SEMARAK ILMU PUBLISHING

Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

Content-Defined Chunking Algorithms in Data Deduplication: Performance, Trade-Offs and Future-Oriented Techniques

Safa Ali Abdo Hussein[1,2,*], R. Badlishah Ahmad[1,3], Naimah Yaakob[1,3], Fathey Mohammed[2,4], Abdul Ghani Khan[5]

[1] Faculty of Electronic Engineering Technology, Universiti Malaysia Perlis, Pauh Putra Campus, 02600 Arau, Perlis Malaysia
[2] Faculty of Engineering and Information Technology, Taiz University, Taiz 6803, Yemen
[3] Centre of Excellence for Advanced Computing, Universiti Malaysia Perlis, Pauh Putra Campus, 02600 Arau, Perlis Malaysia
[4] Sunway Business School, Sunway University, 47500 Subang Jaya, Selangor, Malaysia
[5] School of Computing and Communication, Lancaster University, Bailrigg, Lancaster LA1 4WA, United Kingdom

ARTICLE INFO

Article history:
Received 22 November 2023
Received in revised form 27 May 2024
Accepted 28 August 2024
Available online 1 October 2024

Keywords:

Data deduplication; Chunking method; Content-defined chunking; Hashing-based algorithms; Hash-less algorithms

ABSTRACT

In the digital era, the exponential growth of data presents significant challenges for storage efficiency and processing speed. This paper reviews Content-Defined Chunking (CDC), a cornerstone in data deduplication technology, aimed at addressing these challenges. We systematically examine various CDC algorithms, categorising them into hashing-based and hash-less methodologies, and evaluating their performance in deduplication processes. Through a critical analysis of existing literature, the study identifies the balance between chunking speed and deduplication efficacy as a pivotal area for enhancement. Our findings reveal the need for innovative CDC algorithms to adapt to the evolving data landscape, proposing future research directions for improving storage and processing solutions. This work contributes to the broader understanding of data deduplication techniques, offering a pathway towards more efficient data management systems.

1. Introduction

The proliferation of interconnected devices is expected to substantially extend worldwide networks; according to a previous study [1], the GSM Association projects that by 2025, more than 1.7 billion users will be connected. The projected expansion is anticipated to generate a significant increase in the volume of data, with approximations placing it at 180 zettabytes globally. As a result, the current capacities of transmission and storage systems will be challenged. Significantly, studies suggest that up to 70% of this data could be redundant. This emphasizes the critical necessity for novel approaches to data management, which aim to optimise storage and transmission efficiency in light of the increasing digital overload, as indicated in recent research [2,3].

* *Corresponding author.*
E-mail address: safaali@studentmail.unimap.edu.my

https://doi.org/10.37934/araset.52.1.2134

In this context, data deduplication emerges as a promising technique for reducing data redundancy. By eliminating duplicate data instances and preserving only unique elements, data deduplication optimizes storage capacity and enhances storage platform productivity [4]. Its primary objectives include conserving storage space and reducing network traffic and bandwidth requirements, as thoroughly analysed by Prajapati and Shah [5] and further explored by Randall and Lu [6]. The deduplication process, encompassing stages such as chunking, fingerprinting, indexing, and storage management, is foundational to efficient data management [7,8]. Specifically, the chunking phase, where data is segmented into smaller pieces for more effective duplicate identification and elimination, plays a crucial role in the initial stages of deduplication, as discussed by Khaing and Jeyanthi [9] and Zhang *et al.,* [10]. By following these first steps, the process of indexing chunk fingerprints allows for the accurate detection of duplicate and unique data chunks. This method has been extensively described by Guan [11] and Naresh *et al.,* [12], emphasizing the significance of advanced indexing techniques in the deduplication process.

Data deduplication techniques vary in their approach to segmenting data, with three primary methods based on chunk size: whole file, fixed-size, and variable-size chunking. Whole file chunking treats the entire data object as a single chunk, offering simplicity and speed at the cost of lower deduplication efficiency [13]. Fixed-size chunking segments data into equal-sized chunks, improving deduplication effectiveness compared to whole file chunking, though it struggles with identifying optimal chunk boundaries [14]. Variable-size chunking, which creates chunks of differing sizes, stands out for its superior performance despite its complexity and computational demands [15].

The boundary shift problem, a significant challenge for both whole file and fixed-size chunking methods, occurs when modifications within a file shift the byte offsets of chunks, thereby misidentifying unique chunks as duplicates [16]. This problem highlights the limitations of relying on byte offsets for distinguishing between chunks [17]. In contrast, content-defined chunking (CDC), which segments files based on content rather than byte offsets, maintains consistent chunk identification even when files are altered, significantly improving deduplication ratios [18].

In the context of content-defined chunking (CDC) approaches, the utilization of the Rabin hash sliding window technique is pivotal for determining chunk boundaries. This method, despite its effectiveness, is noted for its computational intensity due to the need for hashing each byte in the data stream, as detailed by Guo *et al.,* [19]. The variability in chunk sizes, a direct outcome of hash function properties, poses efficiency challenges, as discussed by Xu and Zhang [20]. Moreover, existing CDC algorithms confront ongoing hurdles in reducing chunk variance and computational overhead, with notable issues including high computational demand, CPU overhead, and challenges in processing low-entropy strings, as explored in various studies [21-24]. These challenges underscore the necessity for advancements in CDC techniques to balance efficiency and computational resources effectively.

This study narrows its focus on the chunking stage of data deduplication, emphasizing Content-Defined Chunking (CDC) for its potential to increase deduplication ratios by resolving the boundary shift problem inherent in fixed-size chunking. Despite the pivotal role of chunking in the efficiency of data deduplication, there exists a noticeable gap in the literature—a comprehensive comparison and evaluation of CDC techniques. Our work contributes a detailed review of both hashing-based and hash-less CDC algorithms, dissecting their operational methodologies, benefits, and drawbacks. We aim to elucidate the selection of CDC methods for specific applications, address the challenges of chunk size variance and computational overhead, and chart a course for future research, including the integration of CDC in dynamic and heterogeneous data environments. The subsequent sections systematically explore chunking methodologies, recent advancements in CDC algorithms, outline key challenges and future directions for this crucial field and the last section is the review's conclusion.

2. Chunking Methodology

2.1 Chunking Method

The Chunking Method stands as the foundational step in the data deduplication process, playing a critical role in determining the process's overall efficiency. As outlined by Ellappan and Abirami [25], the method encompasses several approaches including whole file chunking, fixed-size chunking, and content-defined chunking, also known as variable size chunking, as we can see in Figure 1. These techniques are essential for categorizing data into manageable and comparable units.

