The Scientific Temper (2024) Vol. 15 (3): 2507-2514
E-ISSN: 2231-6396, ISSN: 0976-8653
Doi: 10.58414/SCIENTIFICTEMPER.2024.15.3.16
https://scientifictemper.com/

RESEARCH ARTICLE

BTEDD: Block-level tokens for efficient data deduplication in public cloud infrastructures

Sabeerath K.*, Manikandasaran S. Sundaram

Abstract

In today's digital era, the exponential growth of data necessitates effective storage and management solutions. The cloud has vast storage possibilities to store huge amounts of data. Public access to the cloud leads to duplicate copies of data stored in the storage. Maintaining a single copy of data in the cloud is most important for efficient data storage management. This paper introduces a groundbreaking strategy for improving the efficacy of cloud storage through innovative data deduplication techniques at the block levels. The block-level duplication verification efficiently identifies the duplicate data in the storage. It helps to protect the duplicate storage in the cloud data storage infrastructure. The block-level deduplication technique uses variable-length blocks based on the duplicate content of the block. Initially, A file is divided into a number of blocks with a size of 5 kb. According to the proposed method, if any block is partially matched with a block already stored in the cloud, then that block is further divided into smaller blocks based on the matching percentage. The smaller blocks help to deduplicate the data more effectively. The work is implemented in a live cloud setting with a C# application hosted on MyASP.NET. The proposed methodology's effectiveness is validated against existing deduplication techniques. The results reveal a marked improvement in storage utilization and data management, affirming the potential of the approach to revolutionize cloud storage efficiency.

Keywords: Cloud storage, Data deduplication techniques, Block-level deduplication, Cloud data security, Data storage management.

Introduction

Cloud storage stands as a pivotal solution for handling vast volumes of data, enabling seamless storage, access, and retrieval across the globe. A critical component of securing data within this virtual environment is encryption, which safeguards data privacy by transforming readable data into encoded information accessible only through a decryption key (H. Shin *et al*., 2018). However, a notable challenge arises when individual users encrypt identical files using distinct keys. This scenario leads to redundant storage of the same file in various encrypted forms, inadvertently consuming excessive storage space bandwidth and generating unnecessary network traffic. This redundancy not only impacts storage efficiency but also complicates data management and access in the cloud (Yongkai Fan A B *et al*., 2019).

Deduplication emerges as a strategic approach to combat these inefficiencies, aiming to eliminate redundant data copies (P M A Kumar *et al*., 2022). This process is straightforward with unencrypted data but becomes complex when dealing with encrypted files, as encryption masks the data's true nature, making duplicates indistinguishable (Nagappan M *et al*., 2023).

Convergent encryption offers an innovative solution to address this. This method generates the encryption key from the data itself using a cryptographic hash function (V. Roshini *et al*., 2023). Consequently, identical files produce identical keys and, thus, identical encrypted outputs, regardless of who encrypts the data. This uniformity allows for effective deduplication, significantly enhancing storage and bandwidth efficiency (Sahaya *et al*., 2023).

Navigating the challenges of encryption and deduplication within cloud storage requires a delicate balance between efficiency and security. Techniques like convergent encryption pave the way for more effective data

PG and Research Department of Computer Science, Adaikalamatha College, Vallam, Affiliated to Bharathidasan University, Thanjavur, Tamil Nadu, India.

***Corresponding Author:** Sabeerath K., PG and Research Department of Computer Science, Adaikalamatha College, Vallam, Affiliated to Bharathidasan University, Thanjavur, Tamil Nadu, India., E-Mail: sabeerathk@gmail.com

How to cite this article: Sabeerath, K., Sundaram, M. S. (2024). BTEDD: Block-level tokens for efficient data deduplication in public cloud infrastructures. The Scientific Temper, **15**(3):2507-2514.
Doi: 10.58414/SCIENTIFICTEMPER.2024.15.3.16

Source of support: Nil

Conflict of interest: None.

