Dynamic Prime Chunking Algorithm for Data Deduplication in Cloud Storage

Manogar Ellappan[1*] and Abirami S[2]
[1] Department of Information Science and Technology, College of Engineering
Anna University, Chennai, India
[e-mail: manogar@auist.net]
[2] Department of Information Science and Technology, College of Engineering
Anna University, Chennai, India
[e-mail: abirami_mr@yahoo.com]
*Corresponding author: Manogar Ellappan

Received May 31, 2020; revised February 11, 2021; revised February 8, 2021; accepted March 13, 2021; published April 30, 2021

Abstract

The data deduplication technique identifies the duplicates and minimizes the redundant storage data in the backup server. The chunk level deduplication plays a significant role in detecting the appropriate chunk boundaries, which solves the challenges such as minimum throughput and maximum chunk size variance in the data stream. To provide the solution, we propose a new chunking algorithm called Dynamic Prime Chunking (DPC). The main goal of DPC is to dynamically change the window size within the prime value based on the minimum and maximum chunk size. According to the result, DPC provides high throughput and avoid significant chunk variance in the deduplication system. The implementation and experimental evaluation have been performed on the multimedia and operating system datasets. DPC has been compared with existing algorithms such as Rabin, TTTD, MAXP, and AE. Chunk Count, Chunking time, throughput, processing time, Bytes Saved per Second (BSPS) and Deduplication Elimination Ratio (DER) are the performance metrics analyzed in our work. Based on the analysis of the results, it is found that throughput and BSPS have improved. Firstly, DPC quantitatively improves throughput performance by more than 21% than AE. Secondly, BSPS increases a maximum of 11% than the existing AE algorithm. Due to the above reason, our algorithm minimizes the total processing time and achieves higher deduplication efficiency compared with the existing Content Defined Chunking (CDC) algorithms.

Keywords: Content Defined Chunking, Dynamic Prime Chunking, Cloud Storage, Data Deduplication, Performance Evaluation, Throughput

1. Introduction

In recent CISCO and International Data Corporation (IDC) studies, three areas have been predicted as important challenges in the datasphere. Firstly, the ever-growing of data storage occurs in the cloud data center's in availability zone and regional zone. Secondly, the data growth happens in edge computing, and finally, the volume of data increases in a social media client system of portable devices and Internet of Things (IoT) devices. According to the study report, from 2018 to 2025, IDC estimates storage exponentially increases from 33 ZB to 173 ZB in the global market [1]. In the above scenario, managing data storage in the cloud becomes a challenging task. Hence, the data reduction technique is introduced, namely 'Data Deduplication', applied to archived and backup storage files [2]. The advantages of data deduplication are to minimize the redundancy and transmission of data, storage cost, and low bandwidth of the network [3]. The deduplication technique stores a single copy of data and the duplicate chunks are removed to achieve storage space efficiency [4, 5]. The method eliminates redundancies at the whole file or chunk level and removes redundant data through their cryptographically protected hash signatures using the SHA1 fingerprint algorithm. The data deduplication process breaks the input in the form of chunks [6, 7].

The input files are divided into chunks, and the duplicates are identified and processed into five stages. The stages are chunking, fingerprinting the hash value, indexing the hash, compression technique [8], and managing the data in an extensive storage system. The compression stage is optional among these stages because it applies only to the traditional compression approach like LZ compression and delta compression [9, 10]. Data deduplication plays a vital role in the final stage of the storage management. Therefore, data storage is categorized into a different specification. They reduce fragmentation, garbage collection, provide reliable data, and secure data [11, 12]. Nowadays, the data deduplication technique is widely applied at the chunk level to reduce the repeated data. The detection of deduplication at the chunk level has been performed in the existing approaches by using a byte-level sliding window. The window is small in size and helps to match the string by comparing a bytes in storage system. Hence, in comparison with file-level deduplication, chunk- level deduplication has more merits [13, 14].

The chunking method is efficient for breaking the data input stream into small pieces or chunks. The chunking method is the first stage of the deduplication system [15]. Chunking in data deduplication is processed in two types, fixed size and variable size [16]. The content position splits each chunk into a fixed size. The Fixed Size Chunking (FSC) is the fastest chunking approach that divides the file or data input stream into equivalent size [17]. The advantage of FSC made the chunking process easy and straightforward approach. The algorithm achieves a high-speed chunking method and high chunk throughput. The chunk level deduplication analysis shows the detection and removal of repeated data that occurs at a level of fine-granularity. FSC method provides a possible solution for the file-level chunking algorithm [18, 19].

The main disadvantage of FSC is less deduplication efficiency and face boundary-shift problems [20, 21]. Due to the boundary shifting problem, FSC fails to find duplication and elimination of chunks. New variable-size chunking methods are proposed, namely, Content-Defined Chunking (CDC) [22, 23] determines the boundary-shifting issues by declaring chunk boundaries depending on the local content of the data input stream. Based on the CDC's internal data content, the chunk will not change the chunk boundaries and data content. Hence, it will not affect the cloud-storage efficiency. Different CDC algorithms are available, among these, we focus on the CDC algorithm, such as Rabin [24], Two-Thresholds-Two-Divisors

(TTTD) [23], MAXP [25], and Asymmetric Extremum (AE) [26]. Pooranian et.al introduced new techniques, namely LEVER and RARE that are different algorithms used to enhance the chunk variance efficiently and the dynamic processing threshold for securing the CDC data deduplication in cloud storage [27, 28]. Next, we compare and analyze different CDC algorithms, the main list of key properties is given below [29].

Content defined: This algorithm solves the boundary shifting problem intelligently by deciding the breakpoint based on the predetermined condition. Here, it avoids the modification in the content of the file. This advantage makes a high impact in determining the duplicate chunks in the data input stream.

Low computational overhead: The CDC techniques involve measuring the chunk boundaries of each byte in a data input stream. The processing time is approximately proportional to the total amount of input data bytes, which can occupy enormous CPU resources. Therefore, it consumes more time and targets high deduplication performance.

Small chunk variance: In the chunking process, the CDC algorithm divides the chunk and stores the chunk into a disk using the deduplication. The long-sized chunk variance affects the performance of storage, and it additionally decreases the deduplication efficiency. CDC produces small chunk variance and improves deduplication elimination ratio.

Rabin degrades in the three issues, such as less throughput, high chunk variance, and minimum deduplication performance [24] because the data stream calculates byte by byte. The TTTD used smaller chunks but could not identify data deduplication for increased chunk size. Therefore, Rabin and TTTD are a time-consuming processes and to solve the issues, Bjørner et al. [25] introduced a new approach, namely MAXP. The MAXP algorithm applies two fixed size windows and an extreme value. The process cuts the chunk based on the breakpoints. Hash value calculation is not performed in MAXP. To overcome the issues of MAXP and Rabin, a new CDC approach, AE [26] is proposed. The AE processes the chunk breakpoint using fixed and variable size window. But, the sliding window is not used in AE. MAXP algorithm has more computational operations. The AE algorithm uses statistical properties and is implemented to reduce the computational overhead and reduces the storage cost [30].

