Frontiers in Health Informatics
ISSN-Online: 2676-7104

www.healthinformaticsjournal.com

2024; Vol 13: Issue 3

Open Access

Novelic Approach For Enhancing Storage Efficiency With Block Size Memory Deduplication

Jibin Joy[1], Dr. S. Devaraju[2]

[1.] jibinjoysamuel@gmail.com , Research Scholar (Ph.D.), Department of Computer Science, Sri Krishna Arts and Science College,Coimbatore, Tamil Nadu, India

[2.] devamcet@gmail.com,Senior Assistant Professor, VIT Bhopal University, Bhopal, Madhya Pradesh, India

Cite this paper as: Jibin Joy, S. Devaraju (2024) Novelic Approach For Enhancing Storage Efficiency With Block Size Memory Deduplication. *Frontiers in Health Informatics, 13* (3), 9714-9727

ABSTRACT:

Block size deduplication is a method employed in storage systems to enhance storage efficiency by recognizing and removing duplicate data blocks. It involves segmenting data into fixed-size blocks or chunks, typically ranging from kilobytes to several megabytes, and then identifying redundant blocks within the dataset. Instead of storing duplicate blocks multiple times, this technique stores only one instance of each unique block and maintains references to that block for subsequent duplicates. By eliminating redundant data, block size deduplication reduces storage overhead and enhances storage efficiency, particularly in environments where multiple copies of the same data are stored. This approach is commonly utilized in backup solutions, file systems, and storage appliances to optimize storage space and streamline data management. Moreover, the document explores practical implementations and real-life examples demonstrating the effectiveness of flat block size deduplication in enhancing scalability, cutting costs, and boosting system reliability within cloud settings. This overview lays the ground-work for a thorough examination of flat block size deduplication methods and their influence on cloud memory management, underscoring their significance and relevance in contemporary computing systems.

Keywords: *Content-Based Page Sharing (CBPS), Text Intersection Deduplication System (TIDS), Feedback Deduplication System (FDS), Virtual Machine (VM).*

1. INTRODUCTION

Cloud memory management plays a vital role in contemporary computing, especially in cloud-based systems where efficient resource allocation is crucial. A prominent technique that has gained considerable attention is flat block size deduplication, which entails identifying and removing duplicate data blocks within cloud memory. This process reduces storage overhead and enhances overall system performance. Introducing flat block size deduplication techniques represents a significant advancement in cloud memory management approaches. By examining data at the block level rather than the file level, these techniques can achieve higher deduplication rates and more detailed resource optimization. This document investigates the principles, obstacles, and advantages of implementing flat block size deduplication in cloud memory management, exploring how these techniques identify duplicate blocks, handle metadata, and maintain data integrity while optimizing storage capacity. Boundary deduplication is a data optimization method that targets and removes duplicate data within specific boundaries or segments, such as files or logical units, rather than at the block level. The process commences by identifying these boundaries, followed by breaking down the data into smaller fragments for comparison using hash functions. Matching hash values are used to pinpoint duplicate fragments, enabling the system to retain only one instance of each unique fragment and substitute duplicates with references or pointers. This strategy saves storage space, boosts data retrieval speed, and streamlines storage efficiency, making it a valuable approach in backup systems, cloud storage, and data archiving where reducing redundancy and enhancing storage utilization are key objectives. Block size deduplication is a method of data deduplication utilized in storage systems to enhance storage efficiency by identifying and removing duplicate data blocks. This technique involves segmenting data into fixed-size blocks or chunks, ranging from kilobytes to megabytes, and then identifying matching blocks within the dataset. Instead of storing redundant blocks multiple times, block size deduplication retains only one copy of each unique block and keeps references or pointers to that block for subsequent duplicates. By doing so, this approach minimizes storage overhead by eliminating redundant data and boosts storage efficiency, particularly in scenarios where multiple copies of the same data exist within the system. Block size deduplication is widely employed in backup solutions, file systems, and storage appliances to save storage space and enhance data management capabilities.

2. RELATED WORKS

Bosman et al. [1] brought attention to the potential risks associated with memory deduplication, highlighting how seemingly harmless features could be exploited by sophisticated attackers. Their study demonstrated that deduplication-based primitives could be manipulated to disclose sensitive information, posing a threat to system security. This specifies the importance of high security measures in deduplication strategies. Cui et al. [3] recently introduced UWare, a middleware aimed at enhancing data transfer efficiency through encrypted cloud storage. Their research aimed to address customer concerns regarding side-channel leakage while retaining the benefits of deduplication. They focused on leveraging similarity features and employing the Proof-of-Work (PoW) protocol to strike a balance between deduplication effectiveness and system throughput. Garg et al. [5] utilized graphics processing units (GPUs) to enhance deduplication efficiency. Their proposed method, Catalyst, offloaded the memory deduplication workload to GPU devices, enabling rapid identification of potentially duplicated pages. Results indicated that Catalyst significantly accelerated data sharing speed from memory compared to traditional methods. Kaur et al. [9] explored data deduplication in cloud computing and its potential to reduce storage costs, network traffic, and power consumption. They stressed the importance of innovative deduplication methods to enhance the efficiency of large-scale storage systems. Ning et al. [11] introduced group-based memory deduplication as a novel defense against covert channel attacks in multi-tenant clouds. Their approach provided group-level isolation, safeguarding virtual machines (VMs) from side-channel attacks by enabling group members to manage guest memory using shared secrets. Raoufi et al. [13] presented PageCmp, a memory-based page comparator that minimizes data sent during deduplication. By leveraging charge-sharing phenomena in DRAM's bulk bitwise operations, they significantly reduced bandwidth usage while maintaining acceptable execution time and power consumption levels.