Received: 27/07/2024 **Accepted:** 07/08/2024 **Published**: 10/09/2024

management, but they must be carefully implemented to safeguard against potential security threats (Sharik *et al*., 2023; Ulthana *et al*., 2023). Ultimately, achieving optimal cloud storage solutions involves continuous innovation and strategic integration of encryption, deduplication, and security measures (Komal *et al*., 2023). This paper proposes an efficient block-level deduplication along with a framework designed to prevent duplicate copies of data in cloud storage.

Related Work

This section discusses related studies in the field. Zhang S *et al*. (Zhang *et al*., 2020) observed large redundant data in picture layers while retrieving photos from the registry, which slows down the launch of non-local containers. In response to this issue, they suggested implementing block-level deduplication, a container deployment architecture that utilizes block-level deduplication. BED holds picture layers as data blocks and a fingerprint list, which is derived from these blocks. When extracting an image, BED obtains the lists of fingerprints, removes any duplicates, and exclusively collects data blocks from the registry that are not stored locally. Subsequently, the image layers for the container are reconstructed by utilizing both local and pulled blocks. Experimental results demonstrate that binary execution diagram (BED) achieves an average reduction of 35% in image pulling time and a decrease of about 48% in data transmission when compared to Docker.

The growing storage needs in cloud data centers, caused by the IT industry's expansion, were the focus of a study by G. Ali *et al*. (G. Ali *et al*., 2020). The study highlighted data redundancy as a major contributor to storage shortages and proposed a block-level data deduplication technique to tackle this issue. Through a comparative analysis between this technology and the commonly used file-level deduplication, the study concluded that block-level deduplication improves storage efficiency by 5%. This method partitions files into smaller segments and produces distinct hash values for each segment, leading to a greater reduction in storage requirements compared to deduplication at the file level. Despite the increased processing time resulting from more computations, tests conducted on a local dataset demonstrated superior efficiency in terms of storage reduction. The authors also addressed future endeavors to maximize these findings and incorporate server-side deduplication to boost data security and efficiency further.

The real-time block-level dual mode data deduplication (RBDD) approach, introduced by Jayashree K *et al*. (Jayashree K *et al*., 2022), combines data deduplication with block-level protection to counteract harmful actions. The solution employs key metrics and access behavioral measure (ABM) to enforce access limits at the block level. Additionally, it preserves a page matrix to record the structure of various pages, encompassing details regarding blocks, keys, and encryption methods. The dual-mode solution involves the re-encryption of user-encrypted data using a system key that is regularly updated and modified during different sessions to counteract various threats effectively. Deduplication is accomplished by validating data at the block level using the block level relational measure (BLRM). This approach improves the efficiency of data deduplication, public auditing, privacy preservation, and access limitation while promoting the advancement of quality of service (QOS).

An innovative method specifically designed for backup applications focused on object storage was introduced by Jackowski *et al*. (Jackowski *et al*., 2023). This work made two primary contributions. Firstly, it compared the characteristics of object storage interfaces with the usage patterns observed in 686 commercial deployments of backup appliances. This investigation identified unique concerns that need resolution to achieve optimal performance in backup systems utilizing block-level deduplication, with a specific focus on metadata management. Additionally, distributed data structures and methods were introduced to manage object metadata within these systems. The technology's implementation served as a storage layer for the HYDRAstor backup system. This method achieved a write throughput that was 1.8 to 3.93 times higher than object storage without in-line deduplication and achieved 5.26 to 11.34 times higher object put operation throughput per unit of time compared to object storage implemented on a cutting-edge file-based backup system.

The hash-indexing block-based deduplication (HIBD) technique was proposed by Viji D *et al*. (Viji D *et al*., 2023) to enhance storage efficiency in cloud environments by minimizing data redundancy. This approach, combined with segmented bind linkage (SBL) techniques, detects and removes redundant data. The system uses sparse augmentation techniques to preprocess files, divide them into blocks, and perform hash indexing. Document similarity evaluation and clustering of related files are achieved through semantic content source deduplication (SCSD) and distance vector weightage correlation (DVWC). Experimental results demonstrate that HIBD significantly improves storage efficiency, surpassing existing deduplication approaches in terms of precision and recall rates. This methodology reduces storage-related expenses and enhances data administration in remote cloud environments, showing exceptional precision and efficiency in execution time.

