symmetry

MDPI

Article

Decentralized and Privacy Sensitive Data De-Duplication Framework for Convenient Big Data Management in Cloud Backup Systems

J. Gnana Jeslin [1] and P. Mohan Kumar [2,*]

[1] Department of Computer Science and Engineering, RMK College of Engineering and Technology, Chennai 601206, India; gnanajeslincse@rmkcet.ac.in or jgjeslin@gmail.com
[2] Department of Computer Science and Engineering, Sri Krishna College of Engineering and Technology, Coimbatore 641008, India
* Correspondence: mohankumarp@skcet.ac.in or mohankumarmohan@gmail.com

Abstract: The number of customers transferring information to cloud storage has grown significantly, with the rising prevalence of cloud computing. The rapidly rising data volume in the cloud, mostly on one side, is followed by a large replication of data. On the other hand, if there is a single duplicate copy of stored symmetrical information in the de-duplicate cloud backup the manipulation or lack of a single copy may cause untold failure. Thus, the deduplication of files and the auditing of credibility are extremely necessary and how they are achieved safely and effectively must be addressed in academic and commercial contexts urgently. In order to tune in this task by using application recognition, data similitude, and locality to simplify decentralized deduplication with two-tier internode and application deduction, we suggest a flexible direct decentralized symmetry deduplication architecture in a cloud scenario. It first distributes application logic to the contents of the directory through implementation-oriented steering to maintain a deployment location and also attributes the same kind of information to the cloud backup node with the storage node specificity by means of a hand printing-based network model to attain adequate global deduplication performance. We build up a new ownership mechanism during file deduplication to ensure continuity of tagging and symmetrical modeling and verify shared ownership. In addition, we plan an effective ownership policy maintenance plan. In order to introduce a probabilistic key process and reduce key storage capacity, a user-helped key is used for in-user block deduplication. Finally, the protection and efficiency audit demonstrate that the data integrity and accuracy of our system are ensured and symmetrically effective in the management of data ownership.

Keywords: deduplication; cloud data storage; convergent encryption; cryptographic hash; routing

Citation: Gnana Jeslin, J.; Mohan Kumar, P. Decentralized and Privacy Sensitive Data De-Duplication Framework for Convenient Big Data Management in Cloud Backup Systems. *Symmetry* **2022**, *14*, 1392. https://doi.org/10.3390/sym14071392

Academic Editor: Alexander Zaslavski

Received: 19 June 2022
Accepted: 2 July 2022
Published: 6 July 2022

1. Introduction

As the dramatic growth in the volume of electronic data created by 2020 is seen, from 12 to 18 zettabytes, and the projected amount of data provided in 2020, to 44 zettabytes, in part, is showing the exponential rise of digital data worldwide. Perhaps, the greatest difficult and critical challenge over large backup devices during the era of Big Data is the management of storage efficiently. Microsoft, as well as EMC workflow, research shows that over 50%, besides 85% of information on main and subordinate devices of output, have been superfluous. One of the previous studies found that almost 80% of the organizations studying data deduction approaches over backup devices are pursuing to eliminate superfluous information to improve the productivity of their storage and to cut backup economy [1].

Data storage is an integral cloud infrastructure branch, allowing database proprietors to keep the data on cloud systems and deliver flexible and profitable storage amenities. In the present decade, most individuals and organizations, due to the benefits of cloud

computing with regards to cost and control, have been storing their personal data with cloud service providers. But this ambitious data management model nevertheless confronts several new safety and reliability problems. The first problem is the quality of cloud services. The cloud volumes are growing exponentially, and the same cloud data is often stored by multiple users, resulting in duplication of data. Recently, 80% of electronic information has been standardized, as shown in published studies. Cloud storage desperately uses efficient deduplication strategies in order to conserve storage capacity and enhance storage performance, that is to say, cloud servers only retain a new instance on any duplicate file and the service provider only requires a connection to access a single file on all DO's in same file [2].

Deduction methods are now approximately divided in two: deduplication from the server and deduplication from the client. Database deductible means that the cloud server tests the replication by acquiring the contracted file through clients and performs a deduplication process. Only the normalization is saved with this form. The client-side deduplication process applies to the client communicating over the cloud server for checking, if the given external file was kept before the file was submitted. If the external data has been doubled, the recipient won't have to store the external data, and also uses the data kept over the server. The service provider deduplication obviously saves server capacity, communications, and latency, which helps both the cloud and the customer. The second problem is cloud storage protection [3].

Only a single copy of the same file is retained by the service provider in a deduplication cloud service. The harm to this particular file copy could lead to hardware or program malfunction, which would result in severe loss of the data of owners and service providers. The authenticity of the very few versions of data security for cloud deduplication storage solutions is also important. The emphasis of data owners and service providers is the fact that outsourced data should be stored securely and intact in the cloud server. Therefore, it is very important to facilitate the credibility auditing of the Cloud deduplication scheme. Owing to their worries over safety and security, owners of data are thus encrypting their own data with the help of their own keys before cloud outsourcing [4].

The process of data deduplication is thus difficult since separate ciphertexts result in the same data, encrypted by keys of multiple data proprietors, and is not duplicated further by the cloud. A hash key has been used as an encryption key to resolve this issue, which otherwise known as convergence encryption approach. The same data is then encrypted into this kind of ciphertext, enabling ciphertext deduction. While this appears to be the perfect option for anonymity and deductibility concurrently, it is sadly influenced by well-known vulnerabilities such as malicious attacks and the issue of accuracy in tags [5].

Big data deductions are a widely flexible automated deduplication strategy used to handle dredging under database infrastructure improvements in order to satisfy cloud storage service level agreement criteria. It commonly supports direct deduplication architecture, as it copies over repositories on the basis of the information cohort, can automatically be identified and discarded, and thus, physical processing needs can be substantially reduced and bandwidth network preserved during data transfer. It works within a standard distributed deduplication architecture to meet large data scaling capacity and efficiency demands. The system comprises the internal assignment of data from clients to various backup systems via data redirection methods, as well as the autonomous removal of intra-node consistency of specific storage nodes [6].

Unfortunately, in inter-node and intra-node contexts, this small-piece, adjustable deduplication system is distributed on a broad scale. Firstly, there is a problem known as the Deduplication Node Knowledge with the multi-node situation separating over decentralized deduplication with a maximized additional workload in the universally similar request. This implies, regardless of the contact overhead requirements, that replication is only achieved within different nodes and the cross-node redundancies remain unaffected. Secondly, the chunk index search disk bottleneck suffers from the intra-node situation.

There is a broad chunk index that maps the fingerprint of every chunk on the disk to classify the mirrored files [7].

Usually, it is an extensive process to fit the small deduplication node storage and contributes to a significant deterioration of the simultaneous deduplication efficiency of many data streams due to the widespread and random Input/Output (I/O) disk indices. The general data deduplication workflow is shown in Figure 1. The documents are first separated into pieces of the same or equivalent size and their individual footprint represents each piece. Doping preserves the specific non-duplicate fractions on the disk by checking the identity of them using signature indexing. The list of partial metadata chunks that would be used to restore the old file are also documented [8].

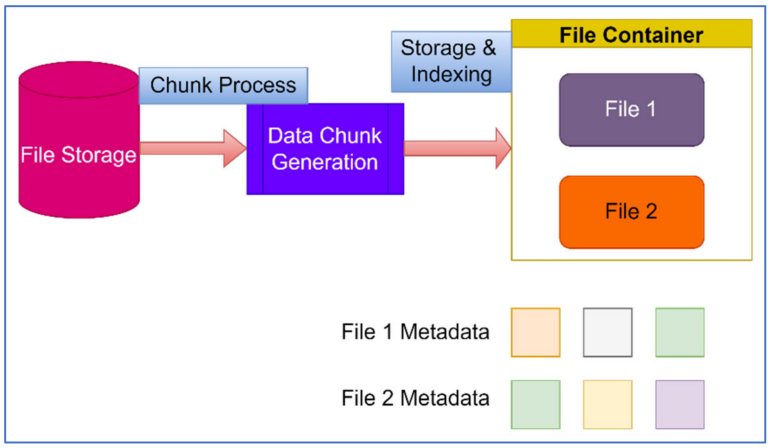

Figure 1. Data Deduplication workflow.

As the volume of signatures exceeds the capacity of Random Access Memory (RAM)in large storage solutions, a range of optimization methods can be proposed to speed up the index on the disk. In general, most chunk data deduplication methods have five major workflows: overall layout, facial recognition, fingerprinting indication, further capacity management, and compression [9]. Further encoding is optional, such as standard non-duplicate transcription of the parts and non-duplicate, but identical, delta compression. Data deduplication storage communication can be characterized in several categories: removal of fragment data restoration, collection of waste, redundancy, protection, etc. [9].