3. RESEARCH METHODOLOGY

The initial phase of this study is dedicated to enhancing cloud memory management through the application of flat block size deduplication techniques. This involves tackling key challenges inherent in cloud memory management and exploring the impacts of deduplication methods specifically designed for flat block sizes. The objective is to provide a comprehensive analysis of how these deduplication techniques affect performance within cloud environments. To achieve this, the study begins with a thorough review of existing literature, aiming to expand on current knowledge, identify gaps, and highlight opportunities for innovation. This review will serve as a foundation, setting the stage for a deeper investigation into the practical implications and benefits of flat block size deduplication in cloud storage systems. The methodology of the research includes the deployment and testing of flat block size deduplication techniques within a cloud memory environment. This involves configuring the cloud storage system to incorporate deduplication algorithms that focus on blocks of uniform size. The performance of these techniques will be assessed based on several critical metrics, including storage savings, data access speeds, and computational load. By rigorously collecting and analyzing data, the study aims to quantify the benefits and limitations of flat block size deduplication. Key performance indicators such as the reduction in storage footprint, the efficiency of data retrieval operations, and the impact on system processing load will be closely monitored. This data will be collected through a series of controlled experiments designed to simulate real-world usage scenarios in cloud environments. The experiments will be structured to measure the direct effects of deduplication on storage utilization and system performance, providing a clear picture of its advantages and potential drawbacks. Furthermore, the research will also delve into the scalability of flat block size deduplication techniques. This involves examining how well these methods can be scaled up to handle larger datasets and more complex storage requirements typically encountered in cloud settings. By evaluating the scalability, the study aims to identify any bottlenecks or performance degradation that might arise as the system expands. The findings from this research are expected to offer valuable insights that can guide the development and optimization of cloud memory systems. The ultimate goal is to enhance the efficiency and management of cloud resources, thereby reducing costs and improving system reliability. By providing evidence-based recommendations, the study aims to contribute to the ongoing evolution of cloud storage technologies, making them more efficient, scalable, and effective in meeting the growing demands of modern computing environments. This research seeks to advance the understanding of flat block size deduplication techniques in cloud memory management, providing a detailed analysis of their performance impacts and practical benefits. Through systematic investigation and data-driven insights, the study aims to pave the way for more efficient and robust cloud storage solutions. Detailed Block Size Deduplication workflow architecture is described in **Figure1.**

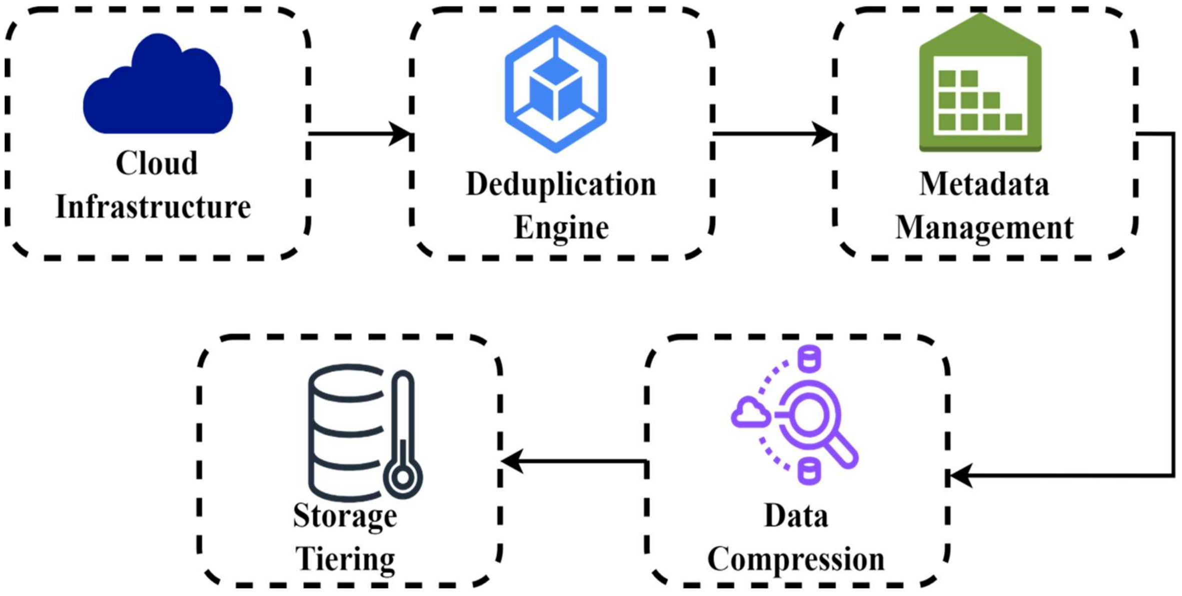

Figure 1: Block Size Deduplication workflow architecture

3.1 Text Intersection Deduplication System (TIDS)

In the Text Intersection and Deduplication System (TIDS), deduplication follows the processing phase, employing a two-tiered intersection approach to classify files within a text corpus as popular or unpopular. Initially, documents are grouped based on text features extracted through named entity recognition (NER) and term frequency (TF) analysis. The process involves four stages: root word detection, data cleansing, stemming, and threshold factor determination.