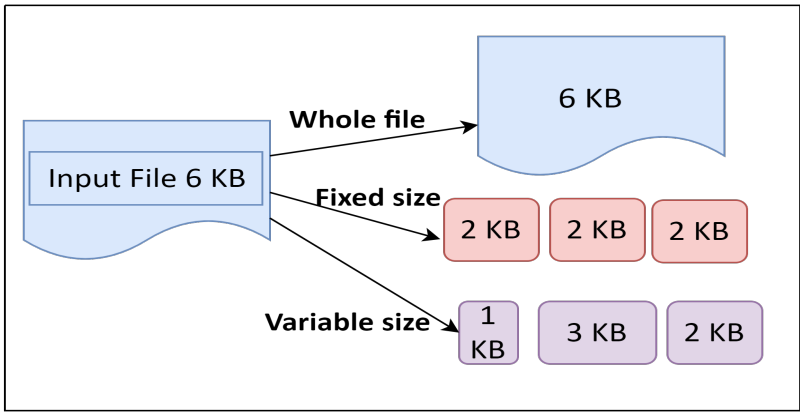

Fig. 1. Chunking Method Categories

2.2 Whole File Chunking

Whole file chunking operates on the principle of identifying redundant data at the file level, contrasting with methods that compare data within smaller segments of files. Cheng *et al.,* [26] describe this approach as comparing entire files against one another to detect duplicates, storing only a single instance of duplicate files while associating further references with metadata pointing to the stored copy, as depicted in Figure 2. This method is particularly advantageous in environments like file servers and archival storage due to its simplicity and the high processing speeds achieved by treating each file as a single chunk. However, Jehlol and George [27] note that the primary limitation of whole file chunking lies in its lower duplication detection ratio at the file level, as minor modifications in files necessitate the creation of new file versions, thereby complicating duplicate identification and potentially leading to inefficiencies.

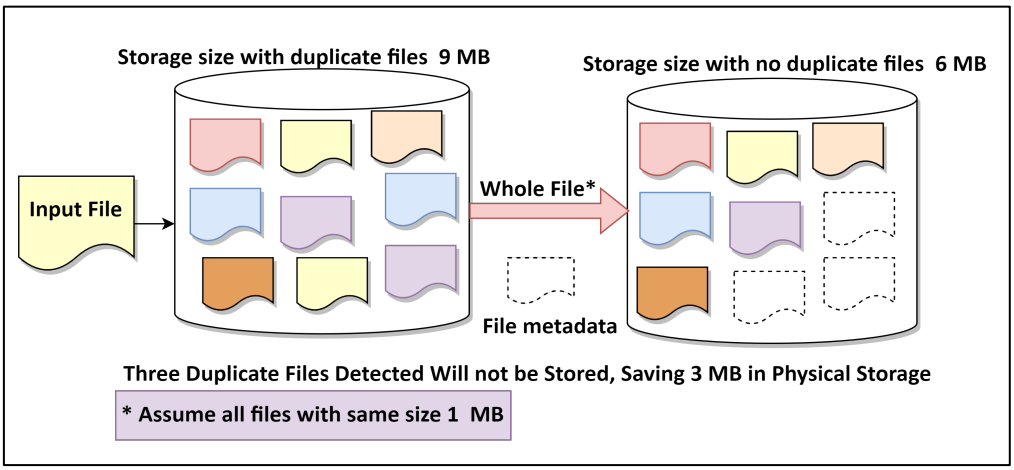

Fig. 2. Whole File Chunking

2.3 Fixed Size Chunking

Fixed-size chunking, as its name implies, involves dividing data into uniformly sized blocks for comparison, a method that Kumar *et al.,* [28] suggest can significantly aid in the detection of duplicate data. This technique enables direct comparison of each block to previously stored blocks of identical size as illustrated in **Error! Reference source not found.**. This method potentially enhancing the process's efficiency in terms of processing time and computational resources. Nevertheless, the approach is not without its drawbacks, including the boundary-shift problem, which arises when changes within a file necessitate the movement of other data chunks.

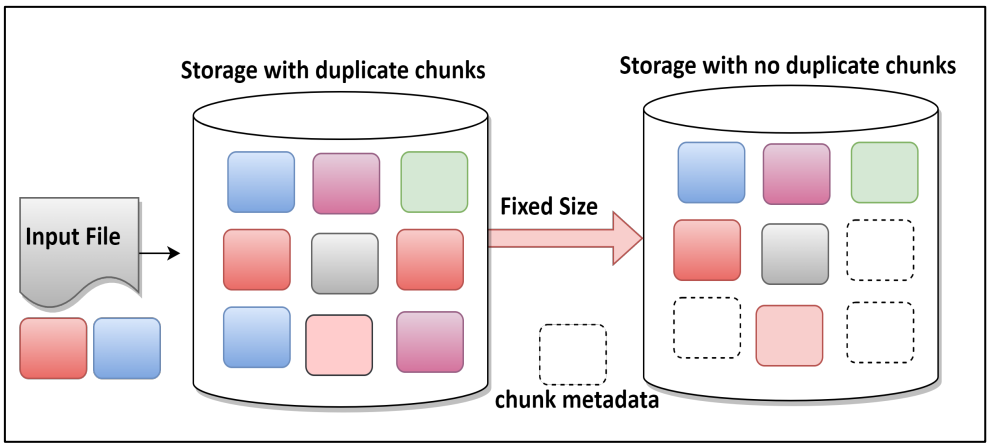

Fig. 3. Fixed Size Chunking

As demonstrated in Figure 4, even a single character insertion can trigger a shift in all subsequent data, rendering previously established block comparisons invalid and thereby compromising duplicate detection accuracy. This limitation, emphasized by Elouataoui *et al.,* [29], becomes particularly problematic when dealing with files of irregular or non-standard sizes. Moreover, the fixed-size chunking method may result in suboptimal storage utilization, as blocks are allocated fully to data elements, even if the actual data occupies only part of the block's capacity.

