes based on the content of the data stream, and when the local content does not change, the chunk limits do not change [27].

a) Fixed-Size Chunk De-duplication: The file is broken into fixed-size chunks, and identical chunks are identified using a standard hash algorithm [28]. The size of chunks can range from 8 to 64 KB [29]. The main drawback of this method is that any modification, even if minimal, in the chunk leads to rewriting the collection of other successive chunks on the drive. For example, if a single byte is entered at the beginning of this data stream, it causes all boundaries of the current chunk defined using FSC to be changed, resulting in less redundant selection and thus less deduplication. In other words, it suffers from what is called a boundary-shifting problem. Nevertheless, this approach is prevalent with a meagre remove data redundancy ratio. Figure (4) shows the deduplication technique with Fixed-size chunk deduplication.

b) Variable-size chunk deduplication: This partition type depends on the file's internal content for dividing the file into chunks [30]. The file is broken into chunks of varying sizes using a method known as Content-Defined Chunking CDC [23]. The boundaries defined in this algorithm are variable in size, which depends on multiple indicators that can change if the content of a file is changed or deleted [31]. The change in the size of the boundaries adopted by this algorithm makes it more resistant to deleting or entering new data [30]. However, this algorithm needs more system resources, such as the CPU, to perform a full file scan and determine the boundaries of each chunk [23]. "Fig. 4" shows the deduplication technique with variable-size chunk data deduplication.

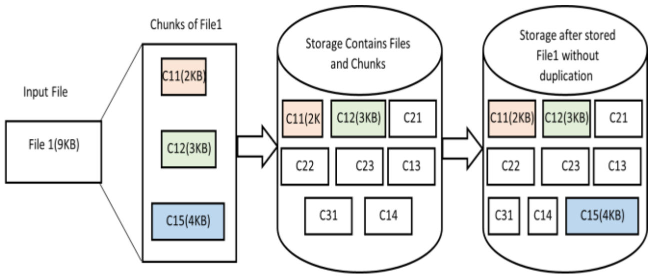

Figure. 4. Variable-size chunk deduplication[11]

3) Granularity based Advantages and Disadvantages: Granularity classification categorizes redundant data according to its Granularity, which describes the influence of this on different storage techniques, the techniques employed in such systems, and the impact of those varied ways on deduplication efficiency, performance, and resource consumption. The main drawback of categorizing data by Granularity is that typical hash storage systems are limited in their ability to reduce data redundancy.

B. **Based on Time deduplication**

In this approach, there are two methods [23].

1) Inline deduplication: Inline deduplication eliminates redundant data during or before it is written to the hard disk, reducing the storage space [23]. This method is flexible and powerful since it processes the data once [8]. Inline deduplication can be done on the client-side or when data is sent from the data source/client to the target/server [10]. However, the inline deduplication approach can only use a fixed-length chunk because it checks the incoming raw chunks and does not know other chunks [17]. The main drawback of this technique is that network efficiency significantly impacts it. However, this method's required storage capacity is less while the computation time is high [32]. "Fig. 5" shows the deduplication technique with Inline Deduplication of Data.

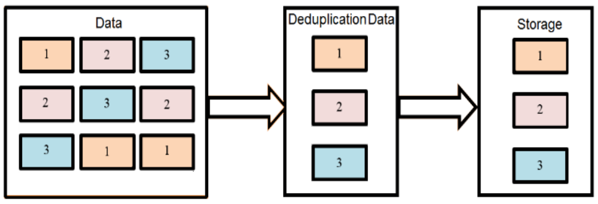

Figure. 5 Inline Deduplication of Data [24].

2) Post-process deduplication: Data is initially written to the storage device, and duplicate data is found and removed [23]. Both file and sub-file levels may benefit from post-process deduplication [17]. This technique's performance is superior to the inline approach [33] because it involves fewer calculations. The major drawback of this approach is that it requires an additional disk cache, which means that it is more expensive than the inline method [32]. "Fig. 6" shows the Post-process Deduplication of Data.

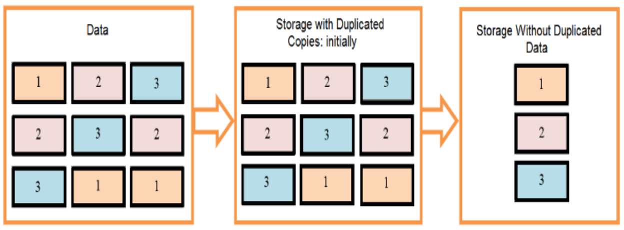
Data
1
2
3
2
3
2
3
1
1
Storage with Duplicated Copies: initially
1
2
3
2
3
2
3
1
1
Storage Without Duplicated Data
1
2
3

Figure. 6. Post-process Deduplication of Data [24].

3) Time Classification Advantages and Disadvantages: In time classification, all deduplication systems depend on when the process occurs, if the process is during the storage process or after the storage process. Inline deduplication occurs during data flow, whereas post-process deduplication occurs after data has been written to disk. Inline deduplication has a slow storage performance, whereas post-process deduplication has a fast storage performance because the hash calculation is deferred. The storage requirement and network traffic are less in inline deduplication and more comparative in post-process deduplication. The storage throughput of inline deduplication is lower than that of post-process deduplication. Inline does not require temporary storage space, while post-process deduplication is required.

C. Based on Side

In this approach, there are two methods source / client deduplication and target deduplication as shown in "Fig. 8" [32].