In terms of decentralized data deduplication, scaling performance and a data reduction ratio comparable to that of a centralized deduplication system are two technological obstacles. We can get an optimum data deduplication ratio by requesting and comparing all of the data worldwide. However, a global index library must be maintained. Overheads in network transmission are caused by index data updates as well as duplicate data detection [10]. In a cloud storage system with hundreds of nodes, such a global deduplication will have a considerable performance decrease. Local deduplication in conjunction with content-aware data routing is an alternate option. One must overcome the hurdle of constructing a data routing method that is both low-complexity and high-dedupe.

There is a list of the important contributions of our research:

- In order to promote privacy protection and control in complex fashion, we suggest an effective and reliable data deduction method;
- We also built a seamless upgrade strategy for reducing the update frequency as well as overhead calculation, aimed at multiple user data deduction through maintaining uploaded data protection;
- According to the application understanding, data similitude and locality, a two-tier routing decision was introduced for directing the file after customers towards the deductions of memory to strike a virtuous equilibrium amongst the competing priorities of better deduction efficacy and reducing workload of the overall system;

- We have designed a universal routing model to ease the chunk search disk constraint for each destination computer with autonomous similitude indexing with extremely fragmented data over the conventional chunk signature indexing schemes;
- The analysis of effectiveness and security shows that the method developed is effective and reliable.

2. Literature Review

2.1. Reducing Data Redundancy

The purpose of information encoding remains to provide an information source file with the least number of bits, as precisely as possible. In general, data compression could be categorized as lossless and lossless in two broad groups. Lossless compression reduces files, as expressed by optimizations such as GZIP and Lempel–Ziv–Welch(LZW), in reversible form by defining and removing statistical redundancies [11]. Loss compression removes information by defining and irretrievably eliminating redundant information as is typical of JPEG compression techniques. This article concentrates on the lossless type of compression, particularly data deduplication as lossless compression for general storage systems is needed [12].

The early methods for data compression use statistical models, also known as entropy coding, which define byte redundancy. The most commonly used entropy codification algorithm is Huffman, which generates the best codes for entropy coding using a frequency-sorted binary tree. In the 60s, ELIA's first suggested arithmetic code converts the whole data over many numbers, aimed at a better compressibility, to meet fundamental data compression ratio limit. The coding of Huffman divides the input into variable symbols, including one with a smaller code [13].

Lempel and Ziv proposed the modern dictionary-model coding method, which is described by the LZ77 and LZ78 algorithms in the 1970s, through a mounting number in technological knowledge globally. It simplifies and speeds data compression by defining string redundancy: it recognizes the strings repeatedly with a descending space then substitutes replicated data with the matched places and spans. Eventually in the 1980s, LZ-compression alternatives were suggested either for the purposes of improving compression or speeding up the encoding operation. Entropy coding methods typically must count all of the information in smaller, non-portable, bits, before coding the occurrence of frequent bytes [14].

Vernacular methods have to look for any line, so that duplicate strings can be matched and eliminated. Consequently, entropy and dictionary coding methods also restrict the compression pane to switch the compression ratio and speed between each other. The compression window sizes DEFLATE that combines LZ77 and Huffman are 64 kB, while for the bzip2 and 7z the maximal windows are 900 kB and 1 GB. There can be broad differences in total compression with the vast array of compression window sizes, although certain techniques are available for increased encoding by analytical response [15].

2.2. Main Characteristics of Data Deduplication

It was recommended that to use variable long chunks in massive file systems in order to detect related data. Deduplication was suggested in the form of a network file system based on this differential chunking concept. When storing separate copies of the same data, such as archival storage and backup schemes, the biggest improvements in data deduplication are made [16]. Consequently, data deduplication for primary storage devices also has been successfully implemented. Digital device processing centers, where data can be deduplicated for virtualization and virtual disk image storage, are another area of use. As discussed earlier, data deduplication from a knowledge, theoretical, point of view has not yet been examined.

Encoding with unidentified alphabets, also based on multi-origins or the null-frequency problem, are the closest problems to informational-theory literature. In reality, the big repetitive blocks in the data source are part of an unfamiliar alphabet that the encoder needs to

understand and explain. In the information-theoretical literature, the associated issue of file synchronization was thoroughly examined. Duplicates of two different copies of almost the same file are an issue with synchronization [17]. Duplication, on the other hand, addresses a vast number of redundant files or data blocks and their communication is unknown beforehand. The most common data deduplication technique at present reduces chunk redundancies in large storage solutions by measuring and fingerprinting data bits. Notice that a deduplication of files was proposed previously, but subsequently, due to improved compression, the approach was overshadowed by chunk-level deduplication [18].

In certain cases, the deduplication of files adopts many of the advantages of finer-grained contrasts. In recent times, the scale of digital data has steadily increased, but the entropy of information is not increasing proportionally. Some data may be continuously copied and preserved on backup and archive storage facilities, for example, by a substantial chunk of high-volume data. In order to identify similar strings for duplicated recognition, conventional compression methods are using a byte sliding window using a small window. In contrast, the data deduplication breaks the data input into non-overlapping, isolated chunks throughout a storage device [19].

In the past, Dropbox has used an established deductibility approach, in which the file hash value is considered to act as evidence that a person has this file. Even worse, the service provider could supply a malicious user with the right hash value to access the file, which ensures that the lack of a sufficient hash value will cause a complete file to leak to an external opponent. That is to say, a major security flaw exists in the deduplication process. Halevi et al., proposed a promising protocol called Proof of Ownership, which allows users to show that Content Security Policy (CSP) owns all of the stored information. Also, the authors have suggested an enhanced Proof of Ownership system, with continuous calculation costs [20].

However, the security of duplicated files is not considered in both of the POW systems. Data owners have more recently began to encrypt contracted files in needed to shield their privacy. Each owner generates distinct ciphertexts because of the varied encryption methods and keys, and hence each owner needs a service provider to store the varied ciphertexts. Therefore, the deduction of a cryptographic outcome is one of the difficult issues. The file deduplication system was first applied to privacy security by Ng et al., To fix this secrecy, Dourceur et al., implemented a primitive cryptography called "Convergent Encryption". DupLESS was investigated for the realistic use of convergent cryptography in safe deduplication [21].

Li et al., used the disperse key approach in order to handle convergent encryption content-based keys. Decentralized network deduplication approaches, with the support of accurate deductions, are transmitted to storage nodes.A hash table has been used, which is distributed in nature, that applies a routing scheme that has no states, by routing customer data to the main data server through equal grid batch coarseness to the deduplication processes [22]. The stateless navigation only contains data about the processing chunk, rather than information about the routing of the previous chunks. While these specific decentralized deduplication systems are able to achieve high-capacity savings, the device output in any store node is still bad due to the weak position. Extreme binning, by using file semblance, is an estimated distributed deduction strategy. The application locates and extracts the similarity fingerprints for all signature files in the file, then sends them to nodes that perform the deduplication process in an insecure stateless manner [23].

This strategy limits deduplication where interfile similarity is low. Increased cache missiles and data skew also affects this. Symantec suggests another data routing method, equivalent to Intense Partitioning, but it just offers a rough architecture. By exploiting data location in backup streams, EMC has built statefull, as well as stateless, models of the routing process. It is installed over DDFS with component-level deduplication with its decentralized deduplication system. The super-split routing is preferred, compared to using single strokes to maximize deduplication, to obtain scalable performance. Decentralized forwarding is an effective and convenient way to create a small deductible cluster. However,

the load balance for a massive, distributed deduplication cannot be saved by high power savings. Stateful routing can accomplish high deletion and load balancing, but it has high fingerprint connectivity overhead [24].

2.3. Security in Deduplication

Douceur et al., recommend and formalize a solution to ensuring data secrecy when the data hash is encrypted, and when an appropriate tag is created from ciphertext. Storer et al., takes the approach to look for secure, anonymous models for deduplication of results. Since this solution is vulnerable to attacks by brute force, Bellare et al., introduces a comment authentication framework and its security concept and suggests one kind of encryption model application [25]. To stable and effective multipathing of vast files, Chen et al., formalizes a notion of frame comment encryption. In addition, Block-Level Message-Locked Encryption (BL-MLE) takes a main administration and PoW into account, at block level. Keelveedhi et al., suggest DUPLESS for further ensuring the theprotection of Convergent Encryption (CE), which encrypts data from a key-server using a forgotten PRF protocol using message-based keys. It works well to avoid the assault of the conventional subjectivist CE from potential brute forces [26].

However, DupLESS participates in significant overhead computations for fine-grained deductions since it takes time to obliviously use PRF protocol. Duan introduces a distributed variant of EwS DUPLESS, in which the consumer would communicate in order to create a convergent key, with the threshold of other customers. It works well to avoid future attacks from the conventional deterministic CE by brute force. Consequently, DupLESS is involved in large overhead measurements for fine grain deduplication, so it takes time to obliviously use the pseudorandom function (PRF) protocol [27].