Tokenization breaks sentences into individual words for root word identification, while data cleansing removes unnecessary words. Stemming refines the text by eliminating morphemes, and a threshold factor identifies key text attributes. TF determines the significance of text components by their frequency within the document, although it may overlook critical elements. NER compensates for this by extracting specific entities such as people, locations, and objects mentioned in the texts. Documents retrieved through TF and NER form a collection of objects pointing to folders containing text files based on these attributes.

In the second tier, the intersection method classifies these objects as popular or unpopular. This classification facilitates the removal of redundant text corpora, optimizing storage space through deduplication. Popular files are retained across all cloud instances, while less popular ones are deleted from secondary servers and preserved only on the main server. This approach ensures efficient storage management and reduces redundancy in the cloud environment, enhancing overall storage efficiency and resource utilization.

3.2 Secure Deduplication in Cloud Storage with Efficient Multilevel Key Management (SDCM)

Hackers can exploit side-channel attacks like data deduplication and file identification to breach cloud-stored data security. To counter these threats, researchers suggest using secure deduplication and robust two-level key management. Secure deduplication enhances privacy by restricting access to authorized users. User authentication for storing and retrieving text files in two cloud services is verified using proof of ownership (POW), as proposed by Upadhyay, D. et al. (2021). The process begins with inline deduplication to identify duplicate files using the SHA-2 method. When a duplicate file is detected, file details and owner information are added to the POW list. A convergent key, generated based on file content, encrypts newly discovered files asymmetrically before storage in the cloud. The POW list is updated with user and file information, two-level keys (one derived from the convergent key and the other random), and an encrypted convergent key generated during the user session. Registered users receive the two-level keys via email to confirm ownership, ensuring secure text file retrieval. Only authorized users can access the services by providing the two-level keys: the random key and the encrypted convergent key stored in their email. The random key decrypts the convergent key, allowing for text file decryption and download. This method enhances data privacy and streamlines file retrieval, improving overall security in cloud storage. By securing deduplication and employing robust key management, cloud storage systems can better protect against unauthorized access and maintain data integrity.

3.3 Feedback Deduplication System (FDS)

This research suggests that maintaining multiple copies of data in the cloud servers can lead to increased requirements of storage. To address this issue and alleviate the strain on public cloud storage, Zhou, B., & Wen, J. T. (2016) propose the implementation of feedback-based deduplication. The feedback technique, akin to text intersection, aims to identify the popularity of files within an input text corpus based on user evaluations. The process involves two main steps. Initially, text documents are grouped based on textual characteristics extracted from a new text corpus. Techniques namely entity recognition and word frequency analysis are employed to identify text attributes, with a threshold factor determining the top text qualities. Named Entity Retrieval (NER) is utilized to extract named entities like people, places, and organizations from the input corpora. Feedback ratings are initiated post-TF and NER processes. Each text file starts with a zero access counter, which is updated with user interactions such as "view," "download," and "share." Upon completion of the user's session, the average access points for each file are calculated to establish a threshold value. Files exceeding this threshold in downloads are deemed popular, while those falling below are considered unpopular. Popular files are retained across all cloud instances, while unpopular files are removed from secondary servers. This feedback-based deduplication process effectively reduces unnecessary storage space by eliminating data redundancies.

3.4 Document Weight Deduplication System (DWDS)

DWDS employs a structured approach for model training using an input text corpus divided into two main components. Initially, documents are stored in the cloud via DropBox. Key features are identified using term frequency (TF) and named entity recognition (NER) techniques, as highlighted by Pugazhendi, E. et al. (2021). Next, the popularity of new text documents is determined by analyzing these text characteristics. Duplicate documents are eliminated after feature identification to facilitate efficient retrieval and analysis.

The research utilizes these identified features to locate additional papers and assess their frequency within the retrieved document set. Document weight, calculated using the average threshold factor, assesses the popularity of each paper. Files with a document weight above the threshold factor are classified as popular, while those below are deemed unpopular. Popular text files are retained across all cloud instances, while unpopular content is removed from secondary servers.

By employing weight-based deduplication, DWDS reduces data redundancies, reclaiming unnecessary storage space efficiently. This approach ensures that only valuable, high-demand documents occupy cloud storage, optimizing resource utilization. The weight-based method not only improves storage efficiency but also enhances document retrieval processes by maintaining a streamlined, relevant dataset. Through this system, DWDS effectively manages cloud-stored documents, balancing the need for accessibility with efficient storage management.

4. ALGORITHM

Upon acceptance of the output (m,n), the Agents Server provides the secret key pair as input. This input includes the file FS and the agent's secret keys (m,n), resulting in the output of the file label and index. The process initiates with the user utilizing EK_sym alongside the file labels w_2 and FS to produce the cipher text CF.The input consists of values such as w_1, w_2, name, FS, and k_fpk, representing file identifiers and labels. This process yields a file tag and the private key pk (EK_sym) for the initial user. Subsequently, the first user forms an authenticator set by utilizing the FPK key k_fpk and the cipher text CF. Upon reception of the input file FS and the secret key pair (m, n) from the agent, the system produces w_1 and w_2. Subsequently, the user generates the symmetric encryption key EK_sym using the file labels and FS, incorporating it into the CF. The user then computes their private key pk with inputs such as file label w_2, file FS, and file identifier name. Furthermore, the user requesting the file employs the POW method on the cloud to authenticate ownership. For every auditing challenge, both the CF and the corresponding authentication set u are necessary. Through this process, an auditing proof is generated to ensure that the entire cipher text CF is stored in the cloud. The auditing challenge is utilized as input for evidence verification. If the proof is valid, the system outputs "true"; otherwise, it outputs "false." This verification is achieved by examining the challenge and the proof.

