IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 22 April 2023, accepted 11 May 2023, date of publication 16 May 2023, date of current version 17 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3276785

RESEARCH ARTICLE

Double Sliding Window Chunking Algorithm for Data Deduplication in Ocean Observation

SHUAI GUO[1], **XIAODONG MAO**[2], **MENG SUN**[2], **AND SHUANG WANG**[2]
[1]College of Science, Qingdao University of Technology, Qingdao 266000, China
[2]College of Information and Control Engineering, Qingdao University of Technology, Qingdao 266000, China

Corresponding author: Shuai Guo (guoshuai@qut.edu.cn)

This work was supported by the Shandong Province Natural Science Foundation under Grant ZR2020QF028.

ABSTRACT As an essential means to eliminate redundant data, data deduplication technology significantly affects today's era of massive data growth. In recent years, due to the rapid development of a series of related industries, such as marine monitoring, the marine monitoring data has exploded, leading to higher storage costs for marine observation stations. In the face of the surge in data size, we first think of using data deduplication technology to reduce the stored data to save storage costs. However, we have many choices for data deduplication technology. Because-block level data deduplication technology can better complete the task, and the core technology of block-level data deduplication technology is how to cut data blocks, this paper proposes a dual sliding window-based segmentation technology. The structure of double sliding windows makes the divided data block size more average to reduce the consumption of the fingerprint table in memory. At the same time, we add a prediction algorithm to the data deduplication system to predict the cutting point of the data block to improve the cutting efficiency. In addition, we propose a more accurate calculation method of the deduplication ratio, which can more accurately compare the algorithm's performance and obtain the final experimental results of this paper by using this calculation method. Moreover, we propose a model based on Markov prediction to store massive ocean data, which can save more resources. At the end of the article, we compared the commonly used segmentation algorithms through careful experiments. Finally, we obtained and will use the public dataset experiment to compare the same checking rate at the end of this article.

INDEX TERMS Double sliding window, ocean observation, data deduplication, content defined chunking, markov chain.

I. INTRODUCTION

It is well known that today is society has come to the age of big data, and all kinds of data are surging. According to the research report, from 2018 to 2025, IDC estimates that the storage capacity of the global market will grow exponentially from 33 ZB to 173 ZB [28]. The ocean, which accounts for about 70% of the total surface area, has entered the era of big data. At present, it has a variety of marine observations. Investigation means, including offshore mapping, island monitoring, underwater exploration, marine fishery operations, marine buoy monitoring, marine scientific research, oil and gas platform environmental monitoring, satellite remote sensing monitoring, etc., forming an extensive marine observation and monitoring system, and accumulating massive marine natural science data, including on-site observation and monitoring data, marine remote sensing data, numerical model data, etc. In recent years, marine observation equipment has been undergoing revolutionary changes. The scale of marine data represented by satellite remote sensing data is growing explosively, and the growth rate of marine data volume is faster than that of other industries. When faced with data deletion, people usually think of compression technology first. However, compression technology retrieves the same data block through string matching, mainly using a string matching algorithm and its variants, which is accurate matching.

The associate editor coordinating the review of this manuscript and approving it for publication was Byung-Seo Kim.

Implementing precise matching is more complex but highly precise and more effective for fine-grained redundancy elimination.

Data deduplication [23] technology finds the same data block through the data fingerprint of the data block. The data fingerprint is calculated by the hash function, which is fuzzy matching. Fuzzy matching is relatively simple, more suitable for large granularity data blocks, and less accurate. Then compression is not suitable for ocean storage data. As a famous and popular storage technology, data duplication can effectively optimize storage capacity in such an era. It eliminates redundant data by deleting duplicate data in the dataset and retaining only one copy. Therefore, data duplication can bring many practical benefits, such as effectively controlling data with a fast growth rate, increasing adequate storage space and improving storage efficiency, saving total storage cost and management cost, and Meet ROI, TCO, etc [23].

The process of data deduplication is that input files are divided into small pieces, and duplicate files are identified and processed into five stages. These stages are blocking, hash value fingerprint, hash index, compression technology, and managing data in various storage systems. The compression phase is optional in these phases because it is only applicable to traditional compression methods, such as LZ compression and delta compression. Data deduplication plays a critical role in the final stages of storage management. In the existing methods, byte-level sliding windows are used to perform block-level deduplication detection. This window is small in size and helps match strings by comparing a byte in the storage system. Therefore, block-level deduplication has more advantages than file-level deduplication. Blocking technology can divide the data stream into data blocks, which is the first step of deduplication.

Currently, the existing blocking technology is divided into two types fixed length blocking and variable-length blocks. Fixed length blocking is to divide the data stream into equal-length blocks according to the set size, such as 4kb or 8kb. FSC is the fastest blocking algorithm at present. The variable length block determines the boundary transfer problem by declaring the block boundary according to the local content of the data input stream. Based on the internal data content of the CDC [11], this block will not change its boundary and data content. Therefore, we also pay more attention to CDC segmentation and intend to improve based on CDC segmentation. As the first and most crucial step in data deduplication technology, segmentation technology will directly lead to high or low efficiency. The subsequent work can be smoother by dividing the data flow into appropriate data blocks.

The existing churning technologies include Fixed-size chunking(FSC), Basic sliding window(BSW) [20], and Two thresholds and two divisor chunking(TTTD) [9]. Fixed-size chunking is the earliest chunking. Because of its fixed-length chunking, it is swift. However, the re-deletion rate is low because of the boundary migration problem. A basic sliding window solves the problem of position offset in Fixed-size chunking. It uses the combination of sliding window and hash function to segment and solves the problem of position offset by judging the characteristics of the hash value. However, due to its characteristics, it cannot control the size of the segmented data block, which also leads to varying data block sizes affecting the re-deletion rate. Two thresholds, two divisors chunking by adding a divisor [9], and two thresholds, the size of the segmented data block in the basic sliding window is limited to conform to the two thresholds strictly. However, adding a divisor and two threshold values for two thresholds and two divisor chunking will inevitably lead to algorithm procrastination. While this paper is committed to solving the shortcomings of a basic sliding window, it will also be more convenient than two thresholds and two divisor chunking algorithms. At the same time, the segmented data blocks strictly conform to a certain range.

In this work, our goal is to make the segmented data blocks strictly conform to a certain range. Therefore, to solve the existing challenges, we propose a segmentation algorithm based on double sliding windows (DSW), which avoids the problem of position offset and limits the size of blocks. DSW is an improved algorithm based on BSW. DSW uses two fixed-length windows to determine whether there is a breakpoint through two conditions. DSW determines the block size limit and is more convenient than TTTD. The contributions of this paper are as follows:

- We propose a segmentation algorithm based on double sliding windows, which limits the minimum and maximum thresholds of data blocks, and reduces the block duplication in storage to reduce the size of the fingerprint table and ultimately reduce the burden of memory.
- We propose a more comprehensive method to calculate the deduplication ratio, which reduces the consumption of fingerprint table and hash chain, making the calculated deduplication ratio more comprehensive.
- Our team is committed to simplifying the data deduplication process, so we propose a prediction algorithm to predict the breakpoints of data blocks so that the overall algorithm can cut data blocks more quickly.

The content of the remaining chapters of this paper is rough as follows: In the second section, we introduce and analyze the advantages and disadvantages of existing algorithms and explain our motivation. The third section will introduce how to carry out relevant work in detail. The fourth section will detail the algorithm architecture and algorithm concept of DSW. The fifth section will detail experiments through different indicators and analyze the advantages and disadvantages of different algorithms. The sixth section summarizes this article and introduces the future work objectives.