Duan offers a distributed EwS variant called DupLESS, in which the customer must communicate with other customers' thresholds to produce a converging key. It is unsatisfactory that all Cloud Service Provider (CSP)authentication tags from separate DOs must be stored in a single register, resulting in large overhead storage for the CSP. Harnik et al., illustrates how information deduplication could be used as a malicious insider attack that contributes to cloud storage data leaking. Halevi et al., are introducing a related scenario of attacks using multiple cloud consumer data deduplication. They add a notion of ownership proof to resist this attack, in order to efficiently show that a recorded owner has the full file in the cloud [28].

In order to produce proofs on a random basis, Pietro and Sorniotti propose the effective and stable proof of ownership schemes called s-PoW. Through using the Bloom filter, Blasco et al., are building the PoW for each block to store cryptographic tags. Note that no privacy security is taken into consideration in any of the systems listed above [29]. Ng et al., introduced a modern POW system for the deduplication of private data that is very computationally slow to produce PoW for encrypted data. For the encrypted data in cloud storing, Xu et al., suggested a leak-resistant deduplication system implemented in client-side that works under PoW. As explained in Li et al., convergence keys can be exchanged and distributed over several systems through the network to ensure the protection and stability of the chosen values for deduplication done over the blocks [30].

Zhou et al., suggest a clever solution; user data as well as indoor deduplication over blocks with a combination key management scheme are implemented in SecDup. In addition, various CE variants are used between clients to reduce overhead measurements [30,31]. Moreover, the keys on the block level are encrypted using a file-level key, to prevent the quantity of shared users from increasing key space. However, because the keys used over the blocks have been provided using the public key, it does not minimize overhead space required for key storage as described in the paper by the multi-level approach [32,33].

Li et al., developed a stable deduplication method for distribution of the data on various remote server machines. Also, Yan et al., suggest a file replication framework focused on the problems of file possession, as well as substitution-based encoding for the adaptive monitoring and revocation of data access [34]. Wen et al., recommend a session-

key-based convergence authentication scheme designated as session-key-based convergent key management scheme (SKC), as well as CKS, to support dynamical key organization in cross-user data deduplication in pervasive computing social systems [35].

But, complex property maintenance problems inherent in the safe deduplication cannot be addressed in these schemes. Hur et al., proposes an encryption-based deduplication approach on the server. They follow randomized convergent encryption (RCE) and an ownership community key for the complex handling of cross-user file levels [35–37]. The throughput will, however, not be preserved and the PoW technique is not considered.

3. Proposed Approach

The proposed data deduplication system is described and built in this section. The deduplication rate ought to be proportionally deduced through the storage nodes and with other numbers of nodes. To hit a high duplication removal ratio, identical data must be redirected to the same deduplication node. To achieve a high duplication elimination ratio, identical data should be transmitted to a certain deduplication node. To achieve strong deduplication performance and a high deduplication ratio while using little computational power, we build an inline deduplication architecture, outlined in this section. The design of our developed framework is seen in the following sections. We introduce our suggested data routing solution to improve high deduplication reliability and scalable performance. The approach has been accompanied with the overview of the implementation-oriented datatypes for fast deduplication performance over the removal of duplication endpoints.

3.1. Overview of Framework

The proposed approach of this paper has been depicted in Figure 2. The architecture proposed in this article is symmetrical in nature with both the end user as well as end service provider. In this symmetrical decentralized deduplication model, there are mainly three modules: user nodes, storage nodes, and manager nodes.

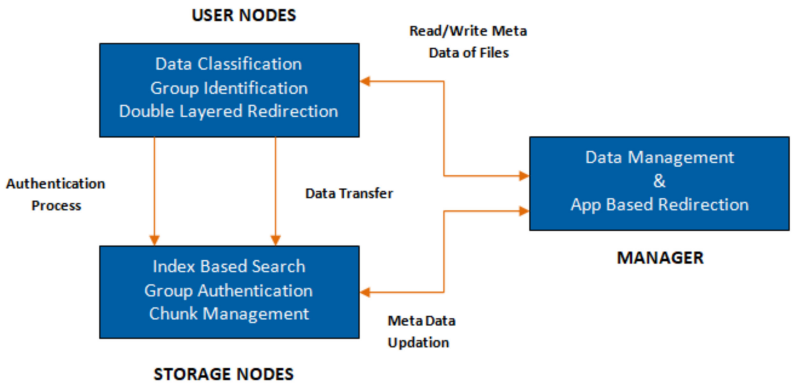

Figure 2. Proposed System model.

User Nodes: A user node has three primary feature components: data categorization, identity verification chunk, and data redirection. The user feature preserves and recovers data files, executes a colliding-resistant hashing algorithm, including SHA and MD5, in the data clustering module, determines the chunk sizes of each data stream, and that redirects from every part to a deduction backup node that is highly similitude via the two-tier routing check. The customer decides whether a data file is duplicated or not by choosing the shock fingerprints over the target node on a data component close to information part transfers, then, the individual components have been transmitted across node links to increase the distributed device interoperability, by preserving the network transmission bandwidth.

Storage Nodes: The storage server section consists of three main components: client similitudes database search, chunk index cache storage, and chunk management. It introduces the core deduplication and network storage logic, along with the return of application-conscious search results for route optimization, stuttering new hot chunk signatures into

the chunk index buffer to speed up the search process, duplicating chunks while storing the single components in larger units known as warehouses in parallel.

Manager: This was responsible for monitoring and handling file platform for data storage and retrieval on the deduplicated storage node. It comprises of administration of file recites and forwarding decision-making. The file receipt management module hosts file mapping for extracting impressions as well as file reconstruction data. The director maintains file-level metadata. The program, understanding the forwarding recommendation framework, selects a collection of applicable storage nodes to every file and provides customers with input on direct super chunk routing. In an effective link failure, the director assists up to two nodes in order to prevent the singular link failures for high access.

Our system generally deducts user data levels by uploading the files from the data holders. Each file has an information module and a file-based key that are determined by the owner of the file. A data user then transfers the label to the supplier. A service provider completes the process of deduplication and generates reports to the data holders. Notice that the data user and service provider ought to provide evidence of the job procedure for ensuring tag accuracy. When the data owner opts to do a block level deduplication in an interior customer, the file will be split into multiple data blocks when not a redundant file.

By executing the user encryption, the converging keys and tags of these blocks are determined. The data proprietor then transmits the label codes to the service provider. The service provider must verify that the block tags are on that data owner's tag index and will return the data manager's results. Afterwards, the data user encrypts all unique frames and sends the service supplier ciphertexts of blocks. We also consider the issue of complex ownership governance and follow a lazy upgrade approach to minimize protection update frequency.

In the proposed approach, we use a bilinear mapping model such as $f: H \times H \rightarrow H_U$ with the bilinear model properties where H and H_U are multiplicative cycle groups with an order q which stands for the order as:

$$\forall h_1, h_2, h_3 \in H \tag{1}$$

$$\forall x, y \in Z_q^*, f\left(h_1^a, h_2^b\right) = f(h_1, h_2)^{ab} and \tag{2}$$

$$f(h_1, h_2).f(h_1, h_3) = f(h_1, h_2.h_3) \tag{3}$$

A novel asymmetric encryption approach for making the transaction more secure has also been proposed. A modern strategy is probabilistic encryption. Probabilistic encryption is highly helpful when it comes to deduplication since the convergent encryption keys only rely on the file and do not change the file itself. The file owner's file serves as a source of data that results in a file key and a file tag being generated. This file key is utilized in the cryptography, while the file tag is just employed to aid with deduplication.

Our application-conscious direction-finding choice is influenced through the repetitive study of applications. It distinguishes between various categories of implementation information through taking the submission's sensitivity with extension of the filename. Also, it chooses a set of deduction storage nodes as applications that have processed the same type of distributed applications with the file. This method relies on a route structure which maps an application type to ID of a backup device. Here, the process knows the routing process which carries out the director's module of routing knowledge. Similarly, a static data routing system is inspired by our super-scale resemblance research. The data routing system is similar. It routes the same superficial processing node with just a few nodes to find storage state information and manages a near-global load balancing without a high overhead scheme.

A file is separated first into c smaller bits in the data splitting module, which are combined into something like a super chunk T. Then, a hash function is used in chunk vulnerability scanning framework to compute all the component identifiers $\left\{g_q^1, g_q^2 \ldots, g_q^d\right\}$. In the client data routing node, the data routing algorithm is performed. For m applications,

our hand printing-based data network architecture will boost load balancing by adjusting the least-loaded node for each super chunk in l candidate nodes. We also prove, since there is a uniform distribution of randomized handprint by cryptographic hash functions, that the global load equilibrium can be reached. Its stable hashing-based data classification is scalable so the addition or deletion of nodes in the server cluster will prevent re-shuffling all subsequently saved data.