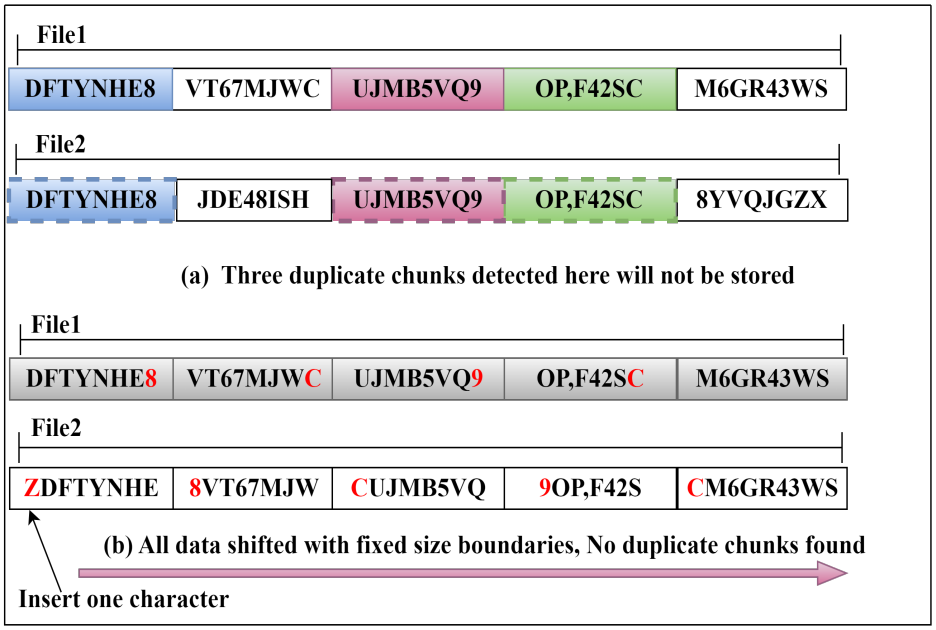

Fig. 4. Boundary Shift Problem

2.4 Variable Size Chunking

Variable size chunking utilizes a content-defined chunking (CDC) method to segment data into chunks of varying sizes. This approach, as described by Babu *et al.,* [30], effectively addresses the byte-shifting issue that plagues fixed-length chunking strategies. By employing the Rabin fingerprint algorithm, CDC defines chunk boundaries, adapting to the unique characteristics of each data segment (Figure 5).

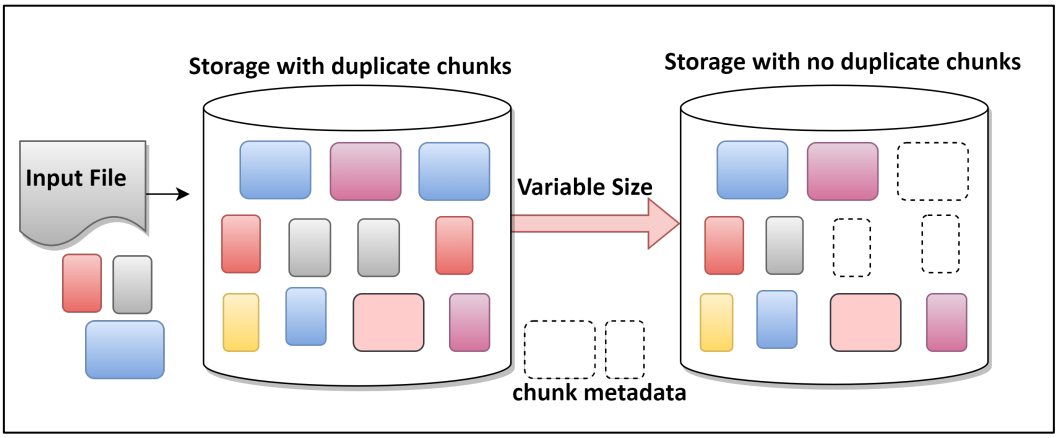

Fig. 5. Variable Size Chunking

As Nakamura *et al.,* [31] detail, this adaptability significantly enhances deduplication efficiency. Each chunk size aligns with the content within, as illustrated in Figure 6. This flexibility ensures resilience against modifications like insertions or deletions within the file, preserving the ability to efficiently identify duplicate chunks and achieve optimal storage utilization. Additionally, CDC can potentially enhance data transfer rates. Despite its advantages in both storage optimization and data management, CDC comes with the trade-off of requiring more intensive computational resources. As Jin *et al.,* [32] point out, the thorough file analysis required to determine individual chunk boundaries can lead to increased CPU utilization.

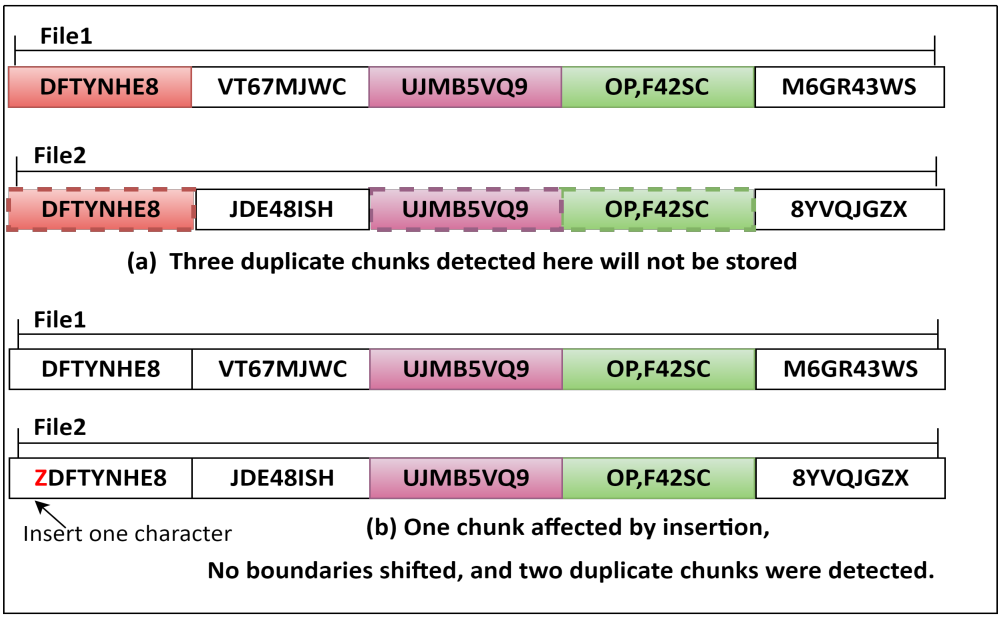

Fig. 6. Variable-Size Content-Based Chunking

2.5 Fixed vs. Variable Size Chunking

The CDC method, by determining chunk boundaries based on the data's local content, circumvents the boundary-shift problem observed with fixed-size chunking. This not only improves storage efficiency but also the overall system performance, by allowing data to be divided into smaller, more manageable chunks for more efficient storage and processing. Xia *et al.,* [33] emphasize CDC's design to withstand the challenges posed by file content alterations, such as insertions and deletions.