A technique to augment data protection and optimize performance in cloud storage systems was proposed by Pavithra M *et al*. (Pavithra *et al*., 2024). This approach integrates Boneh-Goh-Nissim bilinear attribute-based encryption with optimal cache oblivious (BGNBA-OCO) algorithms to offer comprehensive access control and secure deduplication. Before uploading patient files to the cloud, public key encryption is employed to safeguard

data confidentiality. The optimal cache oblivious algorithm uses a random matching coefficient to identify duplicate data, minimizing required storage space. Additionally, an additive linear secret sharing matrix prevents unwanted data alterations, thereby boosting data integrity. Empirical studies demonstrate that the BGNBA-OCO approach surpasses current methods in communication overhead, computation overhead, data confidentiality rate, and data integrity rate, making it a highly efficient alternative for ensuring secure cloud storage.

Methodology

The outlined framework introduces an efficient method for data deduplication in cloud storage, employing a nuanced token generation strategy that adapts to different data storage scenarios. This method leverages the concept of tokenization to assess data uniqueness and redundancy before storage, ensuring that cloud resources are utilized optimally. The framework consists of several key components and cloud services designed to streamline the deduplication process:

Token as a Service (TKNaaS)

Generates unique tokens representing the data content, both at the file and block levels.

Deduplication System as a Service (DSaaS)

This system utilizes generated tokens to check for data redundancy in cloud storage.

Convergent Encryption as a Service (CEaaS)

Ensures that the data is securely encrypted before being stored, using keys derived in a way that supports deduplication.

Cloud Storage as a Service (CSaaS)

It provides virtual space to store the deduplicate data. Figure 1 shows the components involved in the deduplication framework.

The figure shows the overall design of the framework. This paper proposes the methodology which is incorporated into the TKNaaS. The methodology laid out for managing

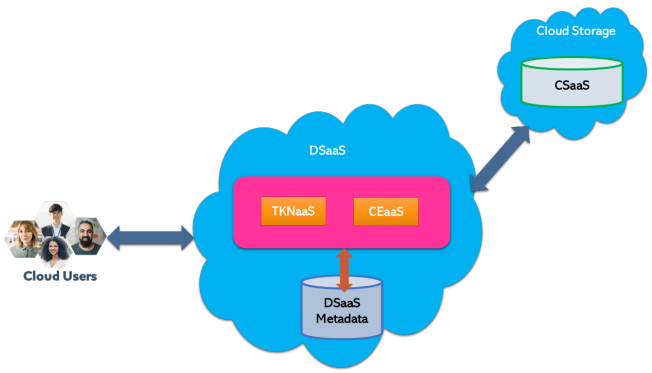

Figure 1: Framework block diagram

data deduplication introduces a streamlined, service-oriented approach aimed at optimizing cloud storage utilization by identifying and eliminating redundant data. This strategy, orchestrated by a cloud service provider specializing in deduplication, offers users a comprehensive deduplication service. Below is a nuanced explanation of this approach, focusing on its structure and the interaction between users and the cloud service provider.

Service-Oriented Deduplication Framework

At the heart of this proposed system is a deduplication service provided by the DSaaS. This service meticulously analyzes user data to detect similarities in file content, employing sophistication to ensure that unique data is stored. This not only conserves valuable storage space but also enhances data retrieval efficiency and reduces bandwidth usage.

User Interaction with the Deduplication Service

A straightforward, secure registration and data management process characterizes the interaction between users and the deduplication service:

Registration

Users initiate their engagement with the service through a registration process. This step is fundamental as it establishes a secure account for each user, facilitating personalized service delivery and secure access to stored data.

Upload

Once registered, users can upload their data to the cloud. During this process, the deduplication service analyses the data blocks, comparing them against existing blocks to identify duplicates. This analysis is not limited to a binary comparison but extends to evaluating the content similarity, ensuring a comprehensive deduplication effort.