In this work, we aim to minimize the computational overhead compared with the previous CDC algorithms. Therefore, to overcome the challenges of existing algorithms, we propose the Dynamic Prime Chunking algorithm (DPC), which reduces the computational overhead and improves chunking throughput. DPC is similar to AE but it uses variable size window rather than fixed size. The DPC applies two different windows, namely, variable size and dynamic variable size window. The DPC algorithm has no backtracking concepts to determine the extreme or maximum value. The bytes are scanned by using two conditional branches and one comparison operation. DPC determines min and max limitations on chunk size and has a higher advantage in reducing the low entropy bytes of other CDC algorithms. Our contributions to this paper are as follows:

(1) We propose DPC that dynamically varies the window size within the prime number based on the min and max threshold and a hash-less CDC algorithm and reduces duplication of chunks in the cloud storage.
(2) Our experimental evaluation of the proposed algorithm for multimedia and operating system datasets are illustrated and compared with other CDC algorithms.
(3) Our DPC analyzes the different performance metrics such as chunking count, processing time, chunking time, throughput, BSPS and DER with other CDC algorithms.
(4) From the above study of performance metrics, our proposed algorithm improves the

overall throughput and processing time in the deduplication system.

(5) Our proposed DPC algorithm minimizes the computational overhead compared with other existing CDC algorithms.

(6) We propose the novel DPC chunking algorithm to increase the performance of deduplication efficiency in cloud storage.

The rest of the paper is organized as follows: Section 2 discusses the background of existing algorithms and motivation. The design and workflow of the DPC algorithm are described in Section 3. Section 4 provides a detailed description of the experimental setup and datasets. Section 5 discusses the results and comparative analysis of the existing CDC with DPC. Finally, Section 6 concludes the research article with scope for future work.

2. Background

Section 2 provides a detailed study on the background of the chunking algorithm, the challenges of the existing CDC algorithm, and the motivation of our proposed work.

CDC method is the essential data element in the data deduplication, which splits the input data stream into small blocks or chunks and the data content window before the cut-point satisfies a predetermined condition. Moreover, the CDC algorithm identifies the duplication chunks, affected deduplication ratio, and performance of the storage system [31]. Muthitacharoen et al. [3] proposed an innovative content-defined chunking algorithm named Low Bandwidth File System (LBFS) to break the chunk based on content to solve the boundary-shifting problem. The algorithm works on the file content and calculates the hash value of the window by applying sliding window techniques. Then, it determines the chunk cut point as per the value of the window, which satisfies the predefined condition.

In CDC, most of the researchers used the Rabin algorithm [32] to break the chunk boundaries. The Rabin Fingerprint algorithm reduces the redundant network traffic and finds the chunk boundary based on Rabin rolling hash [33]. The algorithm computes a sliding window hash value along with a byte of the data, and the window moves every time in entire data bytes. Rabin declares a chunk breakpoint when the hash value of the window is equal to a predefined value. Hence, the new Rabin hash can be quickly determined from the old one, and it requires conditional controls 1 OR, 2 XOR, two left shifts, and two array lookups per byte [12]. Rabin algorithm avoids the small and large chunk size based on the minimum and maximum threshold. The primary deficiencies of Rabin-based CDC produces high chunk size variance and computational overhead, and less accuracy of duplicate detection. The small chunks create more fingerprints and occupy more storage space and high cost. But the long pieces affect the deduplication efficiency in backup storage

Due to the deduplication efficiency, Eshghi et al. [23] introduced the Two-Thresholds-Two-Divisors (TTTD) algorithm to defeat the Rabin chunk size variance. The working principle of the TTTD algorithm uses Basic Sliding Window Algorithm (BSW) technique. This algorithm uses two divisors (main, backup divisor) and implement these divisors to create a minimum and maximum chunk size. TTTD introduces a new threshold, the backup divisor to cut the chunks, which has a maximum probability of breakpoints and reduces the long-sized chunk variance [33]. The main benefit of the TTTD algorithm is the improvement of the deduplication ratio. But, using a backup divisor, it reduces chunking throughput and impacts low deduplication performance.

Next, we discuss the famous CDC algorithm named MAXP or Local Maximum Chunking (LMC) [12], the MAXP algorithm is primarily utilized in remote differential file compression techniques. The Rabin needs to calculate a hash value, but MAXP needs no hash value. It

handles the byte values by converting into digits, which supports minimizing of the computational overhead. MAXP uses two fixed-size windows and maximum local value. This technique applies to break the chunk cut-point to divide the input data stream. The main disadvantage of the MAXP algorithm is while determining an extreme value, the fixed-size window must move towards backtrack direction and checks the higher value of current byte position must be more than the local maximum point and this position helps to break the chunks. Therefore, MAXP using a backtracking approach needs a more conditional branch, and each byte verifies to increase the number of comparison operations. Checking every byte in the input stream and more conditional branch operations based on the resulting algorithm reduces chunking throughput [34].

Yucheng et al. [26] introduced a different CDC algorithm, namely, an Asymmetric Extremum (AE) to perform a fast deduplication method in the storage system of the cloud. AE is based on the maximum local approach or MAXP since it applies a digit as a number in the input stream of data. The chunk uses the two windows and helps AE to become less overhead computation than the MAXP algorithm [35, 36]. The process of AE has two windows, the first one is a fixed window with a byte inserted, and this will change the chunk. The next chunk disturbs the subsequent fragments. The algorithm puts the maximum value of the byte at each cut-point of the chunk and enters a new byte, which will not change the next chunk. Therefore, the process of the AE algorithm reduces the number of affected bytes. AE is intelligent in reducing low entropy bytes as it has a maximum data chunk size and achieves its larger chunk size with a long sequence. The algorithm delivers the maximum efficiency in deduplication and increases the chunking throughput [37].

Table 1. Key Properties of various CDC Algorithms

Chunking Algorithms	Key Properties					
	Content Defined	**Processing Time**	**Chunk Variance**	**Computational overhead**	**Efficient low-entropy strings**	**Limits on chunk Size**
Rabin	Y	H	H	H	N	Y
TTTD	Y	H	H	H	N	Y
MAXP	Y	H	L	H	N	N
AE	Y	M	L	L	H	N
DPC	Y	L	L	L	H	Y

L – Low, M – Medium, H – High, Y – Yes, N – No

Motivation: The primary role of the data deduplication process is chunk based because it applies to the cross-level file redundancy. CDC's higher deduplication system compares with FSC [26]. So the CDC algorithm is mostly used for chunk-level deduplication techniques. The critical issue of low chunking throughput affects deduplication efficiency [18, 39]. The various challenges the existing CDC chunking algorithm faces are the fundamental properties such as chunk variance, computational overhead, etc., and are tabulated in **Table 1**. By comparing the various features of Rabin, TTTD, MAXP, AE, we infer that our dynamic prime chunking algorithm proposed in this work shows the yes, low and high performances. Rabin faces some challenges as variance in chunk size, computational overhead, and it will not detect exact

duplicate. To solve the issue, TTTD introduces the Min-Max threshold to reduce the chunk variance, but the processing time is increased.

The MAXP approach is computationally high, and entropy strings can not be removed. Both the algorithms MAXP and Rabin face the computational overhead high and less chunking throughput. The drawback of AE is, it uses an asymmetric sliding window, and concentration is less for reducing the chunk time variance. AE produced significant chunk variance to decrease deduplication performance. Our proposed work DPC solves the issue of the existing four CDC algorithms. We propose a novel DPC algorithm, which increases the chunking throughput by providing similar deduplication performance. The DPC works are based on the local maximum value and a dynamic variable window without backtracking and solves the boundary shifting issue.

3. Dynamic Prime Chunking Algorithm

Next, we discuss the detailed design, the workflow of the DPC chunking algorithm, and its fundamental vital properties.