The ciphertext and the members of the groups should be changed to ensure an anonymity in future, and backward in the complex ownership management operations. Comprehensive procedures are carried out under particular scenarios:

As the file is updated and then uploaded by μ_j to the service provider, the service provider then attaches μ_j to H_j and changes the community key and re encodes cyphertexts with the lazy upgrade technique. We create an update list for L_j using newly added μ_j uploaders to minimize the overhead for the calculation of often upgraded operations. The Community Key CK_j doesn't instantly upgrade after adding μ_j to the update list L_j. When the service provider files L_j to the service provider, the service provider can first validate L_j's upgrade lists. If μ_j is present, the service provider will conduct updates and refocus operations for the community key. It clears the list afterwards. This enables them to reduce the scale of group operations while ensuring retroactive confidentiality. The following are the comprehensive procedures for major upgrades and re-encryption operations:

CK_j has been used for encrypting encrypted data $C_j^{1'}$ and the corresponding modified data C_j^1. An arbitrary community key CK_j' has been chosen and the encryption process has to be carried out.

The service provider then uses the encryption process for getting all the associated data values found in the table associated with the users U_j^1.

Then, the list is cleared by the service provider and the revised list has been obtained as:

$$CK_j' = \left\{ E_l\left(C_j^{1\prime}\right) \right\}_{l \in l(\, C_j^{1\prime})} \tag{4}$$

While μ_j tries to remove and free the data space of the cloud storage, μ_j transmits a message for removing the data to the service provider. Note that the key used for encrypting blocks must never be associated with the information of the file for complex ownership control, so the updating level of the internal user block will alter the key at the file level. This is distinct from cross-user deduplication of block levels, where chunks only belong to μ_j in internal user deduplication, which makes control of possession much simpler.

In theory, we can generate tags with the necessary bit length to prevent a collision with the cryptographic hash function. However, a huge proportion of hash tags would be created for large volumes of data in the realistic cloud storage scheme. Thus, the possibilities of various data having the same mark due to the hash-collision issue are slim, but possible. There are approaches that have been proposed to handle this situation, they are outlined as follows.

Specific files with the same tag will be saved together under the tag during file level deduplication. For instance, if two separate data D_i and D_i have the same file tag T_i then both D_i and D_i are stored in the T_i tag, in accordance with the order of the upload. In addition, for each file, a command number v is assigned. The service provider shall return the order number—however, according T_i to the data controller of the subsequent editor passing the PoW operation. Therefore, the CSP may conveniently find this information, however, according to $T_i \,||\mu$, mostly during upload or update phase.

This framework will store all blocks $B[i]$ and $B[j]$ in order for the deduplication at the inside-user-level block levels, similar to the deduplication at the file-level, since separate blocks $B[i]$ and $B[j]$ have the same block tag T_i, the order number β is then indicated to each block. Our PoW system is used to determine the block it corresponds to for blocks with the same tag. The service provider generates a sequence value based on T_i with the owner

of the data after the block deduplication procedure. Throughout the update or storage progression, the service provider may simply locate the $T_i \,||\gamma$ block.

3.2. Deduplication Process

The deduplication of the client file can be safely enforced in our program. In particular, when a file TF is obtained by a data owner, the service provider first performs the deduplication test. The user uploads the data if there is no TagF in the cloud and service provider includes TF in its file tag list. If not, the service provider already has a copy, and the administrator executes the service provider POW protocol as shown in Algorithm 1. This user is allowed to access without uploading the file to this saved file, if the protocol is passed. This saves both computing costs and connectivity costs for customer-side file deduplication.

Algorithm 1: Proposed Deduplication Model

Step 1: Client file management with file tag list.
Step 2: Create authentication tag collection as $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$.
Step 3: Estimate Service provider or user behavior modelling as $Q = 1 - (1 - \beta)^d$.
Step 4: Service provider authentication setup phase: $\left\langle G_j, D_j^2, F(GD), Public\ Key \right\rangle$
Step5: Compute Authentication key value as: $L'_j = F\left(E_j\right) and\ M' = L'_j \oplus D_j^2$.
Step 6: Encrypt the file tag information and file chunks.
Step 7: Encryption of keys as $E(PK_j) =$
Step 8: Authenticator thread generation $\alpha_j^{w_j} = D_j^2 \oplus F(GD)$.
Step 9: Execute query for deduplication $Q_d = \left(f'_j, G(i)\right)$.
Step 10: Hash index generation with object indexing process.

Traditionally, the service provider stores the authentication tags created by all data owners for all file blocks, except in that file, which places large $O(F_m)$ storage burden on the service provider over time. This is the standard method of carrying out data integrity verification. Different to the methods above, the service provider will add a tag for the same block and the authentication tag that can only be stored for a collection of authentication tags $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$.

Thus, our system will reduce the overhead storage for the service provider on $O(F_m)$ to $O(m)$. In addition to ensuring the trust of the service provider in the data, the concept of convergent encryption allows one to perform "known information" and compatibilist encryption, which is often deducted encrypted data. Furthermore, the service provider will mask the weighted linear D from the sampled blocks to auditor's random masks S and public key to further avoid privacy leakage against auditor in the course of producing data evidence. Whilst an auditor knows the public key of S and the data owner, the private contents of the data owner under hardness assumptions are computationally unfeasible for an auditor. We explain whether the auditor or the service provider picks m or n blocks arbitrarily in the honesty audit or in the PoW process, to challenge the entire n blocks of file G.

In reality, all blocks for verifying the file's credibility and accuracy are ineffective and inadvisable, since the large volumes of external information are available in the storage of the cloud. Herein, stratified arbitrary process, though, is cheaper and more convenient to detect the wrongdoing of the service provider or the user's dishonesty. The logic and viability of this approach have been shown in previous studies. Assuming that Q is the likelihood of identification for the malfunction of the service provider or consumer, and d denotes quantity of blocks confronted, so the ratio of these quantities meets in the calculation:

$$Q = 1 - (1 - \beta)^d \tag{5}$$

Also, total blocks confronted d, therefore depending on the fraction of chunks damaged β as well as the chance of identification Q. For instance, if the service provider or an unauthorized behaviour has one percent shady blocks, the auditor or the service provider only requires 367 sections to detect broken blocks with 75 per cent likelihood.

$$f(\mu^u, h).f\left(Y, \prod_{k=0}^{u-1} z_k\right) = f(Y, z_k).\prod_{j\in E} \alpha_j^{w_j} \tag{6}$$

where, $\alpha_j^{w_j}$ defines the service provider authentication threads. The service provider retains only a single copy of the same file in a deduplication cloud storage system. The authenticity of a single copy will be harmed if the equipment or program fails to store it. The data owners must also establish, at all times, the credibility of their contracted file. In general, the data owners may entrust an inspector with a separate integrity audit for file F, which is restricted in computational capacity or limited capital.

To provide a maximum deduplication performance with a minimal overhead, the application route table is maintained inside the manager, together with a chunk fingerprint cache and two primary data structures: an application knowledge similarity index and bin. The application route table supports high deduplication with reduced marginal system overhead. In order to make system aware routing decisions, the application routing table is integrated into the manager. Each table entry saves the application type mapping to the node ID and the related capabilities in the storage node for this application. The director will locate the storage node list of an application for a certain form of application and measure the working power for data storage. In view of the system and node, the stage remains together and can conveniently be fitted into the understanding of the server to speed up query operations throughout the application route table. Similarity in-memory data structure is an app-aware similarity index. It comprises of an object index and thin, application-type, hash table-based indices.

The inbound block is referred to as a minor directory of the identical type according to the proceeded type of the file. Separately, all of the access includes a symmetrical correlation between a representative super chunk handprint fingerprint and the package ID where it is stored. Given the very low sampling rate of our hand printing, the index is significantly smaller than that of the conventional block pattern storage directory which maps the corresponding containers from all the chunk fingerprints. To design the index, we used a concurrent framework search index lookup design, as well as index monitoring, that enables the allocation of a lock or a constant number of consecutive hash buckets, which means we can monitor authentication schemes by apportioning a lock by a checksum or a constant amount of successive hash buckets on multicore data storage node.

A container that consists of a data segment to keep data blocks, also a file information tag field to preserve file information, like the signatures, offsets, and span of the chunk, is an independent disk-detailed data structure that protects localities [29]. In parallel with our server deduction architecture, we will assign, distribute, write, read, and efficiently hold chunks simultaneously. During concurrent file storage, with every incoming file, a dedicated open container is retained. As the container fills a new one is generated. The fragment pattern caching similarly performs a major part in improving accuracy of the deduplication process, apart from the above-described data structures. It maintains the chunk of recent RAM container fingerprints. Since a representative fingerprint has been linked to a search request in the application-aware sequence database, the cache pre-fills the entire metadata portion of a mapped container to speed up the fingerprint search. If the cache is complete, an appropriate cache substitution strategy is used to create time for potential prefecture and caching, including Least-Recently Used (LRU) approach.

As shown in Figure 3 the client decides a member node as the path goal node for storing the super-chunk and tells the manager of their node ID by request letter. To make sure chunks are not duplicated, the customer bundles all the chunks from the super chunk into a batch and delivers them to the destination node. The target storage node answers

the customer through a set of special components in the block while searching for chunks of fingerprints. In addition, only the unique bits of the super-compact need to be sent by batch to the target node. For each super-spar, the steps were repeated until the file ends. A customer submission to the manager often serves the purpose of downloading a file. By checking the file method, the manager responds to this request and sends the notification to the user.