However, the computational demand of CDC, particularly due to the processing of content window size and hash value calculations, poses a challenge to deduplication efficiency. Jehlol and George [34] suggest that optimizing chunk size variance could enhance efficiency, noting that a lower variance in chunk size tends to yield better deduplication results. In contrast, fixed-size chunking, while faster and simpler, often results in lower deduplication efficiency due to its inability to adapt to data variability. Neelamegam and Marikkannu [35] argue for simplifying the chunking process to mitigate computational overhead and improve deduplication throughput in CDC implementations.

2.6 Comparison of Chunking Methods

The comparison of chunking methods, as summarized in Table 1, highlights the unique benefits and limitations associated with each technique. Whole file chunking treats each file as a single chunk, offering rapid processing speeds and minimal resource usage due to its straightforward approach. This method is particularly advantageous for environments where execution speed is prioritized over granular deduplication accuracy.

Fixed-size chunking divides data into uniformly sized chunks, facilitating efficient data processing and resource allocation. However, this method may encounter the boundary shift problem during data modifications such as insertions, deletions, or reorganizations, potentially impacting deduplication effectiveness as noted by Chhabra and Bala [36] and Kumar and Puli [37].

Variable-size chunking, or content-defined chunking (CDC), addresses these challenges by dynamically adjusting chunk boundaries based on the data's content, thereby enhancing the precision of duplicate detection. Rajkumar and Dhanakoti [38] demonstrate that CDC's adaptability to content variations results in superior duplication identification. Nonetheless, the complexity of comparing each chunk against multiple others of varying sizes introduces higher computational demands, which may slow processing times and increase resource requirements, as discussed by Ye *et al.,* [39].

Choosing the most suitable chunking method for data deduplication involves weighing factors such as resource utilization, processing speed, data variability, and duplication accuracy, tailored to the demands of the specific application scenario. Whole file chunking might suffice for tasks where deduplication is a secondary concern, like data migration. Fixed-size chunking represents a balanced choice for data backup applications, where it strikes a compromise between efficiency and computational simplicity. Variable-size chunking emerges as the preferred method for scenarios prioritizing high-level deduplication, such as in cloud storage solutions, due to its capacity for handling large datasets with varied content effectively. This assessment underscores the importance of selecting a deduplication strategy that aligns with the operational requirements and objectives of the deployment environment, considering the inherent trade-offs between complexity, efficiency, and resource demands associated with each chunking method.

Table 1
Comparison of Chunking Methods

Metrics	Whole File	Fixed size	Variable size
Chunk Size	Whole file	Fixed size	Variable size
Resource utilization	Less	More	More
Throughput	Low	High	Moderate to High
Deduplication ratio	Low	Low	High
Boundary shift problem	Yes	Yes	No
Computational complexity	Low	Moderate	High

3. CDC Techniques and Algorithms

CDC algorithms aim to strike a delicate balance between chunking speed, which measures the efficiency of dividing large datasets into smaller chunks, and deduplication ratio, which quantifies the data reduction achieved by eliminating duplicate information. However, this quest for optimization often involves a trade-off, where prioritizing one objective can adversely affect the other [40,41].

Smaller chunk sizes tend to lead to increased deduplication ratios by facilitating the identification of more duplicate segments within the dataset. However, this benefit comes at the expense of slower chunking speed due to the larger number of smaller chunks requiring analysis. Conversely, employing larger chunks may enhance chunking speed by reducing the overall number of chunks to process, but this gain often translates to a lower deduplication ratio as fewer opportunities arise to detect duplicates [42]. This inherent trade-off necessitates careful consideration during the design and implementation of CDC algorithms. Finding the optimal equilibrium between chunking efficiency and deduplication effectiveness depends on various factors, including the specific characteristics of the target dataset and the intended use case [43].

Several strategies have been proposed to address this challenge, encompassing both hash-based and hash-less techniques. Hash-based methods like Rabin fingerprinting efficiently identify potential duplicates by exploiting data patterns, but they might introduce computational overhead [44]. Hash-less approaches, like byte pair frequency analysis, offer alternative solutions but may encounter limitations in certain data types or require more complex processing steps [45]. The ongoing development of CDC techniques underscores the dynamic nature of data deduplication challenges. Continuous advancements are observed in:

3.1 Hashing-Based Algorithm

The Content Defined Chunking (CDC) method partitions a file into chunks by analysing the file's content to identify repeatable patterns, such as byte sequences or strings. These patterns act as predetermined breaking points. This analysis transforms a set of input bytes into an output hash, often referred to as a fingerprint [46]. A rolling hash function is employed to pinpoint these specific patterns within the data, generating a hash value for each data block and its adjacent blocks [47].

Research in the field of CDC algorithms has focused on overcoming the issue of low deduplication ratios by introducing hashing-based algorithms for chunking. Various studies have advocated for these algorithms, citing their ability to enhance deduplication ratios [19,20,24,44,49,50]. However, they demand considerable computational resources, particularly for large datasets, as they necessitate scanning the entire data stream to locate duplicate chunks. This intensive process results in increased processing times and reduced chunking speeds.

Rabin-based chunking algorithms, for instance, utilise Rabin fingerprints to determine chunk boundaries. This involves sliding a window of bytes across the data and computing a Rabin fingerprint

for each new position. A change in the fingerprint indicates a new chunk boundary. Despite its effectiveness in detecting duplicates, this technique requires detailed byte-by-byte analysis, resulting in significant time overhead [19].

A chunk boundary is confirmed when the fingerprint matches a specific boundary value, ensuring the chunk's size meets or exceeds the minimum threshold [20]. To achieve higher deduplication rates and efficiency, Xu and Zhang [20] developed the Quick CDC algorithm which uses techniques like chunk boundary jumping and dynamic chunk length adaptation. While these advancements improve resistance to byte shifting, they introduce complexities that could impact chunk performance, particularly where there is considerable variability in chunk sizes [24]. FastCDC offers solutions to speed up the CDC process, achieving processing speeds 3–12 times faster than traditional CDC methods without sacrificing deduplication rate, thereby reducing the computational burden of establishing chunk boundaries [24].