1) Source / Client deduplication: This approach removes duplicate data at the source before sending it [33]. Removing the data takes place on the client/source side before transferring the data to the backup device [32]. One of the essential features of this type of data removal is that it does not require a high bandwidth compared to the Target deduplication. As a result, source/client deduplication has two main advantages: it uses less bandwidth to transmit data and stores unique data [32]. The problem with this method is that it de-duplicates data using the entire client's resources [24]. However, this method's disadvantage is that its overheads the client CPU up to 15% by performing the Deduplication processes. Besides, if large amounts of data need to be processed, then the processing time will be increased, leading to slowing down the servers on the source side [20]. "Fig. 7" (a) shows source/client data deduplication.

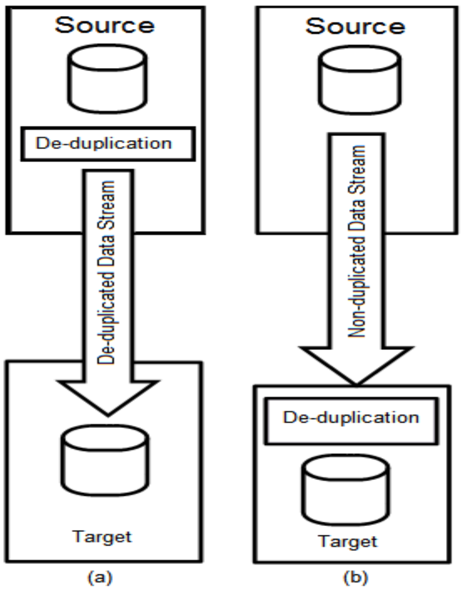

Figure. 7. (a) Source/client data deduplication and (b) target data deduplication [20].

2) Target deduplication: The duplication process occurs at the backup server-side, as all comparable data is completely transferred to the backup server [24]. Target data deduplication is fast and easy to perform the deduplication process on the server side because it contains all the data in its full replica [9]. However, this method has the disadvantage of requiring more bandwidth to transmit data due to the possibility of duplicate data [24]. Figure (8) (b) illustrates target data deduplication.

3) Advantages and disadvantages of side classification: Source deduplication requires bandwidth less than target-based deduplication. The resources that are needed by source-based more than they need for target-based deduplication. The processing overhead at the client for source-based is more than target based. Therefore, the source-based approach is slower than target based.

D. Based on Implementation

In this approach, there are two methods in this principle [32].

1) Hash-based deduplication: Hash-based deduplication is applied to find out if two documents or two chunks are the same [32]. In the beginning, the content of the data is hashed. Next, the created signatures of the chunks are compared to see if these two chunks are redundant or not [24]. If the generated signatures are the same, the two entities are discarded as being too similar. If not, it is saved on the hard drive. Finally, it can calculate the value of a data hash using any of the known hashing algorithms, including but not limited to MD5, SHA-1, SHA-256, and SHA-512 [32].

2) Content or application-aware deduplication: When using the content-aware deduplication method, data is treated as an object in the deduplication application [24]. The process of comparing is performed on the level of objects. After detecting identical parts, it saves only the bytes modified in the two parts [8]. It removes redundant data at the byte

level [24]. The content-aware technique looks for similar fragments or bytes, and only bytes that have changed or are unique are saved [32]. For example, if a backup stream is taking place on a file and it is known where the file boundaries are, knowing the boundaries can be helpful in data deduplication [34].

3) Advantages and disadvantages of implementation classification: This classification requires overcoming the shortcomings of previous classifications by implementing content-based deduplication or application-aware systems to examine and differentiate different systems based on efficiency and speed. Content-aware-based deduplication is faster than content-based deduplication because it only processes and compares data in the same type of objects and does not compare with all. In comparison, the latter is more efficient than the former. The comparison of different data deduplication techniques is shown in Table II.

V. Deduplication Stages

The main stages included in data deduplication can be summarized in four stages: data chunking, fingerprinting, indexing and writing. This research deals with these stages and the techniques used in each stage in detail [18].

Table II Various data deduplication techniques are compared and contrasted.

Deduplication Method	Throughput	Storage	Efficiency	Deduplication Ratio	Bandwidth	Cost
File Level	high	Average	less	a little	a little	a little
Block Level	a little	high	high	high	Average	Average
Source Based	Average	Average	Average	Average	a little	a little
Target- Based	Average	high	Average	Average	high	high
Inline	a little	a little	Average	a little	a little	a little
Post-Process	Average	a little	high	high	high	high

A. Chunking Algorithms

Dividing files or data streams into multiple chunks of fixed or variable length is known as data chunking [12]. A set of different chunking algorithms deals with the process of dividing files into chunks. These algorithms will be analysed and discussed in this section, and their most important advantages and disadvantages are present as follows:

1) Rabin Fingerprint Algorithm: The Rabin fingerprint [35] based on the CDC algorithm was used to eliminate redundant data in deduplication systems and network traffic [36]. The Rabin method establishes minimum and maximum bounds on the size of the chunks to prevent the algorithm's output from being very short or highly lengthy. Tiny chunks contain more fingerprints, need more space to store and process, and are therefore not cost-effective, while too long chunks lead to a decrease in deduplication efficiency [37]. Rabin's algorithm [38] suffers from two main problems; the first is to calculate the fingerprints of all the pieces, which takes a long time [37], and the second is the significant variance in the size of the chunk, which reduces the efficiency of removing duplicate data. "Fig. 8" illustrates the general view of the Rabin fingerprint algorithm [19].

