Distributed and Parallel Databases (2021) 39:201–230
https://doi.org/10.1007/s10619-020-07301-2

A survey on novel classification of deduplication storage systems

Shawgi M. A. Mohamed[1] · **Yongli Wang**[1]

Published online: 16 June 2020

Abstract
The huge blast of information caused a lot of dilemmas in both storage and retrieval procedures. The enlargement in a massive quantity of digital data requirements imposes more storage space, which in turn radically increases performance and cost of backup. Data deduplication is one of the techniques that vanishes replicated data, decreases the bandwidth, and minimizes the disk usage and cost. Since various researches have been broadly considered in the literature, this paper reviews the ideas, categories, and different storage approaches that use data deduplication. Apart from the well-known classification that uses Granularity, Side, Timing, and Implementation for classifying the deduplication approaches, a new classification principle is adopted using the storage location. This classification identifies and describes the diverse methods. Moreover, the deduplication systems are comprehensively described according to the storage location, including Local, Centralized, and Clustered storage systems. Furthermore, the describing objectives, used techniques, features, and drawbacks of the most advanced methods of each type are broadly tackled. Finally, the major deduplication systems' challenges are recognized and illustrated.

Keywords Centralized storage systems · Classification · Clustered storage systems · Data deduplication · Local storage systems

1 Introduction

With the massive expansion data quantity day by day, the storage burden has imposed crucial requirements for data centers [1–3]. Therefore, the perception of cloud computing [4–6] has become increasingly popular. Cloud computing as a

✉ Shawgi M. A. Mohamed
vandam_9000@hotmail.com

Yongli Wang
yongliwang@mail.njust.edu.cn

[1] Nanjing University of Science and Technology, Nanjing, China

well-known industry model is developed from distributed processing [7–9], parallel processing [10], and grid computing [11–13].

At present, Google, Amazon, IBM, Microsoft, Sun [14–16] and other IT giants are all in search of developing cloud computing technologies and products. For example, Google has been devoted to encourage application engines [17, 18] Based on techniques such as MapReduce [19–21] and BigTable [22], which present user methods and ways to process enormous volumes of data. Deduplication technology [23–25] which is a well-known method and universally used in disk-based backup and storage systems intended to decrease the utilization of storage capacity.

The deduplication process is a well-known smart technology in which the duplicate data is removed and only the unique ones are preserved in a way that enables us to fully recover the data at any time [26]. For example, a typical email system might contain 500 instances of the same 1 MB file attachment. Every time the email platform is stored, all 500 replicas of the attachment are kept, demanding 500 MB of storage space. With duplicate data canceled, only one occurrence of the attachment is already stored; the following instances are stated as the saved replica for the duplicate data cancellation ratio of around 500 to 1. The basic components of the deduplication operation are chunking, hashing, and indexing.

Cancellation of duplicate data differs from data compression algorithms, such as LZ77 and LZ78 [27, 28]. While compression algorithms determine the extra data within individual files and symbolize this extra data more efficiently, the purpose of eliminating duplicate data is to examine large amounts of data and identify large sections—such as entire files or large sections of files—identical and replace them with a shared copy.

Deduplication strives at removing both intra-file and interfiles replication over enormous data sets, stored for various periods by inconsistent users and actions, and probably even across several separate storage servers. Research shows that 75% of the digital world is a copy of [29], and 90% of the data is copied into the backup data sets [30].

The main components of the deduplication operation are chunking, hashing, and indexing. Data deduplication technology splits the data into minor blocks and adopts a hash function to give a distinctive hash value to each data block named hash. The hash function picks the data block as an input and makes a cryptographic hash value of the output. The most frequently implemented hash functions are SHA-1 [31–33], MD5, and Rabin's fingerprint hash function [34, 35]. Those hashes are kept after that in a table called a hash table. The data deduplication system matches each signature with all the ones previously saved in the hash table. If the hash value occurs within the system, then the replica block is exchanged with a pointer to that block. Otherwise, the distinctive block is saved to the disk, and the unique hash is saved in the hash table for an additional procedure.

Storing massive amounts of data is so crucial, that is why deduplication systems must be intended consequently. For example, a typical email system might contain 100 copies of the same attached file as 1 MB. Every time the email system is backed up, every 100 copies of the attachments are saved, requiring 100 MB storage space. With duplicate data canceled, only one instance of the attachment is already stored; subsequent instances are reverted back to the saved version for duplicate

data cancellation ratios from around 100 to 1 [36]. Duplicate data cancellation is often paired with data compression to provide additional storage. It is used to cancel duplicate data first to remove large numbers of duplicate data, then, compression is used efficiently to encode each of the stored pieces [37].

Deduplication systems transfer data to the central server due to efficiency and resource shortage. Centralized deduplication systems prone easily to data damage in case of any server crash [38, 39]. Clustered deduplication systems [40, 41] solve this issue by distributing data among several data storage nodes. The matter of performance, throughput, and distributing the load evenly among the nodes became vital in such a type of deduplication system.

Herein, apart from the commonly used classifications based on Granularity, Side, Timing, and Implementation, a novel comprehensive and highly flexible integrated storage location-based classification standard is illustrated. Furthermore, an extensive description of storage location deduplication systems covering Local, Centralized, and Clustered storage systems was carried out. Objectives, techniques, features, and drawbacks of most recent systems were addressed as well.

The rest of the paper is organized as follows: Sect. 2 introduces a brief of exciting deduplication storage systems classifications. Section 3 explains in detail the storage location based deduplication systems. Section 4 discusses and analyzes the efficiency, throughput, and memory consumption of each of the deduplication systems mentioned in Sect. 3. The conclusion is given in Sect. 5 based on the survey.

2 Brief of exciting deduplication systems classifications

From the above, we can briefly describe data deduplication as a process of comparing and eliminating files or chunks, which previously existed in the storage [42, 43]. The deduplication procedure removes chunks that are not distinctive. This procedure consists of five stages:

1) Splitting the incoming data stream into chunks or blocks.
2) Compute a fingerprint of each chunk of a data stream.
3) Using the fingerprint to verify whether a similar chunk of a data stream is there in saved chunk data.
4) Once the replicated chunk exists the reference will be made to the record.
5) According to the outcome, the replicated data is removed; just a distinctive block is saved.

There are various methods for removing duplicated data which is kept in the data store. Nevertheless, most of the association employs the data deduplication methods to save and to decrease the duplication dilemma [44]. The next methods conduct data deduplication as illustrated in Fig. 1.

1) Granularity based deduplication.
2) Time based deduplication.

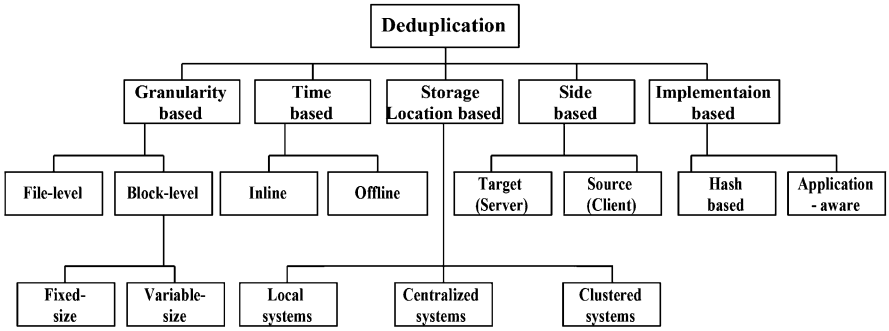

Fig. 1 Classifications of Deduplication systems

3) Side based deduplication.
4) Implementation based deduplication.
5) Storage location based deduplication.

2.1 Classification of deduplication based on granularity

There are two approaches according to this criterion [41]

2.1.1 File-level deduplication

This approach primarily catches the related files and eliminates them [45], and one copy of the file is saved. A reference is employed to point the unique file for the successive copies. This approach doesn't care about the elements within the file. For instance, two text files with plain label alter are saved as two different files. The benefit of this approach is that it is straightforward and rapid. This approach is famous as Single Instance Storage as well [45].

2.1.2 Block or sub-file deduplication

The file is divided into a small number of blocks named chunks [46], and replica chunks are found out by a particular mechanism. If the data is distinctively written to a disk, or else, the only reference is exploited to point the disk position. Depending on the size of a chunk; there are two procedures in chunk deduplication:

2.1.2.1 Fixed-size chunk deduplication It divides the data into fixed-size chunks by Fixed Sized Chunking (FSC) algorithm [47]. The drawback of this technique is its failure to detect replicated data as few alterations in the chunk lead to be rewritten of all successive chunks to the drive. However, this technique is rapid, easy, and requires least CPU overhead.

2.1.2.2 Variable-size chunk deduplication It divides the data into mutable size chunks by Contain Dividing Chunking (CDC) algorithm [48]. The profit of this

technique is its high deduplication ratio, where any modification happens, the border of a block is altered and no changes in successive chunks, and it sets aside more storage room when compared with fixed-size chunks deduplication. However, this technique needs extra CPU cycles to recognize chunk borders and to scan the entire file.

2.1.3 Advantages and disadvantages of granularity classification

It can be easily noticed that this classification divides deduplication systems depending on granularity, which explains the impact of that in the various storage approaches, the techniques used in those systems, and the impact of those diverse methods in deduplication efficiency, resources consumption, and performance. The main disadvantage of this classification is the limitation of traditional hashing deduplication storage systems.