II. BACKGROUND AND MOTIVATION

In the second section, we will briefly introduce various segmentation algorithms, their backgrounds, and principles and introduce the BSW algorithm and its shortcomings. At the end of this section, we will introduce our motivation for this work.

A. CHUNKING TECHNOLOGY

Data deduplication can be divided into two types according to different granularity, specifically block and file level deduplication. The disadvantage of file-level data deduplication technology is pronounced. Because any file byte will produce a new copy of the file when it is changed internally, file-level data deduplication technology cannot save more space. That is why the current deduplication is blocked remember deduplication [29]. If the files transferred to the data deduplication system are transferred in a stream form and cut into pieces of data according to the algorithm, and the data is deleted according to these data blocks, the problem at the file level will be avoided. Moreover, the cutout data blocks can be deduplicated across file types, significantly improving storage efficiency. In addition, the steps of a complete data deduplication system include chunking, duplicate checking, deletion, and recovery [20]. Chunking is the bridgehead of the entire data deduplication system. How to cut it into pieces is also a top priority.

At present, chunking technology includes fixed-length chunking and content-based chunking. Among them, fixed-length segmentation is the earliest and simplest segmentation technology. Its principle is to perform average segmentation according to the fixed size set by the system and determine the data block size manually. This has obvious advantages. Namely, it is simple and fast and can obtain the desired data block size [30], but its disadvantage is more prominent that is, it cannot avoid boundary offset. If the data is modified, fixed-size chunking will have a severe boundary transfer problem. The boundary offset problem is that if a byte is inserted or deleted from the data, fixed-size chunking will generate different blocking results after the byte is inserted or deleted. The best result is that if the file tampers at the end, the data in front of the tampered location does not change, and the effect of data deduplication is the same. The worst result is that if a byte is inserted in the first place during secondary storage [9], the whole file will be stored as a new file if it cannot recognize a new data block after being divided into fixed-length blocks. As a result, the re-deletion rate is too low. This is the boundary offset problem. Due to the existence of such problems, this method is rarely used for segmentation.

B. BASIC SLIDING WINDOW

BSW is the most common chunking algorithm for deduplication, It's based on sliding Windows and hash function, The principle is: First set up a sliding window of fixed width W, a hash function H() to compute the hash value, integers D and r ($D > r$), The window slides forward on the data stream, and the hash function computes the hash value of the data in window W, using the formula H(W)mod(D) = r? To determine whether there is a breakpoint [20]. As shown in the figure below(Fig.1), the sliding window continuously slides on the data stream and calculates the hash value in the window. When the hash value in the window meets the formula, a breakpoint is generated, and the window continues to move backward and repeats the previous steps until the window reaches the end of the data stream.

The advantages and disadvantages of BSW are apparent. The advantage of BSW is that it splits the data according to the data content through the sliding window and the hash function in the calculation window. This can solve the problem of sliding offset in fixed-length segmentation [12]. Therefore, we are not afraid of the problem that duplicate data cannot be identified in a file due to a small quantity of change. Over time, the shortcomings in BSW have also been exposed; that is, the size of the data blocks that can be split and eaten cannot be controlled. Because BSW uses a hash function to identify data block breakpoints, if the data in the window always fails to meet the requirements of the hash function, the data blocks will be too large, which will result in the data blocks being separated by BSW cannot be averaged within an interval.

Therefore, smaller data blocks will occupy more fingerprint tables in the memory, which will cause the memory to bear a more significant burden. Excessive data blocks will affect the efficiency of data deduplication in backup storage. Because BSW cannot control the size of data blocks [14], which affects the efficiency of data deduplication, a double threshold allocation (TTTD) algorithm is introduced to beat Rabin's block size variance [9]. The basic sliding window algorithm (BSW) is the technique employed by the TTTD algorithm. This algorithm refers to BSW and limits the size of data blocks by introducing two thresholds and alternate divisors and using the maximum and minimum thresholds. Finding backup breakpoints involves using the second divisor, half of the primary divisor.

C. MOTIVATION

The commonly used data deduplication is based on the block level rather than the file level. The reason is that the block level can deduce data across file categories [7], and BSW is the best algorithm at the block level. Therefore, our team is committed to solving the shortcomings of BSW. We hope that the segmented data blocks can be averaged within an interval, reducing the burden of the fingerprint table, releasing memory indirectly, and avoiding the problem of boundary offset. We hope to solve the adverse effects of BSW while retaining its advantages. Compared with TTTD, we hope to have a shorter processing time and a better re-deletion rate. We propose that DSW solve the defects in the above algorithms. DSW makes the segmented data blocks more average and keeps them within a specific range through double sliding windows. At the same time, it will have a more convenient and shorter processing time than TTTD.

The concept of DSW is roughly the same as that of BSW, and the same method is adopted to avoid the problem of boundary offset. In addition, we consider a special prediction algorithm to eliminate the sliding process of windows. Imagine that in DSW, if you do not need to slide the window to judge breakpoints but directly select breakpoints, you will

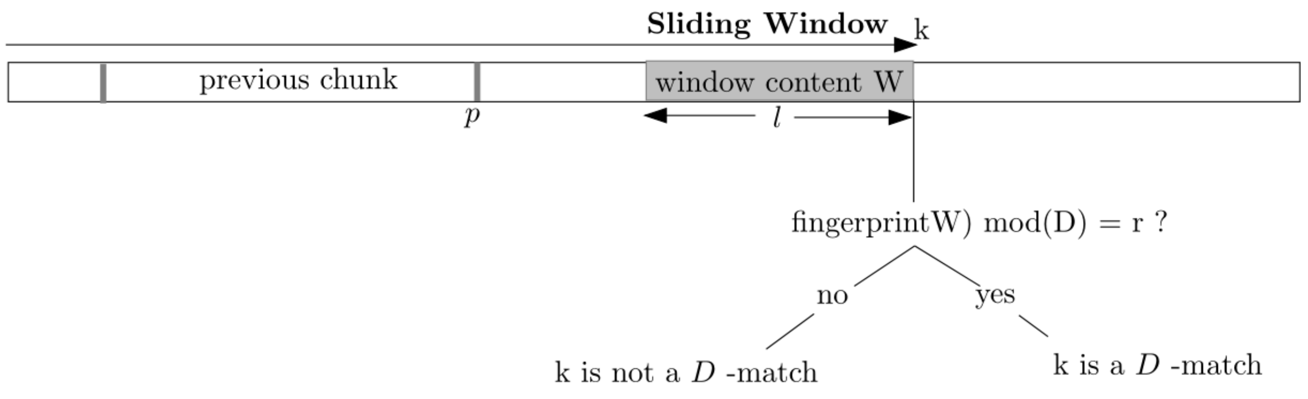

FIGURE 1. Prototype design based on chunking window slicing.

save a lot of running time. At the same time, it can save a lot of traffic consumption in the offline state, which is a considerable amount and worth a try. Currently, the algorithm for saving file size is too single, and different experimental results will be generated under different experimental environments. It is also good to directly calculate the deduplication ratio without various external factors. Therefore, our team wants to better reflect the advantages of each algorithm by discarding the different external experimental environments. Our team will complete the above ideas one by one.

III. RELATED WORK

Fixed size data blocks (FSC) and content definition blocks are two categories of blocking techniques used for data deduplication (CDC). FSC, which divides the input file into fixed size data blocks based on fixed length, is the simplest and fastest approach. FSC first appeared in rsync algorithm [30]. A crucial problem with FSC is that the position offset problem, such as insertion or deletion, even a byte in the file will move all block boundaries outside the edit point, and the new file will only recognize a few duplicate data blocks. The storage system Venti [27] also uses a simple and fast FSC blocking algorithm. CDC is a blocking algorithm based on content definition. It distinguishes breakpoints according to preset conditions rather than a fixed length, and judges whether there are duplicate data blocks by storing the fingerprint table in memory to ensure that only one copy is stored each time. Because CDC distinguishes breakpoints according to set conditions, there is no problem of location offset. However, both CDC and FSC delete duplicate data by identifying duplicate data blocks.