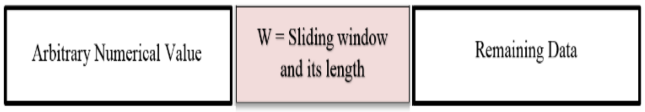

Figure. 8. Operation of Rabin fingerprint algorithm [19].

2) Two Divisors (TD) Algorithm: The TD [39] algorithm is based on using a secondary divisor to determine the breakpoints for dividing large chunks. It has a good chance of getting the duplicated chunks as it is used to overcome the boundary shift problem caused by the BFS algorithm [39]. The TD algorithm starts with checking the stream file, searching for the breakpoint, and creating a fingerprint for each chunk [34]. Then, it checks that the fingerprint of both divisors matches. It will be a breakpoint if the first divider finds a match in the fingerprint before the threshold value. If the first divisor fails to reach these breakpoints during a specific threshold value, the second divisor is used. It uses the secondary divider to try to find a breakpoint [39].

3) TTTD (Two Thresholds, Two Divisors) Algorithm: The TTTD algorithm [39] consists of combining two algorithms, the TD algorithm and the SCM (Small Chunk Merge) algorithm [40]. The TTTD algorithm improves the efficiency of the Rabin algorithm in removing duplicate data. The TTTD algorithm provided an additional backup divisor to reduce the difference in the chunk size, which has a high probability of finding the breakpoints[27]. The TTT algorithm uses four parameters in the process of discovering the breakpoints, which are: Tmin (Minimum Threshold), Tmax (Maximum Threshold), D (Primary Divisor), and D (Primary Divisor). The minimum and maximum threshold values should be set to control for variance in the chunk size so that the second divisor is half of the base divisor [40]. The TTTD algorithm has been improved [41]by adding a new switch condition to improve the time required for execution without affecting the deduplication ratio. If the breakpoint is not reached before 1600 bytes, the values of each major divisor D and second divisor D Dash have been reduced by half [42]. The TTTD algorithm improves [43] processing time by about 6% and reduces chunk size by about 5% [14].

4) MAXP: MAXP [39] is a CDC algorithm that solves the Rabin algorithm's chunk size variance problem by attempting to find local extreme values in a symmetric fixed-size window. MAXP is also recommended for eliminating network redundancy [34]. The MAXP shifts a fixed-size symmetric window over the byte stream on a byte-by-byte basis and checks whether the byte value in the center of the current window is the maximum value. The extreme points are used as a cut point to divide the input stream. The MAXP method [44] uses the strategy of locating local extreme values by rechecking some of the previously compared bytes, which significantly reduces the chunking throughput [4].

5) Bimodal: The bimodal approach combines chunks of varied average sizes and is an improved version of the CDC algorithm[26]. The bimodal algorithm performs a specific split of the size of the expected chunk in a dynamic manner. It works to split the data stream into large chunks, and for non-duplicated chunks, it divides them into smaller chunks. This algorithm is based on two methods to eliminate redundancy in large chunks [45]. The first method works by dividing the data stream into large chunks, and after identifying the areas of the new chunk's content, the data near the boundaries of the changing area chunks are divided into small chunks. The second method uses a flexible algorithm to combine the small chunks from the first method into a large one to solve the boundary shift problem [20].

6) MCDC (Multimodal Content Defined Chunking): The MCDC algorithm [46] is presented to maximize the efficiency of Bimodal Content-Defined Chunking. The MCDC finds the optimal size of chunks by changing the data size of chunks and the ability to compress data in these chunks. This algorithm works in two stages: First, the data is divided into fixed-size chunks, and then the Compression ratio (CR) is found separately for each chunk [42]. Dividing the data into fixed chunks led to the boundary shift problem. In the second stage, the MCDC algorithm has solved this problem by dividing the data stream into variable chunk sizes using Uni-modal chunking and calculating the compression ratio for each of them [42]. The

dividing using variable-size chunks and based on the comparison fingerprint technology reduced the number of chunks and lowered overall system cost while maintaining effective deduplication [45].

7) Leap-Based: Leap-based CDC algorithm [47] add a new control function to see if the window is qualified or not. It is used to improve the algorithms that use the CDC algorithm to remove duplicate data. This algorithm uses a pseudo-random method instead of the methods used in many CDC algorithms. "The Transformation derived from the locality-sensitive hashing and the theorem that the sum or the difference of normal distribution is still a normal distribution" [47]. The leap based has two parameters, M and Pw, and these parameters determine the performance and chunk size of the leap-based CDC, where M is the number of satisfactory windows and Pw is the window interpolation probability. The lead-based CDC algorithm uses two parameters to determine performance [45].

8) AE Algorithm: AE Asymmetric Extremum Algorithm [27] significantly improved the performance and efficiency of existing chunking algorithms. Instead of employing a fixed-size window like the MAXP Algorithm, AE solves the boundary shift issue using an asymmetric variable window[44]. It works to find the maximum local extreme value in the window, does not need backtracking, and needs only one comparison [48]. Therefore, the AE algorithm is high-speed, and the variance in the chunk size is minimal compared to other chunking existing CDC [44] algorithms. It does not impose any restrictions on the size of the chunk size [34]. "Fig. 9" illustrates the AE Asymmetric Extremum Algorithm [19].