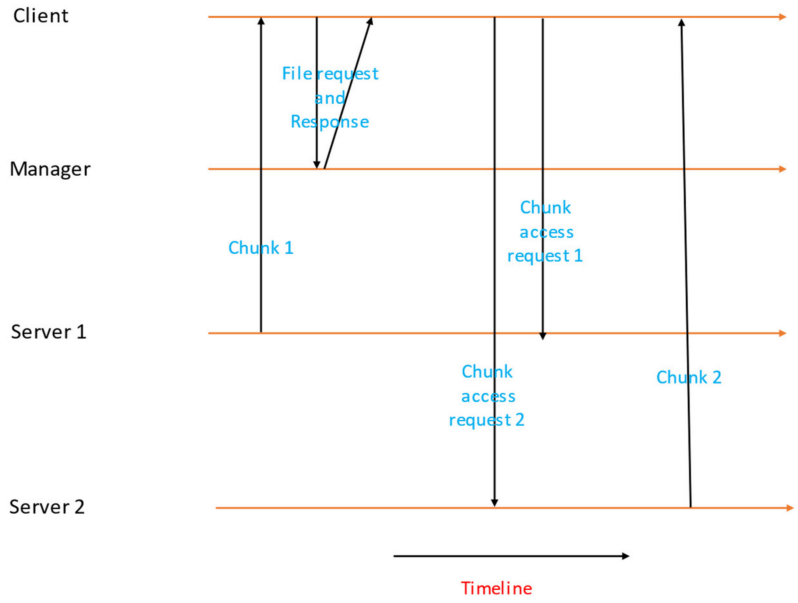

Figure 3. Timeline and communication.

The message includes the block array over a register as well as pointing of the block into deduction cloud storage. The client would then order the required deduct storage node with a super chunk message from any super-chunk in the file. The deduplication server will extract a mega chunk from data warehouses and accelerate the performance of the restoration process. The customer installs per super-chunk to validate the data integrity with a super-chunk search and file ID. The entry data is locally duplicated by a data server. The thread receives and processes the read and write demands from the client in a data server. When the written request is received, the data server begins receiving and saving the data from the thread Block Recipient. Throughout, the Block Receiver thread also operates the deduplication engine as shown in Figure 4.

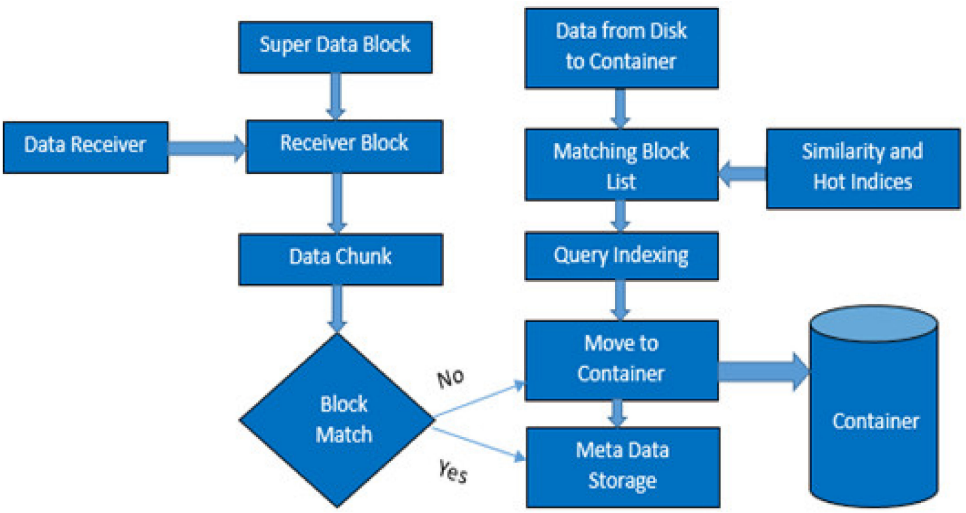

Figure 4. Process flow of deduplication approach.

Firstly, a server receives distinctive fingerprints and metadata from superblock. By searching the hot index and index, the similarity index table decides on the matched data container. Because the package is stored on disk, and an I/O read is needed. An LRU storage caches are used to fit directly and will reduce, to a degree, the number of I/O disk drives. The machine will check the index for the superblock by arranging the container in the index subsets. If this is the case, the original data must not be preserved. It will also opt to preserve the remaining data by choosing an open container. Finally, the storage information of each chunk is written into the disk. After all the chunks have been deduplicated in an extra cube, the remaining data are written in the package at once.

It will lower the quantity of I/O activities and improve server data receiving performance compared to the deduplicated chunk approach. A request from the user for metadata to the metadata server while reading a file. The metadata server then returns to the client the superblock data server address and the superblock ID. The client connects to the server, where each client checks the accompanying block. The superblock information collects the signal, and it is reordered by the data server before returning the rest of the data to the client for processing of the requested read. The machine begins the Block Sender thread, which reads and sends the requested block, when the data transceiver thread accepts read requests.

First, the Block Sender ensures that the metadata of the superblocks has been loaded with the actual address of the information of the superblocks. Block Sender then restructures the superblock, as per the metadata of the superblock. Finally, the reorganized superblock sends Block Sender to the customer. If the next data chunk wasn't a file but the location of the file, this triggers an arbitrary read operation if a data server performs local read operations. In other words, a chunk reads from such data. It only performs storage deduplication for some related containers to remove the bottleneck from a random disk read.

Thus, there will not be so many I/O operations and many data reorganization files will not be opened, thereby significantly lowering the probability of random reading. Based on its LRU cache, the Hot Index is used to boost the deduplication ratio in a single node built on the how often the fingerprints were accessed. With the proposed adaptation, we will switch toward the index value for a secondary deduplication if the newly arriving superblock has not been deduplicated in identical containers. In this cache, when the fingerprint is matched in the memory, we set the frequency for the container fingerprint.

When selecting a matching container by similar fingerprint, a successful replication outcome cannot be achieved since the similar fingerprint does not reflect the data characteristics of the container properly. The data locality function collects duplicate data that promises a strong device deduplication efficiency. Given the fact that such identifiers of a container, in combination with the data locality, are constantly affected, there is no question that there would be a greater likelihood for any other data to be found in the container as duplicate data.

4. Performance Analysis and Results

So as to assess the deduplication effectiveness of a single node deducting server, we use four processing servers. All were run on Ubuntu 14.10 and use a 4-core 8-thread, 2.53 GHz processor and 16GB RAM Intel X3440 CPU and 1TB storage Seagate ST1000DM. Seven desktop computers act as client systems, one system has been a server that acts as the manager and three other servers as the data backup nodes in our prototype deduplication scheme. It uses the ethernet switch Huawei S5700 Gigabit to communicate internally. Our client portion is based on a pipeline architecture powered by events, using an asylum-sensitive RPC deployment by transmitting TCP streams through messages. In order to reduce round trip overheads, all RPC applications are bundled. On one of our four servers, we execute event-driven simulation for deduplication, load delivery, memory allocation, and overhead communication on distributed deduplication techniques.

For our tests, we gather five types of practical data sets and 2 kinds of device traces. The Linux dataset is an assembly that is imported from the open-source code (Linux from versions 2.0 to 3.3.6). The VM database comprises of 2 consecutive monthly complete backups of eight (three windows and five Linux) virtual machines. It is possible to extract sound, video, and image data sets from consumer desktop and laptop computers. The mail and network dataset are two traces that are obtained from the web server and mail server. The column "size" reflects a dataset capability and the column "deduplication ratio" shows the ratio of a logo to the physical size following deduplication with a static chunking fixed chunk size of 4 KB, or an average of 4 KB chunk size in the Cumulus Optional chunks. In our assessment, the preceding evaluation criteria are used to thoroughly evaluate the success of the application of our prototype of the proposed method against the most advanced deductible distributed systems.