2.2 Classification of deduplication based on time

There are two methods under this standard [41]

2.2.1 Inline deduplication

It performs deduplication directly after getting the data at a backup appliance. This method needs less storage capacity required for backup [41]. Inline deduplication can be done in the client-side or when the data is transferring from the client/source to the server.If a block of data arrives into the deduplication destination, it specifies that, whether the data block has been existed already or not. If the data already there, it is considered as a duplicate block then a reference to that block will be written. If it is classified as unique, the deduplication device writes that block into the storage. The main disadvantage of this method is that its performance is affected by network capacity

2.2.2 Offline deduplication

It performs Deduplication after the extradited data is written in disk i.e., Deduplication is listed afterward. This technique needs extra storage space to save backup data [41].

The main feature offline deduplication is that its performance is higher than InLine deduplication. Another merit of this approach is the capability to share index and metadata, and hence aggregation for higher availability (HA) is simpler, the data replication can be more effective. The drawback is that the need for quick disk cache, which usually makes the initial purchase cost more expensive than inline methods.

2.2.3 Features and drawbacks of time classification

It is more general classification type, where all the deduplication systems are classified according to the time of the process if it is during the storage process or after caching it until the availability of time when the flow of data stop, which enable it to cover all the storage systems, both traditional and non-traditional. Whereas, it is found to be stranded in the classification of hybrid deduplication systems, which is modern storage systems eliminating the duplication of some data that meet certain conditions, and postpone other data to remove the redundant data later, thereby increasing their performance and speed such as DIODE [49].

2.3 Classification of deduplication based on side

There are two approaches belong to this principle [41]

2.3.1 Source/client deduplication

The whole deduplication procedure is completed at the source/client part before transferring the data to a backup appliance [41]. Distinctive data is transmitted to the backup device with the lowest existing bandwidth and requires less space.

The two main features of source /client deduplication are that it needs less bandwidth to transmit data and only alerted data is stored.

2.3.2 Target deduplication

The deduplication procedure is completed at the backup device [41]. After it receives the data with its complete replica, this technique requires extra network bandwidth, and it exempts a backup client from the deduplication procedure.

2.3.3 Merits and shortcomings of side classification

It is a general classification also puts under its umbrella all the systems of deduplication depending on where the duplicate is removed to the source and target deduplication, thus clear way to know the techniques used in both types and the extent of their impact on their effectiveness and efficiency and speed of performance, opening the door so to highlight the flaws and the advantages of each, But it ignored the uniqueness of hybrid systems such as SAM [50] and AA-dedupe [51], which are categorized as target deduplication systems, although they eliminate duplication at both side source and target.

2.4 Classification of deduplication based on implementation

There are two methods under this criterion [52]

2.4.1 Hash-based deduplication

From the methods above, i.e. file and chunk level deduplication, this technology is implemented to recognize whether two documents or chunks are similar. A fingerprint is produced by applying a hash function such as MD5, SHA-1 for files or fragments. If the created signatures are equal, then the two entities are identical so that they will be discarded. Otherwise, they are going to be stored in the drive [52].

2.4.2 Content or application-aware deduplication

It splits the data flow into significant segments by knowing the content of the items such as files, database items, and application items [52]. Then it detects the replica segments and saves only the bytes altered in the two parts, thus known as byte-level deduplication [52].

2.4.3 Advantages and disadvantages of implementation classification

This type of classification divides the systems of deduplication according to the implementation into hashing systems and content or application aware-systems in an attempt to overcome the shortcomings in which the previous classifications occurred, thus studying different systems and distinguish them according to their efficiency and speed as the latter type outweighs the former in speed-wise but it came below it in terms of deduplication efficiency due to the need to distinguish between different data according to their type. The main disadvantage of this type of classification is also lack of a classification of non-traditional systems such as genetic deduplication storage systems [53].

3 Classification of deduplication systems based on storage location

As described in the previous section, the second and third classifications of time and side deduplication systems, respectively, are incapable of keeping up with the evolution of modern deduplication systems. The other two categories, the first and fourth, based on the factors of granularity and implementation, are limited only to traditional systems based on Hashing and unable to classify modern systems such as genetic systems. Therefore, we have proposed a new classification standard based on storage location that has two great advantages, and that of inclusiveness and flexibility, as its comprehensiveness lies in its ability to place all deduplication systems under its canopy while its remarkable flexibility is demonstrated in its ability to absorb any new deduplication systems that may appear in the future.

The primary goal of any deduplication system is achieving high deduplication efficiency with minimum possible overheads such as low system resources

consumption, low network bandwidth, and low system latency. We classify deduplication systems depending on storage location into three types :

3.1 Local storage deduplication systems

These systems conduct deduplication and data storage operations entirely in the same device [54]. Their principal function is to achieve a balance between deduplication efficiency and overheads to store as much large data as possible [55]. They exploit their available resources by adopting a lot of techniques in chunking, indexing, and data management. The biggest challenge facing this type of storage system is low performance due to limited resources. The general design of local deduplication storage systems is illustrated in Fig. 2.

REBL [56] was one of the pioneer's deduplication local storage systems. It was proposed to decrease storage overheads like fingerprints calculations and memory consumption to achieve high deduplication efficiency. REBEL combines three methods: compression, removal of corresponding content-defined blocks, and delta compression of the same blocks. It adopts the CDC algorithm with an average 4 KB chunk size by Rabin fingerprints, DERD algorithms [56] and SHA-1 hash function,

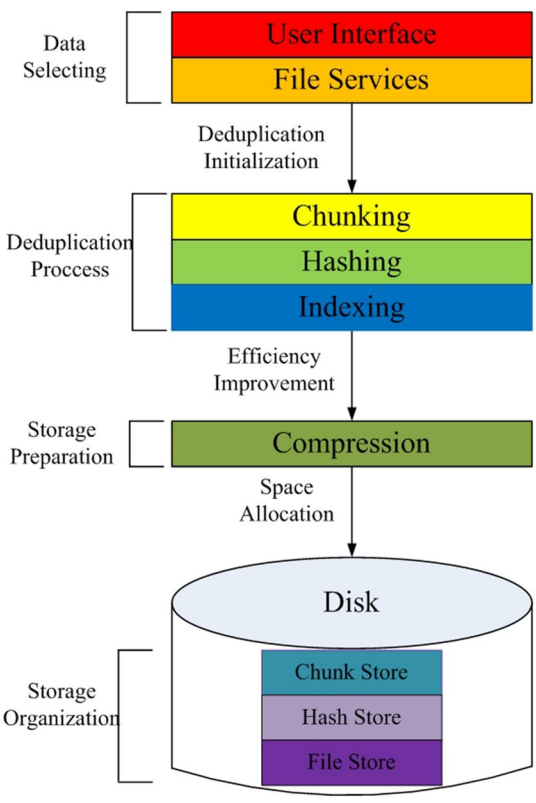

Fig. 2 General Design of Local Deduplication Storage System

(342) super-fingerprints contain (84) signatures with MD5 hash function. The main advantage of REBL is the capability to attain reasonable deduplication efficiency by utilizing relations among similar chunks, instead of just among matching blocks, while maintaining calculation and memory overheads comparable to methods that execute redundancy discovery with a coarser granularity. The main drawback of this system is the high impact of super-fingerprint size in deduplication efficiency where the big size gives low efficiency decreasing the deduplication overheads and vice versa.

Outstanding deduplication efficiency and high productivity with low overheads were the main targets of proposing a High-Performance Deduplication system [57]. This approach composes of client and server elements in one device. This approach adopts the FSC algorithm with 4 KB chunk size, MD-5 hash function, bloom filter [58], Progressive sampled indexing [59], Grouped mark-and-sweep method and an event-driven multi-threaded client–server interaction model [57]. This scheme attained high backup throughput. However, this system exploits fix-sized segments to reduce system latency which led to boundary shifting problems [60] although it achieved high deduplication efficiency with unalterable files.

To address the previous system drawback, also, to boost deduplication speed by entirely creating pipelined and parallel calculations, P-Dedup [61] was proposed. P-Dedupe splits the deduplication procedure into 4 phases: (P1) chunking (CDC algorithm), (P2) fingerprinting (SHA-1 hash function), (P3) indexing the fingerprints, and (P4) writing chunk data and file metadata (including the chunk-to-file mappings). It adopts parallel chunking and fingerprinting mechanisms to more eliminate the hash computation bottleneck of deduplication in high-performance storage systems. P-Dedup adopts a hybrid technique making the implementations of parallel parts synchronous and the executions of parallel chunks asynchronous. This system is capable of making the full utilization of the idle compute resources in a multicore or many core-based computer systems to efficiently eliminate the removal of redundant data bottleneck in deduplication based storage systems. This system succeeded in achieving high deduplication efficiency. However, this came at the account of performance degradation due to the transmission of the bottleneck from the fingerprinting job to writing to the disk and synchronizing the threads with increasing data size.

