INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH AND ANALYSIS

ISSN(print): 2643-9840, ISSN(online): 2643-9875

Volume 08 Issue 11 November 2025

DOI: 10.47191/ijmra/v8-i11-22, Impact Factor: 8.266

Page No. 6086-6096

Enhancing Modern Storage using Chunk-Based Data Deduplication in ABF-HTFC Algorithm

Ashis Kumar Mohapatra

Wisconsin State Department, Sun Prairie WI 53590

ABSTRACT: In modern times, cloud computing has become increasingly popular due to its ease of access, unlimited data storage, and payment capabilities. Additionally, data reduction is a widely used technique to minimize the storage of unnecessary data items and reduce maintenance overhead. Furthermore, research on data reduction in cloud-based systems is increasingly focused on the rapid growth of data volume in cloud storage services. However, valuable storage space is often lost when users upload multiple copies of duplicate data, and it is challenging to identify chunk files. To resolve this problem, we propose an Attribute-based Bloom Filter Hash Table with File Counting (ABF-HTFC) algorithm to remove redundant information and identify the storage using chunk-based data deduplication. Furthermore, boundary detection is speed up using the Fast Content-Defined Chunking (FastCDC) algorithm to achieve high-speed processing and effectively eliminate unnecessary storage. Next, data integrity and reliability in cloud environments can be ensured by using the Cryptographic Hashing - SHA-256 (CH-SHA-256) Algorithm to generate fingerprints and improve indexing efficiency. Finally, we propose an ABF-HTFC algorithm for data deduplication, which removes redundant chunk information and accurately identifies duplicate data in cloud storage. The proposed method outperforms the previous technique in identifying chunk files based on data deduplication. Furthermore, the proposed method was evaluated using storage performance metrics such as latency, throughput, cloud storage capacity, execution time, and deduplication ratio, and the storage efficiency is improved to 94.25%.

KEYWORDS: Data deduplication, cloud storage, chunk file, hash function, bloom filter, FastCDC and CH-SHA-256.

I. INTRODUCTION

Cloud storage service providers cater to the demands of organizations and individuals to store, transfer, back up, and access data from other cloud services at a reasonable cost for an ever-growing volume of data. To provide efficient data storage, cloud service providers generally utilize the widely used drag-and-drop approach, as it enables single data storage and eliminates duplicates, thereby decreasing storage overhead and conserving upload bandwidth [1]. Data duplication occurs in the cloud, where multiple users encrypt and store the same data using different methods. Deduplication, an effective data reduction technology, eliminates redundant data by storing unique copies of data. In addition, duplicate data can be identified by comparing the data fingerprint with previously stored data. The cloud storage system requires a minimum number of copies of all data, referred to as the data duplication factor. Defragmentation reduces storage needs, costs, and bandwidth when data exceeds the replication factor. Data replication, whether at the file or block level, benefits significantly from compression [2], as pricing scales with backup size, offering customers time savings on initial storage and redundant capacity investments.

Data deduplication can reduce storage capacity by removing duplicate data from storage, but it cannot detect and remove redundant data between identical but non-duplicate data. It is helpful for file compression during network transmission and block-level compression based on delta compression [3] to remove redundant data in network storage systems. It not only reduces storage overhead but also reduces file transfer times between servers and clients in a networked storage system. The basic feature of data reduction is to divide the input data stream into pieces, which are divided into two types: (a) fixed-length segmentation and (b) variable-length segmentation [4]. Furthermore, the inference performance can be reduced by using the fixed-length segmentation architecture as a fast and straightforward method. Then, additional redundancies are identified in the derivatives for the data changes, as most of the blocks remain unchanged.

The latency problems associated with public storage's location and bandwidth can significantly impact storage system performance during data transmission; yet, hybrid clouds are known for their outstanding performance. However, storing all the data in the cloud could lead to problems with vendor lock-in or data privacy, potentially rendering the data inaccessible or causing

it to be lost [5]. Large volumes of data must be transferred via the network, and substantial recovery capacity is needed to ensure sufficient robustness in the event of node failures.

A. Objective of the research

- The objective of the proposed ABF-HTFC algorithm is to enhance cloud storage by eliminating redundant information and accurately identifying duplicate data through droplet-based data reduction
- By using the FastCDC algorithm, individuals may create minimum, average, and maximum chunk sizes, standardize chunk sizes, enhance performance, and split fixed-size files into chunk of varying sizes.
- The hashing process also ensures data integrity and reliability in a large-scale cloud environment using CH-SHA-256 to optimize space utilization.
- This method is aimed to improve storage performance, reduce unnecessary space consumption, and effectively manage duplicate particles in a large-scale cloud environment.

B. Motivation of the research

The main purpose of this paper is to introduce the ABF-HTFC algorithm to design an efficient data deduplication system for cloud storage. This approach focuses on eliminating redundant information, optimizing storage utilization through block-based reduction. Therefore, FastCDC method is adopted for fast and accurate border detection and CH-SHA-256 ensures secure fingerprint generation and reliable encryption. By combining attribute-based Bloom filter and file counting, the proposed ABF-HTFC algorithm is able to accurately identify duplicate blocks, improve indexing performance, and ensure data integrity.