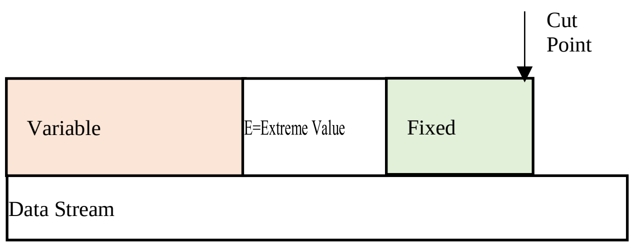

Figure. 9. Operation of Asymmetric Extremum Algorithm [19].

9) Rapid Asymmetric Maximum Algorithm (RAM): The RAM algorithm is a hash-free chunking approach based on AE that declares chunk cut-points using bytes values. It reads data as a byte stream without putting a window at the end of every chunk. Due to the usage of two windows, one fixed and the other variable, RAM employs the same algorithm as AE [19]. However, the RAM method places the fixed-sized window at the start of the chunk, followed by the variable-sized window and the byte with the highest value [12]. The RAM algorithm takes less computation time because it searches for a byte greater or equal to the current maximum value. Unlike the AE algorithm, which searches for data equal to or less than a current value. Since there is a lower probability that a byte is higher than the current value, the RAM algorithm is less overhead than the AE algorithm [49], so RAM's throughput is better than the algorithm AE [48]. "Fig. 10" illustrates the general view of the Rapid Asymmetric Maximum Algorithm (RAM [19].

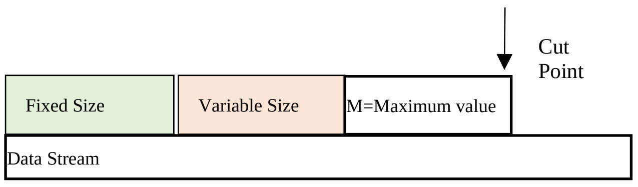

Figure. 10. operation of Rapid Asymmetric Maximum Algorithm (RAM) [19]

10) Minimal Incremental Interval (MII): The MII algorithm[48] was introduced based on incremental data synchronization. Since data is saved on the physical disk, the length of the chunk is considered one of the essential factors in the performance of earlier algorithms such as (AE and RAM). Because it is based on an incremental backup method, chunking is used to pick just the new data, which is not kept on the physical disk. The MII algorithm does not depend on chunk length because it is an incremental synchronization chunking algorithm that has the potential to manage byte shifting problems. MII

compares the byte that was read recently to the byte already existing. The MII can provide better ways to solve the byte shift problem, but the efficiency of this algorithm, in general, is not good, and the performance was poor in the variation of the size of the chunk [48].

11) Parity Check of Interval (PCI): The PCI algorithm solves the MII algorithm problem. It circumvents the boundary shift problem and can locate precisely where the data changes in incremental synchronization. PCI algorithm reads files as a stream of data and consists of a window of length (w) where the window's header is set as the first bye of the file. This method reduces the bandwidth needed to send data across the network, but the speed of dividing data into chunks is lower than the AE and RAM methods [50].

12) Quick CDC: The (Rabin CDC) algorithm takes a long time, and the ratio of deduplication data is low because it depends on byte-by-byte computation. The Quick CDC method employs three techniques to increase cutting speed, deduplication rate, and CDC throughput [12]. In the first technique, the Quick CDC algorithm can jump straight to their chunk boundaries in the case of duplicate chunks that appear several times. The second technique, for the unique chunk, the Quick CDC method overrides the minimum chunk length. Third, The Quick CDC distributes the chunk length into a small area since it can dynamically adjust the mask bits so that chunks are always more significant than the minimum chunk length. As a result, the Quick CDC algorithm improved the chunking speed, and the deduplication ratio was slightly improved [12]. Table III show the Advantage and Disadvantage of different Chunking Methods.

B. Hashing and Fingerprint

The data is broken up into blocks or chunks, and a unique hash value is created for each chunk. A sequence of hash values results from this [51]. The main task of the hash function is to create a unique fingerprint for each file or chunk, and this process aims to convert an extensive data set of variable length into a data set smaller in size and of fixed length [52]. The chunk between the beginning of the file and the breakpoint location, or between the old breakpoint and the new breakpoint position, is passed to the hash function (MD5, SHA-1) for hash value comparison when comparing chunks [34]. Multi-threading expedites the fingerprint process by using multi-core CPUs' capabilities [53].

1) The MD5 Hashing Algorithm: MD5 contains a series of numbers, and it was built based on the md4 algorithm, which is faster than MD5. The MD5 is more secure than the MD4 algorithm. The main objective of the algorithm is to protect the data's integrity and identify any changes made to the data. The results for the MD5 algorithm are always of a fixed size with a hash value of 128 bytes [54]. It produces a string consisting of four 32-bit blocks each. The MD5 method, which includes four processing cycles, is applied to the messages to be encrypted. In a digital signature, encryption, data identification, and data protection applications, the MD5 algorithm is commonly employed [11].