3.2 Centralized storage deduplication systems

They are deduplication storage schemes, which store their unique data in one central server due to deduplication efficiency and resource shortage [62]. Their primary purpose is to attain high deduplication efficiency to save as much great unique data as possible. Deduplication operation involves high costs such as getting files from disk and computing cryptographic hashes. Where the former is an I/O-bound procedure and the latter is a high CPU-intensive operation, and the client may not be capable of computing fingerprints at the required level. Data loss is the primary threat to this type of system because it is stored in a single central server, making it

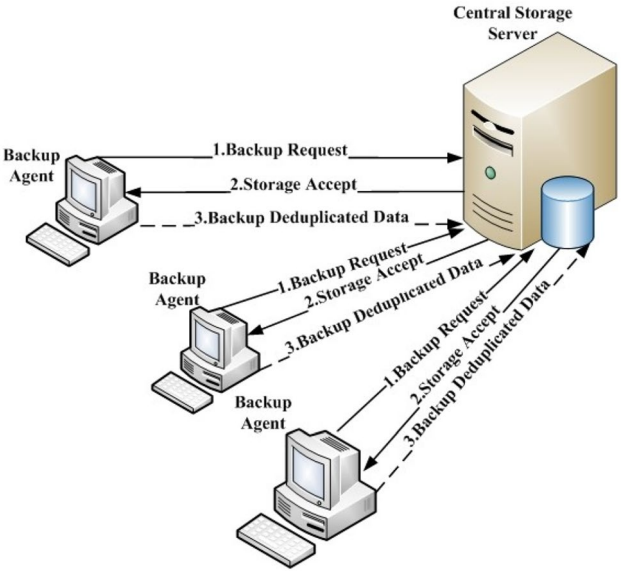

Fig. 3 General Design of Centralized Deduplication Storage System

vulnerable to lose at any time if it collapses. The Fig. 3 illustrates the general design of a centralized deduplication storage system.

In a quest to reach the best tradeoff between the deduplication efficiency and metadata storage overhead achieving a little backup window than previous schemes, MMSD [63] system was proposed. This scheme comprises two significant elements: MMSD client and server. This system utilizes file metadata including file size, type, timestamp [64], path information, and alteration occurrence, in addition to whole file (with a file larger than 1 MB) and chunk level deduplication. In the MMSD system, every globally duplicate file is symbolized with one 16 B MD5 signature, and every variable-size chunk wanted to be treated by locally is expressed with one 20 B SHA-1 fingerprint. MMSD enhanced the overall deduplication efficiency with merely 33.8% of deduplication metadata storage overhead and cut down the storage latency by at least 54.2 %. MMSD conducts deduplication globally for the file more than 1 MB with the WFC policy and chunk deduplication locally which leads to downgrading the deduplication throughput.

To preserve and present effective and rapid similarity discovery, high deduplication productivity with little system overheads in the massive data storage environment, Simdedup [65] was proposed. This near-exact deduplication system utilizes file similarity and block locality by dividing a file entity into multi-segment and exploiting resemblance to discover the most similar portions depend on the Simhash algorithm [66]. The entire deduplication system has two stages of utilizing similarity and locality, which is the similarity detection stage and redundancy removal stage. Simdedup preserves two indexes of fingerprints in RAM (Simhash Fingerprint Index (SFIndex) and chunk fingerprint index (CFIndex)) and disk, respectively to attain outstanding RAM economy and speed up

fingerprint search. SFIndex is responsible for maintaining similarity information of files, while CFIndex is in charge of preserving chunk locality for removing duplicates among documents. The precision of similarity discovery by simhash algorithm increases with a chunk number in one segment. However, this system conducts deduplication only among similar documents, so its efficiency is too low.

In a significant leap in the deduplication systems world, NIDF [67] was designed. This scheme was proposed to enhance replicated data elimination by adopting a genetic programming method to avert collision (false negatives and positives) [68]. NIDF utilizes Sequence Matching Algorithm [69] and Levenshtein's Algorithm [70] for text comparison, and then genetic programming ideas are exploited to identify the most relative match. This system ran in the I-a-a-S layer of the cloud server [71], and it is extracted from the cloud user. NIDF has revealed high deduplication efficiency which can reach up to 100% compared to the hashing method because there is no false-negative probability. The primary disadvantage of this method is the amount of time to execute the comparison with Levenshtein's algorithm, which is about twice the time spent by hashing techniques.

RevDedup [72] system tackles the issue of deduplication overhead reduction and high backup productivity. This approach conducts deduplication in two stages implementing inline deduplication by separating backup data into big-size units called segments and eliminates duplicate segments of new backups. It groups the distinctive sections into containers and saves the containers on the drive. RevDedup reads the containers and implements offline deduplication to little-size data units called chunks. Moreover, it eliminates duplicate pieces of elder backups and points them to the matching fragments in new reserves. RevDedup achieved high deduplication efficiency and throughput. This approach adopts two-level reference administration to remain track of how segments and pieces are shared. RevDedupe does not require examining all segments/chunks as in the conventional mark-and-sweep method [73], so its time drastically decreased. The only drawback of RevDedup is the extra I/Os to recognize and eliminating duplicate chunks on the drive.

DIODE [49] provides significant productivity and space-saving simultaneously for primary storage systems. This system has two innovative mechanisms: Context-aware Threshold Adjustment (CTA) for inline-phase deduplication and Deferred Priority-based Enforcement (DPE) for offline phase deduplication, respectively. CTA mechanism allows selective inline deduplication under a dynamically updated threshold. Data skipped through inline phase is considered as candidates for offline-phase, and is addressed in a ranked arrangement under the control of DPE mechanism. Inline deduplication works in concert with offline deduplication to complement the weakness of each other. DIODE breaks the file to chunk with an average size of 4 KB using the Rabin fingerprinting algorithm. For each chunk, its fingerprint is the SHA1. Though DIODE achieves superior I/O performance to other conventional schemes, the deduplication efficiency of DIODE is inferior to that of Full-inline and Full-offline systems.

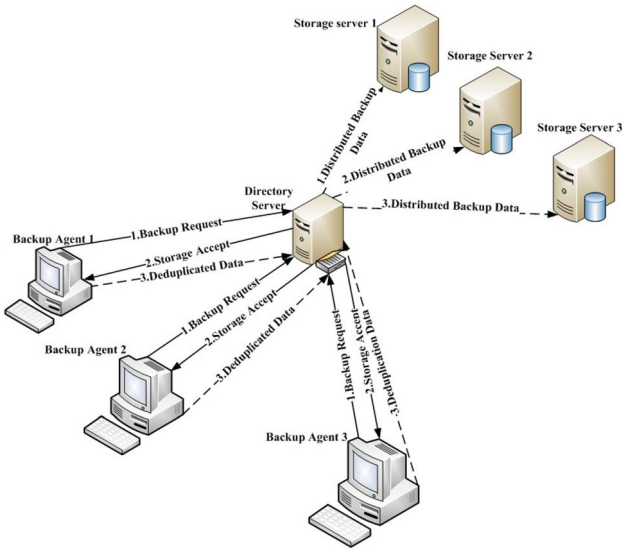

Fig. 4 General Design of Clustered Deduplication Storage System

3.3 Clustered storage deduplication systems

They are the type of deduplication storage systems that save their distinct data by distributing them among multiple storage nodes to avoid data loss in the event of a server crash [74]. The primary purpose of these deduplication storage systems is to achieve that goal with the lowest possible costs. The crucial challenges face such types of systems are high system performance, distributing the load evenly, and keeping reasonable system consistency. To address previous difficulties a lot of techniques were implemented such as distributed load and data replication algorithms. The general design of a clustered deduplication storage system is illustrated in Fig. 4.

To boost deduplication efficiency Fu and co-authors proposed Σ -Dedupe [75] system. This approach was designed to hit a reasonable tradeoff between scalable deduplication productivity and high duplicate efficiency in cluster systems. Σ-Dedupe adopts the Two-Threshold Two-Divisor (TTTD) chunking algorithm [76]. It exploits data similarity and locality by implementing a fingerprinting method at the super-chunk level. Σ-Dedupe creates a similarity index above the conventional container-based locality maintains caching way to mitigate the chunk index searching bottleneck for the deduplication procedure in every deduplication server device. It achieved high parallel deduplication productivity and efficiency. However, it stores some duplicate data, which has some effect in reducing its effectiveness in some way.

In their quest to address the problems of the Σ-Dedupe system and to increase the deduplication effectiveness, Xingyu Zhang and Jian Zhang developed.

Similarity-locality approach [77]. This system breaks data flow into many sections(2 MB) depending on file size, where file bigger than 2 MB is broken into many little parts. It merges the similarity with the locality of the deduplication group. It composes three elements client, a metadata server, and a deduplication server. The locality of the data flow in conjunction with its similarity was used to mitigate the disk bottleneck. Where the former is reserved by collecting adjacent segments in (256 MB) blocks and the latter achieved through bloom filters. In this deduplication scheme, each node has local signature information and a signature summary of all other nodes. When one data segment comes, the node checks its local segment signature information primarily. The node compares the similarity between bloom filters of segments. If bloom filters of segments have more common 1, the two segments have more similar. The node accesses the local block of this segment. If bloom filters of segments have not more common 1, the node checks signature summary quickly. The deduplication efficiency average of this system overcomes Σ-Dedupe scheme. However, the high deduplication throughput due to further fingerprint information and assigning additional resources, the productivity of the system is influenced.

Seeking a scalable rate by many data servers to eliminate redundancy at the same time, with a negligible degrading of deduplication efficiency, Boafft [78] was proposed. This system consists of clients, a metadata server, and data servers. The Boafft uses a data routing mechanism depending on data similarity that decreases the network overhead by rapidly detecting the storage position. It preserves an in-memory similarity indexing in every data server in addition to storage deviation, which in turn increases the memory consumption. The Boafft adopts MinHash function [79] for assigning super-blocks fingerprint. It compares the super-block with the subset of similarity index for data deduplication, and this degrades deduplication efficiency. Due to the global hash checking operation, the system performance decreases sharply especially with the growth of data size.