- Performance of dumping (PD): It is initially specified by having a certain date set to calculate the efficiency of various deduction schemes on the same network. The discrepancy between logical size M and physical size U of the dataset is determined by the time of deduplication Q. Thus, it can be represented by the deductibility efficiency.

$$PD = \frac{M - U}{Q} \tag{7}$$

- Normalized ratio of deductions (NRD): This is a common distributed ratio of deduplication, an accurate deduplication method, and a single node deduplication ratio all referring to the same thing. This is an example of the closeness to an optimal distributed deduplication ratio of the deduplication ratio obtained using a distributed system.
- Standardized deduplication ratio (SDR): This is equal to a standardized deduplication ratio, separated into 1 plus a standard deviation ratio/versus mean consumption of physical capacity on all deduplication servers, analogous to metrics used. This corresponds to standardized deduplication ratio. In other words, the normalization of the deduplication ratio yields a SDR. It demonstrates how effective the data routing techniques are in eliminating the deduplication node's information collection.
- Number of index fingerprint surveys: Due to its lack of intra-node communications, this system uses simulated inter-node communications to perform a chunk fingerprint search, which is easily accessible from our simulation.
- Intra-node RAM use: It is an integral overhead method associated with a deduction server chunk index search. It demonstrates how effective it is to increase the efficiency of the inner node deduplication by optimizing the index chunk lookup.
- Decentralized disk data tilt (DDT): In the deduplication storage server cluster, we describe a measurement for data skewing. The variance between maximum volume L_{max} and minimum volume L_{min} in a storage cluster, separated by L_{mean}, can be expressed and is proportional to the difference.

$$DDT = \frac{L_{max} - L_{min}}{L_{mean}} \tag{8}$$

Prior to the data routing choice, our system enables the customer to perform simultaneous data splitting and fingerprinting. In the case of program files, chunking may be done in two ways. One way is to use SC chunking with fixed-size chunks for each kind of program file, and the other way is to use CDC chunking with variable-size chunks. Hash calculation is also possible for this type of file. Hundreds or thousands of successive little bits are then organized into a super-business for route optimization. The OpenSSL library is based on the implementation of the hash fingerprint. Deduplication outcomes for different Application models are listed in Table 1.

Table 1. Deduplication outcomes for different Application models.

Application	RAM Usage (MB)	Bin Model (MB)	Deduplication (MB)
VM	8.58	24.56	19.65
Mail	55	15.38	22.54
Audio	26.36	9.35	12.74
OS	48.21	14.98	11.35
Photo	11.56	12.55	10.25

We also create a parallel application-wise similitude index searches on individual deduction servers to use the multi-core or multi-core capability of the deduct storage node. With each data stream, we have a deduplication thread that we use for multi-data stream deduplication, but we have an application-aware hash-table-based similitude index that is common to all deduction servers. By splitting the index into the granularity at which the application operates, we allow it to be scanned concurrently. While in the application-aware-similarity-index search, there are as many data streams as CPU logical cores, and there are more locks than digital data, due to the thread context transfer penalty, which forces the data to be loaded into and unloaded from the cache in every thread context transition.

Figure 5 shows the throughput analysis between existing and proposed method. In one deduction storage node, with several data sources, we equate our applications' average likeliness directory over the standard likeliness index for parallel deduplication. The findings demonstrate the concurrent deduplication output of the VM data set with RAMFS data input to minimize storage I/O block performance interference. We are aware of the similarity index and are using a cold cache to measure output and an application to measure output. In this case, "cold cache" refers to the segment fingerprint cache being empty when we first concurrently deduplicate multiple strings on the VM dataset. We revert back to a dataset if we execute parallel deduplication with several streams; "warm cache" refers to duplicate chunk fingerprint processing that has already occurred.

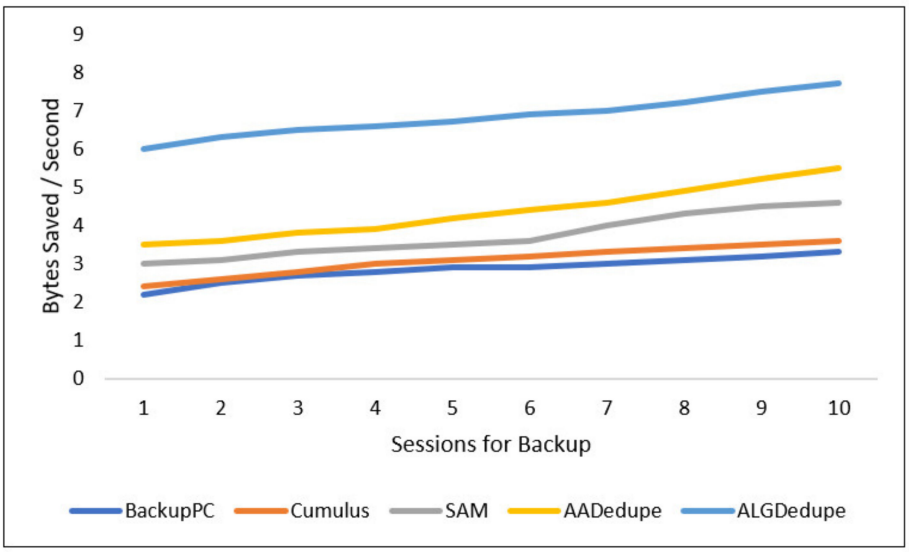

Figure 5. Throughput Comparison analysis.

Figure 6 shows the space complexity analysis. We note that parallel duplication systems with application-aware correlation probably work well over the naive similar duplication approaches, and parallel warm-cache deduplication schemes will achieve higher performance than cold-cache regimes. For both an application-conscious similarity index and a warm supply, the performance of the similar deduplication increases to 5.9 GB/s, and the rapid growth of mobile data flows. With thirteen synchronized channels, the percentage is down by 5.5GB/s because of the overhead competition for Index Location and Disk I/O.

Our scientific findings in each backup session, the providers would need the combined cloud storage capability of each customer for the six online cloud schemes.

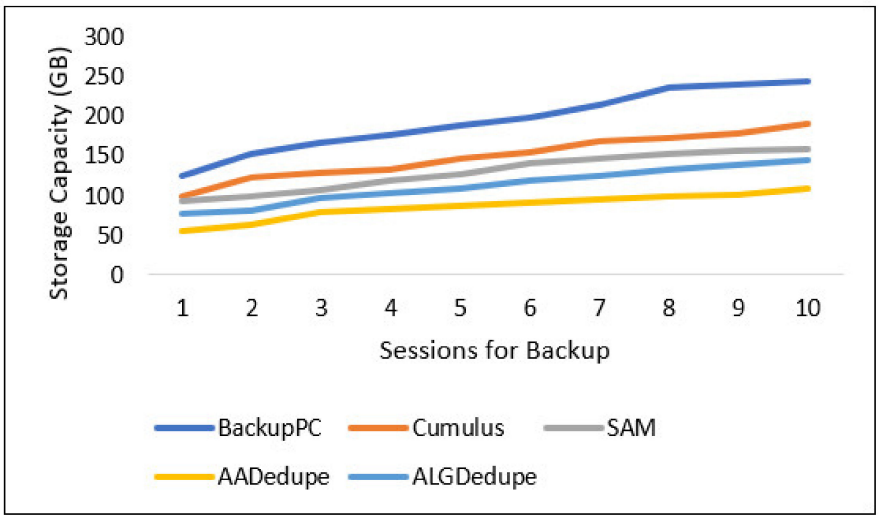

Figure 6. Space Complexity.

Except from origin deduplication systems, Jungle Disk struggles to protect high-cloud storage because it does not erase file copies written at other locations in its incremental backup system. The BackupPC gross graining system cannot find greater consistency than other fine graining mechanisms in the source deduplication schemes. The fine-grained Cumulus carries out a local replication scan only and restricts the quest for unaltered data to blocks in the earlier models of the file, thus saving less space than the local deductible-only AA Dedupe. With more levers of global deduplication with cloud computing, ALG-Dedupe increases the deductible ratio of AA-Dedupe. It also enhances SAM, which blends local chunk-level and global file source deduplication by raising framework knowledge with an overall deductive performance as shown in Figure 7.

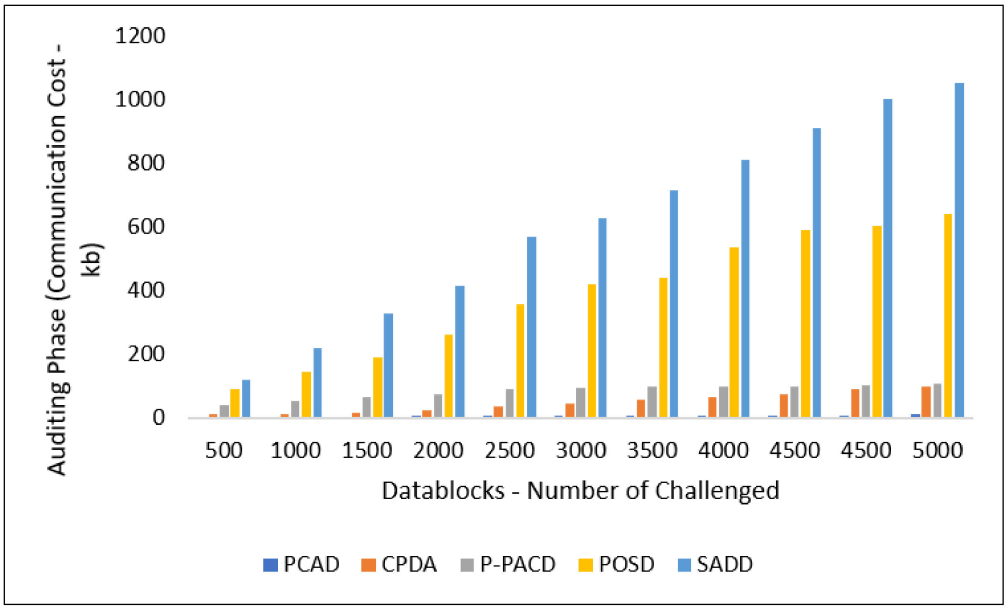

Figure 7. Data transfer cost.