C. Contribution of the research

- Proposed an ABF-HTFC algorithm for efficient removal of redundant information and accurate identification of duplicate fragment files in cloud storage.
- It integrates the FastCDC algorithm to speed up boundary detection, facilitate faster processing, and minimize unnecessary storage.
- Utilize SHA-256 to ensure data integrity, generate unique fingerprints, and enhance cryptographic efficiency in large-scale cloud environments.
- The performance evaluation demonstrates that the proposed ABF-HTFC method outperforms existing techniques in both fragment identification and storage reduction.

II. LITERATURE SURVEY

FuzzyDedup [6] is a new inference technique for safely deduplicating similar data (files, chunks, or blocks). A Fuzzy-Style Exclusion Encryption (FuzzyMLE) method using a specially created encryption algorithm, can reduce ciphertext-based comparable data. However, most of the methods have not been verified using dynamic views in cloud storage environments, and therefore, they all suffer from security vulnerabilities. In this work [7], a chunk-based data extraction scheme has been developed to provide high security to data in the cloud. The Feature-Aware Stateful Routing (FASR) technique [8] reduces system overhead and maintains a high reduction rate in distributed settings, but frequent parallel computations still result in substantial overhead. A technique called data deduplication removes redundant data while lowering costs, bandwidth, and disk utilization. This paper [9] covers the concepts, types, and various storage strategies for utilizing data reduction. They provide a content-level decoding and re-encryption scheme based on Enhanced Randomized Convergent Encryption (ERCE) to protect original data by generating keys from the owner and identifying duplicate data when stored on the cloud platform [10].

Piece-level reduction plays a key role in identifying the correct block boundaries, introducing a new truncation algorithm called Dynamic Prime Chunking (DPC). The goal of DPC is to dynamically change the window size within the key range based on minimum and maximum block sizes [11]. The offered a new cloud storage system with three advanced modules that improve data security and deduplication [12]. The Fixed Window Fixed Bytes Chunking (FWFBC) approach is different from the conventional variable window size approach. A network-based block context-aware model and an initial feature extraction strategy based on individual N sub-blocks comprise the Context-Aware Resemblance Detection (CARD) method introduced in the model [13]. In the novel [14], we aimed to provide a comparative study of different data reduction techniques using optimisation algorithms based on multiple considerations, including data security performance.

Table I. Survey of Various Chunk File Based Data De-Duplication Models

Author	Year	Methodology Used	Limitations	Performance Evaluation
Basappa Kodada [15]	2022	Hash-based indexing technique (HBIT)	Protect cloud storage data	Decryption algorithm-78.9%, Throughput comparison-77.36%
Lee, M [16]	2022	Secure Data Sharing In Cloud (Sedasc) Protocol	Reduces storage space and bandwidth needs.	client computational complexity-23.6ms, server computational complexity-34.5ms
Sreeharsha Udayashankar [17]	2025	Content-defined Chunking (CDC) algorithms	Data replication becomes a significant performance bottleneck.	Throughput-74.6%, Extreme Byte Search-71.26%
Marcel Gregoriadis [18]	2024	CDC algorithm	Reducing Storage and Bandwidth Costs	Throughput-74.23%, deduplication ratio-79.14%, average chunk size-70.19%
Gan [19]	2025	Encrypted deduplication algorithm (ECDedup)	Outsourced data is also vulnerable to offline brute force attacks.	throughput by up to 51.9%
Ellappan [20]	2024	CDC algorithm	Deduplication Efficiency, Reducing The Chunk Variance	Throughput-69.8%
Kavitha, [21]	2025	Data Deduplication-based Efficient Cloud Optimisation Technique (DD-ECOT).	The amount of data continues to grow at an alarming rate.	Storage efficiency-89.21%, latency-42.13%, e data integrity to 81.32%

Table 1 provides a comparative overview of recent research, highlighting the decoupled file-based data reduction techniques in cloud storage, the methods used, the limitations, and the performance evaluation metrics. To address the shortcomings of the current analysis, a method combining Converged Encryption (CE) and data reduction was implemented. Unique encryption keys are generated by using user data to facilitate the secure decryption of encrypted data [22]. The proposed model was implemented more effectively than many existing solutions to enhance performance in cloud infrastructure [23]. Furthermore, the experimental results show that it achieves a 6.7% higher subtraction rate and saves 85.09% storage space through efficient techniques. A Cryptographic Deduplication Authentication Scheme (CDAS) is introduced to enable secure file storage and retrieval while restricting encrypted access to authorised users [24]. Highlights the value of cloud storage and utilises the Elliptic Curve Cryptography (ECC) model, which is recommended for enhanced security, to preserve and mitigate data. Test evaluations and security measures are verified with the suggested technology [25].

III. PROPOSED METHODOLOGY

The ABF-HTFC algorithm's purpose is to effectively identify and eliminate redundant chunk files from cloud storage systems. The method minimizes unnecessary hash Table lookups, accurately identifies duplicate chunks, and accelerates duplicate detection by combining the Bloom filter index with attribute-based data and file counts and managing massive amounts of data in a cloud environment while lowering storage costs, speeding up processing, and preserving data integrity.