CDC was originally used to reduce network traffic when transferring files. Spring et al. [29] designed the first block algorithm using Border's method [6]. It is designed to identify redundant network traffic. Muthitacharoen et al [20]. proposed a file system based on CDC, called LBFS, which enriches the CDC blocking method to reduce and eliminate duplicate data in low bandwidth network file systems. You et al. [34] used the CDC algorithm to reduce data redundancy in the archive storage system. Because Rabin fingerprint in CDC algorithm consumes a lot of time in calculation, people propose many methods to replace Rabin to speed up CDC [1], [32], [37]. The encryption function (such as Rabin) required in the fingerprint identification process can speed up the fingerprint identification through the parallel strategy [31]. And the deformed version of AE [36].

In addition, RapidCDC [21] uses data locality to record the location of each data block, in this way to reduce the amount of calculation for the next duplicate data block. SSCDC [22] proposes a two-phase simultaneous blocking operation based on content definition, but its solution will not affect the de duplication rate. In TTTD algorithm [9], In addition, we have studied the problem of reducing the size difference of data blocks in the CDC algorithm and made a series of improvements. TAPER [12] and REBL [14] show us the effect of combining CDC with delta encoding [25], which can be used for directory synchronization. However, the protocols used in these two schemes are the same. They are both multi-layer protocols. CDC and delta coding are used for multi-level and multi-level redundant data detection. However, it has been proved that the combination of similarity detection [6], [17] and delta coding [8] can only exist as a more radical compression method. Moreover, it is very difficult to find files [4] with similar content or similar content. In [18] and [33], the results of comparison with delta encoding and blocking are provided. In another field, namely the field of Internet traffic analysis, people have discussed the concept of how to identify identity in the crowd at high frequency. Manku et al. [19] deduced a new algorithm to find frequent data items in computer data streams.

However, their work ignored an important point and did not consider that there would be a specific frequency distribution in the counting items. Someone put forward the idea of using parallel flow filter to identify high-base Internet hosts among existing hosts in [7]. Others have designed a special filter in [5] to identify data blocks that occur frequently and obtain estimates of the frequency of the data blocks, and he uses a special parallel bloom filter. It is well known that Bloom filters are used in [38] and [13] to store observed high frequency data blocks. And they only use one bloom filter, which is why they can only see if a duplicate occurs quickly [10], like block-level duplicate data deletion, consumes a lot of metadata overhead. Kruus et al. [13] proposed an idea that a two-stage blocking technique can be used to regroup large, transitional and non-repetitive CDC blocks into smaller ones in the presence of block knowledge. They can significantly reduce the number of blocks while maintaining the same repeat elimination rate as the standard CDC algorithm is their key contribution. Extreme binning [3] uses a representative hash value to identify file-level content similarity and demonstrates that it can perform nearly as well as the original CDC on data sets with no locality between subsequent files. A sparse indexing method is presented in another recent work [15] to find similar large parts inside a stream.

However, with the vigorous development of all walks of life in recent years, the application of data deduplication technology has become more extensive. One of the most striking is the data deduplication technology in cloud storage [16]. But there are also more security issues in the cloud, with PraJapanese et al. [26] surprising statements about security issues in duplicate data deletion technology. Even Yuan et al. [35] proposed block chain-based duplicate data deletion technology in the popular domain block chains. As well as challenges posed by Azad et al. [2] about network edge problems, PG et al. [24] proposed solutions.

IV. DOUBLE SLIDING WINDOWS

The construction of section IV is as follows: In the sectionIV-A, We will introduce the core ideas and design concepts of DSW in detail. In the sectionIV-B, we will deduce the origin of our proposed deduplication ratio, and in the sectionIV-C, we will introduce how our prediction algorithm is implemented and completed.

A. DOUBLE SLIDING WINDOWS

In this section, we will introduce the idea of the DSW algorithm step by step and analyze the algorithm is performance.

1) THE CONVENIENCE OF DOUBLE SLIDING WINDOW

DSW is an improved algorithm based on CDC technology. It aims to improve based on CDC. Compared with CDC, DSW increases the controllable threshold. The performance of the CDC can be judged from three aspects: response time, number of blocks, and re-deletion efficiency. These three points are mutually exclusive; the three essential standards cannot be improved simultaneously. For example, the more blocks cut, the smaller the data blocks, and the higher the re-deletion rate. However, the corresponding processing time is also longer. Because of the addition of data blocks, the hash table in the memory will also store more data blocks to increase the burden on the memory. On the contrary, if the number of blocks is smaller, the larger the data blocks, the lower the re-deletion rate. However, a shorter response time will not bring more memory burden. Therefore, DSW chooses to control the number of blocks. Because the window size in DSW is a variable, it can bring different re-deletion efficiency according to different environments. The three criteria are weighed by limiting the window size in different environments to make it more reasonable.

2) SEGMENTATION TECHNOLOGY BASED ON DOUBLE SLIDING WINDOW

a: DESIGN OF DSW

The algorithm principle of DSW is based on double sliding windows. The algorithm steps are shown in Figure 2. Place two sliding windows (W_1, W_2) of the same size at the starting position of the data stream, and the interval length between W_1 and W_2 is L. While sliding forward continuously, calculate the hash values $H(w_1)$ and $H(w_2)$ in the two windows, and assign the value of $H(w_2)$ in the initial position to t. Because of the anti-collision function of the hash function, the same hash value can be output only when the same data is input. When the Rabin value meets the residue condition, it is determined as a breakpoint and the segmentation is completed. At the same time, because of the uniqueness of the hash value, when $H(w_1) = t$, that is, when the red covers the initial blue position, the condition is met and the chunk is generated. At this time, the data's greatest value block is limited so that the data block is not greater than $[0, 2W + L]$.

DSW can judge breakpoints in the following two ways:

I When the value of the data in window w_1 calculated by Rabin meets the conditions, the blocking conditions are met and the blocking is completed. So we can get the minimum value of the data block.

II When $H(w_1) = t$ is Yes, that is, when the red reaches the initial blue position, it is judged as the tangent point. At this time, we can get the maximum value of the data block. The original intention of DSW is to solve the maximum threshold problem. The data block it cuts can be kept in a stable interval, which can be adjusted. Since the data blocks cut by DSW are always between $[1, 3w + 2L]$, the size of interval L can be manually manipulated to indirectly adjust the maximum threshold. It can be seen from Figure 3 that there are two criteria in DSW. Condition 1 is used to satisfy content-based segmentation, and condition 2 is used to limit the maximum data block. In other cases, the data blocks will be limited to this range, and breakpoints will be determined within this range. The pseudo-code of the algorithm is shown in the following table.

After the breakpoint is determined, the starting point will be reinitialized, and the process will be repeated until the last breakpoint of the data flow is found. Next, we will discuss the implementation of the DWS algorithm based on double sliding windows. Description of DSW algorithm: The main idea of the DSW algorithm is to divide data blocks based on improving the sliding window. The input to the algorithm is the entire string. Three values, window size, block counter, and window interval, are defined in the initial phase of the algorithm. Step 1 manually outputs the window size and window interval to make the final data block size appropriate.