Further innovations include the algorithm by Saeed and George [44], which divides files based on the frequency of byte pairs (BFBC) to improve deduplication speed and storage efficiency. Although BFBC shows improvement, it also introduces issues such as increased processing times and memory requirements, with slight data modifications potentially affecting deduplication ratios. The Two Thresholds Two Divisors (TTTD) algorithm by Eshghi and Tang [49] builds on Rabin's method for eliminating redundant data by adding two divisors and thresholds, thus managing the generation of large chunk sizes. Despite enhancing deduplication performance, TTTD may introduce complexity and reduce effectiveness in some scenarios of redundant data management. Additionally, Yang *et al.,* [50] have developed the Dynamic Asymmetric Maximum (DAM) algorithm, which uses maximum value chunk boundaries and perfect hash techniques to enhance chunk search efficiency in scenarios with large block sizes and low-entropy patterns, though it may face challenges in detecting specific patterns. Table 2 summarises the key features of hash-based algorithms, highlighting the balance between computational efficiency and deduplication performance.

Table 2
Summary of Hash-based Chunking Methods based: Techniques, Advantages, and Disadvantages

Method	Technique	Advantages	Disadvantages
Double Sliding Window (DSW) [19]	Divides the data stream into two fixed size sliding windows, using Rabin fingerprinting for hashing comparison.	Low memory usage, low false-positive rate, high deduplication ratio.	Not suitable for frequently changed data, computationally expensive for large datasets.
QuickCDC [20]	Jumps straight to chunk boundaries in cases of duplicate chunks.	Fast chunking and a higher deduplication ratio with redundant datasets.	Not effective for datasets with a lot of unique data.
Super CDC [21]	Combines two acceleration mechanisms: small chunk cut point skipping and predicting where boundaries are likely to occur.	Reduces chunking computations, achieves a higher deduplication ratio than state-of-the-art gear-based methods.	Requires more memory to store stream history, high processing power needed, cannot recognize small or large duplicate chunks.
FastCDC [24]	Utilizes five different techniques for optimization.	About 3 to 12 times faster than traditional CDC's chunking speed, data removal percentage is close to the classic Rabin method.	Chunk Size Variability complex computations and optimization techniques demand more memory and CPU resources,
Jump-based chunking (JC) [32]	speculative jump technique in the sliding window, jumps based on the hash condition.	Reduces the number of hash calculations and speeds up the chunking process.	May reduce the deduplication ratio, introduces false positives, requires more storage.

Window Chunking Signature Encryption (WCSE) [35]	Separates data, encrypts it with a unique signature using MD5 hash function, and saves it in the cloud.	Efficient segment signature analysis outperforms competing deduplication algorithms.	Increasing computing expenses with large datasets, computational difficulty, and increased processing time
Bytes Frequency Based Chunking (BFBC) [44]	Divides files based on byte pair frequency	Three times faster chunking speed than TTTD, faster hashing algorithm	Computational overhead, high memory requirements, low deduplication ratio
Two Thresholds Two Divisors (TD &TTTD) [49]	Two divisors for boundary identification, Divides data into equal-sized chunks.	High deduplication ratio, efficient entropy removal, and reduces chunk size	High chunk size variance, inefficiency in handling high redundancy data
Dynamic Asymmetric Maximum (DAM) [50]	Uses a maximum value as a chunk boundary, perfect hash algorithm	Detects low-entropy strings in redundant data, slightly improves deduplication rate	Time complexity, incapability to handle large data sets, and low efficiency when using substring comparison

3.2 Hash-Less Algorithm

The hash-less CDC algorithm differs from hashing-based algorithms as it does not use hash values to identify duplicate data chunks. Instead, it employs byte values within a sliding window to perform the chunking process. This approach eliminates the need to compute hash values, significantly reducing the computational complexity involved in deduplication and improving chunking throughput, which has documented the method's efficiency and speed [22,23,25,51-54]. Despite these advancements, the same body of research indicates certain limitations. Notably, the process of boundary shifting—integral to the algorithm's operation—can change the cut points for chunks. Such alterations may inadvertently lead to the exclusion of duplicate chunks, potentially compromising the algorithm's efficacy in deduplication [22,23,25]. Additionally, a critical challenge arises from the potential for significant variance in data chunk sizes. Since the algorithm's chunking mechanism is predicated on the distribution of byte values within the sliding window, an uneven distribution can result in considerable inconsistencies in chunk sizes [51-54]. This variability may undermine the algorithm's efficiency, particularly in its capacity to uniformly identify and eliminate duplicate data chunks. Thus, while the hash-less CDC algorithm offers notable advantages, including reduced computational demands and improved chunking throughput, it also faces challenges in accurately detecting duplicate chunks. These challenges highlight areas for potential improvement and further research, particularly in optimizing the algorithm to address the issues of chunk size variance and the potential omission of duplicates. The existence of such limitations underscores the need for a balanced assessment of the hash-less CDC algorithm's performance compared to traditional hashing-based algorithms. Despite these challenges, the hash-less CDC algorithm's reduced computational demands and increased chunking throughput present notable advantages. Yet, the identified limitations necessitate a balanced evaluation of its performance relative to traditional hashing-based methods. Specifically, addressing chunk size variance and the potential for missing duplicates are critical areas for future enhancement. Table 3 provides a summary of some hash-less algorithms, serving as a resource for comparing these approaches and underscoring the ongoing need for research aimed at refining deduplication technologies.

Table 3
Summary of Hash-less Chunking Methods: Techniques, Advantages, and Disadvantages

Method	Technique	Advantages	Disadvantages
Asymmetric Extremum (AE) [22]	Uses a fixed-size right window and a variable-sized fixed window.	Low computational overhead, high throughput, smaller chunk size, ability to eliminate low-entropy string	Less resistance to byte shifting, more time to process chunks, and a small improvement in deduplication
UltraCDC [23]	Use four techniques compute boundary, skipping sub-minimum size, normalising, and jumping	Reduces overhead of hash calculation, detects low-entropy strings, and has faster processing compared to Rabin and AE.	Low deduplication ratio, possibility of false positives or missed duplicates, storage space requirements
Dynamic Prime Chunking (DPC) [25]	Dynamically adjusts window size based on duplicate chunks	Improves throughput, low computational overhead, reduces processing time, avoids large chunk	High computational complexity, less resistant to boundary problem
Maximum Points /Local Maximum Chunking (MAXP/LMC) [51]	Divides data using local extreme values	Timesaving, high throughput, resistance to data changes	High chunk size variance, limitation in entropy removal, slow processing
Rapid Asymmetric Maximum (RAM) [52]	Uses a fixed-sized left window and a variable-sized right window.	low computational expenses, productivity of chunking is high. High chunking speed.	Problem of boundary shifting.
Minimal Incremental Interval (MII) [53]	based on incremental data synchronization	Reduce time overhead, manage boundary-shift problem	Variance chunk size and inefficient chunking
Double Extreme and Rapid Double Extreme (DE & RDE) [54]	Uses byte values in a sliding window to the determine cut point.	Better handling of low entropy strings improves chunking throughput.	High computational complexity, slow processing time, less resistance to byte shifting, chunk size variance