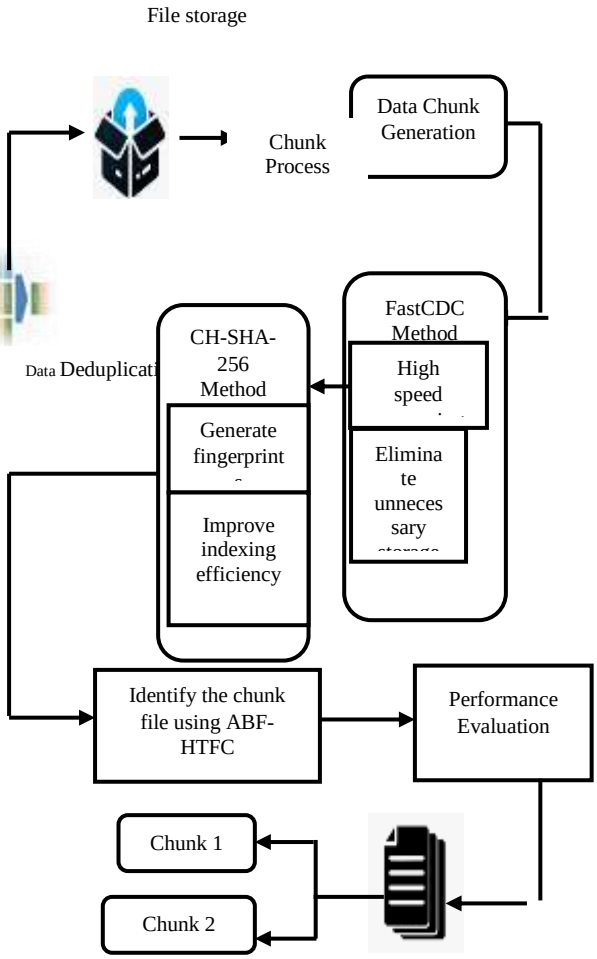

Fig. 1. Architecture Diagram of the Proposed System to Identify Chunk File

Figure 1 shows the structure of the proposed ABF-HTFC approach for chunk-based data reduction in cloud storage. The process starts with saving the file and then dividing it into smaller parts through a chunking process. The FastCDC method is utilized to expedite boundary detection and achieve high-speed processing, thereby minimizing unnecessary storage. Each fragment is processed using the SHA-256 method to create a unique cryptographic fingerprint, ensuring data integrity and improving encryption efficiency. The deduplication process is further enhanced by ABF-HTFC, which quickly identifies duplicate fragments, removes redundant information, and utilizes file counting to manage duplication efficiently. Finally, the optimized particles are stored, and the overall system performance is evaluated to ensure the deduplication rate, thereby guaranteeing scalable and efficient cloud storage management.

A. Fast Content-Defined Chunking (FastCDC)

This section accelerates boundary detection using the FastCDC model to achieve high-speed processing and effectively eliminate unnecessary storage space. The FastCDC model is essential for dividing the input file into variable-sized chunks based on content. The process starts with collecting files of different shapes and sizes, which are then processed using the FastCDC algorithm. This content-aware approach allow s for the identification of similar regions, even when files have been slightly changed or modified, thus improving duplicate detection. The FastCDC method offers speedy processing and efficient removal of unnecessary storage, making it ideal for large-scale cloud environments. Compute the total number of bytes in the input file as illustrated in Equation 1. Let's assume F −input file, b_1 −byte of the fil, and b_n −Total number of bytes.

$$F = (b_1, b_2, \dots, b_n) \quad (1)$$

As shown in equation 2, efficiently compute the hash value of the sliding window data. Let's assume RH_i −rolling hash value, w −sliding window size, α −constant mulitiplier hash stability, p −large prime number, b_{i+j} −byte value position.

$$RH_i = \sum_{j=0}^{w-1} \alpha^j . b_{i+j} \quad (\text{mod } p) \quad (2)$$

As illustrated in Equation 3, calculate the target threshold that defines the boundary selection to limit the size of the average segmentation in boundary detection. Let's assume

$$RH_i \& M = T \quad (3)$$

As indicated in equations 4 and 5, the starting and ending bytecodes of a fragment are evaluated based on the total number of chunks generated. Let's assume C −identified chunks, b_{s_j} −Start byte index of chunk, b_{e_j} −end byte index of chunk, C_m −Total number of chunks formed.

$$C = (C_1, C_2, \dots, C_m) \quad (4)$$

$$C_j = \left(b_{s_j}, b_{s_j+1}, .., b_{e_j}\right) \quad (5)$$

The FastCDC accelerates chunk boundary detection by utilizing a hash function within a sliding window. FastCDC creates variable-sized blocks based on data content, thereby improving the performance of duplicate detection

B. Cryptographic Hashing based on SHA-256 (CH-SHA-256)

The data integrity and reliability in cloud environments can be ensured by using the CH-SHA-256 algorithm to generate fingerprints and improve indexing efficiency. The CH-SHA-256 algorithm is based on calculating the unique digital fingerprint of a data block and calculating a fixed-length 256-bit hash value to create a block. The fingerprint of the new block can be determined and verified using the code, which employs the CH-SHA-256 algorithm to ensure accurate duplicate detection and efficient encryption. Furthermore, SHA-256 implements a collision-prevention mapping between chunk identifiers.