3.1 DPC Design

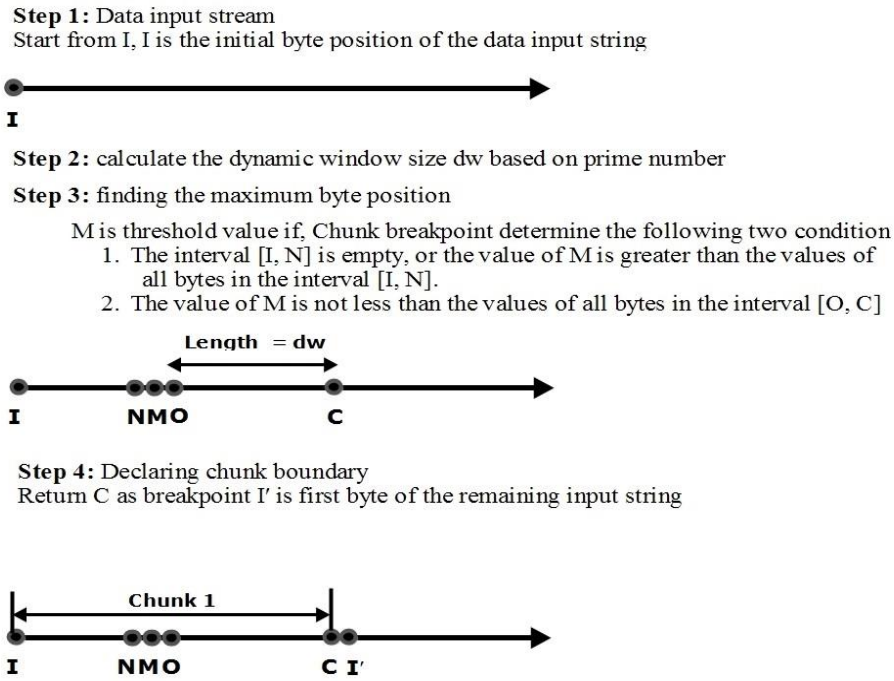

Fig. 1. The Design of DPC algorithm

The dynamic prime chunking algorithm improves the version of AE. DPC mainly applies to two essential properties, position and value. DPC design uses four steps, as presented in **Fig. 1**. First, read the data input stream from the source side. Start from I, where I is the initial byte position of the data input stream. Next, in step 2 we calculate the dynamic window (DW) size using prime numbers by applying step 3 of the algorithm. DPC works with two windows:

variable size and dynamic variable size window. The algorithm determines the maximum value or threshold (M) is the minimum or maximum value of the input stream. The method assigns the threshold value, which is always the extreme or maximum value. The third step determines the maximum byte value and finds the chunk boundary based on the two following conditions:

(1) To verify the interval [I, N] is also empty, or M maximum threshold value is more significant than all byte values within [I, N].
(2) In the dynamic variable-sized window, the extreme value M is not less than the value of all bytes between [O, C].

The initial byte is determined to satisfy the above conditions associated with a threshold value ensuring the highest byte point represented as the maximum local value. In contrast, the maximum byte position has been determined, DPC declaring the right-most byte as a chunk breakpoint in the right side window. Once the chunk boundaries are declared, the algorithm returns the breakpoint position C in step four. Next, the initial byte position sequence continues as I. Repeat the above following steps until finding the last chunk boundary of the input data stream. Next, we discuss the implementation of the DPC algorithm based on workflow.

Algorithm 1: Dynamic Prime Chunking (DPC)

```
Input: input data string, Str; Remaining length of file, RL;
Output: Chunk break point, CPi
Predefined values: Prime_chunk_size [ ] = {2, 3, 5, 7, 11}
                   PRCH 5, // prime chunk value is 5
                   Chunk Counter CC =0
                   e = 2.718281828  // exponential value
                   Chunk_Avg_Size = 6 KB  // (2+3+5+7+11)/5 = 5.6 KB ≈ 6 KB
function DPC_CHUNKING (Str, RL)
    // Calculate Dynamic Window Size
    DW = 1024 * Prime_chunk_size [ CC % PRCH] / (e-1)
    CPi ← 1
    Max_Value ← Str [CPi ].POS
    Max_Position ← CPi
    CPi ← CPi + 1
    while CPi < RL do
        if Str[CPi].POS < = Chunk_Avg_Size then
            if CPi = Max_Position + DW then
                return CPi
            end if
        else
            Max_Value ← Str[CPi ].POS
            Max_Position ← CPi
        end if
        CPi ← CPi + 1
    end while
return RL
end function
```

DPC Algorithm Description: The main goal of the DPC algorithm is to divide the input string into chunks dynamically based on prime numbers. The input of the algorithm takes two parameters the entire input string and the remaining length of the file RL. Five values are predefined in the initial step of the algorithm, such as prime chunk size, prime chunk value, chunk counter, exponential value (e), and average chunk size. Based on existing work [25],

the value e is assigned in the algorithm. Step 3 calculates the dynamic window by using the predefined values. To define the maximum value and position, steps 5 to 6 is processed and assign the max value and position. The current location assigns to 1 and Step 8 to 18 repeats the while loop by comparing the current position and RL until it reached a false condition. In the meantime, the CPi string is compared with the average chunk size as specified in step 8. Steps 10 to 17 determine the CPi by applying maximum value and dynamic window for the given input string. The comparison made for the entire string and step 19 of the DPC function returns the remaining length RL. The computational complexity of DPC is O(n).