Algorithm 1 Double Sliding Window (DSW)

```
Input: input data string, Str;Window size, W;
Output: Chunk break point;
1: w=scanf("d"$a$);
2: int t;
3: int *k=0;
4: int *i=w;
5: int *j=2w;
6: int p=0;
7: for (,j.data!==0,i++ j++ k++) do
8:     t=hash(p+w,p+2w);
9:     Ifhash(k,i)<hash(i,j)
10:    p=j;
11:    print("k,j");
12:    continue;
13:    Ifhash(k,i)==t
14:    p=j;
15:    print("k,j");
16: end for
17: print("p,j")
```

Steps 2-6 create various pointer record strings and record the hash value of the initial position W2. Step 7 enters the loop to keep the window moving forward. Step 8 The hash value of the initial position should be recorded in advance to prevent data disorder caused by the window moving forward. Step 9-14 is the judgment condition 1 to limit the minimum data block. Steps 17-22 are judgment condition 2 to limit the maximum data block. The time complexity of DSW is O (n).

b: WORKFLOW

The starting points of the data stream shown in Figure 4 are A and B, respectively, and the window slides from A to B, where K-I and N-J are the exact sizes. Because the window keeps moving forward and judges the condition, a breakpoint will be generated if it meets the condition. If condition 1 is met, a breakpoint will be generated at J, and the P pointer will point to J. Then, continue to repeat the above steps. Only when the KI window slides to the initial position of the NJ window can condition 2 be triggered, and the cutting is completed; that is, a breakpoint is generated, and then it continues to repeat until it reaches B and stops and outputs the remaining uncut parts. DSW can limit the size of data blocks precisely because of Condition 2, which will generate more data blocks and increase the proportion of data deduplication. This also increases two of the three criteria: the number of cuts and the re-deletion ratio. This is precisely how DSW keeps the size threshold of data blocks within a range to fill in the deficiencies in BSW.

c: ALGORITHM ATTRIBUTE

Next, we will introduce the algorithm attributes of DSW. We will explain in three parts: content based, block size difference and dynamic data block size.

Content-based: Since DSW is an improved algorithm based on BSW, it is also a content-based chunking algorithm. The breakpoint is also determined by taking the remainder of the value calculated by Rabin. By using the content-based algorithm to determine the breakpoint, the content of the data block can have more of the same characteristics so that the proportion of data deduplication can be better increased. Thus better results can be achieved in actual use.

Block size difference: BSW judges breakpoints based on the remainder of the hash value in the window, so it cannot limit the size difference of data blocks. As a result, the data blocks need to be more extensive and occupy much memory. TTTD solves the problem of data block size well, but because TTTD introduces an alternate divisor, it increases the overhead. In order to solve this problem, our team used double sliding windows to limit the size of the data block to $[0, 2W+L]$ by using two equal windows. The size L between windows can be adjusted according to different requirements, and then the size range of data blocks can be adjusted.

Dynamic data block size: As previously described, DSW fixes the data block size between $[0, 2w + L]$. Because the size L between windows can be manually changed, DSW can dynamically select the size of the data block, and can select different L according to multiple experiments to obtain a better deduplication ratio. This is also a more advanced point compared with DSW and TTTD. It is well known that the smaller the average data block, the more significant the proportion of data deduplication will be and the more time consumed. On the contrary, the smaller the proportion of data deduplication, the less time will be consumed. Therefore, we can dynamically select the appropriate data block size according to different scenarios of the algorithm to achieve better results. For example, the online data deduplication system seeks to be more convenient and faster, so we can dynamically select a more significant interval L to obtain faster speed. On the contrary, there is more offline time for re-deletion in an offline data deduplication system, so we can choose a smaller interval L to achieve a higher data deduplication ratio.

B. CALCULATION OF DEDUPLICATION RATIO

In this section, we will derive the specific calculation method of the deduplication ratio and compression ratio we proposed.

First, let's clarify the meaning of several characters:

A is the actual size of the file.

$|A|_c$ is the sum of the sizes of all data blocks generated after the algorithm re deletion of file A.

$|A|_{ME}$ is metadata overhead. The calculation method of $|A|_{ME}$ is:

$$|A|_{ME} = |A|_{HT} + |A|_{HC} \tag{1}$$

where, $|A|_{HT}$ is the metadata cost of the hash table, and $|A|_{HC}$ is the metadata cost of the hash chain. The re deletion ratio

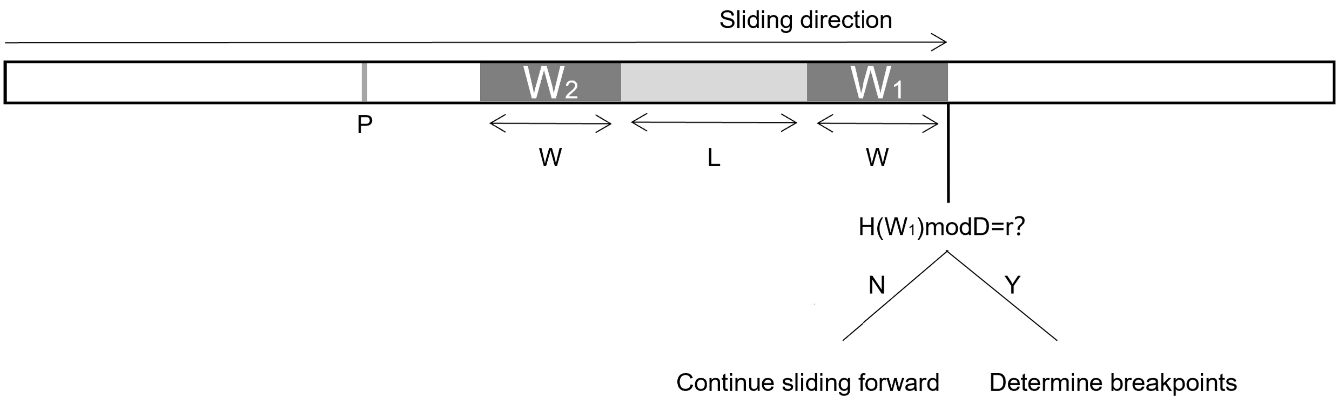

FIGURE 2. **Chunking technique based on double sliding window is a specific case when condition 1 is satisfied.**

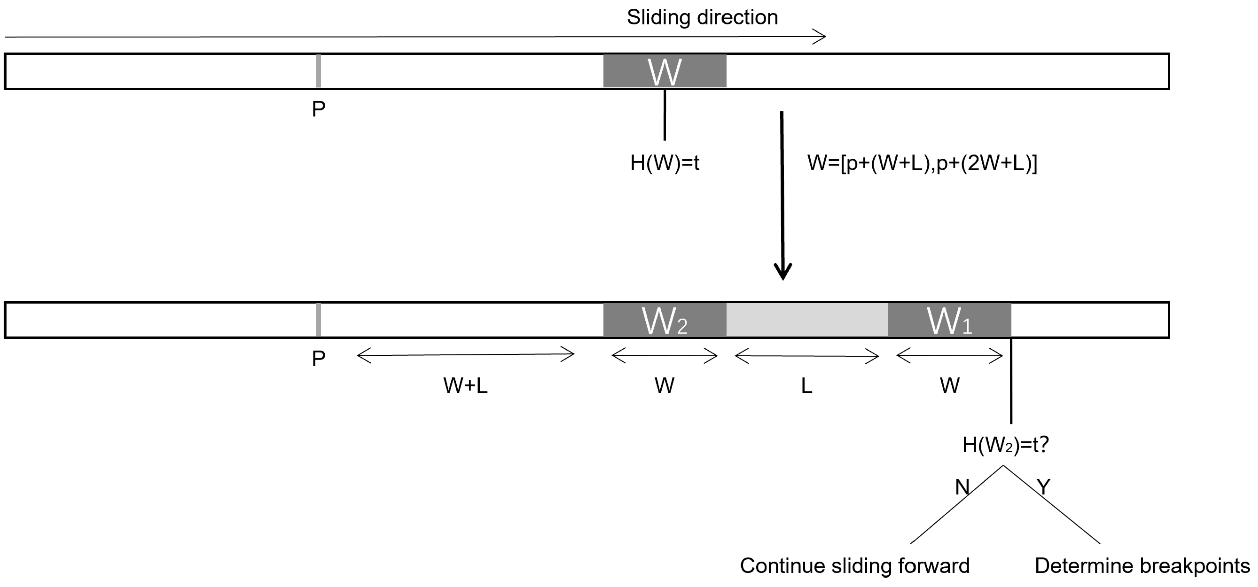

FIGURE 3. **Chunking technique based on double sliding window is a specific case when condition 2 is satisfied.**

