Received September 3, 2021, accepted September 25, 2021, date of publication September 29, 2021, date of current version October 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3116270

Research on Routing Strategy in Cluster Deduplication System

QINLU HE[1], (Member, IEEE), GENQING BIAN[1], (Member, IEEE), WEIQI ZHANG[1], FAN ZHANG[1], SHENGQIANG DUAN[2], AND FENGLANG WU[3]

[1]School of Information and Control Engineering, Xi'an University of Architecture and Technology, Xi'an 710054, China
[2]School of Management, Xi'an University of Architecture and Technology, Xi'an 710054, China
[3]Network Information Department, The First Affiliated Hospital of Xi'an Jiaotong University, Xi'an 710061, China

Corresponding authors: Qinlu He (luluhe8848@Hotmail.com) and Fenglang Wu (wufenglang@xjtufh.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872284; in part by the Industrial Field of General Projects of Science and Technology Department of Shaanxi Province under Grant 2020GY-012; in part by the Industrialization Project of Shaanxi Provincial Department of Education under Grant 21JC017; in part by the "Thirteenth Five-Year" National Key Research and Development Program Project under Project 2019YFD1100901; in part by the Natural Science Foundation of Shaanxi Province, China, under Grant 2014JM2-6127; in part by the Talent Fund Project of Xi'an University of Architecture and Technology under Grant RC1707; in part by the Youth Fund Project of Xi'an University of Architecture and Technology under Grant QN1726; and in part by the Natural Science Project of Xi'an University of Architecture and Technology under Grant ZR18050.

ABSTRACT A cluster deduplication system can coordinate the work of multiple nodes, which can better alleviate the disk index bottleneck existing in the large-scale data backup system. However, there is a problem of isolated islands of information among nodes during data deduplication. When the servers use the query mode to route data, a large amount of system overhead is required to ensure a high deduplication rate and low throughput rate. At the same time, while the servers cannot obtain a higher deduplication rate if the servers adopt the stateless routing method. Data routing strategy can greatly affect the overall performance of the system. The concept of data frequency is proposed in this paper, and the classified routing strategy is designed. In the metadata server, a byte-shaped Bloom filter for recording the occurrence frequency of data blocks is maintained to record the occurrence frequency of data blocks. The values in the Bloom filter are counted. Then the frequency of the data blocks is compared with the configured threshold value to determine whether the data is "hot data". We use stateful routing to send "clod data" to the storage nodes and use stateless routing to send the hot data to the storage nodes. Experimental results show that the classifying routing algorithm based on the frequency of data can greatly reduce the overhead of the system while guaranteeing the deduplication rate of the deduplication system as well as improve system throughput and real-time processing capabilities. Compared with the fully stateful routing scheme, our method only loses less than 2% of the deduplication rate, which reduces the communication query overhead by more than 25% and improves the real-time processing capability of the storage system.

INDEX TERMS Data frequency, classified routing strategy, data deduplication, load balancing.

I. INTRODUCTION

With the rapid development of Internet technology and the advent of big data, the total amount of digital information in the world is growing exponentially. According to the statistics of the Internet Data Center Business (IDC), a market research organization, as early as 2010, the total global data volume has exceeded 1.8ZB [1]. In 2015, IDC predicted that the annual growth rate of the global data volume would remain at 50% in the future. By 2025, the total amount of global data will reach 175ZB, 22 times that of 2011. Among them, my country's data volume will reach 8.6ZB, accounting for about 21% of the world's [2]. The figure indicates that we have to manage a larger scale of data in the future, we will have to manage a larger scale of data. Figure 1 shows the amount of global digital information in recent years and expected in 2025.

Big data refers to the large scale of the data set's capacity which is difficult to store, analyze and process by the traditional technical methods. The big data has brought new challenges to the existing storage systems on the capacity, maintainability, throughput performance, scalability, and

The associate editor coordinating the review of this manuscript and approving it for publication was Qilian Liang.