Download

Users can download their data as needed. The deduplication process streamlines data retrieval by storing just unique instances of data, which reduces the time and bandwidth required for downloads.

Scenario-Based Token Generation

The proposed work delineates three distinct scenarios under which deduplication occurs, although these scenarios are not detailed in the brief. Below is a detailed overview of how the framework operates across the three described scenarios:

Scenario 1: Fresh data upload

- *Situation*

The data being uploaded is new and has no prior copies stored in the cloud.

- *Token generation*

Tokens are created for each block. This comprehensive tokenization allows for a thorough check against existing

cloud data, ensuring that only an entirely new data block is stored.

- *Outcome*

If no duplicates are found, the data is encrypted and uploaded to the cloud, preserving its uniqueness and integrity.

Scenario 2: Duplicate entire file

- *Situation*

The block being uploaded is identical to a block already stored in the cloud.

- *Token generation*

No new tokens are generated for this block, as its initial token (generated during a previous upload) matches one in the cloud, indicating a complete duplication.

- *Outcome*

Instead of storing the duplicate block, the system records a logical link to the already stored block for the current user, conserving storage space and avoiding redundancy.

Scenario 3: Partially modified file

- *Situation*

The block being uploaded contains some blocks that are identical to those in a previously stored file, along with some modified block content.

- *Token generation*

The block is segmented based on its matching content. Tokens are generated only for the block with the modified content, differentiating it from the unchanged portions of the file.

- *Outcome*

Only the modified blocks are encrypted and stored in the cloud. This selective storage ensures that updates or changes to blocks are accommodated without duplicating unchanged data, enhancing storage efficiency.

Block Level Deduplication

The research introduces an innovative approach to minimizing data redundancy in cloud storage by block-level deduplication strategies. This method significantly enhances the efficiency of data storage in the cloud by ensuring that unique data is preserved, thereby optimizing storage use and reducing unnecessary data replication. Let's delve deeper into the methodology and its benefits.

Block Level Deduplication

To address the limitations of file-level deduplication, block-level deduplication breaks down each file into smaller, manageable blocks of data. Each block is then tokenized, creating a unique identifier that represents a much smaller data segment compared to the entire file.

Enhanced Deduplication Accuracy

This granular approach permits a more precise deduplication process. When a file is prepared for upload, its blocks are individually checked against the block tokens stored in DSaaS. If a block's token matches an existing one, indicating that the block's content is already stored in the cloud, it is not uploaded again. This method is particularly effective for files that are largely similar but not identical to existing files, as it ensures that only genuinely new or modified blocks are stored.

Operational Efficiency

Generating tokens and managing the deduplication process using TKNaaS and DSaaS streamlines operations, making the cloud storage system more efficient and cost-effective.

Storage Optimization

This block-level approach maximizes storage efficiency by significantly reducing the volume of duplicated data, ensuring that the cloud storage space is utilized only for truly unique data content.

Cost and Energy Savings

Reducing duplicated data storage not only lowers the cost associated with data storage requirements but also contributes to energy savings by minimizing the computational resources needed for data management.

Token Generation for Blocks Level Deduplication

The process of token generation for file blocks within the context of block level deduplication is a critical component of the proposed research work's strategy to optimize cloud storage usage. This detailed approach ensures that data redundancy is minimized not only at the file level but also at the more granular block level, allowing for even more efficient use of cloud storage resources. Let's examine the procedure for generating and verifying block tokens:

Detailed Procedure of Token Generation at the Block Level

Initial step

The whole file is divided into a number of blocks. The block size is 5 kb each. The decision to use a block size of 5 kb in the proposed deduplication technique is determined by several factors aimed at optimizing performance and storage efficiency in cloud environments. Moreover, (Jayashree *et al*. 2022 and Jackowski *et al*., 2023) have demonstrated the effectiveness of smaller block sizes in achieving better deduplication performance and storage utilization. A block size of 5 kb strikes a balance between the granularity of data deduplication and the overhead of managing numerous small blocks.