Calculate the input representation in bytes of the size of the final chunk, as shown in Equation 6. A 512-bit message block is transformed 64 times using the Merkle–Damgård construction in the SHA-256 algorithm. Let's assume $H^{(t-1)}$ −Previous hash state, M_t −Current 512-bit message block of chunk, CF − Compression function.

$$H^{(t)} = CF\left(H^{(t-1)}, M_t\right) \quad (6)$$

As shown in Equation 7, compute the hash value in the update rule. Let's assume $H^{(t)}$ −Updated hash state after round, $\sum(W, K)$ −Non-linear operations in massage schedule, $(\text{mod } 2^{32})$ −Modular addition over 32-bit words.

$$H^{(t)} = \left(H^{(t-1)} + \sum(W, K)\right) \ (\text{mod } 2^{32}) \quad (7)$$

Calculate the connection operator of the final fingerprint once all blocks have been processed, as indicated by Equation 8. Let's assume $FP(C_j)$ −Final **256-bit fingerprint** of chunk, $(h_0 \| h_1 \| h_2 \| h_3 \| h_4 \| h_5 \| h_6 \| h_7 \|)$ −eight 32-bit words concatenated.

$$FP(C_j) = H^{(T)} = (h_0 \| h_1 \| h_2 \| h_3 \| h_4 \| h_5 \| h_6 \| h_7 \|) \quad (8)$$

As shown in Equations 9 and 10, calculate the index mapping in the deduplication Table for both the collision resistance & Indexing. Let's assume $FP(C_j)$ −Final fingerprint of chunk, C_i, C_j −chunk of file.

$$C_i \neq C_j \ <= FP(C_i) \neq FP(C_j) \quad (9)$$

$$\text{Index}(FP(C_i)) \vdash> C_i \quad (10)$$

The CH-SHA-256 algorithm is used to ensure data integrity, reliability, and uniqueness during derivation. This fingerprint is used for encryption, search, and duplicate detection.

C. Attribute-based Bloom Filter Hash Table with File Counting (ABF-HTFC)

The **ABF-HTFC algorithm** is designed to enhance the process of chunk-based data deduplication in cloud storage. A cryptographic fingerprint identifies each volume and embeds additional attributes such as file type, size, and metadata into the indexing process. Bloom filters are used to quickly check for the existence of a block before performing an extensive check on the hash Table, thereby minimizing unnecessary lookups. A file counting mechanism effectively manages duplicates for each block and verifies that redundant files are removed. This integrated design allows the ABF-HTFC approach to achieve high scalability, reduce search overhead, and optimize cloud storage utilization by combining attribute-based indexing with probabilistic filtering and efficient duplicate monitoring.

Equation 11 illustrates that each region is calculated using both its fingerprint and its properties in attribute embedding for indexing. Let's assume $f_{type}(C_i)$ −file type, f_{size} −file size, $mata(C_i)$ −mata data.

$$A(C_j) = \{FP(C_j), f_{type}(C_j), f_{size}(C_j), mata(C_j)\} \quad (11)$$

As shown in equation 12, evaluate the membership test of Bloom filters using BF for probabilistic duplicate detection. Let's assume BF − Bloom filter array, Number of hash functions.

$$BF\left(FP(C_j)\right) = \begin{cases} 1 & \text{if } FP(C_j) \text{ may ewxist in storage} \\ 0 & \text{if } FP(C_j) \text{ defiently does not exist} \end{cases} \quad (12)$$

Create and update a hash Table with the number of files that indicates the probability that the Bloom filter is present, as shown by Equations 13 and 14. Let's assume HT −hash Table, $\text{count}(C_j)$ −File count for chunk, A −attribute.

$$HT[FP(C_j)] = \left(A(C_j), \text{count}\left(C_j\right)\right) \quad (13)$$

$$\text{count}\left(C_j\right) = \begin{cases} \text{count}\left(C_j\right) + 1 & \text{if duplicate chunk is identified} \\ & \text{if new chunk is inserted} \end{cases} \quad (14)$$

According to Equation 15, calculate the final decision rule for deduplication. Ensures efficient removal of unnecessary files. Let's assume $D(C_j)$ – Deduplication status of chunk, HT –hash Table.

$$D(C_j) = \begin{cases} \text{Duplicate} & \text{if BF}\left(FP(C_j)\right) = 1 \wedge FP(C_j) \in HT \\ \text{Unique} & \text{if BF}\left(FP(C_j)\right) = 0 \vee FP(C_j) \notin HT \end{cases} \quad (15)$$

The ABF-HTFC algorithm enhances chunk-based data deduplication by integrating cryptographic hashing, attribute-aware indexing, Bloom filters for fast membership queries, and file counting for duplicate data.