R_D is calculated as follows:

$$R_D(A) = \frac{1 - (|A|_C + |A|_{ME})}{|A|} \tag{2}$$

The relationship between the re deletion ratio R_D and the compression ratio R_C is:

$$R_C = \frac{1}{1 - R_D} \tag{3}$$

Since $0 \leq R_D \leq 1$ will increase with the increase of R_D. It can be concluded that R_D and R_C are equivalent indicators.

Next, we will expand the formula in more detail. First, let's assume a situation: when file A has been added, deleted, modified and checked in the computer, there are many copies of file A, and when we continue to re delete A, we must consider more carefully.

Let $A = X_1$, $A' = X_K$, and A' is obtained after K times of addition, deletion, modification and query. At this time, we get the theoretical value of $|A'|_C$ as:

$$|A'|_c = \sum_{i=1}^{k-1} Neo(x_i, x_{i+1}) \tag{4}$$

where, X_{i+1} is arrived by X_i after I operations of addition, deletion, modification and query. In addition, $Neo(X_i, X_{i+1})$ represents a set that removes the prefix ($ListIsoP()$) and suffix ($ListIsoS()$) of the same string as X_{i+1}.$Neo(X_i, X_{i+1})$ is expressed as:

$$Neo(X_i, X_{i+1}) = |X_{i+1}| - (|ListIsoP(X_i, X_{i+1})| + |ListIsoS(X_i, X_{i+1})|) \tag{5}$$

Before that, we first define $Sur(X_I, p)$, which represents the remaining part of X_i after P is removed:

$$Sur(X_i, p) = X_i - P \tag{6}$$

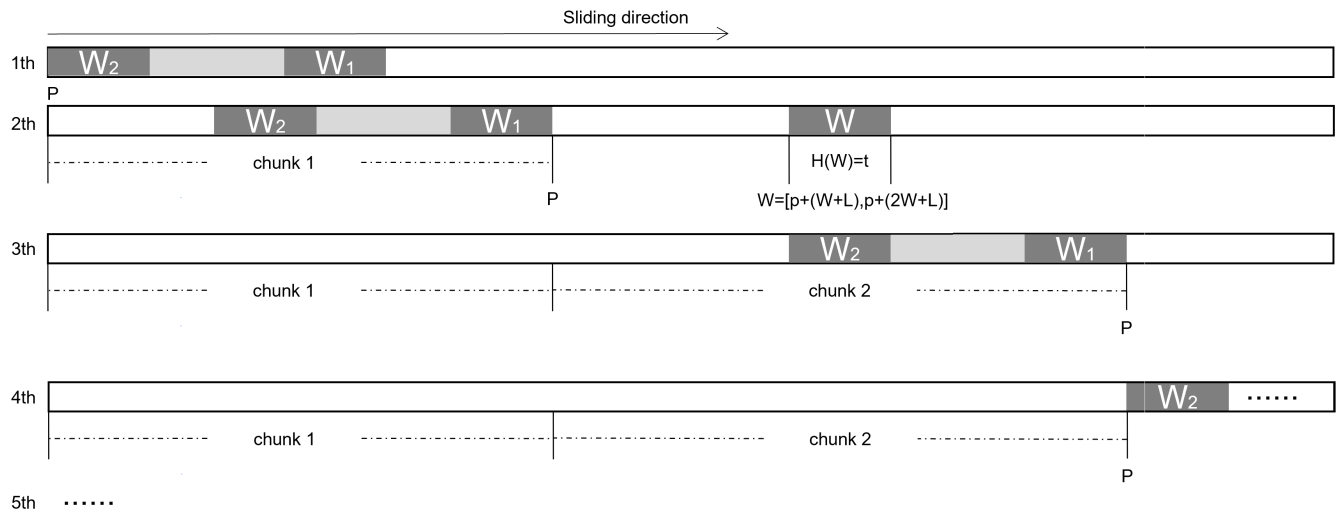

FIGURE 4. **Detailed Workflow of Chunking Technology Based on Double Sliding Window.**

$ListIsoP(X_i, X_{i+1})$ is represented as the longest prefix part of X_i and X_{i+1}. $ListIsoS(X_i, X_{i+1})$ is the longest suffix part of $Sur(X_i, ListIsoP(X_i, X_{i+1}))$ and $Sur(X_{i+1}, ListIsoP(X_i, X_{i+1}))$.

Next, we will analyze the parsing process of $|A'|_{ME}$, Let's first set $y = \{y_1, y_2, y_3 \ldots y_n\}$, Let's say that the function AVG (y) is expressed as the average of the set y, that is,

$$AVG(y) = \frac{y_1 + y_2 + y_3 \ldots y_n}{n} \tag{7}$$

Because $|A|_{ME}$ consists of $|A|_{HT}$ and $|A|_{HC}$.

$|A|_{HT}$ equals:

$$|A'|_{HT} = \frac{AVG_{HT} \sum_{i=1}^{k-1} Neo(x_i, x_{i+1})}{AVG(X_i)} \tag{8}$$

$|A|_{HC}$ equals:

$$|A'|_{HC} = \frac{AVG_{HC} |X_i|}{AVG(X_i)} \tag{9}$$

Substitute Formula (8) and (9) into Formula (1) to get:

$$|A'|_{ME} = \frac{AVG_{HT} \sum_{i=1}^{k-1} Neo(x_i, x_{i+1}) + AVG_{HC} |X_i|}{AVG(X_i)} \tag{10}$$

Finally, we substitute Formula(10) and Formula (4) into Formula (2) to obtain the theoretical re deletion ratio, as in (11), shown at the bottom of the page.

From Formula (11), we can see that if we want to make R_D as large as possible, we can make the size of data blocks more average to increase R_D and reduce data block fluctuations as much as possible, that is, reduce different copies of the same file. If the data blocks are more average, the variance can be reduced, and then the $|A|_{HT}$ and $|H|_{HC}$ stored in memory can be reduced, thus increasing the re-deletion ratio.