- *Efficiency in deduplication*

Smaller block sizes, like 5 kb, increase the likelihood of detecting and eliminating redundant data segments within

files. This granularity helps to achieve higher deduplication ratios, especially in environments where slight changes are made to large datasets, thus maximizing storage savings.

- *Performance considerations*

Managing very small blocks (e.g., 1 kb or smaller) can lead to significant metadata overhead and increased processing time due to the higher number of blocks that need to be indexed and compared. Conversely, larger block sizes (e.g., 64 kb or more) might miss finer redundancy within files. Therefore, a 5 kb block size provides an optimal middle ground, ensuring effective deduplication without overwhelming the system with excessive metadata operations.

- *Storage and network efficiency*

The 5 kb block size also helps balance the trade-off between storage efficiency and network bandwidth utilization. Smaller blocks can be transferred more efficiently over the network, reducing latency and improving the deduplication system's overall performance.

Block token generation

Tokens are generated for each block. The proposed token generation generates a unique token for each block of data.

Block token check

Initially, a token representing the entire block is generated. This token acts as a unique identifier, encapsulating the essence of the block's data.

Verification with DSaaS

The generated block token is then checked against the metadata stored within the DSaaS. This verification step is crucial for determining whether an identical block already exists in the cloud storage.

- *If exists*

If the block token matches an existing entry in DSaaS, indicating the block is already stored, there's no need to proceed further to upload the data. Just take a reference to block and link to the users.

- *If does not exist*

If there's no match, indicating the block is unique to the cloud storage, the process moves to the next step: finding the percentage of blocks matching with existing blocks in the cloud storage. If the percentage of block matching is above 50%, then divide the 5 kb blocks into further small blocks for duplication checks.

Variable size block division

The block is divided into several small blocks based on the matching percentage. The size of this small block is not fixed; smaller blocks are divided into different sizes based on the matching percentage. The division strategy may vary based on the specific requirements of the deduplication system, such as block size or the type of data being processed.

Block token generation

For each small block derived from the 5 kb block, a unique token is generated. This block token serves a similar purpose as the 5 kb block token but on a smaller scale, representing the data contained within each block.

Variable size block token verification with DSaaS

Each generated smaller block token is then verified against the DSaaS to check for the existence of identical blocks within the cloud storage.

- *Duplicate blocks*

If a block token matches an existing entry in DSaaS, it indicates that the block's data is already stored in the cloud. In such cases, there's no need to upload this block again, effectively preventing redundant storage.

- *Unique Blocks*

If a block token does not find a match in DSaaS, it signifies that the block contains unique data not yet stored in the cloud. These blocks are then marked for encryption and upload to ensure the preservation of unique data.

The already stored block in the storage is also divided as it is by the size of the new variable-sized block. The unique block is uploaded to the cloud. The duplicate block is not uploaded. Instead, a reference is generated and linked to the current users.

The proposed method for block-level token generation and verification represents a sophisticated approach to minimizing data redundancy in cloud storage. By extending deduplication checks to the block level, the research work ensures even greater storage optimization, aligning with the goals of reducing storage costs and improving data retrieval performance. Through this meticulous process, each piece of data is evaluated for uniqueness, ensuring that cloud storage is utilized most effectively, storing only what is truly necessary.

The steps involved in the proposed token generation for blocks of a file are as follows.

Step 1: Input gathering

Collects user data that undergoes processing.

Step 2: American Standard Code for Information Interchange (ASCII) conversion

Converts each character of the input data into its ASCII decimal equivalent, allowing for numerical handling.

For understanding, transforming each character of the input data into its ASCII decimal equivalent enables numerical processing by converting characters into corresponding numbers based on the ASCII table (ASCII table, 2024). This table assigns a specific decimal number to each character, such as 65 for 'A', 97 for 'a', 49 for '1', and 32 for a space. By converting characters to their ASCII decimal equivalents, the data can be treated numerically, allowing for

operations like mathematical calculations, sorting, encoding, or encryption. For instance, the string "Hello" translates to the sequence 72, 101, 108, 108, 111. This conversion allows for more efficient and effective data processing in various computer science and data applications.