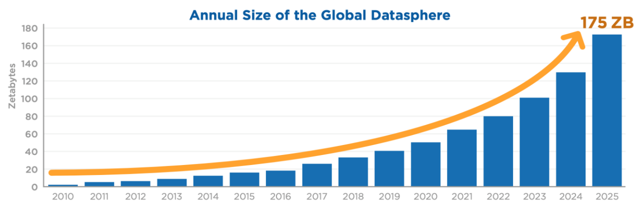

FIGURE 1. The total amount of global digital information between 2010 and 2025 and expected in 2025.

reliability. For the great demand on the storage space for big data, it is necessary to eliminate redundancy on which saving data and optimizing the use of storage space can effectively alleviate the pressure on storage capacity. The existing technologies to eliminate redundant data mainly include data compression [3], [36], [39], [40]–[45] and data deduplication [4], [37], [38], [46]. In the above two technologies, data compression refers to a technology that uses fewer bits of storage to express the original data by using encoding methods. However, although the data compression algorithms can effectively reduce the size of data, this technology has relatively strict space limitations. Using data compression for two files with the same content will still get two identical compressed encoded files. Further complete the reduction of stored data. The difference is that data deduplication technology is a technology that discovers and eliminates duplicate content in the data stream to improve data storage efficiency and reduce data storage costs [32], [47]–[49]. This technology reduces the data storage occupancy of the system by eliminating duplicate data in the data stream and only retaining the data that appears for the first time.

In enterprises, it is necessary to adopt some traditional backup technologies to improve the security of the system, such as periodic backup and snapshot technologies. These technologies increase the data redundancy in the storage space and make the amount of duplicate data in the storage system reaching more than 90% [5], [50]. These duplicate data increase the cost of data storage and processing. The disk-based backup system with the deduplication function can first delete the duplicate data to achieve data compression, thereby greatly reducing the cost of data storage. Therefore, research on data deduplication technology is necessary.

When processing massive amounts of data, a single-node deduplication system has limited computing power and storage space. Therefore, there is a disk index bottleneck problem in the system, which cannot meet the deduplication requirements of massive data. Cluster data deduplication technology based on cloud storage systems is a new research field in recent years [3]–[6]. This technology builds a data deduplication system in the form of a cluster, which can use multiple nodes to coordinate work simultaneously and better alleviate large-scale disk index bottleneck problems in the data backup system. However, when the cluster deduplication system performs deduplication, there is a problem of islands of information among nodes. In order to achieve complete data deduplication, it is necessary to detect duplicate data within the entire storage system. The entire storage system cluster often contains hundreds or thousands of nodes, which makes it impossible to check whether there is duplicate data in the entire storage system. Node deduplication technology cannot be directly applied to this environment, and further research on deduplication technology in a distributed cluster environment is needed.

In a cluster deduplication system, multiple nodes cooperate to complete the task of deduplication and data backup. Because of the large number of nodes, it is impossible to detect duplicate data within the scope of the entire system. For duplicate data detection on a node basis, there may be the same data between different storage nodes, so there is a large amount of duplicate data between nodes, resulting in a low deduplication rate of the entire deduplication system. It is the so-called information island problem between nodes. In this regard, according to the EMC proposal, by using query-style stateful routing, the storage node is queried to ensure that the data to be stored is sent to the most appropriate node for processing before routing data. However, this kind of stateful routine has a large amount of overhead and a low throughput rate, which affects the deduplication performance of the cluster. Through our research, we found that in the process of data deduplication, some "hot data" recurred more frequently, while the other "cold data" recurred less frequently. Due to many repetitions of hot data, stateless routing can also achieve a higher deduplication rate. In contrast, stateful routing can no longer achieve a large increase in deduplication rate but will waste a lot of communication and query overhead. Therefore, the routing strategy studied in this paper reduces the loss of the deduplication rate in the cluster environment, minimizes the system's calculation and communication overhead, and improves the throughput rate and the system's real-time processing capabilities.

This article mainly proposes a classification routing scheme based on the frequency of data. Because there are also two eight rules in the storage field, a small number of commonly used data will appear more frequently, while most of the less commonly used data appear more frequently low. Therefore, this article uses statistical methods to distinguish and process data based on the frequency of data. For cold data, the program uses stateful routing to ensure its deduplication rate; for hot data, the program uses stateless routing. Methods to reduce calculation and query overhead and increase throughput, thereby improving the overall performance of the system.