3.2 Workflow of DPC

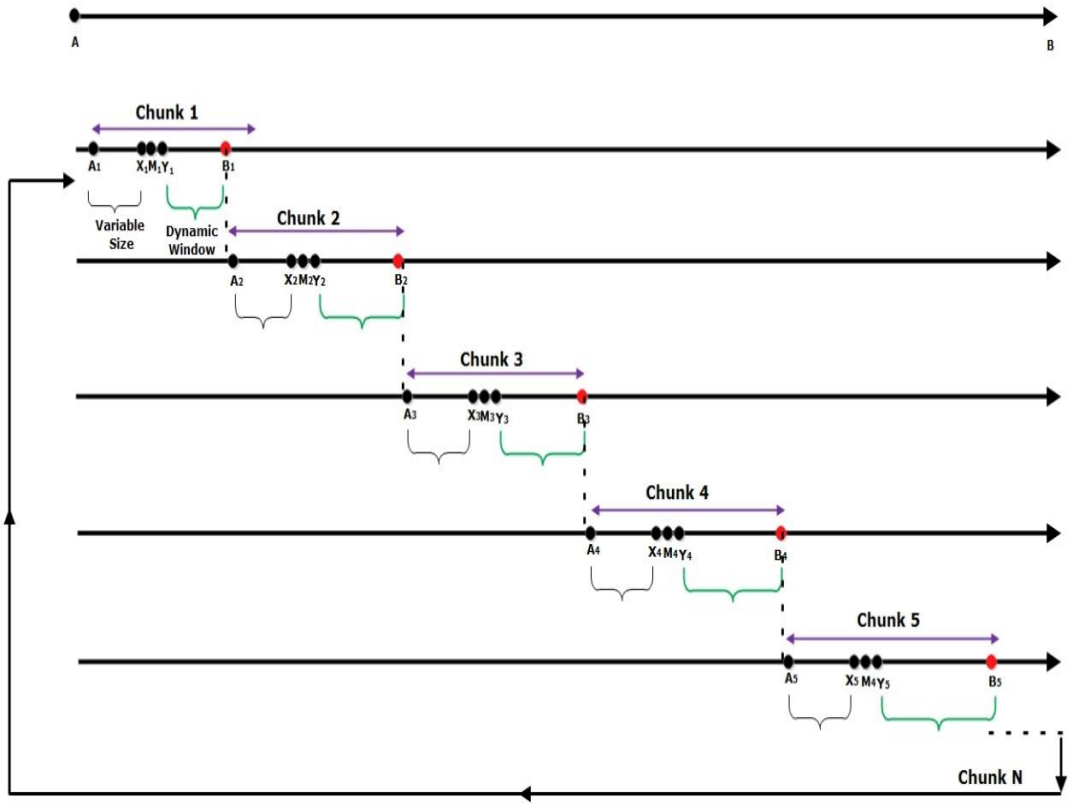

Fig. 2. The workflow of DPC algorithm

In the given **Fig. 2**, the initial byte position is denoted as A_1, and it moves forward in right direction till the end of the byte position B. The entire data stream is divided based on the threshold value M_1. Then the left byte position that is before the threshold must be a variable size window. The interval between the consecutive bytes from A_1 to X_1 is mentioned as M_1. The byte position again moves forward from Y_1 to B_1 as the right move. DPC is also a dynamic variable size window, as specified in Chunk 1. The exact process is repeated from chunk 1 to chunk N. The reason behind the dynamic window is always the break chunk point is variable size. Whereas in AE, the left alone is the variable size, and the right part is fixed. Hence, the consequences are more in AE. Our proposed algorithm DPC utilizes a dynamic window size to overcome this issue, which avoids the long chunk sequence and improves the deduplication throughput.

Table 2. Comparisons of computational overhead difference for CDC

Algorithm	Computational overhead (per byte scanned)
Rabin	2 XORs, 1 OR, 2 array lookups, 2 shifts, 1 conditional branch
TTTD	2 mod,1 comparison, 5 conditional branches
MAXP	2 mod, 2 – 1/p Comparisons, 5 + 1/p conditional branches
AE	1 comparison, 2 conditional branches
DPC	1 comparison, 2 conditional branches

* p - The expected chunk size

3.3 Properties of the DPC algorithm

Now, we are analyzing the behavior of the DPC for the key properties of the CDC.

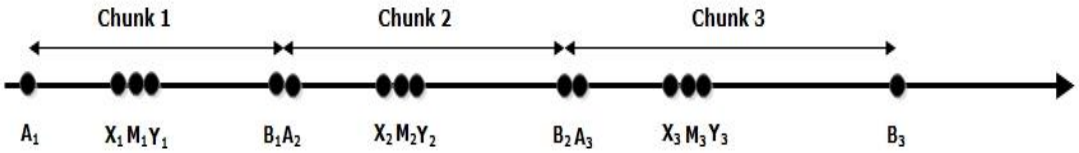

Fig. 3. Example of DPC

Content Defined: The previous algorithm AE delivers the asymmetric window position behind the extreme value and breaks the chunk boundaries. It produces the highest value inside each chunk, preferably by holding them in the chunk boundary. The method can slightly reduce the deduplication performance, but the algorithm is still CDC as the maximum value inside the chunks can additionally reset the chunk boundaries. Unlike AE, the DPC algorithm divides the window position dynamically changes based on prime numbers. The consequent chunk breakpoint window dynamically varies.

Consider **Fig. 3**, for example, M_1, M_2, and M_3 are the three maximum values, Y_1, Y_2, and Y_3, denoting the breaking points from the three identical blocks or chunks. Consider the maximum local value that will not replace all alterations in the example. When there is a byte insert or delete at variable size window the interval [A_1, M_1] in Chunk 1, the second chunk is to be realigned at the maximum point M_1 not changed by the chunk boundary. If the byte insertion is within the interval of the dynamic variable size window (M_1, B_1), the beginning point of second Chunk 2 will be modified, and the maximum value M_2 must realign the boundary to avoid affecting the third chunk. When a string of consecutive chunks has changed, the last updated chunk's efficiency loss has been determined by the modifier's location. If the adjustment is before the extreme value, then there will be no loss of performance. Otherwise, only one duplicate chunk is changed immediately following that modified region will get affected. Our proposed algorithm improves deduplication performance based on chunk variance and bytes saved per second (BSPS).

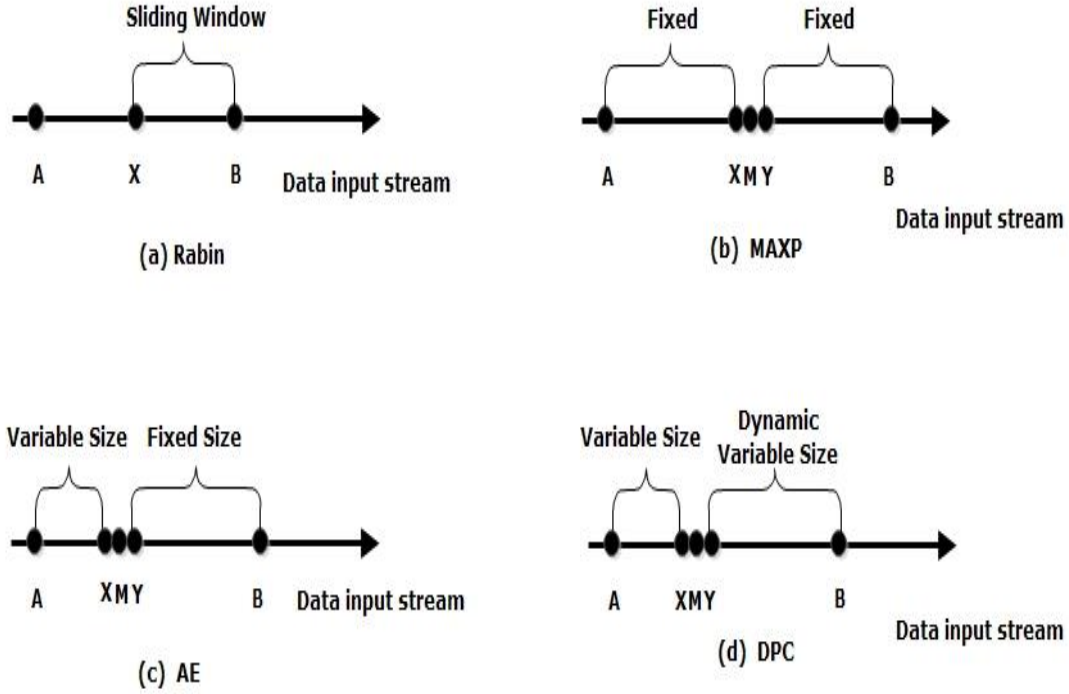

Fig. 4. Design of the strategic difference between CDC algorithm Rabin, MAXP, AE, and DPC

Low Computational overhead: **Table 2** shows the computational overhead and processing time variance of different multimedia data sets with five algorithms: Rabin, TTTD, MAXP, AE, and DPC. The Rabin algorithm requires one OR, two XORs, two shifts, and two more ARRAY lookup to determine the fingerprints and identify the chunk breakpoints, and it needs one conditional branch, as shown in **Table 2**. The most common CDC method Rabin is an effective duplicate detection technique. Still, it takes more time to consume a while because it evaluates the chunk point to the satisfaction of all the operations set out in **Table 2**. TTTD requires two mod operations, one comparison, and five conditional branches. **Fig. 4** shows the comparative analysis of the three existing algorithms such as Rabin, MAXP and AE. TTTD will not be discussed in **Fig. 4**, because the diagrammatic representation of TTTD is a little bit difficult. Hence, we used only three existing algorithms. But, the remaining sections of the work is compared with four existing CDC algorithms.