IV. RESULT AND DISCUSSION

This section compares the ABF-HTFC approach with traditional methods using data deduplication-based chunk file storage performance metrics such as latency, throughput, cloud storage capacity, execution time, and deduplication rate. Moreover, the ABF-HTFC method can be combined with previous FWFBC, ERCE, and CARD techniques to facilitate identification of storage using a chunk file based on a data deduplication model.

Table II. Simulation Parameter

Simulation	Variable
Cloud environment	AWS, EBS cloud storage
Data used and size	Content file type,=<= 5Gb
Simulation framework Visual Studio/ c#.net , VS	Visual Studio/ c#.net , VS
Output type definition	Redundant storage space

Table 2 shows the simulation used to test the ABF-HTFC inference method. The input data contains various content file types up to 5 GB in size. Simulation is implemented using Visual Studio with C#.NET to implement the framework. The simulation output focuses on identifying and minimizing redundant storage space, thereby confirming the effectiveness of the approach.

Table III. Comparison of the Throughput Performance for Proposed System vs. Baseline Models

Chunk Size (Mb)	FWFBC	ERCE	CARD	ABF-HTFC
50	70.12	73.21	75.18	79.56
100	74.10	76.47	79.34	82.16
150	77.15	79.64	82.17	85.23
200	79.31	81.48	85.45	87.28

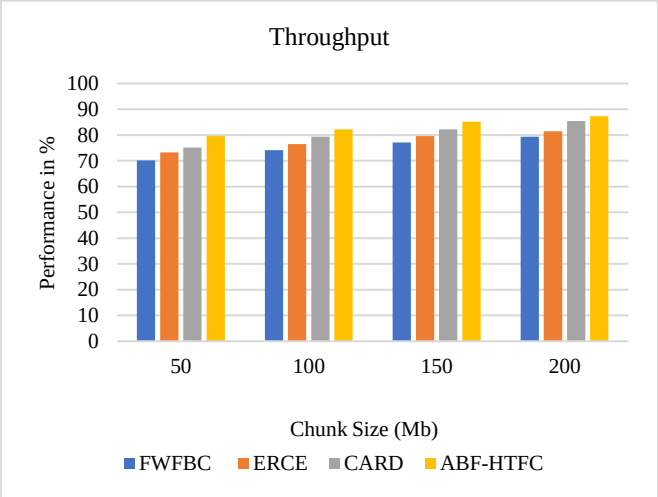

Fig. 2. Comparison of the Throughput Performance using File Chunk Identification Models

As shown in Figure 2 and Table 3, the ABF-HTFC method can identify a chunk file by data duplication using previous techniques throughput performance analysis. Furthermore, the throughput performance analysis shows that the ABF-HTFC methods achieve rates of 79.31%, 81.48%, and 85.45%, respectively, compared to the previous FWFBC, ERCE, and CARD methods. Similarly, the throughput performance analysis shows that the ABF-HTFC methods achieve a rate of 87.28% in identifying the chunk by data duplication, compared to previous techniques.

Table IV. Performance Analysis of the Execution Time based on File Chunk Identification Models

Chunk size	FWFBC	ERCE	CARD	ABF-HTFC
50	41.23	37.4	34.28	30.17
100	37.18	34.21	31.20	25.04
150	35.16	31.12	27.14	21.06
200	31.24	28.14	23.16	17.05

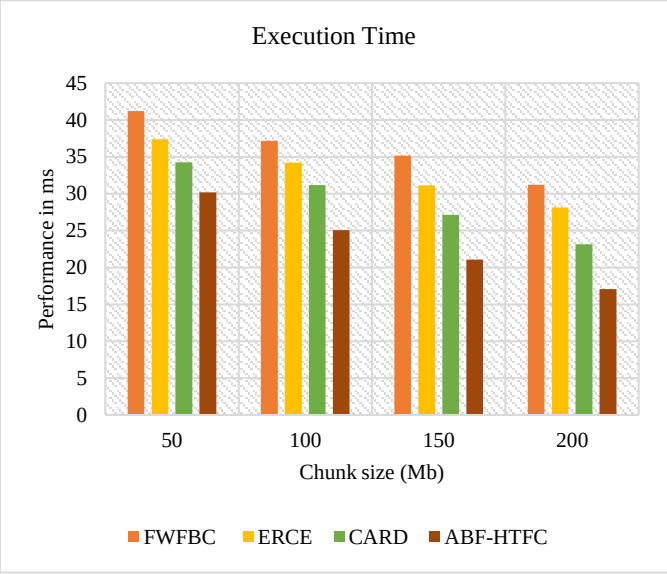

Fig. 3. Analysis of Execution Time during File Chunk Identification

As shown in Figure 3 and Table 4, the ABF-HTFC method can identify a chunk file by data duplication using previous techniques and analyze the execution time performance. Furthermore, the Execution Time analysis shows that the ABF-HTFC methods achieve rates of 31.24ms, 28.14ms, and 23.16ms, respectively, compared to the previous FWFBC, ERCE, and CARD methods. Similarly, the performance analysis of execution time shows that the ABF-HTFC method achieves a rate of 17.05ms in identifying chunks by data duplication, compared to previous techniques.