II. RELATED WORK

HYDRAstor [7] is a deduplication cluster. It uses 64KB data blocks as the granularity, routes them to different deduplication server nodes based on a distributed hash table, and completes deduplication within the nodes according to the block granularity. This technology uses block granularity to better balance the deduplication rate and data overhead and obtain a higher throughput rate. However, it fails to detect similar data by using the locality of the data flow in the cluster deduplication system. The granularity of data blocks larger results in a relatively low deduplication rate when deduplication is performed inside the node. Based on the working principle of the HYDRAstor architecture, researchers designed the HydraFS [10] file system. When this file system uses 64KB data blocks as the routing granularity, the throughput per second can reach more than 100 megabytes.

Bhagwat *et al.* designed an Extreme Binning technology [8]. This technology uses the file as the basic unit in the routing process, performs block processing on the file, uses the SHA1 hash algorithm to calculate the fingerprint of each block, and then selects the smallest fingerprint value among all the block fingerprint values in the file as the file feature are used in subsequent routing decisions. The limit binning method mainly uses a stateless data routing mechanism based on the similarity of files. However, when the similarity in the data stream is not easy to detect, it will greatly affect the deduplication rate of the system, considering the uneven distribution of the file size, the routing mechanism cannot achieve a good load balance between nodes. Changes in data deduplication rate will also be affected.

In order to avoid occupying a large amount of memory space and achieve a better data deduplication effect, Sparse Indexing uses a part of the fingerprint index sequence to represent the fingerprint collection of the entire data stream by sampling. However, these two methods strongly depend on the locality of the data flow.

The DEBAR [9] system is a distributed data de-duplication scheme proposed based on the DDFS scheme to increase the system capacity. Regarding the solution adopted by the previous deduplication system, although cache prefetching technology and bloom filters can reduce most of the block index disk lookups, it is still inevitable that there will be some random disk I/O overhead.

The ChunkStash scheme [17] specifically improves the hit rate of random disk reads and writes, thereby increasing the throughput of the system. This solution mainly adds a level of Flash Memory between the traditional RAM and DISK, which is equivalent to a caching mechanism. Because Flash has a good read and writes speed, the metadata search speed can be improved. The results show that it is better than ordinary disk-based. The index deduplication system can improve the throughput of several to several tens of times. Due to the use of Flash Memory, algorithms and data structures that use Flash memory and Flash memory perception are proposed in the solution to obtain faster index queries. However, since the random update of the Flash memory is quite low, this article overcomes the random update of the Flash by converting the random update of the Flash memory into an extended operation of the Log structure.

PretecTier [14] technology and the routing scheme proposed by EMC. Among them, EMC uses the principle of data locality to propose a routing strategy based on Superchunk [15]. It divides the data stream into data blocks and then assembles multiple consecutive data blocks into superblocks. The superblock is used as the basic unit of routing processing to maintain locality [16] and supports the cluster deduplication system. Scalability. In the storage node, data deduplication is performed according to more fine-grained data blocks to increase the data deduplication rate. This scheme can obtain a high deduplication rate under the premise of maintaining a balanced data distribution. However, similar to the broadcast system communication overhead and frequent block fingerprint query, it seriously affects the deduplication performance of the cluster [18], [19].

Due to the exponential growth of the total amount of global digital information, single-node deduplication systems have become increasingly unable to meet the needs of large-scale data processing, and cluster deduplication technologies have emerged. While the cluster deduplication system alleviates the disk bottleneck problem, it also brings information islands between nodes. In order to query more storage nodes and eliminate more duplicate data, the system needs to pay a lot of communication and query overhead, which affects the overall performance of the system. At the same time, when the cluster is performing data distribution, it is also necessary to ensure load balance among storage nodes as much as possible. The above problems can be effectively solved by studying and using appropriate data routing strategies. Therefore, the research in this area has important theoretical significance and application value for the development and application of storage technology. This paper takes the twenty-eight rule existing in the storage field as a research entry point. In order to reduce the query communication overhead of stateful routing in the cluster deduplication system, the concept of data frequency is introduced, and a classification routing strategy based on data frequency is proposed. In order to alleviate the problem of load imbalance, the classification routing strategy is improved, and the concept of stateless admission threshold is used to dynamically control the routing admission conditions of each storage node to avoid the blind aggregation of some high-frequency data and obtain a more uniform load-distribution result.