In **Fig. 4**, the existing algorithms MAXP and AE to find the local extreme value using comparison operations and both approaches are different. However, AE is fast chunking algorithm in comparison with MAXP. Based on the difference between both algorithms as presented in **Fig. 4**. MAXP determines the local maximum value in the fixed-size region [A, B]. When the byte value M is the center position between the two windows, it has the local maximum value of the window, while the M value must be higher than any byte in the fixed windows [A, X] and [Y, B]. If every byte in the region [A, B] has been processed and M has the local extreme value, then it is declared as a chunk boundary or cut-point. Another fixed window of bytes in range of [Y, B] also to be scanned repeat if the MAXP process of the byte N. That means MAXP requires the data content of the byte to be stored in the fixed window immediately just before the current byte position. For this reason, MAXP needs two modular operations to change the array and two more operations, a comparison of 2-1 / p and a conditional branch of 5 + 1 / p to determine the local extreme value.

By contrast, AE needs to find the local extreme point in the asymmetric window [A, B], which includes a fixed window [Y, B] and a region of variable size [A, X], the size of which is determined by the input data stream content. The result, AE requires storing only at a local extreme value and maximum value of the position. AE does not need backtracking and keeps the same maximum point. Hence, the method requires only one comparison and two conditional branches. **Fig. 4** shows the difference between AE and DPC algorithm. DPC uses two variable-size windows. DPC algorithm finds the maximum value M between the intervals of [A, B], which includes dynamic window region [Y, B] and variable size region [A, X], it is defined by the input of data stream content. The DPC algorithm needs one comparison and two conditional operations, as specified in an AE algorithm. However, the main difference between the two algorithms is that DPC uses the dynamic variable size window, conditional branch for time-consuming and table lookup operations, then the other three algorithms. The advantage of DPC reduces the long chunk variance based on maximum chunk size and improves the throughput of the deduplication system. The key idea of DPC is to reduce the processing time with minimum computational resource-intensive tasks. The DPC algorithm has framed two comparisons to achieve, as listed in **Table 2**.

Chunk Size Variance: Chunk size variance is defined as a chunk size difference between the long chunk and small chunk in the length of the file. In AE, the throughput efficiency is high because fixing the minimum chunk size is 2 KB and maximum is 16 KB. To improve the performance slightly higher, our DPC works with the principle of prime chunk to determine the dynamic window size. In the entire scenario, the chunk breakpoint lies between the prime of 2KB with 11KB as an average of 6KB. Hence, the long chunk sequence avoids DPC, and the total number of chunks is reduced to improve the throughput and processing time.

4. Experimental Setup and Datasets

Our proposed work DPC has been implemented in the prototype of an open-source deduplication system named Destor [11, 38] on two machines, namely client and backup server. In the specification, the configuration of the two machines follows. The client machine runs on a virtual machine running the Ubuntu 12.04 operating system and has an 8-core Intel ® core TM i7-770CPU@3.60 GHz with 16.0 GB of RAM. Another and powerful backup server has an 8-core Intel@ xeon®E5-2640V2/2GHz with 64 GB RAM to run the Ubuntu 14.04 operating system. To identify and remove the redundant chunk, Destor applies the SHA-1 cryptographic hash function. The system is experimentally evaluated to analyze the impact of the distribution of chunks and throughput. The performance of processing time and chunk time executes in the system.

Multimedia and OS Datasets: Our experiments are performed on four CDC algorithms with different real-world multimedia and operating system.

Multimedia: The data sets of our experiments include personal files for the home folder and media files. The files have been extracted from the resource of our Department Lab, Research Workstation, and Personnel PC. The media datasets are mainly movies, sports videos, online course materials videos, and video surveillance, which has been maintained by the Department server. Video file formats are MP4, avi, mkv, etc.

Operating system: The datasets include the various version of Linux operating system images and many software-related compressed files. The datasets are more similar content between the data. The details of the dataset is shown in **Table 3**.

Table 3. Multimedia and OS Datasets

Type	Size (GB)	No. of Files
Home	46.3	9010
Media files	29.3	259
Linux Files	35.2	16
Compressed	32.3	44

5. Results and Discussion

This section provides the comparative result analysis of CDC algorithms along with DPC. The performance metrics total chunk count, chunking and processing time, throughput, BSPS, and DER are tabulated in **Table 4**.

Table 4. Comparative analysis of Rabin, TTTD, MAXP, AE and DPC

Datasets	Algorithm	Total Chunk Count	Chunk Time (s)	Processing Time (s)	Throughput (MB/s)	BSPS (MB/s)	DER
Home	Rabin	10009200	1855.29	4157.36	15.03	288.44	1.0321
	TTTD	8999200	2367.72	3845.47	16.54	299.75	1.0425
	MAXP	7976320	2880.16	3584.34	15.56	189.14	1.0589
	AE	7776320	2298.93	3284.34	14.45	267.34	1.0569
	DPC	6765980	1796.09	3015.70	17.74	328.15	1.0661
Media	Rabin	7241570	1332.95	3832.39	33.68	745.78	1.3139
	TTTD	7031570	2585.29	3602.32	30	524.55	1.3345
	MAXP	7652990	3240.10	3360.55	10.76	480.83	1.3414
	AE	7452990	2805.81	3160.55	9.72	688.90	1.342
	DPC	6994540	2386.62	2952.65	35.27	755.14	1.3534
Linux	Rabin	9240750	1744.09	2767.59	15.37	249.91	1.0495
	TTTD	9430750	2475	2626.27	14.57	380.45	1.0495
	MAXP	9083490	3750	3062.07	15.92	343.34	1.0894
	AE	8834880	3346.63	2862.07	9.35	417.55	1.1806
	DPC	8537010	1280.19	1906.59	18.93	900.11	1.1869
Compressed	Rabin	7549770	1416.33	4001.55	15.48	39.65	1.0000
	TTTD	7439770	2744.25	3500.41	20.54	40.15	1.0068
	MAXP	8322600	3390	3750.97	16.55	20.21	1.0091
	AE	8122600	3067.76	3450.97	9.6	25.66	1.0054
	DPC	5630230	2435.49	2890.69	30.59	59.07	1.0094

5.1 Total Chunk Count

Fig. 5 compares the Rabin, TTTD, MAXP, AE, and DPC algorithms executed in all four datasets with the chunk count metrics. **Fig. 5** represents the following observations. Firstly, to analyze total chunk counts, DPC outperforms Rabin, TTTD, MAXP, and AE. DPC reduces the average AE chunk count by 22 % and 25 %, respectively. Second, the average DPC reduces the AE chunk count by 3%. We tested for four datasets, DPC reduces the chunk count since the window size dynamically fits based on min and max chunk size.