Step 3: Binary transformation

Transforms these ASCII values into binary code, preparing them for bitwise operations.

Step 4: Binary size calculation

Totals the number of bits to determine if padding is necessary.

Step 5: Padding

This step ensures that the binary data length aligns with a standard block size (64 bits in this case), adding zeros if necessary.

Step 6: Block division

This step splits the binary data into uniformly sized blocks, which is essential for consistent bitwise operations.

Step 7: XOR operation

Sequentially reduces the data blocks down to a single block using the XOR bitwise operation, which combines the data blocks.

Step 8: ASCII conversion (Back)

Converts the binary results back into ASCII decimal values.

Step 9: Character code generation

Transforms these decimal values back into readable ASCII characters.

Step 10: Token finalization

Combines the final set of characters into a single string that acts as a unique identifier or Token for the file, helping to prevent duplicate storage in systems like cloud databases.

Pseudo-code for Block-level Token Generation

```
Procedure GenerateBlockToken
  Input: userData
  Output: blockToken
  Begin
    // Step 1: Input Gathering
    rawData := userData
    // Step 2: ASCII Conversion
    asciiValues := []
    For each character in rawData do
      Convert character to ASCII decimal representation
      Append ASCII value to asciiValues
    EndFor
    // Step 3: Binary Transformation
    binaryStrings := []
    For each value in asciiValues, do
      Convert value to binary string
      Append binary string to binaryStrings
    EndFor
    // Step 4: Binary Size Calculation
    totalBits := 0
    For each binary in binaryStrings do
      totalBits += Length of binary
    EndFor
    // Step 5: Padding
    While totalBits mod 64 != 0 do
      Append "0" to the last binary string in binaryStrings
      Increment totalBits by 1
    EndWhile
    // Step 6: Block Division
    blocks:= Divide binaryStrings into 64-bit blocks
    // Step 7: XOR Operation
    xorResult := blocks[0]
    For i from 1 to Count(blocks) - 1 do
      xorResult := XOR(xorResult, blocks[i])
    EndFor
    // Step 8: ASCII Conversion (back)
    asciiResult := []
    For each 8-bit segment in xorResult do
      Convert 8-bit segment to decimal
      Append decimal to asciiResult
    EndFor
    // Step 9: Character Code Generation
    characters := []
    For each value in asciiResult do
      Convert decimal value to ASCII character
      Append character to characters
    EndFor
    // Step 10: Token Finalization
   blockToken := Concatenate characters into a single string
    Return fileToken
  End
EndProcedure
```

The research contributes to the field of cloud storage by offering a more robust and efficient method for data deduplication. The innovative approach of generating tokens at block levels addresses a critical need for improved storage optimization techniques. This work not only enhances the efficiency of cloud storage systems but also sets a new benchmark for future research in data deduplication technologies.

Implementation Setup and Importance

The implementation of deduplication research as a cloud application service is a practical step towards demonstrating the effectiveness and real-world applicability of proposed token generation techniques for block-level data. Utilizing C# to develop this cloud-based application and deploying it on a platform like MyASP.NET showcases the feasibility and operational benefits of the approach. Here's an overview of this implementation phase and its significance:

Development and Deployment

Cloud application service

By coding the deduplication process in C# and creating a robust application that embodies the research's theoretical advancements. This choice ensures that the application benefits from C#'s versatility and the rich ecosystem of .NET for cloud development.

Hosting on MyASP.NET

Choosing MyASP.NET as the deployment environment highlights the application's compatibility with cloud-based platform as a service (PaaS) offerings. MyASP.NET is known for its ease of use and support for .NET applications, making it an ideal choice for hosting a deduplication service.

Key Features of the Implementation

Real-time deduplication

The application actively demonstrates the process of deduplicating cloud storage in real time, offering tangible insights into how duplicate data can be identified and eliminated before storage.