Table V. Performance Evaluation Of Deduplication Ratio In File Chunk Identification

Chunk Size (Mb)	FWFBC	ERCE	CARD	ABF-HTFC
50	73.24	75.26	78.23	81.25
100	75.17	77.23	80.33	84.23
150	79.12	80.16	84.24	86.37
200	81.10	83.19	85.14	89.46

As shown in Figure 4 and Table 5 the ABF-HTFC method can identify a chunk file by data duplication using previous techniques and analyze the performance of deduplication ratio. Furthermore, the deduplication ratio analysis shows that the ABF-HTFC methods achieve rates of 81.10%, 83.19%, and 85.14%, respectively, compared to the previous FWFBC, ERCE, and CARD methods. Similarly, the performance analysis of the deduplication ratio shows that the ABF-HTFC method achieves a rate of 89.4% in identifying duplicated chunks, compared to previous techniques.

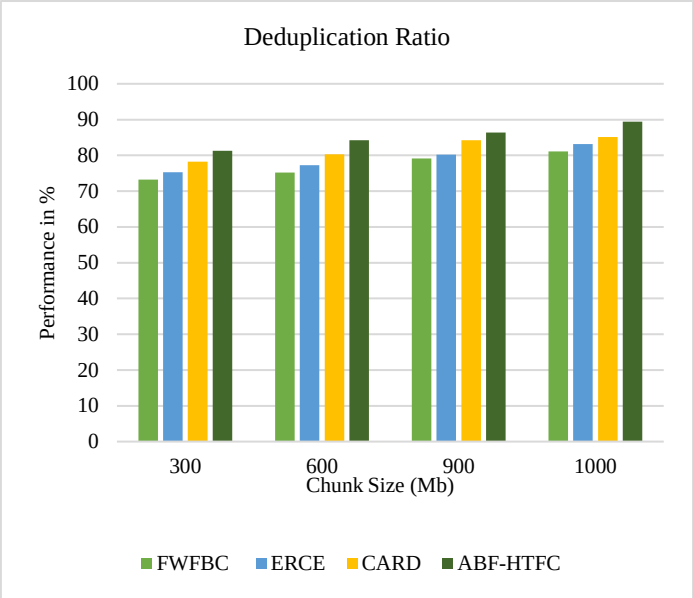

Fig. 4. Detailed Analysis of Deduplication Ratio Based on File Chunk Identification

Table VI. Performance Evaluation Of Storage Efficiency Through File Chunk Identification

Chunk Size	FWFBC	ERCE	CARD	ABF-HTFC
50	74.2	77.24	80.17	83.24
100	79.24	80.13	84.07	86.22
150	81.29	84.28	85.23	89.17
200	85.21	89.14	90.13	94.25

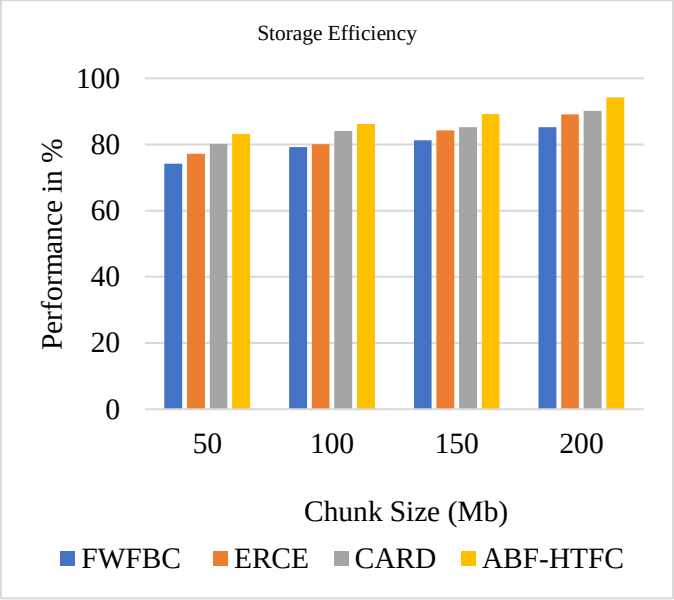

Fig. 5. Analytical Study on Storage Efficiency Achieved via File Chunk Identification

The suggested method is to evaluate the storage efficiency performance and identify a chunk file by data duplication utilizing prior techniques, as illustrated in Figure 5 and Table 6. Additionally, compared to the earlier FWFBC, ERCE, and CARD approaches, the suggested methods obtain rates of 85.21%, 89.14%, and 90.13%, respectively, according to the storage efficiency analysis. Additionally, the storage efficiency performance analysis reveals that the suggested approach outperforms earlier methods in detecting duplicate chunks, with a 94.25% identification rate.