2) The SHA-1 Hashing Algorithm: The National Institute of Standards and Technology (NIST) developed the (SHA-1) algorithm as a security mechanism based on the results of the (SHA) algorithm. The MD4 method is the basis for the hash algorithm, SHA-1 [55]. The (SHA-1) algorithm always outputs 160 bits, regardless of the size of the message. The algorithm (SHA-1) uses complex methods to transform data and logical functions [56]. For processing units, this arithmetic process is decomposed into the 32-bit words of 512-bit size, with four loop operators and 20 cycles for each circuit, for 80 cycles [11]. SHA-1 is more potent in encryption when compared to MD5, but it takes more time for data encryption. Algorithm (SHA-1) contains 80 iterations, while algorithm (MD5) contains 64 iterations, so it is slower than (MD5). One of the essential applications of the (SHA-1) and (MD5) algorithm is deduplication, where the chunk hash computation expresses the bottle in deduplication [11].

Table III Advantages and Disadvantages of The Chunking Methods

Method	Advantage	Disadvantage
Rabin Fingerprint Algorithm	Eliminate redundant data deduplication systems. Reduce network traffic.	The chunking output is low. It takes a long time. Significant variance in variance size. Data removal efficiency is low
TD Two Divisors Algorithm	Reduces chunk size. Good chance to find duplicated chunks.	Duplicate chunk. Detection problem.
TTTD	Improves the efficiency of the deduplication ratio.	The chunking output is low.
MAXP	Computational overhead is generally reduced. Reduces the contrast between chunks	Throughput of chunking is low.
Bimodal	More duplicate data is eliminated.	Suffers from shifting boundaries
MCD	Boundary shift with the best chunk size.	Boundary shift problem.
Leap-Based	Improvement to deduplication performance.	Additional overheads in the calculation.
AE Asymmetric Extremum Algorithm	gains high performance. Very fast. Smaller chunk variance.	Less resistance on byte shifting. It takes more time to process chunks.
RAM Rapid Asymmetric Maximum Algorithm	Reduce computational expenses. The productivity of chunking is high. High chunking speed. The cost of chunking is low.	Boundary shift problem.
PCI Parity Check of Interval	It has a greater ability to resist byte shifting.	The variance in size was very poor. The algorithm's efficiency is insufficient.
MII Minimal Incremental Interval	Manage byte shifting problem.	Adjusting the chunk size is difficult. The efficiency of the method is low.
Quick CDC	Enhance chunking speed and enhance the throughput of CDC.	The deduplication ratio is slightly improved.

3) The Mathematical Bounded Linear Hashing Algorithm: The linear hash method comprises mathematical boundaries formed by multiplying distinct random values by a predefined quantity of non-repeatable zero bytes. It's enough to produce distinct unique signatures to identify the plaintext contents of the chunks by using different number sequences to obtain different short hash values. The hash functions of massive data are described by mathematical signatures, which have algebraic features and a low collision probability [6]. Furthermore, compared to typical security hash functions, the arithmetic operations utilized to produce hash code are fundamental, resulting in a relatively minimal processing cost. This approach's computational cost is minimal compared to classic hash algorithms like MD5 and SHA-1. A 16-bit mathematical function is used to produce each hash. Using several hash functions to represent the data content can help decrease collisions and enhance the lookup stage [32].

C. Indexing and Matching

The hashing and indexing process consists of a temporary lookup table to store the name of the chunks and their hash values[20]. The new hash is compared to the previously stored hash values in indexing to identify duplicate data chunks. Two or more chunks are considered duplicates if their fingerprints match since the duplicates are removed, and only the unique chunks are stored [34]. Every hash value at location (i) is compared with all hashes from location (i + 1) to the end of the sequence. A new reference is created when the hash values are equal. The sequence of hash values is extended by duplicate identification tags and backward references at the end of this stage [51]. In deduplication matching steps. If the hash values are the same, the procedure compares the two chunks byte by byte; if they are the same, the system removes the new chunk and adds a logical reference to the location of the old one. This operation takes much time and overhead the system [14]. One of the important challenges facing deduplication is the possibility of expanding the fingerprint indexing table. If the size of the fingerprint table is more than the whole amount of RAM, the hard disk index search becomes a bottleneck [57].

D. Writing on Disk

Each unique chunk is added to the system and requires a corresponding (hash, location) entry to be inserted into the system's fingerprint index. Even for modest data sets, the fingerprint index size can exceed the system's RAM size. Let's consider a chunk store with 20TB of unique data: if the fingerprint index only stores each chunk's SHA-1 hash (20B), an average chunk size of 4KB would result in a 100GB index! In general, caching is the technique we use to improve our performance whenever our data structures exceed the bounds of our memory [37]. The standard caching techniques rely on good locality to be effective (spatial and/or temporal locality) [58]. Unfortunately, seen that SHA-1 fingerprints are independently and uniformly distributed, and as a result, fingerprint index queries have no locality of reference. The fingerprint index performs poorly when normal caching methods are naively used, and each lookup still necessitates a costly disk search. The Data Domain deduplication solution addresses this issue, known as the disk index bottleneck problem [58].

VI. CONCLUSION

Many companies and organizations use different techniques to remove redundant data to get rid of redundant data. In this study, many redundant data reduction approaches are discussed, like the many types of data deduplication techniques categorized according to granularity-based, time-based, side-based deduplication, and Implementation has been studied and clarified. The most important characteristics related to these types are discussed. The challenges and solutions to issues related to data duplication are covered. In this survey, the most important advantages and disadvantages of using these types on a large scale are also reviewed and discussed. The research included an overview of the methods for splitting data into fixed and variable chunks and provided tips on maximizing the efficiency and productivity of data deduplication. Various hashing approaches have been examined, and their primary methodologies have been varied. The most important types of hashing methods studied are MD5, SHA-1, and mathematical model-based hashing. The study also looked at the indexes used to find duplicate data chunks and save unique ones. In addition, comparisons were made between the different methods and algorithms according to the criteria of time, efficiency, and the percentage of de-duplicating data.