At this stage, the reliable Key Distribution Service (KDS) establishes the essential public parameters for the system, along with the public key (n) for the cloud provider and the private key (m) for the users (PS). Key generation in cloud computing is a fundamental concept that depends on the specific characteristics of the system. KDS plays a crucial role in converting encrypted text and transmitting confidential information. Key selection is performed by the KDS to facilitate the conversion of encrypted text.

Assuming F1 represents an order p multiplicative group, the KDS unit defines four hash functions to select keys for public and private parameters, as follows:

- HH1: $\{0,1\}^* \rightarrow F1$
- HH2: $F1 \rightarrow \{0,1\}^p$ (p-Security Parameter)
- hh1: $\{0,1\}^* \rightarrow \{0,1\}^p$
- hh2: $\{0,1\}^* \rightarrow W_p^*$

These four hash functions are determined using the pseudo-random function f: $\{0,1\}^* \times K_{prf} \rightarrow W_p^*$ and the security parameter. A private key ($m \in W_p^*$) is randomly selected, and the public key (n) is calculated as r^m.

Through the utilization of the security parameter and the pseudo-random function f: $\{0,1\}^* \times K_{prf} \rightarrow W_p^*$, these hash functions are established, leading to the random selection of a private key ($m \in W_p^*$) and the computation of the public key ($n = r^m$).

During data uploads to the cloud, processes such as inter- and intra-deduplication, along with intra-tag creation, are carried out. Each user in every domain (D=1, 2, 3,..., n) must generate a unique intra-tag to enable data deduplication. Intra-deduplication is performed by the first agent when a user uploads the file FS to check for duplicates within the same domain. Once intra-deduplication is conducted within domains and duplicates are identified, users can utilize the original private key to upload files with the same name in the same domain. The data upload phase comprises data encryption/key recovery intra-deduplication, intra-tag creation, and inter-deduplication.

Before uploading data FS, user U in domain Di (where I=1, 2,..., n) adds an intra-tag to prevent data duplication. Intra-deduplication is performed by agent Ai to detect duplication within the domain Di. If no duplicates are found, cross-domain inter-deduplication is required by the Central Processing System (CPS). Upon detecting duplication, user U acquires the convergent key generated by the original uploader.

When the first user, U_1, in domain D_i^'s wishes to upload file FS, they select a random integer r_m and use the symmetric key private key ($pk=K_{sym}$, MAC_{sym}) to create an intra-file tag as follows: $\alpha_{d_i} = (d_i^{hh_1} g^{r_m})$ ----------------- (4.5)

The initial user uploads a file to the agent using the parameters $length_{FS}$ and α_{d_i}. The file's size is denoted by $Length_{FS}$. To check for the existence of a file copy (FS), a comparison is made between a_{d_i} and the previously recorded tag value from D_i.

4.1 Algorithm Setup

The main task of the Agent is to generate file indexes (w_1) and file labels (w_2) for users in each domain (D_1, D_2, D_3, etc.). The file index is responsible for identifying duplicates stored in the cloud. Authentication keys have been created and encrypted based on the file label. Each domain accommodates both one-time users and returning users, with the first user utilizing an agent to upload the file index to the cloud. Upon data delivery to the cloud through an agent, a check is performed to determine if a previous file with the same file index exists, with the agent maintaining a copy of the file index table.

When there is no cloud storage, the first user encrypts the file FS using the symmetric encryption key (EK_sym). To begin, a blindfolded hash function R' is computed using a random integer c ∈ W_p*. The hash value calculation is defined by Equation (4.6): R^' = HH_1(FS) r^c -------------------- (4.6)

Upon receiving the hash value (R') from the original user, the agent uses a private key (m) to compute the value H' = R^'m, which is then communicated back to the original user. The initial user calculates δ = H^' n^(-c) for the Agent public key n, and proceeds to generate (w_1) and (w_2):

- ω_1 = HH_2(δ‖2┤) ------------- file index for duplicate data check
- ω_2 = HH_2(δ‖2┤) ------------- file label for symmetric encryption keys and MAC keys (EK_sym, K_sym, MAC_sym)

After creating the file index (w_1), it is transmitted to the agent and stored. If a file already exists with the same index, a notification is sent to the original user, and no new file is uploaded to the cloud. The file index is uploaded to the cloud if there are no similar entries maintained by the agent. Cloud computing verifies the file index value. To encrypt the file as C_FS = Enc(FS, EK_sym), the first user determines the symmetric key value EK_sym, calculated as hh_1 multiplied by w_2 FS, assuming the cloud retains the file index value. The encrypted text consists of several smaller blocks.

4.2 Authentication Algorithm

The first user generates a file tag, authenticator set u, private key pk=(K_sym, MAC_sym), and encryption key file, all of which are stored in the cloud. The original user has saved the private key and symmetric key pair (pk=(K_sym, MAC_sym)). During data retrieval, the file is decrypted using EK Sym. A random value (qa) and a file tag (prf) are produced using the private key (pk). The Proof Verification method is completed after validating the file tag ε, decrypting the encrypted content, recovering the PRF key kprf, and generating a random number ΰ.

K_sym=hh1(identifier‖ω_2|(|FS|)|1┤) and K_MAC=hh1(identifier‖ω_2|(|FS|)|1┤) are generated by the original user, where the identifier denotes the file's name and w_2 represents the file label. The private key is represented as pk=(K_sym, K_mac). Random values ω and K_prf are randomly generated with ω ∈ W_p^* and K_prf ∈ W_p^*, respectively.