To attain low storage overhead, whereas maintaining acceptable deduplication throughput and resource consumption, DEDIS was proposed [80]. This system is a reliable and completely decentralized system that implements cluster-wide offline deduplication of virtual machines primary volumes. DEDIS composes of three modules: Distributed Duplicates Index (DDI), Duplicate Finder (D. Finder) and Garbage Collector (GC). DEDIS design is entirely decentralized, avoiding any single point of failure or contention, thus, safely scaling out. This system can run simultaneously in a fully decentralized and scalable system while keeping low latency and throughput overhead, less than 14, and a baseline single-server deduplication throughput with low-end hardware. As for the offline deduplication approach, this scheme needs some extra space to store data before deduplication operation, as well as low deduplication efficiency.

In a search for a solution to address the apparent shortcomings of the system mentioned above, RMD [81] was proposed. This system leverages a bloom filter array and a data resemblance algorithm to boost system throughput dramatically. RMD structure consists of five main components: Dynamic Bloom Filter Array (DBA), Bin Address Tables (BATable), Fingerprint Bin Buffer (FpBinBuffer), RAM Hit Table and on-disk Fingerprint Bins (FpBin). The DBA and Bin Address Table (BATable) work

with each other to catch the identical fingerprint bin to the given reference one. Each fingerprint bin has fingerprints from similar segments sharing the same reference fingerprint, and the FpBinBuffer is used to buffer fingerprint bins. RMD achieves good fingerprint index performance (1.2 MF) with minimal RAM consumption because of rapidly detecting the chunk comparison candidates. Moreover, RMD outperforms the DEDIS deduplication approach regarding deduplication efficiency and throughput. This scheme is near exact deduplication system, so it stores some redundant data.

As part of the relentless search for a practical compromise between the differing objectives of replicable repeatability productivity and high system efficiency, AppDedup [82] was proposed. It is an application-aware scalable inline distributed deduplication framework in a cloud environment. AppDedup adopts a two-tiered data routing scheme the file-level application-aware routing decision in director and the super-chunk level similarity aware data routing in clients. It consists of three main components: clients, dedupe storage nodes, and a director. AppDedup executes data chunking with fixed or variable chunk size and super-chunk grouping in the data dividing module for each data flow and computes chunk fingerprints by a collision-resistant hash function, like MD5, SHA-1 or SHA-2 [83]. AppDedup optimizes distributed deduplication by taking advantage of application awareness, data similarity, and locality in streams. AppDedup slightly improves the deduplication efficiency in terms of memory consumption increase, in addition to low deduplication throughput.

4 Discussion and analysis

In this section, we will discuss and analyze in detail the efficiency, throughput, and memory consumption of 1 TB data set of each of the deduplication systems mentioned above. The ability of any redundancy removal system is the percentage of the amount of redundant data removed by that system [84]. It is calculated from the following equation:

$$\left[(1 - RemovedDataSize/OrignalDataSize) * 100\right] \tag{1}$$

The performance of a redundancy system is the amount of redundant data that a system removes in a given time unit, which measures its speed and we calculate it here in MB/s. The system consumption memory, which is an essential factor for any deduplication system. It significantly affects the effectiveness, performance, and cost of the system. To calculate the amount of memory consumed by any method, we need to know the size of the blocks that divide the data and the fingerprint of each one [85]. We can calculate memory consumption from the following equation:

$$[(DataSetSize/BlockSize) * FingerprintSize] \tag{2}$$

4.1 Analysis of local storage deduplication systems

Figure 5 illustrates that P-Dedupe is the highest local deduplication storage system in terms of efficiency followed by the H.P.Dedup system and finally REBL

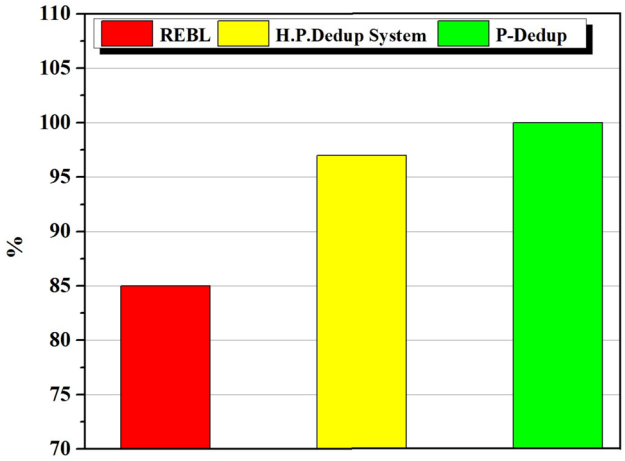

Fig. 5 Efficiency of Local Storage Deduplication Systems

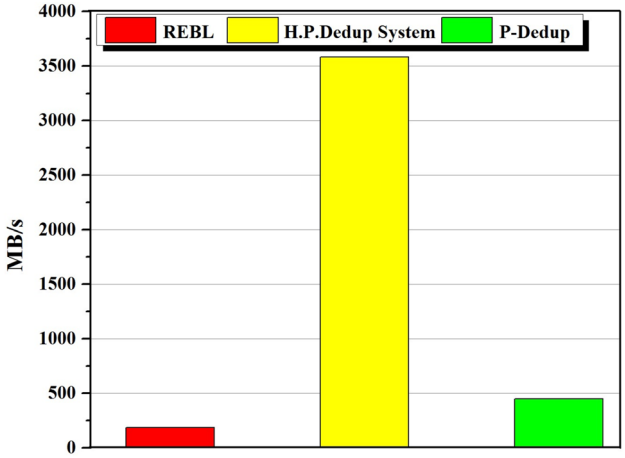

Fig. 6 Throughput of Local Storage Deduplication Systems

with values around 100%, 97%, and 85%, respectively. The adopted techniques by P-Dedupe such as the CDC chucking algorithm, similarity, and locality caches were the major reasons for this achievement. Despite the use of the CDC algorithm by the REBL system, its deduplication efficiency was less than H.P.Dedup due to super fingerprints implementation by the former.

H.P.Dedupe is the highest local storage system in terms of deduplication throughput followed by the P-Dedupe system and finally, REBL with values around 3584 MB/s, 450 MB/s, and 182.72 MB/s, respectively as it is illustrated in Fig. 6, H.P.Dedupe throughput overcame P-Dedupe and REBL due to the implementation of the FSC algorithm in addition to small group files in 2 MB segments. P-Dedupe

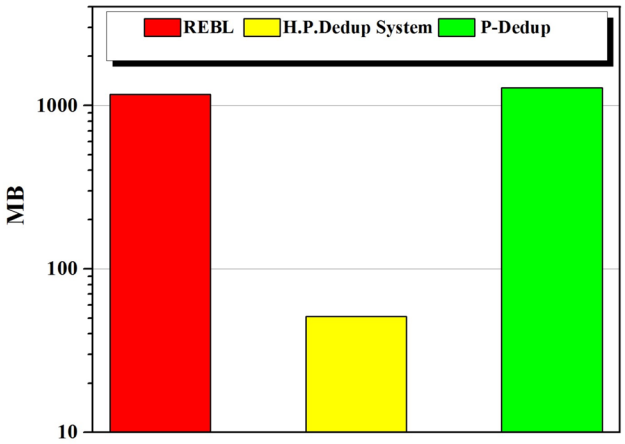

Fig. 7 Memory Consumption of Local Storage Deduplication Systems

throughput outperformed REBL because of parallel chunking (CDC) and fingerprinting algorithms in addition to in-memory similarity hash table and locality cache.

Contrarily, H.P.Dedupe is the lowest local storage system in terms of memory consumption followed by REBL, and finally, PDedupe with values around 51.2 MB, 1170.28 MB, and 1282 MB, respectively as it is illustrated in Fig. 7. The Progressive Sampled Indexing technique adopted by H.P.Dedupe was the primary reason for this achievement, which led to reduce the memory consumption of 1 TB data set to only 51.2 MB. While super fingerprints technique made REBL overcome P-Dedupe in terms of memory consumption at the account of deduplication efficiency as shown previously in Fig. 5.

From the above, we can notice that balancing deduplication efficiency with overheads is the major dilemma of local storage deduplication systems as summarized in Table 1. They struggled to achieve the perfect compromise between them. From inserting variable-sized chunks in a fixed-sized super piece as in REBL to the executions of parallel parts synchronous and the implementations of parallel sections asynchronous as in P-Dedup. However, the ideal solution still missing.