ACKNOWLEDGMENT

The authors would like to thank the Iraqi commission for computers and informatics\ Informatics Institute of Postgraduate Studies (www. iips.edu.iq) Baghdad-Iraq for its support in the present work.

REFERENCES

[1]. G. Sujatha and J. R. Raj, "A Comprehensive Study of Different Types of Deduplication Technique in Various Dimensions," A Compr. Study Differ. Types Deduplication Tech. Var. Dimens., vol. 13, no. 3, pp. 316–324, 2022.

[2]. S. T. Ahmed and L. E. George, "Lightweight hash-based de-duplication system using the self detection of most repeated patterns as chunks divisors," J. King Saud Univ. - Comput. Inf. Sci., 2021, doi: https://doi.org/10.1016/j.jksuci.2021.04.005.

[3]. H. Kambo and B. Sinha, "Secure data deduplication mechanism based on Rabin CDC and MD5 in cloud computing environment," in 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT), May 2017, pp. 400–404. doi: 10.1109/RTEICT.2017.8256626.

[4]. Y. Zhang et al., "A Fast Asymmetric Extremum Content Defined Chunking Algorithm for Data Deduplication in Backup Storage Systems," IEEE Trans. Comput., vol. 66, no. 2, pp. 199–211, 2017, doi: 10.1109/TC.2016.2595565.

[5]. Y. Cui, Z. Lai, X. Wang, and N. Dai, "QuickSync: Improving Synchronization Efficiency for Mobile Cloud Storage Services," IEEE Trans. Mob. Comput., vol. 16, no. 12, pp. 3513–3526, 2017, doi: 10.1109/TMC.2017.2693370.

[6]. A. S. M. Saeed and L. E. George, "Data deduplication system based on content-defined chunking using bytes pair frequency occurrence," Symmetry (Basel)., vol. 12, no. 11, pp. 1–21, 2020, doi: 10.3390/sym12111841.

[7]. A. V. and K. S. Sankar, "Study of Chunking Algorithm in Data Deduplication," Adv. Intell. Syst. Comput., vol. 398, pp. 319–329, 2016, doi: 10.1007/978-81-322-2674-1.

[8]. N. Sharma, A. V. Krishna Prasad, and V. Kakulapati, "Data deduplication techniques for big data storage systems," Int. J. Innov. Technol. Explor. Eng., vol. 8, no. 10, pp. 1145–1150, 2019, doi: 10.35940/ijitee.J9129.0881019.

[9]. M. K. Yoon, "A constant-time chunking algorithm for packet-level deduplication," ICT Express, vol. 5, no. 2, pp. 131–135, 2019, doi: 10.1016/j.icte.2018.05.005.

[10]. S. M. A. Mohamed and Y. Wang, "A survey on novel classification of deduplication storage systems," Distrib. Parallel Databases, vol. 39, no. 1, pp. 201–230, 2021, doi: 10.1007/s10619-020-07301-2.

[11]. A. S. M. Saeed and L. E. George, "Fingerprint-based data deduplication using a mathematical bounded linear hash function," Symmetry (Basel)., vol. 13, no. 11, pp. 1–19, 2021, doi: 10.3390/sym13111978.

[12]. Z. Xu and W. Zhang, "QuickCDC: A Quick Content Defined Chunking Algorithm Based on Jumping and Dynamically Adjusting Mask Bits," in 2021 IEEE Intl Conf on Parallel Distributed Processing with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom), 2021, pp. 288–299. doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00049.

[13]. W. Xia et al., "The design of fast content-defined chunking for data deduplication based storage systems," IEEE Trans. Parallel Distrib. Syst., vol. 31, no. 9, pp. 2017–2031, 2020, doi: 10.1109/TPDS.2020.2984632.

[14]. H. A. S. Jasim and A. A. Fahad, "New techniques to enhance data deduplication using content based-TTTD chunking algorithm," Int. J. Adv. Comput. Sci. Appl., vol. 9, no. 5, pp. 116–121, 2018, doi: 10.14569/IJACSA.2018.090515.

[15]. N. Kumar, R. Rawat, and S. C. Jain, "Bucket based data deduplication technique for big data storage system," in 2016 5th International Conference

on Reliability, Infocom Technologies and Optimization, ICRITO 2016: Trends and Future Directions, 2016, pp. 267–271. doi: 10.1109/ICRITO.2016.7784963.

[16]. K. Akhila, A. Ganesh, and C. Sunitha, "A Study on Deduplication Techniques over Encrypted Data," Procedia Comput. Sci., vol. 87, pp. 38–43, 2016, doi: 10.1016/j.procs.2016.05.123.

[17]. A. Kaur and S. Sharma, "An Efficient Framework and Techniques of Data Deduplication in Cloud Computing," Int. J. Comput. Sci. Technol., vol. 8491, pp. 27–31, 2017.