The file tag is defined as α=α_0‖MAC┤ *(K_MAC)(α_0), where α_0=n‖Enc(K_prf‖φ,┤K_sym)┤. The initial user computes the authenticator I for each block c_i∈W_p^* (i∈[1,n]) of the ciphertext as follows:* ℵ_*i=f*(K_prf)(i)+ωc_i. The set of authenticators is denoted by φ={ℵ_i}1≤i≤n. The initial user uploads {C_FS} and the file tag to the cloud before deleting the messages from local storage and keys. The cloud then creates a link to the file F for the first user.

4.3 Proof Generation

When a user requests proof from the cloud, they send an auditing challenge to the cloud. Upon receiving the challenge, the cloud provides the user with auditing proof. The user randomly selects an i-elements subset I (I⊆[1,n]) and assigns a random value, v_i∈W_p^*, to each i. This subset I and the corresponding values {v_i}_i∈I form the auditing challenge sent to the cloud.

Upon receiving the auditing challenge from the user, the cloud calculates a linear combination of encrypted data blocks: $\Psi=\sum_{l\in I} v_i q_i$ ------------------- *(4.8) The authenticator combination set η is computed as:* $\eta=\sum_{l\in I} v_i \eta_i$ After receiving the auditing evidence and the file tag, the user undergoes proof verification using the symmetric key and MAC key extracted from the file tag. The user evaluates the verification equation to determine its validity: $\eta=\sum_{l\in I} v_i f(k\_prf)(i) +\Psi\varphi$ ------------------ (4.9)

If the data exists in the user's entire file, the equation is correct, demonstrating that the user's files are securely stored in the cloud.

```
Algorithm : Flat Block Size Deduplication
Input:
    Data to be deduplicated (Data)
Steps:
1.  Divide Data into fixed-size blocks (Block_Size).
2.  Compute a hash (Hash) for each block using a cryptographic hash function.
$\sum_{l\in[1,n]} \aleph_i = \sum_{l\in[1,n]} f_{k_{prf}}(i) + \varphi \sum_{i\in[1,n]} c_i$
3.  Compare the Hash of each block with existing hashes in the system.
4.  If Hash already exists:
    o   Create a reference or pointer to the existing block.
    o   Skip storing the duplicate block.
5.  If Hash is new:
    o   Store the block along with its Hash.
Output:
    Deduplicated data (Deduped_Data)
```

5. RESULTS AND DISCUSSION

Assessing the effectiveness and efficiency of systems through performance evaluation is crucial for understanding how well a system or component meets its objectives. This involves systematically measuring and analyzing various metrics and parameters to provide a comprehensive picture of system performance. In the context of cloud memory management utilizing flat block size deduplication techniques, performance evaluation plays a pivotal role. Cloud environments demand efficient resource utilization and optimal performance due to their large-scale and dynamic nature. Flat block size deduplication techniques, which segment data into fixed-size blocks to identify and eliminate duplicates, offer potential benefits in terms of storage efficiency and cost savings. However, the actual impact of these techniques must be thoroughly assessed to understand their true value and effectiveness.

Key performance metrics such as storage space savings, data access times, and computational overhead are critical in this evaluation process. Storage space savings indicate how effectively the deduplication technique reduces the data footprint, which directly correlates with cost savings in cloud storage. Data access times reflect the efficiency of data retrieval operations, influencing overall system responsiveness and user experience. Computational overhead measures the additional processing required to perform deduplication, impacting the system's overall performance and scalability.

By carefully analyzing these metrics, researchers can gain valuable insights into how flat block size deduplication influences the efficiency and resource utilization in cloud environments. This evaluation helps identify any trade-offs between storage savings and performance impacts, guiding the optimization of deduplication techniques for better cloud memory management. Ultimately, performance evaluation ensures that deduplication methods contribute positively to the overall efficiency of cloud storage systems, supporting informed decision-making and continuous improvement.

Table 1: Encryption time comparison table

File Size (KB)	**Encryption Time**		
	Block Size Deduplication	**Boundary Deduplication**	**Flat Block Size Deduplication**
1	0.6	0.7	0.3
2	0.21	0.33	1.11
4	1.57	1.22	1.01
8	2.14	1.4	0.92