4. Challenges and Open Issues of CDC Algorithms

Despite the effectiveness of numerous CDC algorithms, several challenges and open issues still necessitate systematic study and further development. This section addresses significant challenges in CDC algorithms that impact deduplication efficiency, which in turn affects storage space utilization and network transmission efficiency. The creation of an efficient chunking algorithm can enhance storage utilization, cost efficiency, and presents substantial utility and research value. The gap between the throughput of existing algorithms and the capabilities of current storage devices represents a significant challenge, highlighting the crucial role of data chunking research in improving data deduplication system performance, conserving CPU resources, and optimizing storage device capacity. The following sections outline pivotal areas for future investigation aimed at overcoming these challenges and propelling CDC technology forward:

4.1 Variance in Chunk Size

The variability in chunk sizes presents a significant challenge for CDC techniques, which strive to deduplicate similar data by segmenting it based on content. Balancing chunk granularity with

deduplication efficiency is a complex task: excessive chunking can increase processing time, whereas insufficient chunking may reduce throughput. Future research should focus on developing adaptive chunking strategies that dynamically adjust to data characteristics. Investigating machine learning models that predict optimal chunk sizes could offer groundbreaking improvements in CDC efficiency.

4.2 Dynamic Data Changes

CDC algorithms must be designed to adapt to changes in data content, especially in environments characterized by frequent updates. This adaptability is essential for maintaining efficient chunking and deduplication processes over time. Research in this area could explore the development of algorithms that more effectively identify and adjust to data mutations, thereby enhancing the longevity and efficacy of deduplication strategies.

4.3 Reducing CPU Overhead

The intensive process of detecting chunk boundaries in CDC algorithms significantly contributes to CPU resource consumption. Addressing this challenge requires exploring methods to reduce computational demands. Future investigations could explore the potential of hardware acceleration, leverage distributed computing architectures, or optimize algorithmic efficiency to reduce CPU load.

4.4 Stream Data Processing

Adapting CDC algorithms for real-time streaming data processing poses distinct challenges due to the continuous nature of the data. Developing strategies for efficient, real-time chunking and deduplication is an important area of research.

5. Conclusions

This review offers a comprehensive examination of CDC algorithms, elucidating their core concepts, methodologies, advantages, and limitations. It has identified critical challenges facing these algorithms, such as chunking speed, chunk size variance, chunk boundary detection, and deduplication ratio. Proposed solutions to these challenges are discussed, along with observations on their effectiveness. The review also highlights the complex implementation issues faced by CDC algorithms and outlines state-of-the-art research aimed at addressing these challenges to improve CDC algorithm efficiency and performance. The need for focused research on designing efficient CDC algorithms that navigate the complexities of hashing and hash-less byte shifting to achieve superior performance while reducing storage demands is evident.

Acknowledgement
Funding for this research was provided by Taiz University. The author expresses gratitude for this support, which has facilitated her PhD research without financial constraints. Special thanks to her PhD advisor, Prof. Ir Dr R. Badlishah Ahmad, for his invaluable guidance and mentorship, which have significantly contributed to the development of her research skills and critical thinking.

References

[1] Iqbal, Muhammad Saqib, Zulhasni Abdul Rahim, and Naoki Ohshima. "Enhancing Workforce Performance and Applications Toward Industry 5.0 with the 5G Conceptual Framework in Malaysia." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 31, no. 3 (2023): 282-289. https://doi.org/10.37934/araset.31.3.282289

[2] Adhab, Ayad Hasan, and Naseer Ali Hussien. "Techniques of Data Deduplication for Cloud Storage: A Review." (2022). https://doi.org/10.31695/IJERAT.2022.8.4.2

[3] Jagadeesh, Selvaraj, Sabna Machinchery Ali, Soundara Pandian Gnana Selvan, Mohammad Aljanabi, and Manimaran Gopianand. "Hybrid AES-Modified ECC Algorithm for Improved Data Security over Cloud Storage." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 32, no. 1 (2023): 46-56. https://doi.org/10.37934/araset.32.1.4656

[4] Huang, Darong, Jinfu Zhou, Bo Mi, Fengtian Kuang, and Yang Liu. "Key-based data deduplication via homomorphic NTRU for internet of vehicles." *IEEE Transactions on Vehicular Technology* 72, no. 1 (2022): 239-252. https://doi.org/10.1109/TVT.2022.3205627

[5] Prajapati, Priteshkumar, and Parth Shah. "A review on secure data deduplication: Cloud storage security issue." *Journal of King Saud University-Computer and Information Sciences* 34, no. 7 (2022): 3996-4007. https://doi.org/10.1016/j.jksuci.2020.10.021

[6] Randall, Owen, and Paul Lu. "Predicting Deduplication Performance: An Analytical Model and Empirical Evaluation." In *2022 IEEE International Conference on Big Data (Big Data)*, pp. 319-328. IEEE, 2022. https://doi.org/10.1109/BigData55660.2022.10020871

[7] Conde-Canencia, Laura, and Belaid Hamoum. "Deduplication algorithms and models for efficient data storage." In *2020 24th International Conference on Circuits, Systems, Communications and Computers (CSCC)*, pp. 23-28. IEEE, 2020. https://doi.org/10.1109/CSCC49995.2020.00013

[8] Godavari, Amdewar, Chapram Sudhakar, and T. Ramesh. "Hybrid deduplication system—a block-level similarity-based approach." *IEEE Systems Journal* 15, no. 3 (2020): 3860-3870. https://doi.org/10.1109/JSYST.2020.3012702

[9] Khaing, Me Me, and N. Jeyanthi. "DaDe: A Study of Chunking Algorithm and Hashing Function in Block Level Chunk based Data Deduplication." In *2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT)*, pp. 1-7. IEEE, 2022. https://doi.org/10.1109/GCAT55367.2022.9972150

[10] Zhang, Changjian, Deyu Qi, Wenlin Li, and Jing Guo. "Function of content defined chunking algorithms in incremental synchronization." *IEEE Access* 8 (2020): 5316-5330. https://doi.org/10.1109/ACCESS.2019.2963625

[11] Guan, Haocheng. "Research on Deduplication Technology Based on B+ Tree Index and Hash Index." In *2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)*, pp. 887-894. IEEE, 2023. https://doi.org/10.1109/EEBDA56825.2023.10090509

[12] Naresh, V., K. Neelima, V. Sai Teja, B. Swathi, and Y. Vamsi. "Content based Hash Indexing Approach to Handle Data Deduplication over Cloud Data."