Thanks to the framework architecture and world-wide duplication detection, the Jungle Disc will surpass 64%, save 43% of Cumulus space and minimize the use of SAM for third storage. It achieves a 27 percent improved space utilization, compared with the

current implementations. The high efficiency of the deduplication of data from finely grained or world-wide deductibility systems is greatly hampering system success. As shown in Figure 8 our method, the mutual Hash Table application- based archive layout that is stored on the client's side in RAM is parallel to local duplication detectors. We use a horizontal division model to separate the entire unpredicted index into several tiny, individual domains, which are divided by a file-type application category.

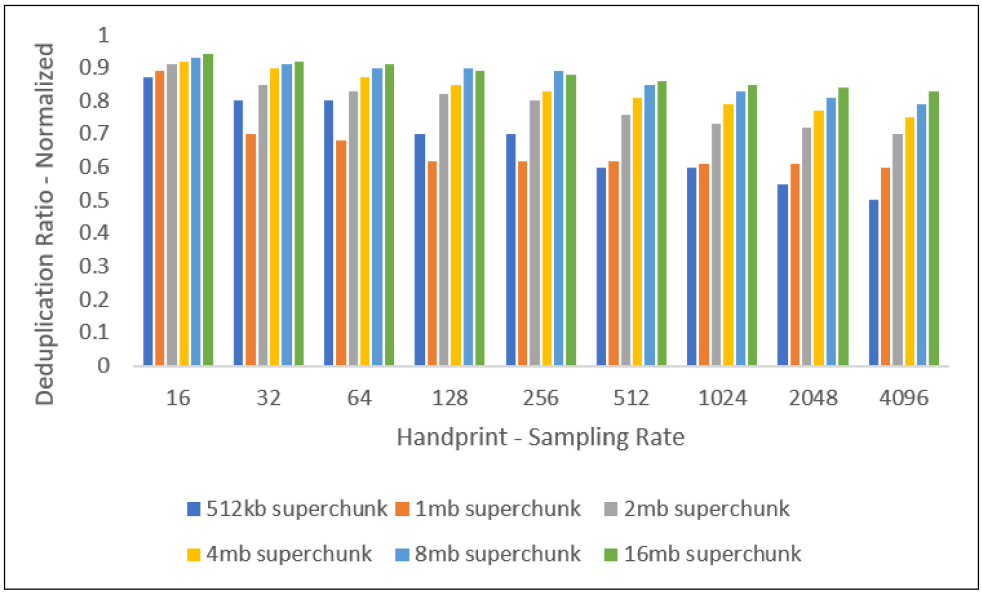

Figure 8. Deduplication process performance.

Including high parallel global replication checks, we apply Despite the large WAN latency, through batch I/O and parallel query, we greatly increase the system's overall deduplication efficiency as shown in Figure 9. In the same cloud-related storage network, we compare the five cloud backup systems and use our current metric to calculate the performance of various deduplication methods.

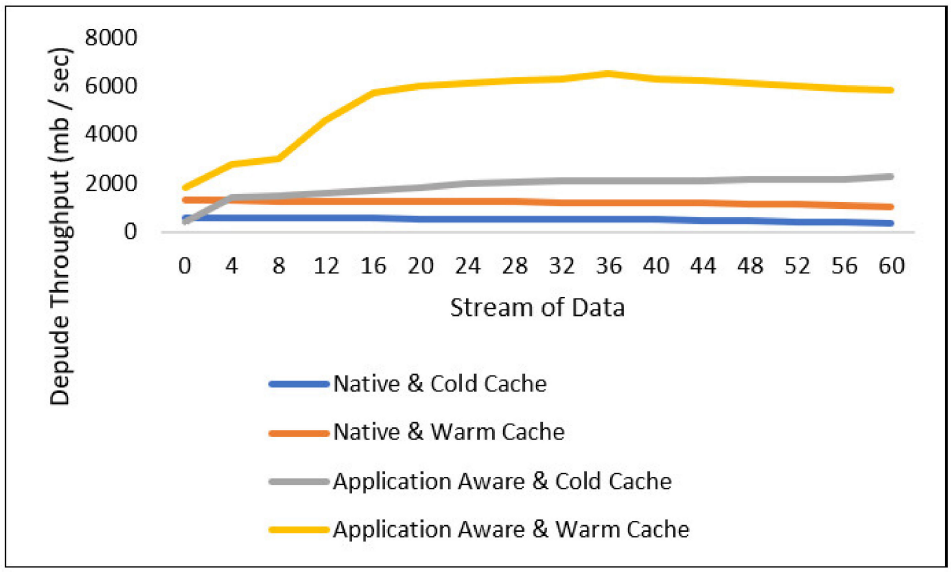

Figure 9. Deduplication throughput analysis.

The deduplication performance measurement as shown in Figure 10 is much better than other backup systems with a low overhead. The main benefit of this is the knowledge of its use and the global identification of duplicates in the deduplication process. Because

of its benefits in global architecture, it is approximately 1.6 times the application-informed SAM and 1.9 times the local dumping Cumulus quality, which is 14% higher than the existing local system, AA-Dedupe.

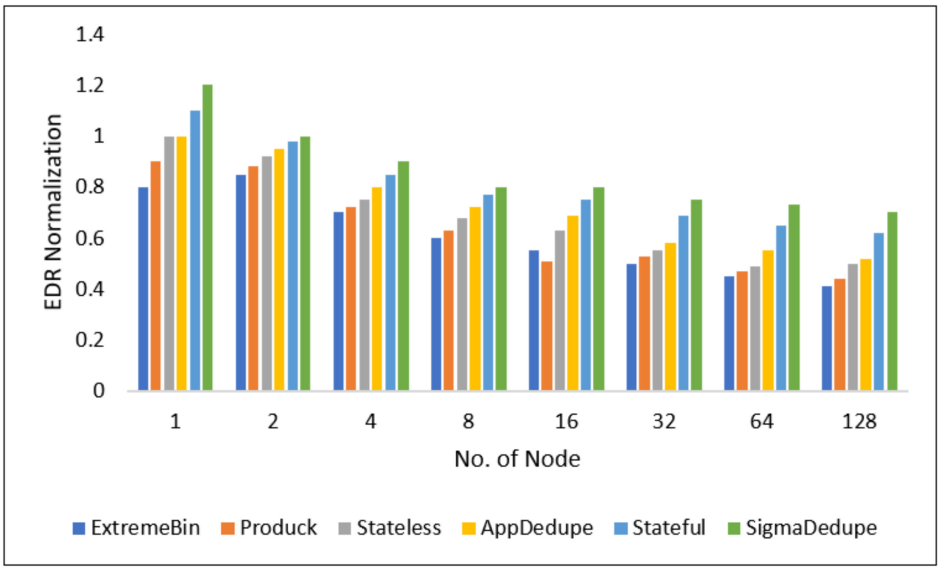

Figure 10. Distributed deduplication effectiveness.

5. Conclusions

In this article, we defined an application-conscious flexible inline decentralized large-data management deduplication architecture, which achieves a balance between scalable efficiency and deduplication efficiencies through the use of application sensitivity, data uniqueness, and localization. In order to minimize cross-node information duplication with controlled overhead and strong load balancing, a two-level information redirection organization is adopted to optimize the routing data on the super-chunk granularity, and use an application-aware sequence index to increase the performance of deduplication in each node with low RAM use. Initially, it surpasses the stateful close coupling framework in the large deduplication ratio of the cluster with a marginally larger overhead organization compared to the strongly mountable slack groupings. Secondly, it greatly increases the loose stateless connectiveness strategies in the powerful deductibility ratio across the cluster, while preserving the latter's high overhead scalability. Furthermore, we use the user-supported framework to subtract internal block amounts. The analyses of security reveals that our proposed system will ensure security conditions, other than a side channel assault, as we use the PoW check mechanism to achieve reciprocal PoW checking when bandwidth is being saved. The study of the performance reveals only a five percent excess overhead, compared with the others, is essential for the proposed scheme when joint PoW verification and management is allowed. Finally, we officially demonstrated the efficiency of the proposed technique and numerical analysis and experimental findings are proof that our scheme is effective by compared communications, calculation, and storage costs with modern schemes.