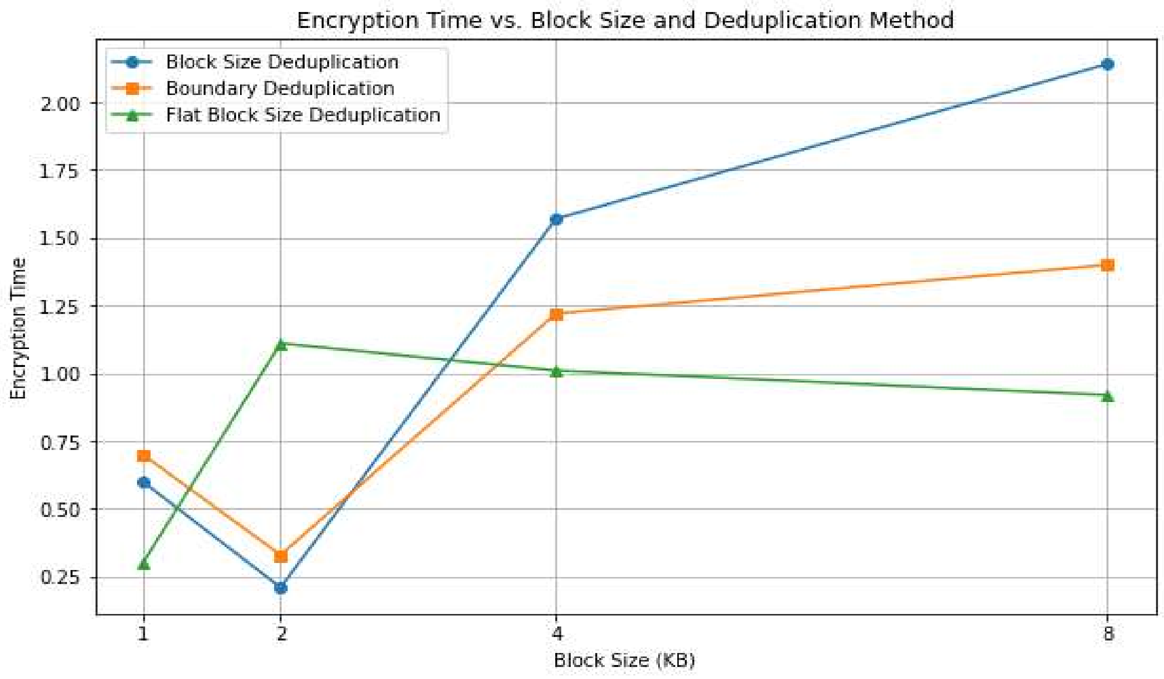

Figure 2: Encryption time comparison chart

The data presented in **Table 1 and Figure 2** highlights the encryption time (in seconds) for various file sizes (in kilobytes) using three deduplication methods: Block Size Deduplication, Boundary Deduplication, and Flat Block Size Deduplication. For a 1KB file, Block Size Deduplication took 0.6 seconds, Boundary Deduplication took 0.7 seconds, and Flat Block Size Deduplication took 0.3 seconds. As the file size increased to 2KB, Block Size Deduplication's encryption time decreased significantly to 0.21 seconds, while Boundary Deduplication's time increased to 0.33 seconds, and Flat Block Size Deduplication's time increased notably to 1.11 seconds.

This pattern continued with larger file sizes, indicating that Block Size Deduplication generally had the shortest encryption times. At a 4KB file size, Boundary Deduplication briefly had a longer encryption time than Block Size Deduplication before the times aligned again for larger files. Overall, the data suggests that Block Size Deduplication consistently delivers the fastest encryption times across various file sizes compared to the other deduplication techniques.

Table 2: Decryption time comparison table

File Size (KB)	**Decryption Time**		
	Block Size Deduplication	**Boundary Deduplication**	**Flat Block Size Deduplication**
1	0.1251	0.0203	0.0148
2	0.0398	0.0288	0.0244
4	0.0514	0.0423	0.0229
8	0.0923	0.0829	0.0560

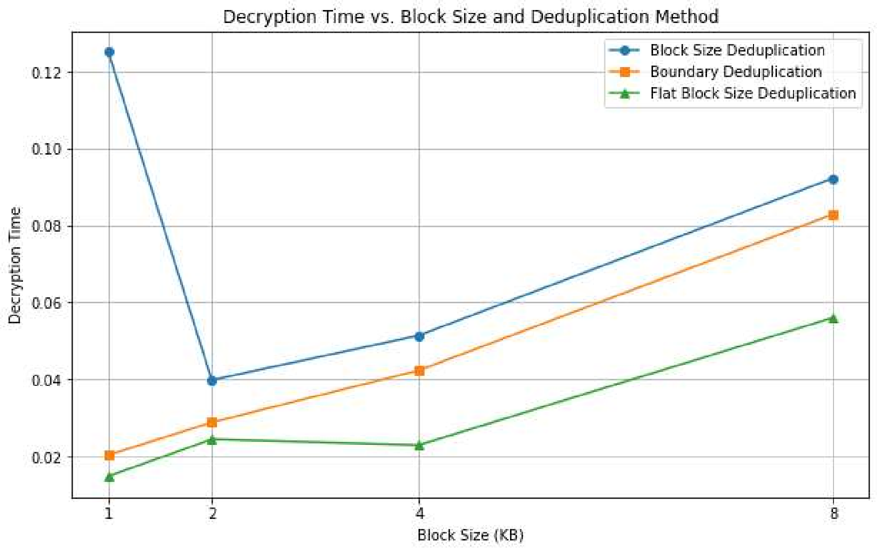

Figure 3: Decryption time comparison chart

The data provided in **Table 2 and Figure 3** showcases the decryption time (in seconds) for various file sizes (in kilobytes) using three different deduplication techniques: Block Size Deduplication, Boundary Deduplication, and Flat Block Size Deduplication. For a 1KB file size, Block Size Deduplication required 0.1251 seconds for decryption, Boundary Deduplication took 0.0203 seconds, and Flat Block Size

Deduplication took 0.0148 seconds. With an increase in file size to 2KB, all methods saw a notable decrease in decryption times, with Block Size Deduplication at 0.0398 seconds, Boundary Deduplication at 0.0288 seconds, and Flat Block Size Deduplication at 0.0244 seconds. This pattern persisted, with decryption times generally decreasing as file sizes increased. Notably, Block Size Deduplication consistently showed higher decryption times compared to the other methods, especially noticeable at larger file sizes where the differences became more pronounced. Conversely, Flat Block Size Deduplication consistently exhibited the lowest decryption times, highlighting its efficiency in decrypting data across various file sizes when compared to other deduplication techniques.