4.2 Analysis of centralized storage deduplication systems

Figure 8 shows that NIDF is the highest centralized storage system in terms of deduplication efficiency with a value of 100%. It adopts genetic programming concepts without any negative or positive false, which enabled it to attain this great feature among all previously mentioned systems. NIDF is followed by RevDedupe, which achieved 96% due to two-phase deduplication technology. MMSD overcame remained other systems with an efficiency of 91% due to its deep mining of application awareness and file metadata information. DIODE tried to balance deduplication throughput and efficiency by coordinating dynamically between online and post-deduplication. It classifies files to several categories and selectively removes

Table 1 Local Storage Deduplication Systems

Name	Objectives	Techniques	Features	Drawbacks
REBL	Reducing storage overheads	Super-FingerprintsCDC chunkingAlgorithm and Rabin fingerprint	Reasonable deduplication efficiency 85%. Reasonable memory consumption 1170.28 MB	Bounded by the size of a super fingerprint. Low system throughput 182.72 MB/s
H.P.Dedupe System	Optimizing single-node performance. Scale to Large capacity. Good deduplication efficiency	Progressive Sampled indexing. FSC 4 KB. Segment grouping (2 MB). SHA-1 hash function	High deduplication efficiency of 97%. High system throughput 3584 MB/s. Low memory consumption 51.2 MB	Boundary shifting problem
P-Dedupe	Achieving high deduplication throughput	Parallel chunking(CDC) and fingerprinting algorithms (SHA-1). In memory similarity hash table and locality cache	High deduplication efficiency of 100%. Reasonable system throughput 450 MB/s. Reasonable memory consumption 1282 MB	Disordering the chunks inside data sections

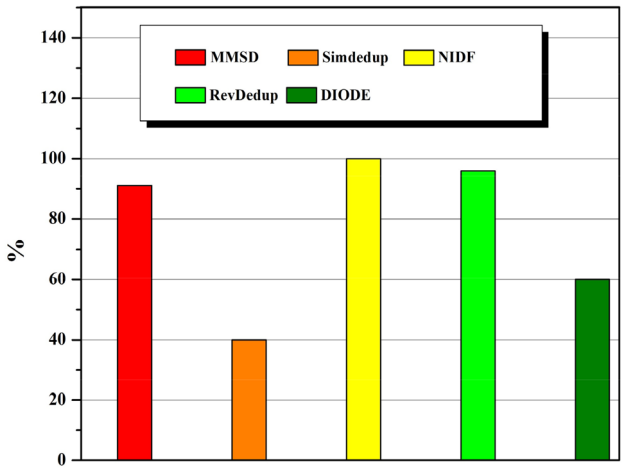

Fig. 8 Efficiency of Centralized Storage Deduplication Systems

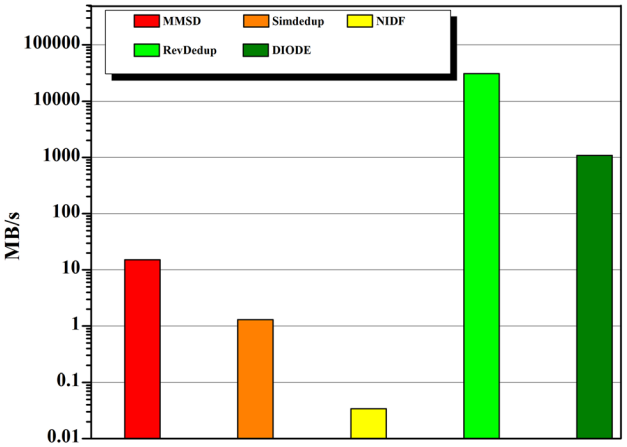

Fig. 9 Throughput of Centralized Storage Deduplication Systems

duplicates among some of them and postpone others to the post-deduplication stage. DIODE determines a specific threshold before launching the offline removing phase. This previously mentioned strategy led to a decrease in deduplication efficiency to 60% beneath the full online and offline systems. Finally, SimDedupe removes redundant data among similar chunks by the simhash algorithm. That is why its deduplication efficiency is too low at 40%.

Figure 9 displays that RevDedup and DIODE are the highest centralized deduplication storage systems regarding productivity up to 30,720 MB/s and 1083 MB/s, respectively. They achieved this feature by adopting two phases of deduplication operation entirely in the high- performance server as it is explained previously in

Sect. 3. RevDedup outperformed DIODE because it is a static post-deduplication system that does not change with the load variation, while DIODE is a dynamic system that varies with the load, which takes longer in the process of removing the duplicate data.

The techniques applied by the MMSD system, which were designed to increase the effectiveness of this scheme. This method led to a decrease in the system productivity significantly from the approaches mentioned earlier. MMSD system implemented the elimination of redundancy in the client first. Taking advantage of the distinctive features of any file such as type, size, timestamp, and modification frequency. MMSD removes redundant data locally via the CDC algorithm for small documents and through the WFC algorithm for large ones, which led to a reduction in productivity to 15 MB/s.

The significant decline in the productivity of the last two systems (SimDedup, NIDF) was due to the increase in the efficiency of the NIDF system and the increment of deduplication overheads in terms of system latency by the SimDedup system. In addition to that, SimDedup implemented the CDC algorithm and the procedure of removing the repetition in two stages detection of similar chunks first and then eliminate the redundancy among them, resulting in lower productivity of the system to 1.3 MB/s. The adopting of the NIDF to the concepts of genetic programming and especially the algorithm of Levenshteins Edit Distance, which consumes considerable time led to the significant reduction of this system productivity to 0.034 MB/s.

Figure 10 illustrates that the least centralized memory consumption system of 1 TB data size is Simdedup, which consumes about 0.032 MB for a quantity of data of 1 TB. This small amount of memory due to a large segment size of 250 MB with very low simhash fingerprint 64 bit. The amount of memory consumption (5 MB) by RevDedup refers to its use of 4 MB chunk size and hash function SHA-1 20 byte. It is also noted that MMSD consumes a small amount of memory which

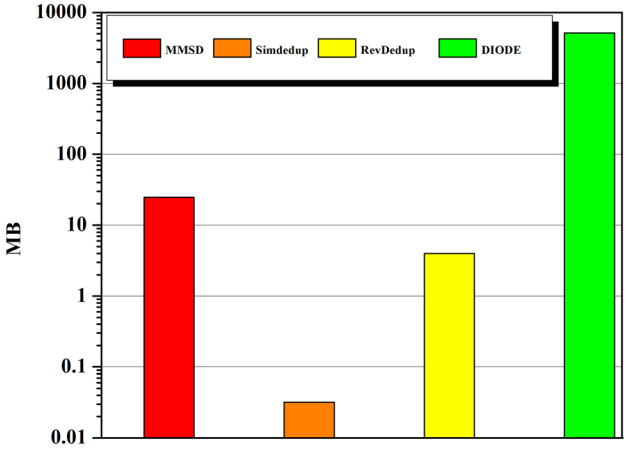

Fig. 10 Memory Consumption of Centralized Storage Deduplication Systems

does not exceed 24.7 MB. The low MMSD consumption is due to its combination of global file-level redundancy removal algorithms WFC for files larger than 1 MB and locally CDC with 8-KB chunk size. With 16 bytes MD5 fingerprint for the first algorithm and 20 bytes SHA-1 for the latter one.

The highest memory consumption by DIODE compared to its previously mentioned counterparts (Fig. 10), is due to the techniques used by this system. Where DIODE, consumes 5120 MB for the same size of the previous data set due to its 20-byte fingerprint and the average chunk size 4 KB. It can be noticed that NIDF was excluded from Fig. 10 because it is not a hashing deduplication system, so it does not have a chunk size.

Based on the diverse approaches of centralized deduplication storage systems as it is summarized in Table 2. It is noticeable that all these systems struggle to attain high deduplication efficiency with minimum overheads. MMSD adopts the CDC algorithm locally and WFC one globally to accelerate system throughput. RevDedup collects chunks into large segments and conducts deduplication among new pieces first then offline among old ones. MMSD failed to obtain its goal, whereas its throughput was only 15.11 MB/s. RevDedup increases network bandwidth to achieve high deduplication efficiency. DIODE as the state of the art of centralized deduplication storage systems complements the weakness of online and offline deduplication systems by working concretely with each other. However, this comes at the account of deduplication efficiency.

4.3 Analysis of clustered storage deduplication systems

As shown in Fig. 11, AppDedup, S.L.Dedupe, RMD, and Σ-Dedupe are the most efficient in removing the repetition, 95%,94.5%,94.3%, and 92.5% respectively. It is noted that the efficiency of these systems above 90%, which is very high value. What the scheme comes at the forefront is AppDedup that divides the data into chunks, collects them into groups and calculates super-chunk then distributes them to the suitable device depending on the type of file and the similarity of the fingerprint to obtain the highest possible effectiveness.

Thus, we see that the AppDedup system eliminates repetition individually depending on the type and similarity of the data, making it outperform another scheme regarding efficiency. The efficiency of S.L. Dedupe is less than the first system, although it benefits from the similarity and locality of the data but differs as it does so by relying on the similarity in the bloom filter vector. Thus reduces the effectiveness of determining the similarity of the data due to the false positive probability. RMD uses the similarity algorithm only to combine similar fingerprints into one unit and uses the data space to accelerate its performance, making its average efficiency slightly lower than previous systems. Although Σ-Dedupe used the minimum sampling algorithm to obtain the reference footprint as RMD, the latter had collected the high-frequency reference fingerprints in one unit, which increased its efficiency and speed while the former used them individually.