III. CLASSIFICATION ROUTING STRATEGY BASED ON DATA FREQUENCY

An efficient data routing strategy is the key to a cluster deduplication system with good indicators including ensuring a high deduplication rate, low system overhead, high throughput, and load balancing. However, it is difficult to balance the indicators as some conflicts occurred among them. There are conflicts between these indicators, so they need to be focused on. In the scheme proposed by EMC, to increase the deduplication rate of the deduplication system, the stateful routing used has a high cost, which affects the deduplication performance of the cluster. Although stateless routing reduces the cost, it cannot be guaranteed a deduplication rate. According to the twenty-eight rule in the storage field, the study found that by distinguishing the frequency of data occurrence, the part of the data that appears more frequently due to a large number of repetitions, even if stateless routing is used, the deduplication rate can be obtained. It is guaranteed that stateful routing can no longer achieve a large increase in the deduplication rate. The continued use of stateful routing will waste a lot of communication and query overhead. Therefore, the statistics of the frequency of data blocks proposes data frequency. Taking the data frequency as a parameter, a data routing strategy in the deduplication cluster based on frequency is designed. This solution maintains a byte bloom filter and a frequency array that records the frequency of data blocks in the server node, which is used to determine whether the data is ''hot data'', and uses query-style stateful routing for cold data Send to the deduplication node and send the hotspot data to the deduplication node using a stateless routing method.

A. SYSTEM STRUCTURE

The architecture of the deduplication cluster backup system used in this article is shown in Figure 2. The system is mainly composed of a client, a metadata server, and a cluster of storage nodes. The client is responsible for the following tasks including collecting user backup data, dividing the data into blocks separately, making up the superblock, completing the calculation of fingerprints, selecting the characteristic fingerprints, sending the superblock and its fingerprint information to the metadata server.

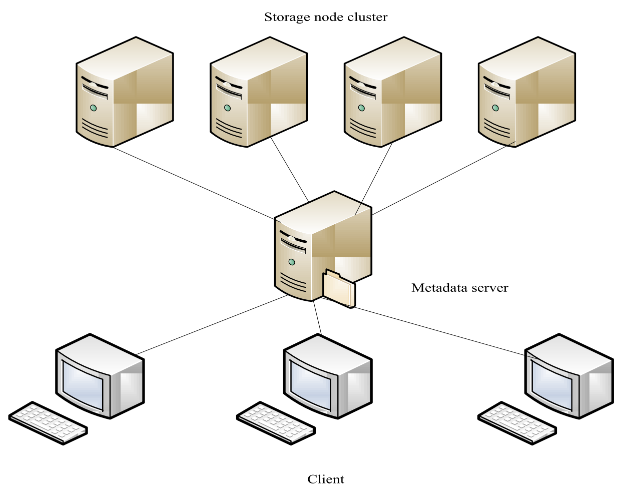

FIGURE 2. Cluster backup system architecture.

The metadata server is also a front-end processor or server node, which is responsible for sending the data blocks of the client to the corresponding nodes in the storage node cluster to complete data backup. The server node maintains a byte bloom filter. This byte bloom filter records how often the processed data block fingerprint hits the bloom filter, it represents the frequency of the data block. According to this frequency, determine the routing strategy for sending data blocks and fingerprints to different storage nodes on the backend for backup. Server nodes are mainly used to maintain frequency information of data block fingerprints, load information of storage nodes, and adopt different data routing strategies. It does not require many computing resources. The system can use one or more server nodes according to the distribution of client nodes.

The storage node cluster is composed of multiple storage nodes, that is, data servers. Each storage node is equivalent to a data storage layer in a single deduplication system architecture and is responsible for data backup. Each storage node maintains a Bloom Filter and it is the part of the fingerprint caches in the memory aimed at quick finding if the fingerprint of the data block exists. When the storage node receives the superblock to be processed, it first divides it into small data blocks before assembly according to fixed-length blocks and then queries the fingerprints of these data blocks. First, look up the fingerprint cache to see if the fingerprint exists. If it does not exist, continue to query Bloom Filter. If the fingerprint of the data block to be processed already exists, it indicates that this is a duplicate data block and can be deleted. If the fingerprint cannot be found in the Bloom Filter, it indicates a new data block, store the data block and update the Bloom Filter at the same time.

B. CLASSIFICATION ROUTING STRATEGY BASED ON THE FREQUENCY

1) BYTE BLOOM FILTER

In this paper, the routing of data blocks needs to be based on the frequency information of the data blocks. The improved bloom filter, that is, the byte bloom filter, is used at the server node to maintain the frequency information of the fingerprint of the data block. Bloom filter is a data structure proposed by Burton Howard Bloom in 1970 to store data in the form of a bitmap, which can quickly determine a specified data object [20], [21]. Its basic structure is shown in Figure 3, including a bitmap with a length of m and k independent hash functions, in which the value range of each hash function h(x) is 1 to m.