6. CONCLUSION AND FUTURE SCOPE

This paper explores cloud memory management with a focus on implementing flat block size deduplication techniques. The primary aim is to streamline data storage in cloud environments by identifying and eliminating redundant data blocks. By adopting flat block size deduplication, this approach seeks to optimize memory utilization, reduce storage costs, and enhance overall performance of cloud-based systems. The paper examines the specific methodologies and strategies involved, highlighting both their benefits and potential limitations. It provides valuable insights into how flat block size deduplication can significantly improve the efficiency of cloud memory management processes. Looking ahead, future research could investigate more advanced deduplication techniques tailored to various data formats, integrate quantum-resistant cryptographic protocols to enhance data security, and develop dynamic access control systems for flexible data sharing. Additionally, incorporating blockchain technology could improve data governance, while AI-driven optimization and automation could streamline cloud operations. Addressing ethical and regulatory issues in cloud computing, as well as focusing on hybrid and multi-cloud frameworks, could enhance interoperability and effective resource management across diverse cloud landscapes. Embracing these future research directions has the potential to optimize cloud infrastructures, strengthen security protocols, refine data governance strategies, and promote responsible and efficient cloud computing practices

ABBREVIATIONS

CBPS - Content-Based Page Sharing

TIDS - Text Intersection Deduplication System

FDS - Feedback Deduplication System

VM- Virtual Machine (VM)

DECLARATION OF COMPETING INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

DATA AVAILABILITY

No data was used for the research described in the article. The implementation is mainly based on the hardware dependency so that execution of the work is mainly dependent on the time and data complexity.

AUTHOR CONTRIBUTIONS

Mr. Jibin Joy wrote the entire article and responsible for data pre-processing. Dr. Devaraju S read and approved the final manuscript.

AUTHORS' INFORMATION

Mr. Jibin Joy is a Research Scholar(Ph.D) in Sri Krishna Arts and Science College Coimbatore,India, His research area is cloud computing and published various Scopus and research papers.

Dr. S. Devaraju is Senior Assistant Professor, VIT Bhopal University, India. He has more than 15 years of teaching experience with various Scopus and WoS research papers and also have association with various funded projects.

FUNDING

None

ETHICS APPROVAL AND CONSENT TO PARTICIPATE

This article does not contain any studies with human participants or animals performed by any of the authors.

COMPETING INTERESTS

The authors declare that they have no conflict of interest

AUTHOR DETAILS

Jibin Joy, Research Scholar(Ph.D), Sri Krishna Arts and Science College, Coimbatore,India

Dr. S. Devaraju, Senior Assistant Professor, VIT Bhopal University, Bhopal, Madhya Pradesh, India

REFERENCES

1. Bosman, E., Razavi, K., Bos, H., & Giuffrida, C. (2016). Dedup Est Machina: Memory Deduplication as an Advanced Exploitation Vector. 2016 IEEE Symposium on Security and Privacy (SP). doi:10.1109/sp.2016.63
2. Cui, H., Duan, H., Qin, Z., Wang, C., & Zhou, Y. (2019). SPEED: Accelerating Enclave Applications Via Secure Deduplication. 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). doi:10.1109/icdcs.2019.00110
3. Cui, H., Wang, C., Hua, Y., Du, Y., & Yuan, X. (2018). A Bandwidth-Efficient Middleware for Encrypted Deduplication. 2018 IEEE Conference on Dependable and Secure Computing (DSC). doi:10.1109/desec.2018.8625127
4. Fu, Y., Xiao, N., Jiang, H., Hu, G., & Chen, W. (2017). Application-Aware Big Data Deduplication in Cloud Environment. IEEE Transactions on Cloud Computing, 1–1. doi:10.1109/tcc.2017.2710043
5. Garg, A., Mishra, D., & Kulkarni, P. (2017). Catalyst. Proceedings of the 13th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments - VEE '17. doi:10.1145/3050748.3050760
6. Huang, H., Yan, C., Liu, B., & Chen, L. (2017). A survey of memory deduplication approaches for intelligent urban computing. Machine Vision and Applications, 28(7), 705–714. doi:10.1007/s00138-017-0834-6
7. Jagadeeswari, N., & Mohanraj, V. (2017). A survey on memory deduplication employed in cloud computing. 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing