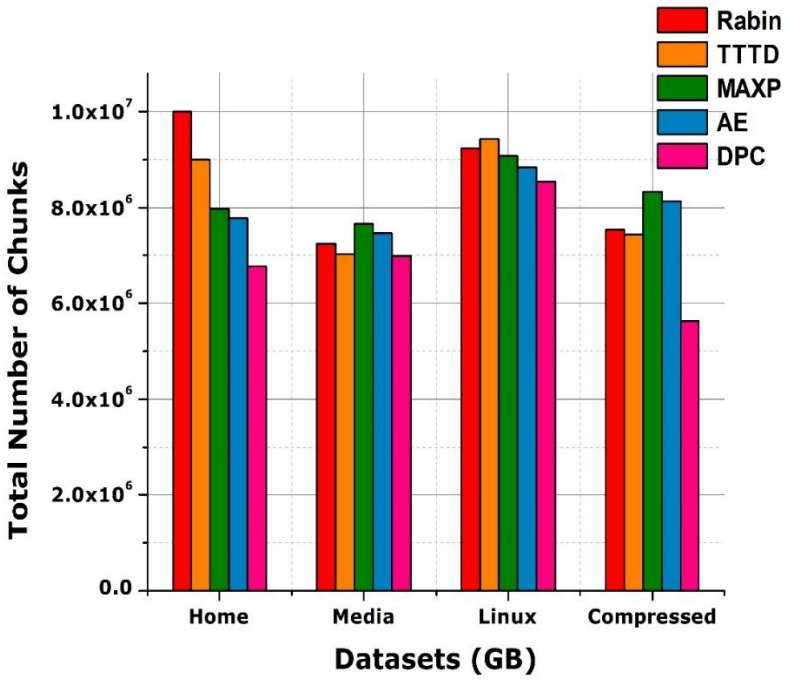

Fig. 5. Total Number of Chunks for different datasets

5.2 Chunking Time

Fig. 6 briefly discusses the chunking time on the four different datasets such as home folder, media, Linux, and compressed files. DPC achieves lower chunking time than AE among the four algorithms. The reason for DPC inferior chunk time to AE is due to a reduction in chunk count, as shown in **Fig. 5**.

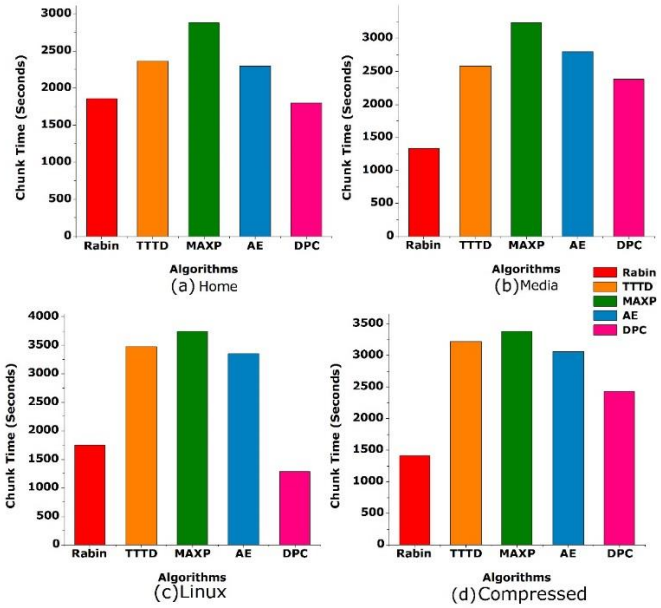

Fig. 6. Chunking Time Variance with datasets of CDC Algorithm

5.3 Chunking Throughput

The chunking throughput has been determined by splitting the amount of data processed and the amount of total time taken to chunk the files. **Fig. 7** shows the impact of different chunking algorithms such as Rabin, TTTD, MAXP, AE, and DPC on the entire throughput of the system. For the easy way of description, we analyze the individual datasets along with the chunking algorithm. We observed that, DPC provides higher throughput as 18, 35, 19, 31 Mbps for the home, media, Linux, and compressed files in comparison with other CDC algorithms.

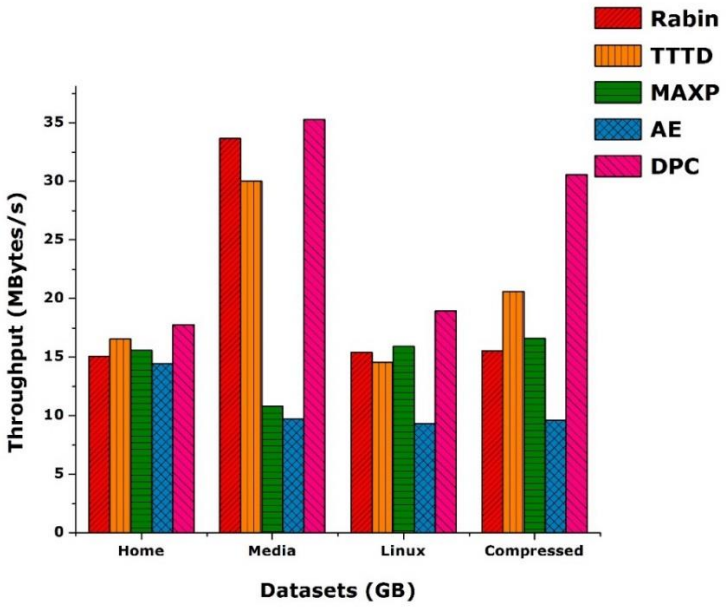

Fig. 7. Chunking Throughput with different CDC algorithms

5.4 Processing Time

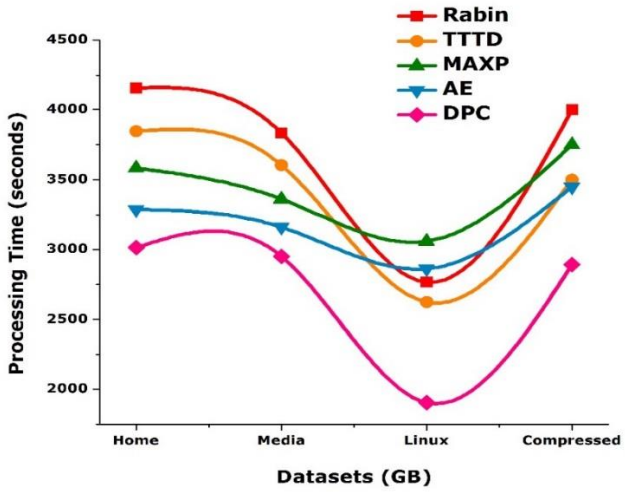

Fig. 8. Processing time for CDC algorithms

Fig. 8 evaluates the processing time for all CDC algorithms with various datasets. In **Fig. 8**, we depict the home folder, and the media file reveals a slight reduction in processing time compared with the existing AE algorithm. Similarly, for Linux and compressed files, our DPC provides maximally reduced processing time as compared to other CDC algorithms.

5.5 Bytes Saved per Second (BSPS)

The BSPS definition has been measured by dividing the number of duplicates found by the number of files processed and multiplying it by the throughput. The result for different datasets for the four chunking algorithms is considered in the proposed works illustrated in **Fig. 9**. Our results indicate that DPC has a higher throughput than other CDC algorithms, which helps to achieve higher bytes saved per second.