The significant reduction in Boafft efficiency from previous systems, although it also used the similarity algorithm, is due to the aggregation of the reference

Table 2 Centralized Storage Deduplication Systems

Name	Objectives	Techniques	Features	Drawbacks
MMSD	Achieving shorter backup windows, high deduplication throughput, and low system overheads	Files < 1 MB WFC policy. Globally WFC is Symbolized with 16 B. MD5, CDC, and SHA-1	High deduplication efficiency of 91%. Shorten the backup window by 54.2%. Low memory consumption 24.70 MB	Low deduplication throughput 15.11 MB/s
Simdedup	Achieving high deduplication throughput and low system overheads	File similarity and chunk locality. In-memory cache. SHA-1 hash function. CDC algorithm (4 KB, 250 MB segment)	Very low memory consumption 0.032 MB MB	Very low deduplication efficiency of 40%. Very low deduplication throughput of 1.3 MB/s
NIDF	Providing a reliable system for identifying duplicates. Introducing GP deduplication system	A Sequence Matchingand Levenshtein Algorithms. Genetic Programming concepts	No false positives and false negatives. Very high deduplication efficiency of 100%	Very low deduplication throughput 0.034 MB/s
RevDedup	High backup and restore Throughput. Low deletion overhead for expired backups	CDC algorithm. Checking only non-shared segments for chunk removal	Very high deduplication throughput 30720 MB/s. High deduplication efficiency of 96%. Very low memory consumption 5 MB	Extra I/Os of identifying and removing duplicate chunks on disk
DIODE	Providing salient read/write performance and space-saving simultaneously	SHA-1 hash function. CDC algorithm (4 KB)	High deduplication throughput 1083 MB/s	Low deduplication efficiency of 60%. High memory consumption 5120 MB

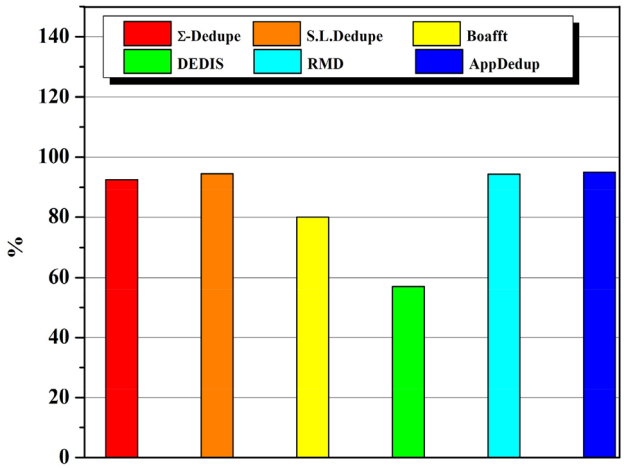

Fig. 11 The efficiency of Clustered Storage Deduplication Systems

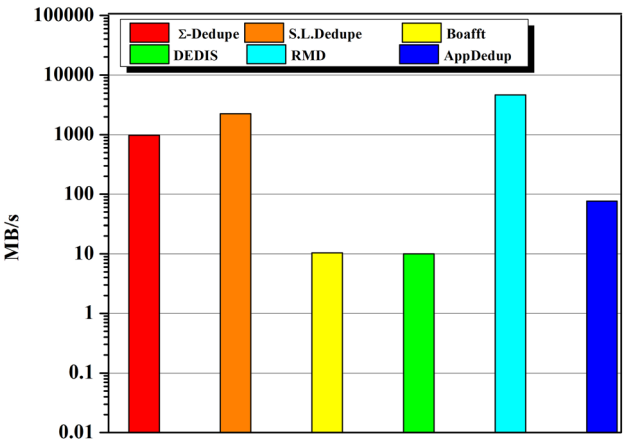

Fig. 12 Throughput of Clustered Storage Deduplication Systems

fingerprint in the form of a large segment average size is 512 KB on which to select the server for where the redundancy will be performed locally there. The efficiency of DIDES is 57% because it is a selective offline deduplication system. whereas DIDES removes the repetition of non-common chunks only, which led to the reduction of its efficiency.

The RMD and S.L.Dedupe are found to be the highest clustered deduplication systems in terms of throughput as shown in Fig. 12 With an average performance of 4687.5 MB/s and 2251 MB/s, respectively.

The reason why RMD is superior to other systems is its ability to collect similar segments and mix the most frequent in one unit, which significantly limits the scope of the search in addition to use the Dynamic bloom filter matrix, which substantially

increased the speed of the system. The following S.L.dedupe, with lower performance than RMD, although it is still high due to its reliance on data similarity and locality to increase its speed, while the role of its bloom filter was only to increase its effectiveness as mentioned earlier.

The use of Σ-Dedupe for the data similarity and its capacity at the level of superchunk to be directed to the server, which will achieve the highest amount of redundancy elimination, has reduced its performance more than previous systems because of the time consumed but still maintained within the acceptable range of 980 MB/s. The previously explained deduplication procedure of AppDedup, which depends on distributing the load according to file type and fingerprint similarity after chunking and grouping it in super fingerprints has a significant impact on performance reduction to 76.41 MB/s.

Finally, the performance of Boafft and DEDIS systems is very low and very close although the former is 10.41 MB / s slightly higher than the later 10 MB / s, because it comprehensively eliminates redundancy based on data similarity only, while the other is not connected to the duplicate data cancellation system, This makes its value a little less than the first

Figure 13 illustrates that the least clustered memory consumption system of 1 TB data size is S.L.Dedupe with memory consumption of 0.078125 MB due to its choice of one SHA-1 fingerprint for 256 MB data block to increase system speed. RMD came after that with memory consumption of no more than 5 MB for a category of same data set size.

As a result of using units containing 512 KB segments with an average size of 4 KB and a footprint of 10B, Σ-Dedupe consumes 20 MB of memory because it combines the data sections in super-block with a capacity of 16 MB each represented by 40 bytes for every eight fingerprints. The Boafft and AppDedup consumed the same amount of memory (40 MB) as shown in Fig. 13. Where Boafft divides the data into a superblock at a rate of 512 KB and each selects the SHA-1 reference footprint, and AppDedup collects the data in a super chunk with a size of 1 MB

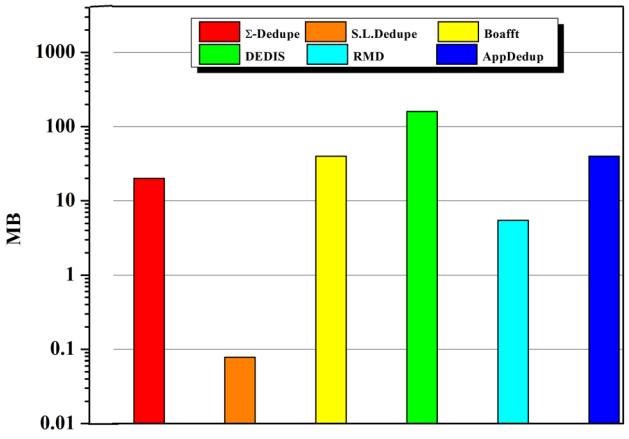

Fig. 13 Memory Consumption of Clustered Storage Deduplication Systems

Table 3 Clustered Storage Deduplication Systems

Name	Objectives	Techniques	Features	Drawbacks
Σ-Dedupe	Achieving scalable deduplication throughput and efficiency tradeoff	Data similarity and locality. Similarity index. TTTD chunking algorithm. MD5 or SHA-1 hash function. CDC or FSC algorithm	High deduplication efficiency of 92.5%. High deduplication throughput of 980 MB/s. Low memory consumption 20 MB	Storing some redundant data
S.L.Dedupe	Achieving high deduplication throughput and efficiency	Similarity and locality deduplication cluster. Bloom filter. MD5 or SHA-1 hash function	High deduplication efficiency of 94.5%. High deduplication throughput 2251 MB/s. Very low memory consumption 0.0781 MB	Exciting the false positive probability
Boafft	Achieving scalable throughput	Similarity routing algorithm. The similarity index in each data server. Frequency Fingerprint cache. CDC algorithm(4 KG). MinHash function	Reasonable deduplication efficiency of 80%. Acceptable memory consumption of 40 MB	Low deduplication throughput 10.41 MB/s
DEDIS	Achieving low I/O storage overhead and acceptable deduplication throughput	FSC algorithm (4 KG). In-memory cache. SHA-1 hash function	Reasonable memory consumption of 160 MB	Low deduplication efficiency of 57%. Very Low deduplication throughput 10 MB/s
RMD	Achieving high deduplication throughput	CDC algorithm. SHA-1 hash function. Dynamic Bloom Filter Array (DBA). Bin Address Tables (BATable). Fingerprint Bin Buffer (FpBinBuffer). RAM hittable. On-disk Fingerprint Bins (FpBin)	High deduplication throughput 4687.5 MB/s. Low memory consumption 5.5 MB. High deduplication efficiency of 94.3%	Storing some redundant data
AppDedup	Achieving tradeoff between scalable deduplication throughput and efficiency	Application awareness, data similarity, and locality. Global application route table. TTTD chunking algorithm(8 KB). MD5 hash function	High deduplication efficiency of 95%. Low memory consumption 40 MB	Low deduplication throughput of 76.41 MB/s

electing a 40-byte representative fingerprint. At last, DEDIS also distributes data almost evenly between its storage servers, so that the memory consumed between 32 servers for 1 TB data set after being divided into blocks of 4 KB and 20-byte fingerprints is 160 MB per server.