(ICECDS). doi:10.1109/icecds.2017.8390074
8. Jia, S., Wu, C., & Li, J. (2017). Loc-K: A Spatial Locality-Based Memory Deduplication Scheme with Prediction on K-Step Locations. 2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS). doi:10.1109/icpads.2017.00049
9. Kaur, R., Chana, I., & Bhattacharya, J. (2017). Data deduplication techniques for efficient cloud storage management: a systematic review. The Journal of Supercomputing, 74(5), 2035–2085. doi:10.1007/s11227-017-2210-8
10. Kim, D., Song, S., & Choi, B.-Y. (2016). Existing Deduplication Techniques. Data Deduplication for Data Optimization for Storage and Network Systems, 23–76. doi:10.1007/978-3-319-42280-0_2
11. Ning, F., Zhu, M., You, R., Shi, G., & Meng, D. (2016). Group-Based Memory Deduplication against Covert Channel Attacks in Virtualized Environments. 2016 IEEE Trustcom/BigDataSE/ISPA. doi:10.1109/trustcom.2016.0063
12. Niu, Y., Liu, W., Xiang, F., & Wang, L. (2015). Fast Memory Deduplication of Disk Cache Pages in Virtual Environments. 2015 IEEE Fifth International Conference on Big Data and Cloud Computing. doi:10.1109/bdcloud.2015.50
13. Raoufi, M., Deng, Q., Zhang, Y., & Yang, J. (2019). PageCmp: Bandwidth Efficient Page Deduplication through In-memory Page Comparison. 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). doi:10.1109/isvlsi.2019.00023
14. Difference Engine: Harnessing Memory Redundancy in Virtual Machines by D. Gupta, S. Lee, M. Vrable, S. Savage, A. C. Snoeren, G. Varghese, G. M. Voelker, and A. Vahdat
15. Data Deduplication with Encrypted Big Data Management in Cloud Computing, Nahlah Aslam K.P., Dr. Swaraj K.P. (2019- IEEE Xplore ISBN: 978-1-7281-1261-9)
16. J.Joy Ensuring Data Integrity And Security In Diverse Cloud Environments To Prevent Duplicacy. November 2023, Tuijin Jishu/Journal of Propulsion Technology 44(4):1001-4055, DOI:10.52783/tjjpt.v44.i4.1798
17. Cui, H., Wang, C., Hua, Y., Du, Y., & Yuan, X. (2018). A Bandwidth-Efficient Middleware for Encrypted Deduplication. 2018 IEEE Conference on Dependable and Secure Computing (DSC). doi:10.1109/desec.2018.8625127
18. Fu, Y., Xiao, N., Jiang, H., Hu, G., & Chen, W. (2017). Application-Aware Big Data Deduplication in Cloud Environment. IEEE Transactions on Cloud Computing, 1–1. doi:10.1109/tcc.2017.2710043
19. An Effective Data Storage Model for Cloud Databases using Temporal Data De-duplication, S. Muthurajkumar, M. Vijayalakshmi, A. Kannan (2016 IEEE Eighth International Conference on Advanced Computing (ICoAC), 978-1-5090-5888-4/16/$31.00@2016 IEEE)
20. J.Joy, Avoidance Of Duplicacy And Compelling Cloud Security In Different Cloud Situations, November 2023, International Journal Of Creative Research Thoughts 11(11):2320-2882, DOI: 10.56975/1k890v13
21. Enhanced Storage Optimization System (SoS) for IaaS Cloud StorageS. Muthurajkumar, M. Augustus Devarajan A, SudalaiMuthu T (2020, Proceedings of the Fourth International Conference on Inventive Systems and Control (ICISC 2020) .IEEE Xplore Part Number: CFP20J06-ART; ISBN: 978-1-7281- 2813-9)
22. C. Bo, Z. F. Li, and W. Can, \Research on chunking algorithm of data deduplication,"American Journalof Engineering and Technology Research, vol. 11, no. 9, pp. 1353 {1358, 2011.
23. E. Kave and H. K. Tang, \A framework for analyzing and improving content based chunking algorithms," tech. rep., International Enterprise Technologies Laboratory, HP Laboratories Palo Alto, Sept 2005.
24. A. Muthitacharoen, B. Chen, and D. Mazieres, \A low-bandwidth network le system," in Proceedings of the eighteenth ACM symposium on Operating systems principles, SOSP '01, (New York, NY, USA), pp. 174{187, ACM, 2001.

25. C. Wang, Z. guang Qin, J. Peng, and J. Wang, \A novel encryption scheme for data deduplication system," in Communications, Circuits and Systems (ICCCAS), 2010
26. International Conference on, pp. 265 {269, july 2010. [20] S. Parez, \Bitcasa: Innite cloud storage." http://techcrunch.com/2011/09/18/bitcasa- explains-encryption/, Sept 2011. Online.
27. J. Dinerstein, S. Dinerstein, P. Egbert, and S. Clyde, "Learning-based fusion for data deduplication," in Machine Learning and Applications, 2008. ICMLA '08. Seventh International Conference on, pp. 66 – 71, dec. 2008.
28. B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file system," in Proceedings of the 6th USENIX Conference on File and Storage Technologies, FAST'08, (Berkeley, CA, USA), pp. 18:1–18:14, USENIX Asso- ciation, 2008.
29. M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezise, and P. Camble, "Sparse indexing: large scale, inline deduplication using sampling and locality," in Proccedings of the 7th conference on File and storage technologies, FAST '09, (Berkeley, CA, USA), pp. 111–123, USENIX Association, 2009.
30. Jibin Joy, & Dr. S. Devaraju. (2024). Securing Cloud Memory Through Efficient Deduplication Using Ecc Algorithm. Educational Administration: Theory and Practice, 30(5), 9421–9429. Retrieved from https://kuey.net/index.php/kuey/article/view/4583
31. Saharan, S., Somani, G., Gupta, G., Verma, R., Gaur, M. S., & Buyya, R. (2020). QuickDedup: Efficient VM deduplication in cloud computing environments. Journal of Parallel and Distributed Computing. doi:10.1016/j.jpdc.2020.01.002
32. Vano-Garcia, F., & Marco-Gisbert, H. (2018). How Kernel Randomization is Canceling Memory Deduplication in Cloud Computing Systems. 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA). doi:10.1109/nca.2018.8548338
33. Jibin Joy, S. Devaraju (2024) Ensuring Secure Cloud Data Sharing Through Blockchain-Based Auditing For Authentication And Fuzzy Identity-Based Proxy Re-Encryption For Access Control. Library Progress International, 44(1s), 134-146.