Author Contributions: Conceptualization, P.M.K.; Visualization, J.G.J.; Writing—original draft, J.G.J.; Writing—review & editing, P.M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Premkamal, P.K.; Pasupuleti, S.K.; Singh, A.K.; Alphonse, P.J.A. Enhanced attribute based access control with secure deduplication for big data storage in cloud. *Peer-to-Peer Netw. Appl.* **2021**, *14*, 102–120. [CrossRef]
2. Shynu, P.G.; Nadesh, R.K.; Menon, V.G.; Venu, P.; Abbasi, M.; Khosravi, M.R. A secure data deduplication system for integrated cloud-edge networks. *J. Cloud Comput.* **2020**, *9*, 61. [CrossRef]
3. TMalleswari, Y.J.N.; Vadivu, G. Adaptive deduplication of virtual machine images using AKKA stream to accelerate live migration process in cloud environment. *J. Cloud Comput.* **2019**, *8*, 3. [CrossRef]
4. Saharan, S.; Somani, G.; Gupta, G.; Verma, R.; Gaur, M.S.; Buyya, R. QuickDedup: Efficient VM deduplication in cloud computing environments. *J. Parallel Distrib. Comput.* **2020**, *139*, 18–31. [CrossRef]
5. Jiang, S.; Jiang, T.; Wang, L. Secure and Efficient Cloud Data Deduplication with Ownership Management. *IEEE Trans. Serv. Comput.* **2020**, *13*, 1152–1165. [CrossRef]
6. Begum, B.R.; Chitra, P. ECC-CRT: An Elliptical Curve Cryptographic Encryption and Chinese Remainder Theorem based Deduplication in Cloud. *Wirel. Pers. Commun.* **2021**, *116*, 1683–1702. [CrossRef]
7. Zheng, Y.; Yuan, X.; Wang, X.; Jiang, J.; Wang, C.; Gui, X. Toward Encrypted Cloud Media Center with Secure Deduplication. *IEEE Trans. Multimed.* **2017**, *19*, 251–265. [CrossRef]
8. Wang, Y.; Miao, M.; Wang, J.; Zhang, X. Secure deduplication with efficient user revocation in cloud storage. *Comput. Stand. Interfaces* **2021**, *78*, 103523. [CrossRef]
9. Zhang, Y.; Mao, Y.; Xu, M.; Xu, F.; Zhong, S. Towards Thwarting Template Side-Channel Attacks in Secure Cloud Deduplications. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 1008–1018. [CrossRef]
10. Kan, G.; Jin, C.; Zhu, H.; Xu, Y.; Liu, N. An identity-based proxy re-encryption for data deduplication in cloud. *J. Syst. Archit.* **2021**, *121*, 102332. [CrossRef]
11. Saraswathi, S.S.; Malarvizhi, N. Distributed deduplication with fingerprint index management model for big data storage in the cloud. *Evol. Intell.* **2021**, *14*, 683–690. [CrossRef]
12. Prajapati, P.; Shah, P. A Review on Secure Data Deduplication: Cloud Storage Security Issue. *J. King Saud Univ. Comput. Inf. Sci.* **2020**, *34*, 3996–4007. [CrossRef]
13. Hou, H.; Yu, J.; Hao, R. Cloud storage auditing with deduplication supporting different security levels according to data popularity. *J. Netw. Comput. Appl.* **2019**, *134*, 26–39. [CrossRef]
14. Tan, Y.; Jiang, H.; Sha, E.H.M.; Yan, Z.; Feng, D. SAFE: A source deduplication framework for efficient cloud backup services. *J. Signal Process. Syst.* **2013**, *72*, 209–228. [CrossRef]
15. Gao, X.; Yu, J.; Shen, W.T.; Chang, Y.; Zhang, S.B.; Yang, M.; Wu, B. Achieving low-entropy secure cloud data auditing with file and authenticator deduplication. *Inf. Sci.* **2021**, *546*, 177–191. [CrossRef]
16. Kaur, R.; Chana, I.; Bhattacharya, J. Data deduplication techniques for efficient cloud storage management: A systematic review. *J. Supercomput.* **2018**, *74*, 2035–2085. [CrossRef]
17. Shen, W.; Su, Y.; Hao, R. Lightweight Cloud Storage Auditing with Deduplication Supporting Strong Privacy Protection. *IEEE Access* **2020**, *8*, 44359–44372. [CrossRef]
18. Wu, J.; Li, Y.; Wang, T.; Ding, Y. CPDA: A Confidentiality-Preserving Deduplication Cloud Storage with Public Cloud Auditing. *IEEE Access* **2019**, *7*, 160482–160497. [CrossRef]
19. Wang, S.; Wang, Y.; Zhang, Y. Blockchain-based fair payment protocol for deduplication cloud storage system. *IEEE Access* **2019**, *7*, 127652–127668. [CrossRef]
20. Fu, Y.; Xiao, N.; Jiang, H.; Hu, G.; Chen, W. Application-Aware Big Data Deduplication in Cloud Environment. *IEEE Trans. Cloud Comput.* **2019**, *7*, 921–934. [CrossRef]
21. Li, X.; Li, J.; Huang, F. A secure cloud storage system supporting privacy-preserving fuzzy deduplication. *Soft Comput.* **2016**, *20*, 1437–1448. [CrossRef]
22. ElkanaEbinazer, S.; Savarimuthu, N.; Bhanu, S.M.S. ESKEA: Enhanced Symmetric Key Encryption Algorithm Based Secure Data Storage in Cloud Networks with Data Deduplication. *Wirel. Pers. Commun.* **2021**, *117*, 3309–3325. [CrossRef]
23. Li, S.; Xu, C.; Zhang, Y. CSED: Client-Side encrypted deduplication scheme based on proofs of ownership for cloud storage. *J. Inf. Secur. Appl.* **2019**, *46*, 250–258. [CrossRef]
24. Liang, X.; Yan, Z.; Deng, R.H. Game theoretical study on client-controlled cloud data deduplication. *Comput. Secur.* **2020**, *91*, 101730. [CrossRef]
25. Luo, S.; Zhang, G.; Wu, C.; Khan, S.U.; Li, K. Boafft: Distributed Deduplication for Big Data Storage in the Cloud. *IEEE Trans. Cloud Comput.* **2020**, *8*, 1199–1211. [CrossRef]
26. Wu, H.; Wang, C.; Fu, Y.; Sakr, S.; Lu, K.; Zhu, L. A differentiated caching mechanism to enable primary storage deduplication in clouds. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *29*, 1202–1216. [CrossRef]
27. Tian, G.; Ma, H.; Xie, Y.; Liu, Z. Randomized deduplication with ownership management and data sharing in cloud storage. *J. Inf. Secur. Appl.* **2020**, *51*, 102432. [CrossRef]
28. Fan, Y.; Lin, X.; Liang, W.; Tan, G.; Nanda, P. A secure privacy preserving deduplication scheme for cloud computing. *Future Gener. Comput. Syst.* **2019**, *101*, 127–135. [CrossRef]
29. Hovhannisyan, H.; Qi, W.; Lu, K.; Yang, R.; Wang, J. Whispers in the cloud storage: A novel cross-user deduplication-based covert channel design. *Peer-to-Peer Netw. Appl.* **2018**, *11*, 277–286. [CrossRef]

30. Joe, C.V.; Raj, J.S.; Smys, S. Mixed Mode Analytics Architecture for Data Deduplication in Wireless Personal Cloud Computing. *Wirel. Pers. Commun.* **2021**, *116*, 939–954. [CrossRef]
31. Zhang, G.; Yang, Z.; Xie, H.; Liu, W. A secure authorized deduplication scheme for cloud data based on blockchain. *Inf. Process. Manag.* **2021**, *58*, 102510. [CrossRef]
32. Yang, X.; Lu, R.; Shao, J.; Tang, X.; Ghorbani, A.A. Achieving Efficient and Privacy-Preserving Multi-Domain Big Data Deduplication in Cloud. *IEEE Trans. Serv. Comput.* **2021**, *14*, 1292–1305. [CrossRef]
33. Yu, C.M.; Gochhayat, S.P.; Conti, M.; Lu, C.S. Privacy Aware Data Deduplication for Side Channel in Cloud Storage. *IEEE Trans. Cloud Comput.* **2020**, *8*, 597–609. [CrossRef]
34. Zheng, X.; Zhou, Y.; Ye, Y.; Li, F. A cloud data deduplication scheme based on certificateless proxy re-encryption. *J. Syst. Archit.* **2020**, *102*, 101666. [CrossRef]
35. Keke, G.; Meikang, Q.; Xiaotong, S.; Hui, Z. Smart data deduplication for telehealth systems in heterogeneous cloud computing. *J. Commun. Inf. Netw.* **2016**, *1*, 93–104. [CrossRef]
36. Geeta, C.M.; Shreyas Raju, R.G.; Raghavendra, S.; Buyya, R.; Venugopal, K.R.; Iyengar, S.S.; Patnaik, L.M. SDVADC: Secure Deduplication and Virtual Auditing of Data in Cloud. *Procedia Comput. Sci.* **2020**, *171*, 2225–2234. [CrossRef]
37. Jeslin, J.G.; Kumar, P.M. Implementing an Efficient Data Deduplication Framework for Cloud Storage. *Indian J. Comput. Sci. Eng.* **2022**, *13*, 136–144. [CrossRef]