C. USING MARKOV MODEL TO PREDICT

During the operation of ocean observation data, we found that we need to consume more resources to delete redundant data more frequently due to a large amount of data. Therefore, our team introduced Markov model prediction to identify the cutting points of data flow more effectively. The characteristics of Markov are precisely what we think. Our team first solved how to convert the actual problem into data that can be applied by Markov prediction. Before we want to know how to predict Markov, we must first understand the Markov process, which is a random process. The Markov process is an essential technique for examining the state space of discrete event dynamic systems. Stochastic process theory provides its mathematical foundation. This time, we will use the most straightforward Markov process, the first-order Markov process. The system state S (t+1) is the only state to which it is connected (t), not to the previous state. The formula can be expressed as:

$$p(X_{t+1}|X_t, \ldots, X_1) = p(X_{t+1}|X_t) \tag{12}$$

When we deal with the state problem, the first thing we think of is to divide the size of the data block in the actual problem into three states: large data block, medium data block, and small data block. We mark them as L, M, and S. Mark blocks smaller than 7000 as S, blocks more significant than 7000 and more significant than 15000 as M, and blocks larger than 15000 as L. At this point, we have solved the

$$R_D(A') \approx \frac{1 - (\sum_{i=1}^{k-1} Neo(x_i, x_{i+1}) + \frac{AVG_{HT} \sum_{i=1}^{k-1} Neo(x_i, x_{i+1}) + AVG_{HC} |X_i|}{AVG(X_i)})}{|A|} \tag{11}$$

abstraction of practical problems. The result our team wants is to input the status of the previous data block. The system can give us the status of the following data block. If it is S, the window will continue to move backward. If it is M, the window will directly slide 7000 without calculating the hash value in the window to save resources. If it is L, the window will directly slide 15000 to save resources and response time. Next is the actual operation of our team. The first step of Markov prediction is to calculate the state transition probability matrix. If the Markov chain is in state j at time $(t-1)$ and moves to state I at time t, the transition probability is recorded as

$$p_{ij} = (X_t = i | X_{t-1} = j), \quad i = 1, 2, \ldots; j = 1, 2, \ldots \tag{13}$$

among $p_{ij} \geq 0, \sum_i p_{ij} = 1$ The state transition probability matrix is composed of state transition probability:

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \cdots \\ p_{21} & p_{22} & p_{23} & \cdots \\ p_{31} & p_{32} & p_{33} & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \tag{14}$$

Consider Markov chain $X = \{X_0, X_1, X_2, \ldots, X_t, \ldots\}$ The probability distribution at time t is called the state distribution at time t, which is recorded as

$$\Pi(t) = \begin{bmatrix} \pi_1(t) \\ \pi_2(t) \\ \vdots \end{bmatrix} \tag{15}$$

among,

$$\pi_i(t) = P(X_t = i), \quad i = 1, 2, 3, \ldots \tag{16}$$

The initial distribution vector $\Pi(0)$ typically only has one component that is 1, with the remaining components being zero, suggesting that the Markov chain begins from a particular state. The state distribution at the time (t − 1) and the transition probability distribution can be used to calculate the state distribution of Markov chain X at time t.

$$\pi(t) = P\pi(t-1) \tag{17}$$

Recursive:

$$\pi(t) = P^t \pi(0) \tag{18}$$

P^t here is called the t-step transition probability matrix,

$$P^t_{ij} = P(X_t = i | X_0 = j) \tag{19}$$

The pseudocode of this method is shown below:

Algorithm 2 Markov Prediction

```
Input: Previous chunking type;
Output: Next chunking type;
1:  data=["S","M","L","S","M",
    "S","M","L","M","S","M","L","S",
    "M","S","M","S","M","L","S","S","S","S"]
2:  Sdata=[]
3:  for i in range(len(data)-1) do
4:      if $data[i] = “S” \rightarrow Sdata.append(data[i+1])$
5:  end for
6:  Similarly, Mdata=[] and Ldata=[] can be obtained
7:  SS=0,SM=0,SL=0
8:  for i in range(len(Sdata)) do
9:      if $Sdata[i] = S \rightarrow SS = SS + 1$
10:     if $Sdata[i] = M \rightarrow SM = SM + 1$
11:     if $Sdata[i] = L \rightarrow SL = SL + 1$
12: end for
13: P11=SS/len(Sdata),P12=SM/len(Sdata)
14: P13=SL/len(Sdata)
15: In the same way, we can get p21, p22, p23, p31, p32, p33
16: P=[[p11,p12,p13],[p21,p22,p23],[p31,p32,p33]]
17: paifirst=[]
18: for i in range() do
19:     pai=pai.dot(P)
20:     a=list(pai).index(max(list(pai)))
21:     if a=0 return S
22:     if a=1 return M
23:     if a=2 return L
24: end for
```

V. EXPERIMENTAL RESULTS AND DISCUSSION

Next, we will introduce our experimental environment and data set sources and compare FSC, BSW, and DSW regarding the total number of blocks, data block distribution, and deduplication ratio.

TABLE 1. Specific operating environment of the experiment.

Device name	DELL XPS 8950
Processor	12th Gen Intel(R)Core(TM)i7-12700 2.10 GHz
RAM	64.0GB(63.7GB Available)
System type	Windows11/Ubnutu 22.04
Display adapter	NVIDIA GeForce RTX 3060 12GB

TABLE 2. Details of all data sets used in this experiment.

	Name	Source	Size
1	Ocean observation dataset 1(OD1)	Ocean observation station collection	1.94GB
2	Ocean observation dataset 2(OD2)	Ocean observation station collection	1.83GB
3	Ocean observation dataset 3(OD3)	Ocean observation station collection	1.01GB
4	Ocean observation dataset 4(OD4)	Ocean observation station collection	1.92GB
5	General Dataset 1(GD1)	networkrepository.com	57.3MB
6	General Dataset 2(GD2)	networkrepository.com	661MB
7	General Dataset 3(GD3)	networkrepository.com	852MB
8	General Dataset 4(GD4)	networkrepository.com	878MB

A. EXPERIMENTAL ENVIRONMENT AND DATA SET SOURCE

The computers used in this experiment are shown in the table below:

And the data set used in this experiment is shown as follows. The data sets 1-4 used in this paper are all from publicly

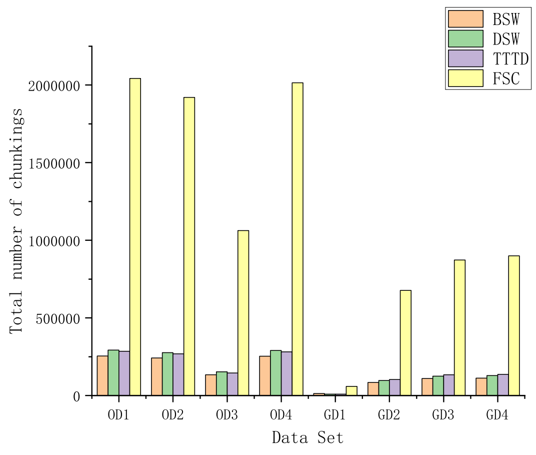

FIGURE 5. Total number of data blocks identified by each deduplication system.

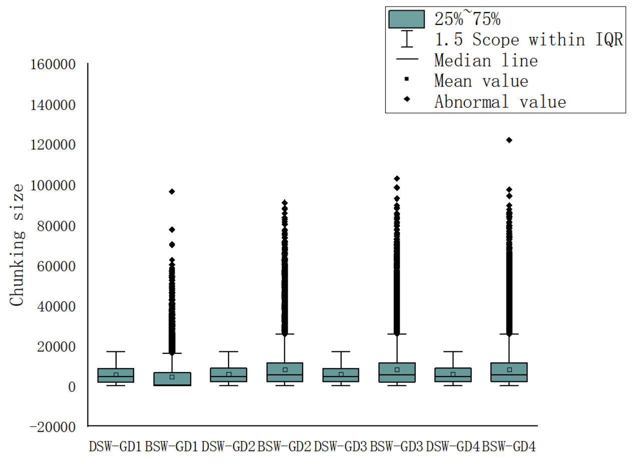

FIGURE 6. Performance of common data and ocean observation data in different systems.

generated marine ship trajectory data, which are a set of time series data sets generated by time changes, while the data set 5-8 is a data set generated by public daily network life. The more important reason for listing different types of data sets is to observe whether DSW is more suitable for deleting data generated by time series.