[13] Sujatha, G., and Jeberson Retna Raj. "A Comprehensive Study of Different Types of Deduplication Technique in Various Dimensions." *International Journal of Advanced Computer Science and Applications* 13, no. 3 (2022). https://doi.org/10.14569/IJACSA.2022.0130339

[14] Xue, Zilong, Huixun Qian, Lingling Shen, and Xiaotong Wu. "A comprehensive study of present data deduplication." In *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, pp. 1748-1754. IEEE, 2021. https://doi.org/10.1109/HPCC-DSS-SmartCity-DependSys53884.2021.00257

[15] Viji, D., and S. Revathy. "Comparative analysis for content defined chunking algorithms in data deduplication." *Webology* 18, no. SpecialIssue2 (2021): 255-268. https://doi.org/10.14704/WEB/V18SI02/WEB18070

[16] Lou, Hao, and Farzad Farnoud. "Data deduplication with random substitutions." *IEEE Transactions on Information Theory* 68, no. 10 (2022): 6941-6963. https://doi.org/10.1109/TIT.2022.3176778

[17] Kwon, Hyungjoon, Yonghyeon Cho, Awais Khan, Yeohyeon Park, and Youngjae Kim. "DeNOVA: Deduplication extended nova file system." In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1360-1371. IEEE, 2022. https://doi.org/10.1109/IPDPS53621.2022.00134

[18] Joe, C. Vijesh, and S. Shinly Swarna Sugi. "Comprehensive analysis of content defined de-duplication approaches for big data storage." In *2022 Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pp. 454-458. IEEE, 2022. https://doi.org/10.1109/I-SMAC55078.2022.9987434

[19] Guo, Shuai, Xiaodong Mao, Meng Sun, and Shuang Wang. "Double sliding window chunking algorithm for data deduplication in ocean observation." *IEEE Access* (2023). https://doi.org/10.1109/ACCESS.2023.3276785

[20] Xu, Zhen, and Wenbo Zhang. "Quickcdc: A quick content defined chunking algorithm based on jumping and dynamically adjusting mask bits." In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pp. 288-299. IEEE, 2021. https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00049

[21] Wan, Binzhaoshuo, Lifeng Pu, Xiangyu Zou, Shiyi Li, Peng Wang, and Wen Xia. "Supercdc: A hybrid design of high-performance content-defined chunking for fast deduplication." In *2022 IEEE 40th International Conference on Computer Design (ICCD)*, pp. 170-178. IEEE, 2022. https://doi.org/10.1109/ICCD56317.2022.00034

[22] Zhang, Yucheng, Hong Jiang, Dan Feng, Wen Xia, Min Fu, Fangting Huang, and Yukun Zhou. "AE: An asymmetric extremum content defined chunking algorithm for fast and bandwidth-efficient data deduplication." In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1337-1345. IEEE, 2015. https://doi.org/10.1109/INFOCOM.2015.7218510

[23] Zhou, Peng, Zhenyu Wang, Wen Xia, and Haotong Zhang. "UltraCDC: A Fast and Stable Content-Defined Chunking Algorithm for Deduplication-based Backup Storage Systems." In *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pp. 298-304. IEEE, 2022. https://doi.org/10.1109/IPCCC55026.2022.9894295

[24] Xia, Wen, Yukun Zhou, Hong Jiang, Dan Feng, Yu Hua, Yuchong Hu, Qing Liu, and Yucheng Zhang. "{FastCDC}: A fast and efficient {Content-Defined} chunking approach for data deduplication." In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pp. 101-114. 2016.

[25] Ellappan, Manogar. "Dynamic Prime Chunking Algorithm for Data Deduplication in Cloud Storage." *KSII Transactions on Internet & Information Systems* 15, no. 4 (2021). https://doi.org/10.3837/tiis.2021.04.009

[26] Cheng, Geyao, Deke Guo, Lailong Luo, Junxu Xia, and Siyuan Gu. "LOFS: A lightweight online file storage strategy for effective data deduplication at network edge." *IEEE Transactions on Parallel and Distributed Systems* 33, no. 10 (2021): 2263-2276. https://doi.org/10.1109/TPDS.2021.3133098

[27] Jehlol, Hashem B., and Loay E. George. "Big Data De-duplication Using Classification Scheme based on Histogram of File Stream." In *2022 International Conference on Intelligent Technology, System and Service for Internet of Everything (ITSS-IoE)*, pp. 1-7. IEEE, 2022. https://doi.org/10.1109/ITSS-IoE56359.2022.9990942

[28] Kumar, PM Ashok, E. Pugazhendhi, and K. Vara Lakshmi. "Cloud Data Storage Optimization by Using Novel De-Duplication Technique." In *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 436-442. IEEE, 2022. https://doi.org/10.1109/ICSSIT53264.2022.9716508

[29] Elouataoui, Widad, Imane El Alaoui, Saida El Mendili, and Youssef Gahi. "An End-to-End Big Data Deduplication Framework based on Online Continuous Learning." *International Journal of Advanced Computer Science and Applications* 13, no. 9 (2022). https://doi.org/10.14569/IJACSA.2022.0130933

[30] Venkatesh Babu, S., P. Ramya, and Jeffin Gracewell. "Content Deduplication with Granularity Tweak Based on Base and Deviation for Large Text Dataset." *Scientific Programming* 2022 (2022). https://doi.org/10.1155/2022/9515181

[31] Nakamura, Yuta, Raza Ahmad, and Tanu Malik. "Content-defined merkle trees for efficient container delivery." In *2020 IEEE 27th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, pp. 121-130. IEEE, 2020. https://doi.org/10.1109/HiPC50609.2020.00026

[32] Jin, Xiaozhong, Haikun Liu, Chencheng Ye, Xiaofei Liao, Hai Jin, and Yu Zhang. "Accelerating Content-Defined Chunking for Data Deduplication Based on Speculative Jump." *IEEE Transactions on Parallel and Distributed Systems* (2023). https://doi.org/10.1109/TPDS.2023.3290770