[18]. J. Malhotra and J. Bakal, "A survey and comparative study of data deduplication techniques," in 2015 International Conference on Pervasive Computing: Advance Communication Technology and Application for Society, ICPC 2015, 2015, pp. 1–5. doi: 10.1109/PERVASIVE.2015.7087116.

[19]. D. Viji and D. S. Revathy, "Comparative Analysis for Content Defined Chunking Algorithms in Data Deduplication," Webology, vol. 18, no. SpecialIssue2, pp. 255–268, 2021, doi: 10.14704/WEB/V18SI02/WEB18070.

[20]. H. A. Jasim and S. By, "An Improved Technique to Enhance De-Duplication using Content-Based TTT-D Chunking Algorithm A," Univ. Baghdad - Coll. Sci. Comput. Sci. Dep., no. March, 2018.

[21]. R. Vikraman and A. S, "A Study on Various Data De-duplication Systems," Int. J. Comput. Appl., vol. 94, no. 4, pp. 35–40, 2014, doi: 10.5120/16334-5616.

[22]. R. Misal and B. Perumal, "Data deduplication for efficient cloud storage and retrieval," Int. Arab J. Inf. Technol., vol. 16, no. 5, pp. 922–927, 2019.

[23]. P. M. Kumar, G. Usha Devi, S. Basheer, and P. Parthasarathy, "A Comprehensive Study on Data Deduplication Techniques in Cloud Storage Systems," Int. J. Grid Util. Comput., vol. 11, no. 4, pp. 509–516, 2020, doi: 10.1504/IJGUC.2020.108450.

[24]. G. Sujatha and J. R. Raj, "A Comprehensive Study of Different Types of Deduplication Technique in Various Dimensions," Int. J. Adv. Comput. Sci. Appl., vol. 13, no. 3, pp. 316–323, 2022, doi: 10.14569/IJACSA.2022.0130339.

[25]. L. Conde-Canencia and B. Hamoum, "Deduplication algorithms and models for efficient data storage," Proc. - 24th Int. Conf. Circuits, Syst. Commun. Comput. CSCC 2020, pp. 23–28, 2020, doi: 10.1109/CSCC49995.2020.00013.

[26]. A. El-Shimi, R. Kalach, A. Kumar, A. Oltean, J. Li, and S. Sengupta, "Primary data deduplication - Large scale study and system design," Proc. 2012 USENIX Annu. Tech. Conf. USENIX ATC 2012, pp. 285–296, 2019.

[27]. Y. Zhang et al., "AE: An Asymmetric Extremum content defined chunking algorithm for fast and bandwidth-efficient data deduplication," in Proceedings - IEEE INFOCOM, Apr. 2015, vol. 26, pp. 1337–1345. doi: 10.1109/INFOCOM.2015.7218510.

[28]. A. Bhalerao, "A Survey : On Data Deduplication for Efficiently Utilizing Cloud Storage for Big Data Backups," Int. Conf. Trends Electron. Informatics, no. August 2019, 2017, doi: 10.1109/ICOEI.2017.8300844.

[29]. D. Kim, S. Song, and B. Y. Choi, Data deduplication for data optimization for storage and network systems. 2016. doi: 10.1007/978-3-319-42280-0.

[30]. E. Manogar and S. Abirami, "A study on data deduplication techniques for optimized storage," 6th International Conference on Advanced Computing, ICoAC 2014. pp. 161–166, 2015. doi: 10.1109/ICoAC.2014.7229702.

[31]. K. Gnana Sambandam and E. Kamalanaban, "Proceedings of the International Conference on Soft Computing Systems," Adv. Intell. Syst. Comput., vol. 398, pp. 319–329, 2016.

[32]. S. M. A. Mohamed and Y. Wang, "A survey on novel classification of deduplication storage

systems," Distrib. Parallel Databases, vol. 39, no. 1, pp. 201–230, 2021.

[33]. K. Vijayalakshmi and V. Jayalakshmi, "Analysis on data deduplication techniques of storage of big data in cloud," Proceedings - 5th International Conference on Computing Methodologies and Communication, ICCMC 2021. pp. 976–983, 2021. doi: 10.1109/ICCMC51019.2021.9418445.

[34]. A. Bhalerao and A. Pawar, "A survey: On data deduplication for efficiently utilizing cloud storage for big data backups," Proc. - Int. Conf. Trends Electron. Informatics, ICEI 2017, vol. 2018-Janua, no. May, pp. 933–938, 2018, doi: 10.1109/ICOEI.2017.8300844.

[35]. Rabin, "Fingerprinting by random polynomials.pdf." 1981.

[36]. A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee, "Redundancy in network traffic: Findings and implications," SIGMETRICS/Performance'09 - Proc. 11th Int. Jt. Conf. Meas. Model. Comput. Syst., vol. 37, no. 1, pp. 37–48, 2009, doi: 10.1145/1555349.1555355.

[37]. B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file system," FAST 2008 - 6th USENIX Conf. File Storage Technol., pp. 269–282, 2008.

[38]. E. Kruus, C. Ungureanu, and C. Dubnicki, "Bimodal content defined chunking for backup streams," Proc. FAST 2010 8th USENIX Conf. File Storage Technol., pp. 239–252, 2010.