B. TOTAL CHUNK COUNT

It can be seen from the information in the figure that DSW can always identify more data blocks than BSW, regardless of the time series data set facing the track of ocean ships or the messy data set in daily life, and since FSC always divides more data blocks according to the fixed length than the other two. Among them, DSW can recognize 14.9%, 14.6%, 14.8% and 14.4% more than BSW when facing the time series data set of ocean ship track.

C. MAX/MIN CHUNK SIZE AND CHUNKING DISTRIBUTION

Next, we will discuss the size of data blocks. From Figure 6, we can see the difference between DSW and BSW. The size of data blocks in DSW is all concentrated in the range of 1.5 IQR, and there are no outliers. However, there are too many outliers in BSW. Thanks to limiting the size of data blocks in DSW, data blocks are concentrated in one area, as shown in the figure. On the other hand, BSW must allow the size of data blocks, which makes it impossible to centralize a large amount of data, and the final results also show that there are a large number of outliers.

In addition, we can view the distribution of data block size from another angle. As shown in Figure 7, we selected eight different results for comparison, including the combination of DSW and GD1, GD2, OD1, and OD2, and the combination of BSW and GD1, GD2, OD1, and OD2. Two classifications and eight groups of data can be viewed. From the figure, we can see that the combination of DSW decreases with the increase

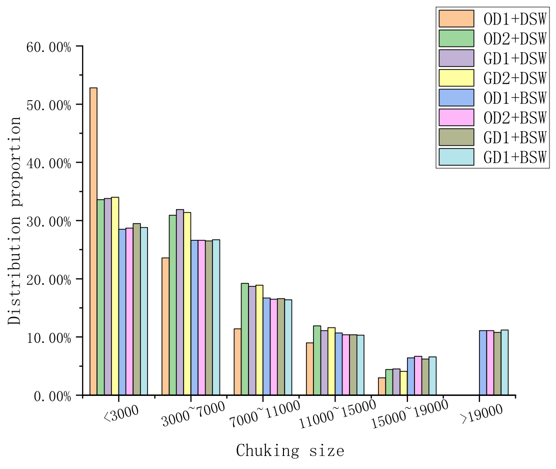

FIGURE 7. Distribution of common data and ocean observation data in different systems.

of the data block size due to the size of the data block being limited by DSW. On the contrary, the BSW decreases with the increase in the data block size.

D. DEDUPLICATION RATIO

Next, we will compare the re-deletion ratio of DSW and BSW. Figure 8 shows that DSW performs better than BSW in the face of ocean data. Regarding common data, DSW and BSW have advantages and disadvantages. This is because most of the ocean data are time series data, and it is advantageous to limit the size of data blocks when facing time series data. As shown in Figure 9, we compared TTTD and DSW in various datasets and found that DSW was slightly stronger than TTTD when faced with public datasets. However, this does not represent the advantages or disadvantages of the two algorithms, as TTTD requires setting spare divisors and thresholds, which will result in different results when faced with different spare divisors and thresholds. When facing

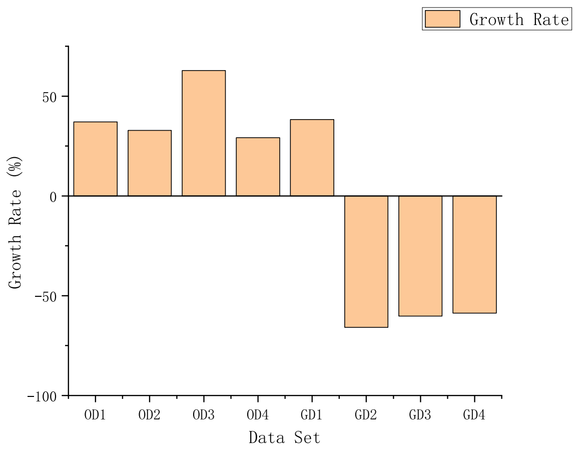

FIGURE 8. The improvement of DSW over BSW under different data types.

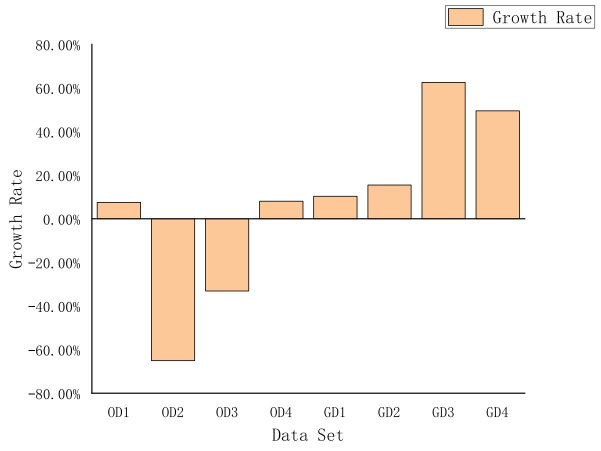

FIGURE 9. The improvement of DSW over TTTD under different data types.

TABLE 3. Accuracy of Markov prediction and proportion of saved resources under ocean observation data.

Data set	Accuracy	Reduce the proportion of sliding resources	Save steps
OD1	55.73%	1.897%	316107309
OD2	57.21%	1.901%	299051235
OD3	58.69%	1.901%	165406500
OD4	57.72%	1.901%	313630548

ocean observation data, the two algorithms each have different performances, and for the same reason, this does not represent the final result of TTTD.

E. MARKOV PREDICTION CONCLUSION

This section will describe the results of adding Markov prediction to the data deduplication system. First, we take a part of the known data set that has yet to be applied in other experiments as input to train the probability matrix of the Markov model. Our team will evaluate the accuracy of the model, the impact on the re-deletion rate, and the most critical saving steps. As shown in the following table, we can conclude that the data deduplication system, after adding the Markov model, can reduce the resource waste by 1.9% on average.

VI. CONCLUSIONS AND FUTURE PROSPECTS

We propose a new method to calculate the deduplication ratio. We find that BSW does not perform well in the face of ocean data, so we introduce a new algorithm, DSW, which performs much better in the face of ocean data than existing algorithms, and has an absolute size limit on the block size. This algorithm can improve the performance of applications that use content-based blocking. In future work, we will be committed to solving the optimization operation of DSW in the face of ordinary data so that it can play an outstanding role in various scenarios.