[33] Xia, Wen, Xiangyu Zou, Hong Jiang, Yukun Zhou, Chuanyi Liu, Dan Feng, Yu Hua, Yuchong Hu, and Yucheng Zhang. "The design of fast content-defined chunking for data deduplication based storage systems." *IEEE Transactions on Parallel and Distributed Systems* 31, no. 9 (2020): 2017-2031. https://doi.org/10.1109/TPDS.2020.2984632

[34] Jehlol, Hashem Bedr, and Loay E. George. "Big Data Backup Deduplication: A Survey." (2022). https://doi.org/10.32628/IJSRSET229425

[35] Neelamegam, G., and P. Marikkannu. "Health Data Deduplication Using Window Chunking-Signature Encryption in Cloud." *Intelligent Automation & Soft Computing* 36, no. 1 (2023). https://doi.org/10.32604/iasc.2023.031283

[36] Chhabraa, Nipun, and Manju Balab. "An optimized data duplication strategy for cloud computing: Dedup with ABE and bloom filters." *International Journal of Future Generation Communication and Networking* 13, no. 1 (2020): 824-834.

[37] Kumar, Mr. M. A. R., and Mrs. S. Puli. "Finding Data Deduplication Using Cloud." YMER Digital 21, no. 05 (2022): 136-142. https://doi.org/10.37896/YMER21.05/17

[38] Rajkumar, K., and V. Dhanakoti. "Methodological Methods to Improve the Efficiency of Cloud Storage by applying De-duplication Techniques in Cloud Computing." In *2020 2nd International Conference on Advances in Computing,*

Communication Control and Networking (ICACCCN), pp. 876-884. IEEE, 2020. https://doi.org/10.1109/ICACCCN51052.2020.9362940

[39] Ye, Xuming, Jia Tang, Wenlong Tian, Ruixuan Li, Weijun Xiao, Yuqing Geng, and Zhiyong Xu. "Fast variable-grained resemblance data deduplication for cloud storage." In *2021 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pp. 1-8. IEEE, 2021. https://doi.org/10.1109/NAS51552.2021.9605398

[40] Ellappan, Manogar, and Abirami Murugappan. "A smart hybrid content-defined chunking algorithm for data deduplication in cloud storage." *Soft Computing* (2023): 1-16. https://doi.org/10.1007/s00500-023-09290-7

[41] Begum, MD Jareena, and B. Haritha. "Data Deduplication Strategies in Cloud Computing." *International Journal of Innovative Science and Research Technology* 5, no. 8 (2020): 734-738. https://doi.org/10.38124/IJISRT20AUG238

[42] Zhang, Zihao, Huiqi Hu, Zhihui Xue, Changcheng Chen, Yang Yu, Cuiyun Fu, Xuan Zhou, and Feifei Li. "Slimstore: A cloud-based deduplication system for multi-version backups." In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 1841-1846. IEEE, 2021. https://doi.org/10.1109/ICDE51399.2021.00164

[43] Saharan, Shweta, Gaurav Somani, Gaurav Gupta, Robin Verma, Manoj Singh Gaur, and Rajkumar Buyya. "QuickDedup: Efficient VM deduplication in cloud computing environments." *Journal of Parallel and Distributed Computing* 139 (2020): 18-31. https://doi.org/10.1016/j.jpdc.2020.01.002

[44] Saeed, Ahmed Sardar M., and Loay E. George. "Data deduplication system based on content-defined chunking using bytes pair frequency occurrence." *Symmetry* 12, no. 11 (2020): 1841. https://doi.org/10.3390/sym12111841

[45] Saeed, Ahmed Sardar M., and Loay E. George. "Fingerprint-based data deduplication using a mathematical bounded linear hash function." *Symmetry* 13, no. 11 (2021): 1978. https://doi.org/10.3390/sym13111978

[46] Sujatha, G., and Jeberson Retna Raj. "Digital data identification for deduplication process using cryptographic hashing techniques." In *2021 International Conference on Intelligent Technologies (CONIT)*, pp. 1-4. IEEE, 2021. https://doi.org/10.1109/CONIT51480.2021.9498307

[47] Fazlina, M. A., R. Latip, H. Ibrahim, and A. Abdullah. "Systematic Literature Review: Leveraging Data Deduplication Strategies & Hashing Techniques to Eliminate Data Redundancy in Cloud Environments." International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE) 8, no. 1.4 (2019): 332-339. https://doi.org/10.30534/ijatcse/2019/5181.42019

[48] Ye, Xuming, Xiaoye Xue, Wenlong Tian, Zhiyong Xu, Weijun Xiao, and Ruixuan Li. "Chunk content is not enough: Chunk-context aware resemblance detection for deduplication delta compression." *arXiv preprint arXiv:2106.01273* (2021). https://doi.org/10.1109/DCC52660.2022.00103

[49] Eshghi, Kave, and Hsiu Khuern Tang. "A framework for analyzing and improving content-based chunking algorithms." *Hewlett-Packard Labs Technical Report TR* 30, no. 2005 (2005).

[50] Yang, Ye, Xiaofang Li, Dongjie Zhu, Hao Hu, Haiwen Du, Yundong Sun, Weiguo Tian, Yansong Wang, Ning Cao, and Gregory MP O'Hare. "A resource-constrained edge IoT device data-deduplication method with dynamic asymmetric maximum." *Intell. Autom. Soft Comput* 30, no. 2 (2021): 481-494. https://doi.org/10.32604/iasc.2021.019201

[51] Bjørner, Nikolaj, Andreas Blass, and Yuri Gurevich. "Content-dependent chunking for differential compression, the local maximum approach." *Journal of Computer and System Sciences* 76, no. 3-4 (2010): 154-203. https://doi.org/10.1016/j.jcss.2009.06.004

[52] Widodo, Ryan NS, Hyotaek Lim, and Mohammed Atiquzzaman. "A new content-defined chunking algorithm for data deduplication in cloud storage." *Future Generation Computer Systems* 71 (2017): 145-156. https://doi.org/10.1016/j.future.2017.02.013

[53] Zhang, Changjian, Deyu Qi, Zhe Cai, Wenhao Huang, Xinyang Wang, Wenlin Li, and Jing Guo. "MII: A novel content defined chunking algorithm for finding incremental data in data synchronization." *IEEE Access* 7 (2019): 86932-86945. https://doi.org/10.1109/ACCESS.2019.2926195

[54] Chao, Yun, and JinDian Su. "An Effective Way To Reduce Network Transmission In Backup System." In *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*, pp. 125-127. IEEE, 2022. https://doi.org/10.1109/MDM55031.2022.00038