Clustered storage deduplication schemes (as it is briefed in Table 3) distributing the load evenly among its several storage devices play a crucial rule in designing such a type of deduplication storage scheme. In addition to achieving single node performance with the growth of the system. In the way of speeding up system performance, these systems adopt many techniques like WFC, FSC algorithms, bloom filters, and LRU cache, which impact their deduplication efficiency. They alleviated the influence of this issue by exploiting a combination of CDC in fixed super-chunk technique as in scalable data routing storage scheme. However, this method did not overcome the efficiency dilemma completely. The Boafft addressed this issue differently by adopting the MinHash similarity function. Although this solution still had a significant impact on system efficiency, AppDedup as state of the art in clustered deduplication handles this shortage by distributing files according to their types to the same node exploiting locality and similarity of the streams. This approach neglects the common data among diverse data types, which affects its deduplication efficiency too.

4.4 Challenges of storage deduplication systems

From all what is mentioned above, it could be easily noticed that there are various challenges facing deduplication systems:

Metadata management: The problem of the enormous amount of data management, where local deduplication systems struggle to reach high system efficiency with fewer overheads as possible. The limited resources impede the local deduplication storage systems from achieving their goals despite the implementation of many techniques to do so as explained earlier.

The balance of efficiency and overhead: The primary problem in deduplication systems is the compromise between the highest possible deduplication efficiency and the overhead on local and centralized deduplication storage systems. For instance, lesser chunk lengths raise the space-saving profits of deduplication but result in more significant index structures that are extra expensive to preserve. Preferably, the index would be loaded entirely into RAM, but for big storage and a reasonably little chunk length, the index is excessively huge and has to be saved partly on the drive. The matter raises the quantity of disk I/O processes required by deduplication, which may initially impair the system I/O performance as was the case with P-Dedupe and DIODE despite both of which are being cutting-edge in their category.

Scalability: The most efficient can be achieved when any chunk can, in standard, be evaluated with any other piece and be eliminated if found identical. However, such complete matching is more laborious as the quantity of data and components in a large-scale storage scheme rise. Mainly, a central index answer is possible to turn out to be itself extremely big, and its treatment a bottleneck on deduplication

productivity [86]. Fractional indexes can be equivalent merely to a partition of redundant enhanced scalability but did just fracstional deduplication. Nevertheless, the number of chunks that cannot be identical can be decreased by discovering data locality and by collecting pieces jointly with a more prominent resemblance [87] as described earlier in systems RMD and AppDedup.

5 Conclusion

Data deduplication is one of the well-known techniques, which eliminates redundant data, decreases the bandwidth, and also minimizes the disk usage and cost. Diverse research papers have been deliberated from the literature; as a result, this paper had reviewed the ideas, categories and the different storage approaches that use data deduplication Regardless of the common classification that uses Granularity, Side, Timing, and Implementation to categorize duplicate data cancellation methods, a new classification principle has been adopted using the storage location. This classification divides deduplication approaches to local, centralized, and clustered storage systems.

Over and above, deduplication systems were comprehensively explained according to a storage location. The objectives, techniques used, advantages, and disadvantages of the most advanced methods of each type were widely dealt with. The efficiency, throughput, and memory consumption of each studied system under the novel classification criterion were analyzed and clarified.

Finally, main deduplication challenges (metadata management, efficiency balance with overhead, and scalability) were recognized and discussed, leading to conclude that there are still a lot of open issues that should be solved and handled in the deduplication storage world.

Acknowledgments This research was supported by the Nanjing Municipal Government-Nanjing University of Science and Technology Joint Scholarship for International Student. We thank our colleagues from the School of Computer Science and Engineering who provided insight and expertise that greatly assisted the research, although they may not agree with all of the conclusions of this paper.

References

1. Al-Fares, M., Loukissas, A., Vahdat, A.: ACM SIGCOMM computer communication review, pp. 63–74. ACM, New York (2008)
2. Greenberg, A., Hamilton, J.R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D.A., Patel, P., Sengupta, S.: ACM SIGCOMM computer communication review, pp. 51–62. ACM, New York (2009)
3. Kandula, S., Sengupta, S., Greenberg, A., Patel, P., Chaiken, R.: Internet measurement, pp. 202–208. ACM, New York (2009)
4. Mell, P., Grance, T., et al.: The NIST definition of cloud computing, Computer Security Division Information Technology Laboratory. National Institute of Standards and Technology, Gaithersburg (2011)
5. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Commun. ACM **53**, 50–58 (2010)
6. Antonopoulos, N., Gillam, L.: Cloud computing. Springer, London (2010)

7. Kilov, H., Linington, P.F., Romero, J.R., Tanaka, A., Vallecillo, A.: The reference model of open distributed processing: Foundations, experience, and applications. Comput. Stand. Interfaces **35**, 247–256 (2013)
8. Rumelhart, D.E., Hinton, G.E., McClelland, J.L., et al.: A general framework for parallel distributed processing. Parallel distributed processing: explorations in the microstructure of cognition, vol. 1, pp. 45–76. MIT Press, Cambridge (1986)
9. McClelland, J.L., Rumelhart, D.E.: Explorations in parallel distributed processing: a handbook of models, programs, and exercises. MIT Press, Cambridge (1989)
10. Kumar, R., Marinov, D., Padua, D., Parthasarathy, M., Patel, S., Roth, D., Snir, M., Torrellas, J.: Parallel Computing Research at Illinois The UPCRC Agenda. University of Illinois, Champaign (2008)
11. Berman, F., Fox, G., Hey, A.J.: Grid computing: making the global infrastructure a reality, vol. 2. Wiley, Hoboken (2003)
12. Bote-Lorenzo, M.L., Dimitriadis, Y.A., GmezSnchez, E.: Grid computing, p. 291298. Springer, Berlin (2003)
13. G. Mittal, D. Kesswani, K. Goswami, et al," A survey of current trends in distributed, grid and cloud computing", arXiv preprint arXiv:1308.1806,(2003).
14. Barroso, L.A., Dean, J., Holzle, U.: Web search fora planet: the Google cluster architecture. IEEE Micro **23**, 2228 (2003)
15. Linden, G., Smith, B., York, J.: Amazon. com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. **7**, 7680 (2003)
16. Tsai, W.-T., Sun, X., Balasooriya, J.: Service-oriented cloud computing architecture. Information technology: new generations (ITNG) Seventh International Conference. IEEE, Piscataway (2010)
17. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. **30**, 107–117 (1998)
18. Chu, H., Rosenthal, M.: Search engines for the worldwide web: A comparative study and evaluation methodology. Proc. Ann. Meeting-Am. Soc. Inform. Sci. **33**, 127135 (1996)
19. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM **51**, 107113 (2008)
20. Shim, K.: Mapreduce algorithms for big data analysis. Proc. VLDB Endow. **5**, 2016–2017 (2012)
21. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: a distributed storage system for structured data. ACM Trans. Comput. Syst. (TOCS) **26**, 4 (2008)
22. Geer, D.: Reducing the storage burden via data deduplication. Comput. IEEE **41**(12), 15–17 (2008)
23. Paulo, J., Pereira, J.: A survey and classification of storage deduplication systems. ACM Comput. Surveys (CSUR) **47**, 11 (2014)
24. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: a survey. IEEE Trans. Knowl. Data Eng. **19**, 116 (2007)
25. Burrows, J.H.: Secure hash standard. Department of Commerce Washington DC Tech. Rep, Washington DC (1995)
26. Kim, D., Song, S., Choi, B.-Y.: SAFE: structure-aware file and email deduplication for cloud-based storage systems, pp. 130–137. Piscataway, IEEE (2013)
27. Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. IEEE Trans. Inf. Theory **23**(3), 337–343 (1977)
28. Ziv, J., Lempel, A.: Compression of individual sequences via variable-rate coding. IEEE Trans. Inform. Theory **24**(5), 530–536 (1978)
29. Biggar, H.: Experiencing data deduplication: improvingefficiency and reducing capacity requirements, White paper. The Enterprise Strategy Group, Milford (2007)
30. Manager, N., et al.: Demystifying data deduplication. Proceedings of the ACM/IFIP/USENIX Middleware'08 Conference Companion, pp. 12–17. ACM, New York (2008)
31. ONeill, M., et al.: Low-cost Sha-1 hash function architecture for Rfid tags. RFIDSec. **8**, 4151 (2008)
32. Michail, H., Kakarountas, A.P., Koufopavlou, O., Goutis, C.E.: A low-power and high-throughput implementation of the Sha-1 hash function. International Symposium on Circuits and Systems ISCAS, p. 40864089. IEEE, Piscataway (2005)
33. Deepakumara, J., Heys, H.M., Venkatesan, R.: FPGA implementation of md5 hash algorithm. IEEE Electr. Comput. Eng. Can. Conf. **2**, 919924 (2001)
34. Rabin, M.O., et al.: Fingerprinting by random polynomials. Center for Research in Computing Technology, Aiken Computation Laboratory, University, Cambridge (1981)