[39]. K. Eshghi and H. K. Tang, "A framework for analyzing and improving content-based chunking algorithms," Hewlett-Packard Labs Tech. Rep. TR, 2005, [Online]. Available: http://shiftleft.com/mirrors/www.hpl.hp.com/techreports/2005/HPL200530R1.pdf%5Cnpapers3://publication/uuid/053B1556-804C-4F39-BD0B-2EBD9C047F30

[40]. N. Kumar, S. Antwal, G. Samarthyam, and S. C. Jain, "Genetic optimized data deduplication for distributed big data storage systems," in 2017 4th International Conference on Signal Processing, Computing and Control (ISPCC), 2017, pp. 7–15. doi: 10.1109/ISPCC.2017.8269581.

[41]. T. S. Moh and B. C. Chang, "A running time improvement for the two thresholds two divisors algorithm," Proc. Annu. Southeast Conf., 2010, doi: 10.1145/1900008.1900101.

[42]. S. O. Majed and S. K. Thamer, "Cloud based industrial file handling and duplication removal using source based deduplication technique," AIP Conf. Proc., vol. 2292, no. October, 2020, doi: 10.1063/5.0030989.

[43]. D. Datta, S. Mishra, and S. S. Rajest, "Quantification of tolerance limits of engineering system using uncertainty modeling for sustainable energy," Int. J. Intell. Networks, vol. 1, no. May, pp. 1–8, 2020, doi: 10.1016/j.ijin.2020.05.006.

[44]. A. Bhalerao and A. Pawar, "Two-threshold chunking (TTC): Efficient chunking algorithm for data deduplication for backup storage," Int. J. Sci. Technol. Res., vol. 8, no. 9, pp. 754–757, 2019.

[45]. S. H. A. H. Algorithms, H. Abdulsalam, and A. A. Fahad, "Evaluation of Two Thresholds Two Divisor Chunking Algorithm Using Rabin Finger print, Adler, and SHA1 Hashing Algorithms," Iraqi J. Sci., vol. 58, no. 4C, 2017, doi: 10.24996/ijs.2017.58.4c.19.

[46]. J. Wei, J. Zhu, and Y. Li, "Multimodal Content Defined Chunking for Data Deduplication," Available: https://www.researchgate.net/publication/261286019, Research gate., 2014.

[47]. C. Yu, C. Zhang, Y. Mao, and F. Li, "Leap-based Content Defined Chunking — Theory and Implementation," in 2015 31st Symposium on Mass Storage Systems and Technologies (MSST), May 2015, pp. 1–12. doi: 10.1109/MSST.2015.7208290.

[48]. C. Zhang et al., “MII: A novel content defined chunking algorithm for finding incremental data in data synchronization,” IEEE Access, vol. 7, pp. 86932–86945, 2019, doi: 10.1109/ACCESS.2019.2926195.

[49]. P. K. Krishnaprasad and B. A. Narayamparambil, “A Proposal for Improving Data Deduplication with Dual Side Fixed Size Chunking Algorithm,” in 2013 Third International Conference on Advances in Computing and Communications, Aug. 2013, pp. 13–16. doi: 10.1109/ICACC.2013.10.

[50]. C. Zhang, D. Qi, W. Li, and J. Guo, “Function of Content Defined Chunking Algorithms in Incremental Synchronization,” IEEE Access, vol. 8, pp. 5316–5330, 2020, doi: 10.1109/ACCESS.2019.2963625.

[51]. P. Sobe, D. Pazak, and M. Stiehr, “Parallel Processing for Data Deduplication,” PARS-Mitteilungen, vol. 32, pp. 109–118, 1AD.

[52]. L. E. G. 2 Ahmed Sardar M. Saeed, “symmetry Data Deduplication System Based on Frequency Occurrence,” Symmetry (Basel)., vol. 12, no. 11, p. 1841, 2020.

[53]. Y. Zhang, Y. Wu, and G. Yang, “Droplet: A distributed solution of data deduplication,” Proc. - IEEE/ACM Int. Work. Grid Comput., pp. 114–121, 2012, doi: 10.1109/Grid.2012.21.

[54]. S. Kumar and E. P. Gupta, “A Comparative Analysis of SHA and MD5 Algorithm,” Int. J. Comput. Sci. Inf. Technol., vol. 5, no. June 2014, pp. 4492–4495, 2014.

[55]. A. Kshemkalyani, “An Efficient Implementation of SHA-1 Hash Function,” IEEE Int. Conf. Electro-Information Technol., vol. 43, no. 1, pp. 47–52, 2006.

[56]. X. Chan and G. Liu, “Discussion of One Improved Hash Algorithm Based on MD5 and SHA1,” Lect. Notes Eng. Comput. Sci., vol. 2167, no. 1, pp. 270–273, 2007.

[57]. W. Xia, D. Feng, H. Jiang, Y. Zhang, V. Chang, and X. Zou, “Accelerating content-defined-chunking based data deduplication by exploiting parallelism,” Future Generation Computer Systems, vol. 98. pp. 406–418, 2019. doi: 10.1016/j.future.2019.02.008.

[58]. A. Venish and K. S. Sankar, “Survey Paper for Dedup,” Indian J. Sci. Technol., vol. 8, no. October, pp. 1–7, 2015, doi: 10.17485/ijst/2015/v8i26/.

Cite this article as :

Hashem Bedr Jehlol, Loay E. George, "Big Data Backup Deduplication : A Survey", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 9 Issue 4, pp. 174-191, July-August 2022. Available at doi : https://doi.org/10.32628/IJSRSET229425
Journal URL : https://ijsrset.com/IJSRSET229425