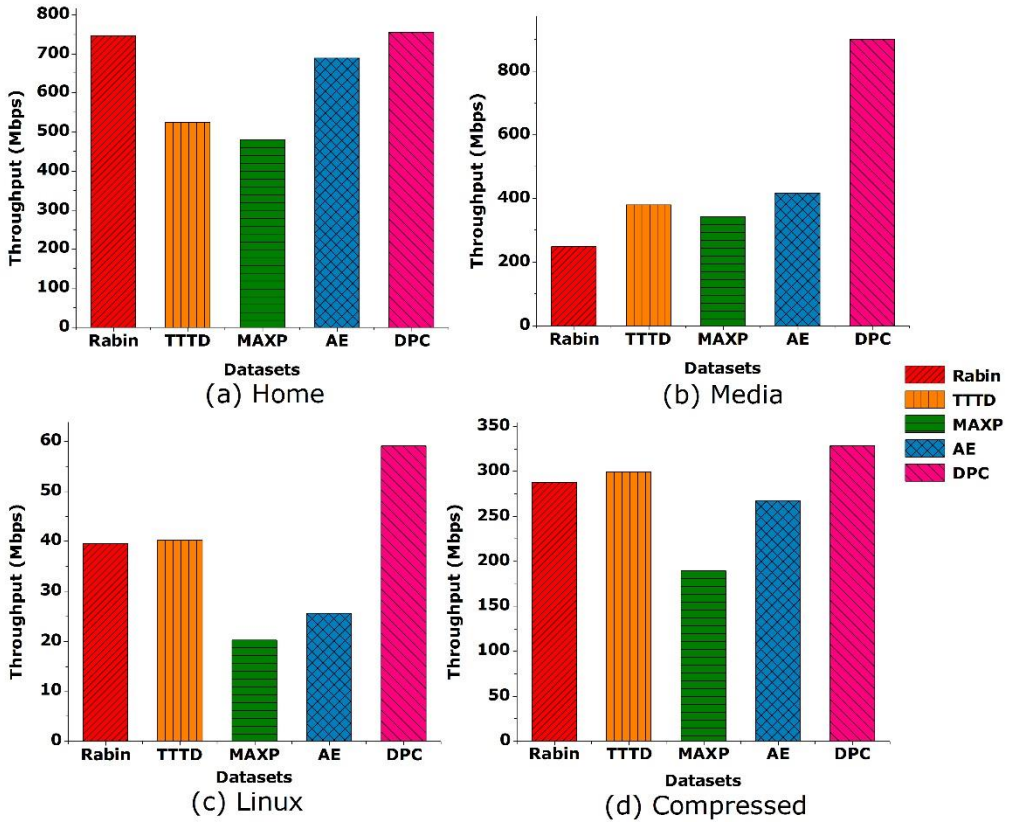

Fig. 9. BSPS for the four datasets

Deduplication Elimination Ratio (DER) is defined as the ratio between the input data to the stored data. The greater the value of DER infers high deduplication efficiency. The values are experimentally evaluated and are plotted in **Table 4**. Section 5 and **Table 4** provides the evaluation of performance metrics such as chunk count, chunk time, throughput, processing time, BSPS, and DER. The metrics analyzed from different datasets of the existing four CDC algorithms are Rabin, TTTD, MAXP, and AE. While discussing the results, our proposed work DPC is compared only with AE. The main reason is, AE achieves higher throughput and it is a fast chunking method. Because it finds the maximum value and chunks boundary without backtracking. Similarly, DPC also has higher throughput and minimizes the chunk variance.

Hence, DPC uses a dynamic window, min and max chunk size, maximum threshold, and finds the chunk breakpoint without backtracking. Rabin, TTTD, and MAXP are existing algorithms compared with DPC. These algorithms face the challenges of high computational overhead and low chunking throughput. Hence, DPC reduces the computational overhead and minimizes the processing time. DPC algorithm does not find more duplicate chunks in cloud storage because it is a hash-less chunking algorithm. Hence, in the next phase, we will improve the efficiency of the algorithm.

6. Conclusion

We presented DPC, a novel CDC algorithm that efficiently sets the dynamic variable size window to avoid the long chunk variance and quick process of the chunk breakpoint. As a result, DPC achieves low computational overhead, high throughput and reduces the processing time, chunk size, and existing CDC algorithms. The overall experimental results based on multimedia and operating system datasets demonstrate DPC's robust performance over the other existing algorithms in terms of BSPS and the efficiency of the backup storage system. In the future, we will focus on the next stage of indexing in deduplication to reduce the time-consuming process of cloud storage.

References

[1] D. Reinsel, J. Gantz, and J. Rydning, "Data age 2025: The evolution of data to life-critical don't focus on big data," *Framingham: IDC Analyze the Future*, 2017. Article (CrossRef Link)

[2] S. Quinlan and S. Dorward, "Venti: A New Approach to Archival Storage," *FAST 2002 Paper*, vol. 2, pp. 89-101, 2002. Article (CrossRef Link)

[3] A. Muthitacharoen, B. Chen, and D. Mazieres, "A low-bandwidth network file system," in *Proc. of the 18th ACM Symposium on Operating Systems Principles*, pp. 174-187, 2001. Article (CrossRef Link)

[4] D. T. Meyer and W. J. Bolosky, "A study of practical deduplication," *ACM Transactions on Storage(ToS)*, vol. 7, no. 4, pp. 1-20, 2012. Article (CrossRef Link)

[5] A. El-Shimi, R. Kalach, A. Kumar, A. Ottean, J. Li, and S. Sengupta, "Primary data deduplication-large scale study and system design," in *Proc. of USENIX Annual Technical Conference*, pp. 285-296, 2012. Article (CrossRef Link)

[6] B. Zhu, K. Li, and H. Patterson, "Avoiding the Disk Bottleneck in the Data Domain Deduplication File System," in *Proc. of the 6th USENIX Conference on File and Storage Technologies (FAST'08)*, vol. 8, pp. 1-14, 2008. Article (CrossRef Link)

[7] R. Vestergaard, Q. Zhang, and D. E. Lucani, "Lossless Compression of Time Series Data with Generalized Deduplication," in *Proc. of IEEE Global Communications Conference (GLOBECOM)*, pp. 1-6, 2019. Article (CrossRef Link)

[8] Y. Zhang, W. Xia, D. Feng, H. Jiang, Y. Hua, and Q. Wang, "Finesse: fine-grained feature locality based fast resemblance detection for post-deduplication delta compression," in *Proc. of the 17th {USENIX} Conference on File and Storage Technologies*, pp. 121-128, 2019. Article (CrossRef Link)

[9] W. Xia, H. Jiang, D. Feng, L. Tian, M. Fu, and Y. Zhou, "Ddelta: A deduplication-inspired fast delta compression approach," *Performance Evaluation*, vol. 79, pp. 258-272, 2014. Article (CrossRef Link)

[10] W. Xia, H. Jiang, D. Feng, and L. Tian, "Combining deduplication and delta compression to achieve low-overhead data reduction on backup datasets," in *Proc. of IEEE Data Compression Conference*, pp. 203-212, 2014. Article (CrossRef Link)