35. Whiting, D. L., Dilatush, T.: System for backing up files from disk volumes on multiple nodes of a computer network. Google Patents 5778395, US,(1998)
36. Bigelow, S., Crocetti, P.: Compression, deduplication and encryption: what's the difference?. TechTarget, Newton (2018)
37. Constantinescu, C., Glider, J., Chambliss, D.: Mixing deduplication and compression on active data sets. 2011 Data Compression Conference, p. 393402. IEEE, Piscataway (2011)
38. Rehomed, A.: Data security and reliability in cloud backup systems with deduplication. Ph.D. dissertation. Chinese University of Hong Kong, Hong Kong (2012)
39. Dong, W., Douglis, F., Li, K., Patterson, R.H., Reddy, S., Shilane, P.: Tradeoffs in scalable data routing for deduplication clusters. FAST **11**, 1529 (2011)
40. Coates, J. L., Bozeman, P. E., Patterson, D. A.: Distributed storage cluster architecture. Google Patent 7590747, US,(2009)
41. Manogar, E., Abirami, S.: A study on data deduplication techniques for optimized storage. Advanced Computing (ICoAC) 2014 Sixth International Conference, p. 161166. IEEE, Piscataway (2014)
42. Zhu, B., Li, K., Patterson, R.H.: Avoiding the disk bottleneck in the data domain deduplication file system. Fast **8**, 114 (2008)
43. Heckel, P.C.: Minimizing remote storage usage and synchronization time using deduplication and multi-chunking: Syncany as an example. Technical Report TR-CS-96-05. Universitat Mannheim, School of Business Informatics and Mathematics Laboratory for Dependable Distributed Systems University, Mannheim (2012)
44. He, Q., Li, Z., Zhang, X.: "Data deduplication techniques", in Future Information Technology and Management Engineering (FITME). Int. Conf. IEEE **1**, 430433 (2010)
45. Vikraman, R., Abirami, S.: A study on various data deduplication systems. Int. J. Comput. Appl. **94**(4), 35–40 (2014)
46. Vanish, A., Sankar, K.S.: Study of chunking algorithm in data deduplication. Proceedings of the International Conference on Soft Computing Systems, p. 1320. Springer, New Delhi. (2016)
47. Xia, W., Zhou, Y., Jiang, H., Feng, D., Hua, Y., Hu, Y., Liu, Q., Zhang, Y.: Fastcdc: a fast and efficient content defined chunking approach for data deduplication. USENIX Annual Technical Conference, p. 101114. USENIX, Berkeley (2016)
48. Malhotra, J., Bakal, J.: A survey and comparative study of data deduplication techniques. Pervasive Computing (ICPC), International Conference, p. 15. IEEE, Piscataway (2015)
49. Tang, Y., Yin, J., Dengand, S., Li, Y.: Diode: Dynamicinline-offline deduplication providing efficient space-saving and read/write performance for primary storage systems. Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), p. 481486. IEEE, Piscataway (2016)
50. Tan, Y., et al.: SAM: A semantic-aware multitiered source de-duplication framework for cloud backup. 2010 39th International Conference on Parallel Processing, pp. 614–623. IEEE, Piscataway (2010)
51. Fu, Y., et al.: AA-dedupe: an application-aware source deduplication approach for cloud backup services in the personal computing environment. 2011 IEEE International Conference on Cluster Computing, p. 112120. IEEE, Piscataway (2011)
52. Naga Malleswari T. Y., D. Malathi, and G. Vadivu," Deduplication techniques: A technical survey." International J Innovative Res Sci Technol 1.7 (2014).
53. Montana, D.J.: Strongly typed genetic programming. Evolut. Comput. **3**(2), 199230 (1995)
54. Kaurav, N.: An investigation on data de-duplication methods and its recent advancements. Proceedings of the International Conference on Advances In Engineering And Technology-ICAET. Engineering And Technology ICAET, Talegaon Dabhade (2014)
55. Talk, G., Keswani, V., Parab, N., Mace, J.: Disaster recovery using local and cloud spanning deduplicated storage system. Google Patent App.12/942,988, US(2011)
56. Kulkarni, P., Douglis, F., LaVoie, J.D., Tracey, J.M.: Redundancy elimination within large collections of files. USENIX Annual Technical Conference, General Track, p. 5972. USENIX, Berkeley (2004)
57. Guo, F., Efstathopoulos, P.: Building a high-performance deduplication system in USENIX annual technical conference. USENIX annual technical conference. USENIX, Berkeley (2011)
58. Broder, A., Mitzenmacher, M., Mitzenmacher, A.B.I.M.: Network applications of bloom filters: A survey. Internet Mathematics. Citeseer, Princeton (2002)

59. Efstathopoulos, P., Guo, F., Shah, D.: Progressive sampling for deduplication indexing. Google Patent8311964, US,(2012)
60. Meister, D., Kaiser, J., Brinkmann, A., Cortes, T., Kuhn, M., Kunkel, J.: A study on data deduplication. Proceedings of the International Conference on High-Performance Computing, Networking, Storage and Analysis, p. 7. IEEE Computer Society Press, Washington, DC (2012)
61. Xia, W., Jiang, H., Feng, D., Tian, L., Fu, M., Wang, Z.: P-dedupe: Exploiting parallelism in data deduplication system. Networking, Architecture, and Storage (NAS), IEEE 7th International Conference, p. 338347. IEEE, Piscataway (2012)
62. Yan, H., Li, X., Wang, Y., Jia, C.: Centralized duplicate removal video storage system with privacy preservation. Sensors **18**(6), 1814 (2018)
63. Meng, H., Li, J., Liu, W., Zhang, C.: Mmsd: a metadata-aware multi-tiered source deduplication cloud backup system in the personal computing environment. Comput. Softw. **8**, 542 (2013)
64. T. Knutson and R. Carbone, "Filesystem timestamps: What makes them tick?", GIAC GCFA Gold Certification,(2016).
65. Yao, W., Ye, P.: Simdedup: a new deduplication scheme based on simhash. International Conference on Web-Age Information Management, p. 7988. Springer, Berlin (2013)
66. Fu, Z.-J., Shu, J.-G., Wang, J., Liu, Y.-L., Lee, S.-Y.: Privacy-preserving smart similarity search based on simhash over encrypted data in cloud computing. Internet Technol. J. **16**(3), 453460 (2015)
67. Madhubala, G., Priyadharshini, R., Ranjitham, P., Baskaran, S.: Nature-inspired enhanced data deduplication for efficient cloud storage. Recent Trends in Information Technology (ICRTIT), International Conference, p. 16. IEEE, Piscataway (2014)
68. Henson, V.: An analysis of compare-by-hash. HotOS, p. 1318. USENIX, Berkeley (2003)
69. Brown, R. A.: Sequence matching algorithm. 2015, Google Patent 8965935, US(2015)
70. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Phys. Dokl. **10**, 707710 (1966)
71. Mell, P., Grance, T., et al.: The NIST definition of cloud computing. National Institute of Standards and Technology, Gaithersburg (2011)
72. Li, Y.-K., Xu, M., Ng, C.-H., Lee, P.P.: Efficient hybrid inline and out-of-line deduplication for backup storage. ACM Trans. Storage (TOS) **11**(2), 1–21 (2015)
73. Zorn, B.: Comparing mark-and-sweep and stop-and-copy garbage collection. Proceedings of the ACM conference on LISP and functional programming ACM, p. 8798. ACM, New York (1990)
74. Retnamma, M. K. V., Kottomtharayil, R., Attard, D. R.: Distributed deduplicated storage system. Google Patent 9020900, US,(2015)
75. Fu, Y., Jiang, H., Xiao, N.: A scalable inline cluster deduplication framework for big data protection. Proceedings of the 13th international middleware conference, p. 354373. Springer-Verlag Inc, New York (2012)
76. Eshghi, K., Tang, H.K.: A framework for analyzing and improving content-based chunking algorithms. Hewlett-Packard Labs Tech. Rep. TR **30**, 2005 (2005)
77. Zhang, X., Zhang, J.: Data deduplication cluster-based on similarity-locality approach. Green Computing and Communications(GreenCom), IEEE and Internet of Things iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing, p. 21682172. IEEE, Piscataway (2013)
78. Luo, S., Zhang, G., Wu, C., Khan, S., Li, K.: Boafft: Distributed deduplication for big data storage in the cloud. IEEE transactions on cloud computing. IEEE, Piscataway (2015)
79. Sarawagi, S., Kirpal, A.: Efficient set joins on similarity predicates. Proceedings of the ACM SIGMOD International Conference on Management of Data ACM, p. 743754. ACM, New York (2004)
80. Paulo, J., Pereira, J.: Efficient deduplication in a distributed primary storage infrastructure. ACM Trans. Storage (TOS) **12**(4), 20 (2016)
81. Zhang, P., Huang, P., He, X., Wang, H., Zhou, K.: Resemblance and mergence based indexing for high-performance data deduplication. J. Syst. Softw. **128**, 1124 (2017)
82. Fu, Y., Xiao, N., Jiang, H., Hu, G., Chen, W.: Application-aware big data deduplication in cloud environment. IEEE Transactions on Cloud Computing. IEEE, Piscataway (2017)
83. Sklavos, N., Koufopavlou, O.: Implementation of the sha-2 hash family standard using fpgas. J. Supercomput. **31**, 227248 (2005)
84. Clements, A.T., Ahmad, I., Vilayannur, M., Li, J., et al.: Decentralized deduplication in san cluster file systems. USENIX Annual Technical Conference, p. 101114. USENIX, Berkeley (2009)
85. Mike, D.: Understanding data deduplication ratios. SNIA Data Management Forum, p. 7. SNIA, Daryaganj (2008)

86. He, S., Zhang, C., Hao, P.: Comparative study of features for fingerprint indexing. 16th IEEE International Conference on Image Processing ICIP, pp. 2749–2752. IEEE, Piscataway (2009)
87. Douceur, J.R., Adya, A., Bolosky, W.J., Simon, P., Theimer, M.: Reclaiming space from duplicate files in a serverless distributed file system. Distributed Computing Systems,Proceedings. 22nd International Conference, p. 617624. IEEE, Piscataway (2002)