REFERENCES

[1] B. Agarwal, A. Akella, A. Anand, A. Balachandran, P. Chitnis, C. Muthukrishnan, R. Ramjee, and G. Varghese, "EndRE: An end-system redundancy elimination service for enterprises," in *Proc. NSDI*, 2010, pp. 419–432.
[2] M. W. Al Azad and S. Mastorakis, "The promise and challenges of computation deduplication and reuse at the network edge," *IEEE Wireless Commun.*, vol. 29, no. 6, pp. 112–118, Dec. 2022.
[3] D. Bhagwat, K. Eshghi, D. D. E. Long, and M. Lillibridge, "Extreme binning: Scalable, parallel deduplication for chunk-based file backup," in *Proc. IEEE Int. Symp. Modeling, Anal. Simulation Comput. Telecommun. Syst.*, Sep. 2009, pp. 1–9.
[4] D. R. Bobbarjung, S. Jagannathan, and C. Dubnicki, "Improving duplicate elimination in storage systems," *ACM Trans. Storage*, vol. 2, no. 4, pp. 424–448, Nov. 2006.
[5] A. Broder and M. Mitzenmacher, "Network applications of Bloom filters: A survey," *Internet Math.*, vol. 1, no. 4, pp. 485–509, Jan. 2004.
[6] A. Z. Broder, "On the resemblance and containment of documents," in *Proc. Compress. Complex. SEQUENCES*, 1997, pp. 21–29.
[7] J. Cao, Y. Jin, A. Chen, T. Bu, and Z.-L. Zhang, "Identifying high cardinality Internet hosts," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 810–818.
[8] F. Douglis and A. Iyengar, "Application-specific delta-encoding via resemblance detection," in *Proc. USENIX Annu. Tech. Conf., Gen. Track*, San Antonio, TX, USA, 2003, pp. 113–126.
[9] K. Eshghi and H. K. Tang, "A framework for analyzing and improving content-based chunking algorithms," Hewlett-Packard Labs, Palo Alto, CA, USA, Tech. Rep. TR 30.2005, 2005.
[10] K. Eshghi, M. Lillibridge, L. Wilcock, G. Belrose, and R. Hawkes, "Jumbo store: Providing efficient incremental upload and versioning for a utility rendering service," in *Proc. FAST*, vol. 7, 2007, pp. 123–138.
[11] D. Geer, "Reducing the storage burden via data deduplication," *Computer*, vol. 41, no. 12, pp. 15–17, Dec. 2008.
[12] N. Jain, M. Dahlin, and R. Tewari, "TAPER: Tiered approach for eliminating redundancy in replica synchronization," in *Proc. Fast*, vol. 5, 2005, p. 21.
[13] E. Kruus, C. Ungureanu, and C. Dubnicki, "Bimodal content defined chunking for backup streams," in *Proc. Fast*, 2010, pp. 239–252.
[14] P. Kulkarni, F. Douglis, J. D. LaVoie, and J. M. Tracey, "Redundancy elimination within large collections of files," in *Proc. USENIX Annu. Tech. Conf., Gen. Track*, 2004, pp. 59–72.
[15] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezis, and P. Camble, "Sparse indexing: Large scale, inline deduplication using sampling and locality," in *Proc. Fast*, vol. 9, 2009, pp. 111–123.
[16] B. Mahesh, K. P. Kumar, S. Ramasubbareddy, and E. Swetha, "A review on data deduplication techniques in cloud," in *Proc. Embedded Syst. Artif. Intell. (ESAI)*, Fez, Morocco, 2020, pp. 825–833.
[17] U. Manber, "Finding similar files in a large file system," in *Proc. USENIX Winter*, vol. 94, 1994, pp. 1–10.

[18] N. Mandagere, P. Zhou, M. A. Smith, and S. Uttamchandani, "Demystifying data deduplication," in *Proc. ACM/IFIP/USENIX Middleware Conf. Companion*, Dec. 2008, pp. 12–17.
[19] G. S. Manku and R. Motwani, "Approximate frequency counts over data streams," in *Proc. 28th Int. Conf. Very Large Databases (VLDB)*, 2002, pp. 346–357.
[20] A. Muthitacharoen, B. Chen, and D. Mazières, "A low-bandwidth network file system," in *Proc. 18th ACM Symp. Oper. Syst. Princ.*, Oct. 2001, pp. 174–187.
[21] F. Ni and S. Jiang, "RapidCDC: Leveraging duplicate locality to accelerate chunking in CDC-based deduplication systems," in *Proc. ACM Symp. Cloud Comput.*, Nov. 2019, pp. 220–232.
[22] F. Ni, X. Lin, and S. Jiang, "SS-CDC: A two-stage parallel content-defined chunking for deduplicating backup storage," in *Proc. 12th ACM Int. Conf. Syst. Storage*, May 2019, pp. 86–96.
[23] T. R. Nisha, S. Abirami, and E. Manohar, "Experimental study on chunking algorithms of data deduplication system on large scale data," in *Proc. Int. Conf. Soft Comput. Syst.*, L. P. Suresh and B. K. Panigrahi, Eds. New Delhi, India: Springer, 2016, pp. 91–98.
[24] P. G. Shynu, R. K. Nadesh, V. G. Menon, P. Venu, M. Abbasi, and M. R. Khosravi, "A secure data deduplication system for integrated cloud-edge networks," *J. Cloud Comput.*, vol. 9, no. 1, pp. 1–12, Dec. 2020.
[25] C. Policroniades and I. Pratt, "Alternatives for detecting redundancy in storage systems data," in *USENIX Annu. Tech. Conf., Gen. Track*, 2004, pp. 73–86.
[26] P. Prajapati and P. Shah, "A review on secure data deduplication: Cloud storage security issue," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 7, pp. 3996–4007, Jul. 2022.
[27] S. Quinlan and S. Dorward, "Venti: A new approach to archival data storage," in *Proc. Conf. File Storage Technol. (FAST)*, 2002.
[28] D. Reinsel, J. Gantz, and J. Rydning, "Data age 2025: The evolution of data to lifecritical. Don't focus on big data; focus on the data that's big," Int. Data Corp. (IDC), White Paper, 2017.
[29] N. T. Spring and D. Wetherall, "A protocol-independent technique for eliminating redundant network traffic," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, Aug. 2000, pp. 87–95.
[30] A. Tridgell et al., "The rsync algorithm," Tech. Rep., 1996.
[31] W. Xia, D. Feng, H. Jiang, Y. Zhang, V. Chang, and X. Zou, "Accelerating content-defined-chunking based data deduplication by exploiting parallelism," *Future Gener. Comput. Syst.*, vol. 98, pp. 406–418, Sep. 2019.
[32] W. Xia, H. Jiang, D. Feng, L. Tian, M. Fu, and Y. Zhou, "Ddelta: A deduplication-inspired fast delta compression approach," *Perform. Eval.*, vol. 79, pp. 258–272, Sep. 2014.
[33] L. You and C. T. Karamanolis, "Evaluation of efficient archival storage techniques," in *Proc. MSST*, 2004, pp. 227–232.
[34] L. L. You, K. T. Pollack, and D. D. E. Long, "Deep store: An archival storage system architecture," in *Proc. 21st Int. Conf. Data Eng. (ICDE)*, 2005, pp. 804–815.
[35] H. Yuan, X. Chen, J. Wang, J. Yuan, H. Yan, and W. Susilo, "Blockchain-based public auditing and secure deduplication with fair arbitration," *Inf. Sci.*, vol. 541, pp. 409–425, Dec. 2020.
[36] 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 Trans. Comput.*, vol. 66, no. 2, pp. 199–211, Feb. 2017.
[37] 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. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 1337–1345.
[38] B. Zhu, K. Li, and R. H. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file system," in *Proc. Fast*, vol. 8, Feb. 2008, pp. 269–282.

SHUAI GUO received the B.S. degree from Taishan University, Taian, China, in 2012, and the M.S. degree in computer applications and the Ph.D. degree in computer architecture from the Ocean University of China, Qingdao, China, in 2014 and 2019, respectively.

Since 2019, he has been a Faculty Member with the Qingdao University of Technology. His research interests include the Internet of Things, data deduplication, artificial intelligence, and big data.

XIAODONG MAO received the bachelor's degree in software engineering from Harbin Commercial University, in 2021. He is currently pursuing the master's degree with the Qingdao University of Technology. His research interests include data deduplication, big data, and artificial intelligence.

MENG SUN received the degree in computer science and technology from Shenyang Agricultural University, in June 2022. She is currently pursuing the master's degree in computer science and technology with the Qingdao Institute of Technology. Her research interests include trajectory prediction and the Internet of Things.

SHUANG WANG received the bachelor's degree in mathematics and applied mathematics from the Qingdao University of Technology, in June 2022, where she is currently pursuing the master's degree in computer science and technology. Her research interests include quantum machine learning, quantum error correction, and image processing.