User interaction

By allowing users to deploy their applications within this environment, provide a hands-on experience with the deduplication process, further validating the research's practical benefits.

Impact and Evaluation

Proof of concept

This implementation serves as a concrete proof of concept for this research, showing that the theoretical token generation methods for deduplication can be successfully applied in a live cloud environment.

Performance insights

Deploying the application on a cloud platform like MyASP.NET not only demonstrates its functionality but also provides valuable data on its performance in a real-world setting, including efficiency in storage management and processing speeds.

By successfully implementing and deploying deduplication research as a cloud application service, it has taken a significant step towards validating the proposed token generation techniques for improving cloud storage efficiency. This practical application not only underscores the research's relevance and potential impact on cloud storage solutions but also opens avenues for further enhancements based on real-world usage and performance data.

Results and Discussion

The research presents a comprehensive study on enhancing cloud storage efficiency through advanced deduplication techniques, focusing on the novel approach of block-level token generation. Deploying this method in a real cloud environment and subsequent performance analysis against existing techniques are critical steps in validating the proposed solution's effectiveness and efficiency.

Table 1: Performance comparison for block-level token generation

Size (kb)	*(Jayashree K., et al., 2022)*	*(A. Jackowski, et al., 2023)*	*BTEDD*
	Milliseconds		
5	15	19	9
10	29	37	17
20	57	76	34
30	88	114	55
40	118	150	70
50	146	188	88

Table 2: Performance comparison for block-level token generation

Scenarios	*(Jayashree K., et al., 2022)*	*(A. Jackowski, et al., 2023)*	*BTEDD*
	Milliseconds		
Scenario1	184	264	78
Scenario2	119	163	53
Scenario3	159	224	69

The information from Table 1 details the performance comparison of the proposed block-level token generation with existing approaches (Jayashree K. *et al*. 2022; A. Jackowski *et al*. 2023). It underscores the efficiency and effectiveness of the proposed deduplication technique. Notably, the superior results of the proposed method compared to the existing method highlight its robustness and adaptability in improving cloud storage.

The comparative analysis provided in Table 2 demonstrates the proposed deduplication technique's superiority in handling diverse scenarios of data redundancy. By efficiently processing new uploads, avoiding the storage of complete duplicates, and intelligently managing partially modified blocks, the proposed technique presents a promising approach to optimizing cloud storage. It's success in producing better results across these tests not only validates the effectiveness of the proposed method but also sets a benchmark for future deduplication research and development.

Conclusion

The innovative approach of generating tokens at block levels addresses a critical need for improved storage optimization techniques. This work not only enhances the efficiency of cloud storage systems but also sets a new benchmark for future research in data deduplication technologies. This token-based deduplication framework presents a highly effective approach to managing cloud storage, significantly

reducing unnecessary data duplication through intelligent, scenario-specific token generation. By precisely identifying and storing only new or modified data, the framework optimizes cloud storage usage, supports secure data encryption, and ensures efficient data management. This approach not only conserves storage space and bandwidth but also aligns with sustainable computing practices by minimizing the cloud infrastructure's energy consumption and operational costs.

Acknowledgment

We sincerely acknowledge the department head, Dr L. Nagarajan, and Dr N. Sumathi, the institution's principal, for providing the facility to complete this paper Successfully.

References

Andrzej Jackowski, Lukasz Slusarczyk, Krzysztof Lichota, Michał Welnicki, Rafal Wijata, Mateusz Kielar, Tadeusz Kopec, Cezary Dubnicki, Konrad Iwanicki. (2023). ObjDedup: High-Throughput Object Storage Layer for Backup Systems With Block-Level Deduplication. IEEE Transactions on Parallel & Distributed Systems, 34(07): 2180-2197.https://doi.org/10.1109/TPDS.2023.3250501

ASCII table. (2024). Table of ASCII codes, characters, and symbols. Retrieved June 7, from https://www.ascii-code.com

G. Ali, M. Ilyas Ahmad and A. Rafi. (2020). Secure Block-level Data Deduplication approach for Cloud Data Centers. IEEE *International Conference on Computing, Mathematics and Engineering Technologies*, Pakistan:1-6.https://doi.org/10.1109/iCoMET48670.2020.9074109.