Table VII. Performance Evaluation Of Latency In File Chunk Identification Process

Chunk Size	FWFBC	ERCE	CARD	ABF-HTFC
50	54.21	50.14	46.18	40.29
100	50.32	45.06	41.27	36.17
150	46.23	41.17	37.15	31.25
200	41.07	37.02	33.26	27.09

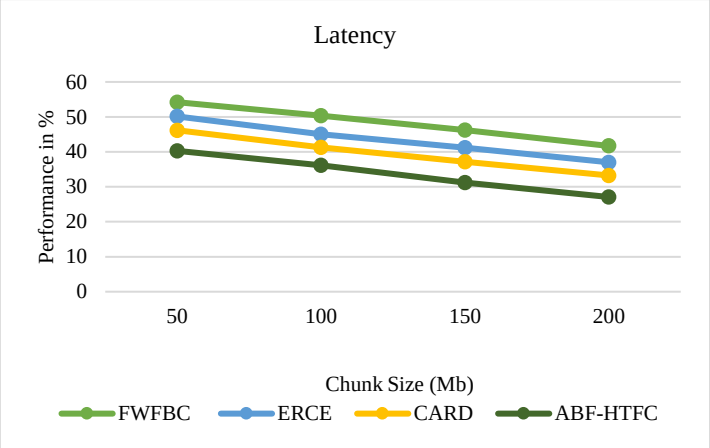

Fig. 6. Analytical Study on Latency Impact during File Chunk Identification

The suggested ABF-HTFC method is to evaluate latency performance and identify a chunk file by data duplication using prior techniques, as illustrated in Figure 6 and Table 7. Additionally, compared to the previous FWFBC, ERCE, and CARD approaches, the suggested methods obtain rates of 41.07%, 37.02% and 33.26%, respectively, according to the latency analysis. Additionally, the latency performance analysis reveals that the suggested ABF-HTFC approach outperforms earlier methods in detecting duplicate chunks, with a 29.09% identification rate.

V. CONCLUSION

In conclusion, the proposed ABF-HTFC algorithm effectively removes redundant information and optimizes storage through fragment-based data reduction. Integration of the FastCDC algorithm accelerates boundary detection, enables faster processing, and eliminates unnecessary storage overhead. To ensure integrity and reliability, the CH-SHA-256 algorithm is used for efficient fingerprint generation and encryption. The ABF-HTFC method further enhances the derivation by embedding attributes, utilizing Bloom filters for efficient searching, and managing duplicates based on file count. Experimental evaluation demonstrates that the proposed approach outperforms existing methods in accurately identifying duplicate particles, with improvements in latency, efficiency, storage capacity, execution time, and detection rate. Overall, the ABF-HTFC algorithm achieves a storage efficiency of 94.25%, proving its effectiveness as a scalable and reliable solution for cloud storage environments.

A. Limitation of the Research

- Albeit the suggested Attribute-based Bloom Filter Hash Table with File Counting (ABF-HTFC) algorithm shows a tremendous enhancement in the detection accuracy and storage efficiency, it has a number of limitations.
- First, the computational costs of the fingerprinting and chunk identification process, especially that of the Fast Content-Defined Chunking (FastCDC) and Cryptographic Hashing (CH-SHA-256) phases, are expensive in large scale or high-performance cloud environments. This can affect the processing latency and energy usage of operations that are data intensive.
- Second, dynamic workload behavior of cloud systems which involve the addition, modification and deletion of files have not been fully tested. This may impact the stability and live capability of the deduplication system proposed.
- Third, although the approach improves the efficiency of storage, the security implications of hash-based fingerprinting in multi-tenant environments should be explored further, in particular, in relation to data leakage, as well as unauthorized access to hash Tables.
- Final, the system has not been fully tested to operate in a heterogeneous or hybrid cloud environment where the differences in infrastructure, network bandwidth and storage policies can affect the overall deduplication and retrieval performance.

B. Future Research

To address these limitations and widen the scope of the proposed model, future studies could be directed towards several directions. The development of lightweight or adaptive versions of Bloom Filter that are still accurate with low memory and computation overheads is just one of the possible directions. This should result in a more efficient system that could perform real-time cloud operations. The other direction is to adjust the ABF-HTFC model to workloads in dynamical and distributed clouds, in a way that will make it viable to identify and deduce chunks even in circumstances of frequent data updates and multi-user dynamics. Moreover, the implementation of the suggested approach into the hybrid and multi-cloud architectures can increase the levels of scalability, interoperability, and resilience, allowing the seamless deduplication of the data that spreads across the geographically distributed data centers. Then, advanced security controls, e.g., homomorphic encryption or blockchain-based audit trail, will enhance the integrity, privacy, and management of trust in the deployment of heterogeneous cloud environments and enhance the resilience of the system in the future.