[11] M. Fu, D. Feng, Y. Hua, X. He, Z. Chen, W. Xia, Y. Zhang, and Y. Tan, "Design tradeoffs for data deduplication performance in backup workloads," in *Proc. of the 13th USENIX Conference on File and Storage Technologies*, pp. 331-344, 2015. Article (CrossRef Link)
[12] W. Xia, H. Jiang, D. Feng, F. Douglis, P. Shilane, Y. Hua, M. Fu, Y. Zhang, and Y. Zhou, "A comprehensive study of the past, present and future of data deduplication," in *Proc. of the IEEE*, vol. 104, no. 9, pp. 1681-1710, 2016. Article (CrossRef Link)
[13] X. Xiaolong and Q. Tu, "Data deduplication mechanism for cloud storage systems," in *Proc. of International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp. 286-294, 2015. Article (CrossRef Link)
[14] W. Xia, Y. Zhou, H. Jiang, D. Feng, Y. Hua, Y. Hu, Q. Liu, and Y. Zhang, "Fastcdc: a fast and efficient content-defined chunking approach for data deduplication," in *Proc. of USENIX Annual Technical Conference*, pp. 101-114, 2016. Article (CrossRef Link)
[15] H. Wu, C. Wang, K. Lu, Y. Fu, and L. Zhu, "One size does not fit all: The case for chunking configuration in backup deduplication," in *Proc. of the 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing(CCGRID)*, pp. 213-222, 2018. Article (CrossRef Link)
[16] E. Manogar and S. Abirami, "A study on data deduplication techniques for optimized storage," in *Proc. of the 6th International Conference on Advanced Computing(ICoAC)*, pp. 161-166, 2014. Article (CrossRef Link)
[17] G. Lu, Y. Jin, and D. Du, "Frequency based chunking for data de-duplication," in *Proc. of IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, vol. 1, pp. 287-296, 2010. Article (CrossRef Link)
[18] A. Venish and K. S. Sankar, "Study of chunking algorithm in data deduplication," in *Proc. of International Conference on Soft Computing Systems*, pp. 13-20, 2016. Article (CrossRef Link)
[19] I. Lkhagvasuren, J. M. So, J. G. Lee, C. Yoo, and Y. W. Ko, "Byte-index Chunking algorithm for data deduplication system," *International Journal of Security and its Applications*, vol. 7, no. 5, pp. 415-424, 2013. Article (CrossRef Link)
[20] D. Bhagwat, K. Eshghi, D. Long, and M. Lillibridge, "Extreme binning: Scalable, parallel deduplication for chunk-based file backup," in *Proc. of IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems*, pp. 1-9, 2009. Article (CrossRef Link)
[21] P. Kulkarni, F. Douglis, J. LaVoie, and J. M. Tracey, "Redundancy elimination within large collections of files." in *Proc. of USENIX ATC*, 2004. Article (CrossRef Link)
[22] A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee, "Redundancy in network traffic: findings and implications," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1, 2009. Article (CrossRef Link)
[23] K. Eshghi and H. K. Tang, "A framework for analyzing and improving content-based chunking algorithms," *Hewlett-Packard Labs Technical Report TR*, vol. 30, pp. 1-10, 2005. Aritcle (CrossRef Link)
[24] M. O. Rabin, "Fingerprinting by random polynomials," Technical report, 1981. Article (CrossRef Link)
[25] N. Bjørner, A. Blass, and Y. Gurevich, "Content-dependent chunking for differential compression, the local maximum approach," *Journal of Computer and System Sciences*, vol. 76, no. 3-4, pp. 154-203, 2010. Article (CrossRef Link)
[26] Y. Zhang, H. Jiang, D. Feng, W. Xia, M. Fu, F. Huang, and Y. Zhou, "AE : An asymmetric extremum content defined chunking algorithm for fast and bandwidth-efficient data deduplication," in *Proc. of IEEE Conference on Computer Communications*, pp. 1337-1345, 2015. Article (CrossRef Link)
[27] Z. Pooranian, K. C. Chen, C. M. Yu, and M. Conti, "RARE: Defeating side channels based on data-deduplication in cloud storage," in *Proc. of IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops*, pp. 444-449, 2018. Article (CrossRef Link)
[28] Z. Pooranian, M. Shojafar, S. Garg, R. Taheri, and Rahim Tafazolli, "LEVER: Secure Deduplicated Cloud Storage with Encrypted Two-Party Interactions in Cyber-Physical Systems," *IEEE Transactions on Industrial Informatics*, 2020. Article (CrossRef Link)

[29] Y. Zhang, D. Feng, H. Jiang, W. Xia, M. Fu, F. Huang, and Y. Zhou, "A fast asymmetric extremum content defined chunking algorithm for data deduplication in backup storage systems," *IEEE Transactions on Computers*, vol. 66, no. 2, pp. 199-211, 2016. Article (CrossRef Link)
[30] R. Widodo, H. Lim, and M. Atiquzzaman, "A new content-defined chunking algorithm for data deduplication in cloud storage," *Future Generation Computer Systems*, vol. 71, pp. 145-156, 2017. Article (CrossRef Link)
[31] C. Yu, C. Zhang, Y. Mao, and F. Li, "Leap-based content defined chunking—theory and implementation," in *Proc. of the 31st Symposium on Mass Storage Systems and Technologies(MSST)*, pp. 1-12, 2015. Article (CrossRef Link)
[32] W. Zhanjie and S. Lang, "Research on Distributional Stability of Chunk Sizes in Data Chunking," *International Journal of Digital Content Technology and its Applications*, vol. 7, no. 5, pp. 443-450, 2013. Article (CrossRef Link)
[33] T. S. Moh and B. C. Chang, "A running time improvement for the two thresholds two divisors algorithm," in *Proc. of the 48th Annual Southeast Regional Conference*, pp. 1-6, 2010. Article (CrossRef Link)
[34] C. Zhang, D. Qi, Z. Cai, W. Huang, X. Wang, W. Li, and J. Guo, "MII: A novel content defined chunking algorithm for finding incremental data in data synchronization," *IEEE Access*, vol. 7, pp. 86932-86945, 2019. Article (CrossRef Link)
[35] N. Kumar and S. C. Jain, "Efficient data deduplication for big data storage systems," *Progress in Advanced Computing and Intelligent Engineering*, pp. 351-371, 2019. Article (CrossRef Link)
[36] R. Vinoth and L. J. Deborah, "A Survey on Efficient Storage and Retrieval System for the Implementation of Data Deduplication in Cloud," in *Proc. of International Conference on Computer Networks, Big data and IoT*, pp. 876-884, 2019. Article (CrossRef Link)
[37] S. Saharan, G. Somani, G. Gupta, R. Verma, M. S. Gaur, and R. Buyya, "QuickDedup: Efficient VM deduplication in cloud computing environments," *Journal of Parallel and Distributed Computing*, vol. 139, pp. 18-31, 2020. Article (CrossRef Link)
[38] W. Xia, H. Jiang, D. Feng, L. Tian, M. Fu, and Z. Wang, "P-dedupe: Exploiting parallelism in data deduplication system," in *Proc. of the 7th International Conference on Networking, Architecture, and Storage*, pp. 338-347, 2012. Article (CrossRef Link)
[39] S. Al-Kiswany, A. Gharaibeh, E. Santos-Neto, G. Yuan, and M. Ripeanu, "Storegpu: exploiting graphics processing units to accelerate distributed storage systems," in *Proc. of the 17th International Symposium on High Performance Distributed Computing*, pp. 165-174, 2008. Article (CrossRef Link)

Manogar Ellappan received M.Sc (Computer Science) degree from the Department of Mathematics, College of Engineering, Anna University in 2004. Currently, pursuing research as a Full time in the Department of Information Science and Technology, College of Engineering, Anna University, Chennai. His research interests include Data Deduplication, Cloud Storage, and Big Data.

Abirami S received her Ph.D. in the area of Document Imaging from Anna University, Chennai. Currently, she is working as an Assistant Professor (Sr. Grade) in the Department of Information Science and Technology, Anna University, Chennai. She is having more than 16 years of research experience. She has authored more than 120 research publications in the International Journals and Conferences and 5 book chapters. She has mentored many sponsored research projects from Amazon, Tamil Virtual University, and Centre for Technology and Development. Her current research area focuses on the use of machine learning, data mining, computational intelligence, cognitive computing, and video analytics.