H. Shin, D. Koo, Y. Shin and J. Hur. (2018). Privacy-Preserving and Updatable Block-Level Data Deduplication in Cloud Storage Services. *IEEE International Conference on Cloud Computing (CLOUD)*, San Francisco: 392-400.https://doi.org/10.1109/CLOUD.2018.00056.

Jayashree K, Narayana KE (2022). Real Time Efficient Block Level Dual Mode Data Deduplication Towards Mitigating Side Channel Attack in Cloud Storage. Research Square, 1: 1-18. https://doi.org/10.21203/rs.3.rs-2276184/v1

Komal Kshirsagar, Pallavi Patekar, Saurav Kolhe, Neha Nimbalkar, Prof. P. N. Pathak. (2023). Secure Cloud Storage With Deduplication Technique. International Research Journal of Modernization in Engineering Technology and Science, 05(5): 7290-7295. https://www.doi.org/10.56726/IRJMETS40666

Nagappan Mageshkumar A, J. Swapna B, A. Pandiaraj C, R. Rajakumar G, Moez Krichen D, E, Vinayakumar Ravi. (2023). Hybrid cloud storage system with enhanced multilayer cryptosystem for secure deduplication in the cloud. International Journal of Intelligent Networks, 4: 301-309. https://doi.org/10.1016/j.ijin.2023.11.001

P. M. A. Kumar, E. Pugazhendhi and R. K. Nayak. (2022). Cloud Storage Performance Improvement Using Deduplication and Compression Techniques. *IEEE International Conference on Smart Systems and Inventive Technology*, India: 443-449. https://doi.org/10.1109/ICSSIT53264.2022.9716524.

Pavithra, M., Prakash, M. & Vennila, V. BGNBA-OCO based privacy preserving attribute based access control with data duplication for secure storage in cloud. Journal of Cloud Computing, 13(8): 1-18. https://doi.org/10.1186/s13677-023-00544-1

Sahaya Jenitha A, & Prakash, S. J. (2023). A general stochastic model to handle deduplication challenges using hidden Markov model in big data analytics. The Scientific Temper, 14(04), 1398–1403. https://doi.org/10.58414/SCIENTIFICTEMPER.2023.14.4.50

Sharik Ahmad and Ms. Sneha Deokate. (2023). Enhance Cloud Data Compression using Duplication. International Journal of Research Publication and Reviews, 4(8): 1982-1989.

Ulthana ASR, Dr.K.Juliana Gnanaselvi. (2023). A New Data Security with Privacy-Preserving and Deduplication based Cloud Storage through Public Cloud Auditing in Cloud Computing. European Chemical Bulletin, 12(12): 836-842. https://doi.org/10.48047/ecb/2023.12.12.57

Vadukapuram Roshini, P. Sathish, Dr. V. Bapuji. (2023). Cloud Data Deduplication: Ensuring Integrity and Eliminating Duplication. Dogo Rangsang Research Journal, 13: 108-115.

Viji, D., Revathy, S. (2023). Hash-indexing block-based deduplication algorithm for reducing storage in the cloud. Computer Systems Science and Engineering, 46(1), 27-42. https://doi.org/10.32604/csse.2023.030259

Yongkai Fan A, B, Xiaodong Lin B, Wei Liang C, Gang Tan d, Priyadarsi Nanda E. (2019). A secure privacy-preserving deduplication scheme for cloud computing. Future Generation Computer Systems, 101: 127–135. https://doi.org/10.1016/j.future.2019.04.046

Zhang, S., Wu, S., Fan, H., Zou, D., Jin, H. (2020). BED: A Block-Level Deduplication-Based Container Deployment Framework. LNCS, Springer, 12398. https://doi.org/10.1007/978-3-030-64243-3_38