REFERENCES

1) Ujjwal Rajput, Sanket Shinde, Pratap Thaku, "Analysis on Deduplication Techniques for Storage of Data in Cloud," International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056, Volume: 09 Issue: 05 | May 2022.
2) J Gnana Jeslin, Mohan Kumar, "Implementing An Efficient Data Deduplication Framework For Cloud Storage," Indian Journal of Computer Science and Engineering (IJCSE), DOI: 10.21817/indjcse/2022/v13i1/221301046 Vol. 13 No. 1 Jan-Feb 2022.
3) Wang, C., Wang, K., Li, M., Wei, F., & Xiong, N. (2023). Chunk2vec: A novel resemblance detection scheme based on Sentence-BERT for post-deduplication delta compression in network transmission. IET Communications, 18(2), 145-159. https://doi.org/10.1049/cmu2.12719
4) Ahmed, Saja Taha, and Loay E. George. "Lightweight hash-based de-duplication system using the self detection of most repeated patterns as chunks divisors." Journal of King Saud University-Computer and Information Sciences 34.7 (2022): 4669-4678.
5) Yolchuyev, Agil, and Janos Levendovszky. "Data Chunks Placement Optimization for Hybrid Storage Systems." Future Internet 13.7 (2021): 181.
6) T. Jiang et al., "FuzzyDedup: Secure Fuzzy Deduplication for Cloud Storage," in IEEE Transactions on Dependable and Secure Computing, vol. 20, no. 3, pp. 2466-2483, 1 May-June 2023, doi: 10.1109/TDSC.2022.3185313.
7) Ruba, S., Kalpana, A.M. Advanced chunk-based data deduplication framework for secure data storage in cloud using hybrid heuristic assisted optimal key-based encryption. Wireless Netw 31, 3467–3489 (2025). https://doi.org/10.1007/s11276-025-03947-x
8) Yao, Wenbin, et al. "FASR: An efficient feature-aware deduplication method in distributed storage systems." IEEE Access 10 (2022): 15311-15321.
9) S. M. A. Mohamed and Y. Wang, “A survey on novel classification of deduplication storage systems,” Distrib. Parallel Databases, vol. 39, no. 1, pp. 201–230, Mar. 2021, doi: 10.1007/s10619-020-07301-2..
10) B. Venkatesan and S. Chitra, "Data De-Duplication Process and Authentication Using ERCE with Poisson Filter in Cloud Data Storage," Intelligent Automation & Soft Computing, DOI:10.32604/iasc.2022.026049
11) Ellappan, Manogar. "Dynamic Prime Chunking Algorithm for Data Deduplication in Cloud Storage." KSII Transactions on Internet & Information Systems 15.4 (2021).
12) Ganesh, D. "Super-Fast and Secure Deduplication System based on Fixed Window Fixed Byte Chunking and Semantic Weight Poisson Process Filter in Cloud Storage." Frontiers in Health Informatics 13.5 (2024).
13) Ye, Xuming, et al. "Context-aware resemblance detection for data deduplication with neural network." Engineering Applications of Artificial Intelligence 144 (2025): 110116.
14) Arora, R. I. C. H. A., and D. Vetrithangam. "Advancements in Deduplication Techniques for Efficient Data Storage." Journal of Theoretical and Applied Information Technology 102.5 (2024).
15) Basappa Kodada . FSAaCIT: Finite State Automata based One-Key Cryptosystem and Chunk-based Indexing Technique for Secure Data De-duplication in Cloud Computing. TechRxiv. August 16, 2022.
16) Lee, M., & Seo, M. (2022). Secure and Efficient Deduplication for Cloud Storage with Dynamic Ownership Management. Applied Sciences, 13(24), 13270. https://doi.org/10.3390/app132413270.
17) Sreeharsha Udayashankar, Abdelrahman Baba, Samer Al-Kiswany, "Accelerating Data Chunking in Deduplication Systems using Vector Instructions", Submitted on 7 Aug 2025, https://doi.org/10.48550/arXiv.2508.05797.

18) Marcel Gregoriadis, Leonhard Balduf, Björn Scheuermann and Johan Pouwelse, "A Thorough Investigation of Content-Defined Chunking Algorithms for Data Deduplication," arXiv:2409.06066v3 [cs.DC] 28 Sep 2024.
19) Gan, Chuang, et al. "Coupling Secret Sharing with Decentralized Server-Aided Encryption in Encrypted Deduplication." Applied Sciences 15.3 (2025): 1245.
20) Ellappan, Manogar, and Abirami Murugappan. "A smart hybrid content-defined chunking algorithm for data deduplication in cloud storage." Soft Computing 28.15 (2024): 9037-9052.
21) Kavitha, Ranga, et al. "Data Deduplication-based Efficient Cloud Optimisation Technique: Optimizing Cloud Storage through Data Deduplication." This work is open access and licensed under the Creative Commons CC BY 4.0 License. Volume 17 (2025), Issue 2.
22) Ahmad, Shahnawaz, et al. "Convergent encryption enabled secure data deduplication algorithm for cloud environment." Concurrency and Computation: Practice and Experience 36.21 (2024): e8205.
23) Mohiuddin, Mohd Hasan, and Latha Tamilselvan. "Optimizing cloud infrastructure efficiency through advanced multimedia data deduplication techniques." Bulletin of Electrical Engineering and Informatics 14.4 (2025): 2823-2836.
24) Periasamy, J. K., et al. "Enhancing cloud security and deduplication efficiency with SALIGP and cryptographic authentication." Scientific Reports 15.1 (2025): 30112.
25) Ahamed, N. Niyaz, and N. Duraipandian. "Secured Data Storage Using Deduplication in Cloud Computing Based on Elliptic Curve Cryptography." Computer Systems Science & Engineering 41.1 (2022)