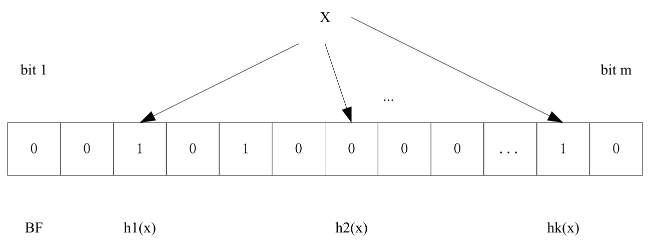

FIGURE 3. The structure of the bloom filter.

When using a Bloom filter, first initialize all its bits to 0. For each element x, calculate the corresponding k hash function values and use these hash functions in the Bloom filter respectively. The bit where the value is located is assigned the value 1. Judging whether a certain element x belongs to this set, it is also implemented according to the k hash function values in the Bloom filter. If the values of these bits are all 1, then x may belong to this set; otherwise, it can be determined that the element x does not belong to the object set.

The byte-type Bloom filter used by the server node is a modification of the Bloom filter, and its structure is shown in Figure 4.

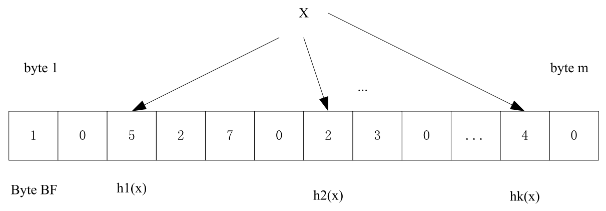

FIGURE 4. The structure of byte bloom filter.

The difference with the Bloom filter is that the data structure used by the byte Bloom filter is a Byte array instead of the original bitmap. When describing the object collection, the bits of this Byte array is also initialized to 0. The update method is as follows: Whenever a block fingerprint is mapped to the corresponding bit, we take the self-increment operation of this bit instead of the original bloom filter. The operation of setting the bit to 1, in this way, the value of the bit in the array indicates to some extent the frequency of the corresponding data block fingerprint, as for judging whether a certain data block fingerprint x exists, as long as the unit's value is mapped to it is non-zero. If all bit units corresponding to x are greater than 0, which means that the data block fingerprint may already exist in the object set; otherwise, it can be determined that the data block fingerprint x does not exist. At this time, the value on the bit is automatically incremented, and then the data block is processed accordingly.

2) FREQUENCY ARRAY

The frequency information of the data block required in this article can be stored by byte-type Bloom filters, and how to use the frequency information needs to use a frequency array to complete. The specific structure of the frequency array is shown in Table 1.

TABLE 1. Frequency array table.

Frequency	Number of digits
0	X0
1	X1
2	X2
…	….
125	X125
126	X126
127	X127

Since the byte bloom filter we use is a Byte array with a length of 100 million, it contains 100 million byte bits, and the maximum value that can be counted by each byte bit without exceeding 127, the maintained frequency array is an array of length 128. In the frequency array, the value of each bit with a non-zero frequency is initialized to 0. The value of the bit with a frequency of 0 is initialized to the length of the byte bloom filter, here is 100 million, whenever the data block fingerprint is maintained if the byte bloom filter of the frequency information is updated, and the frequency array is also updated at the same time. The specific update method is as follows: When a byte bit of the byte bloom filter performs an auto-increment operation, the value of the byte bit changes from count to count+1, and the frequency subscript in the frequency array is the bit of count. The value is subtracted by one, and the value of the digit whose frequency subscript is count+1 is added by one so that the total value of each bit of the entire frequency array remains 100 million.

The main function of the frequency array is to provide a threshold for distinguishing hot data from cold data. When the frequency information corresponding to the characteristic fingerprint information of the data block is greater than the set threshold, the data block corresponding to the frequency information can be determined as hot data. Accordingly, the frequency information corresponding to the characteristic fingerprint information of the data block is smaller than the threshold data. The block is determined to be cold data. The method for determining the threshold is: arrange the frequency array from low to high frequency (the frequency is 0 is not included) and select the top P% of the sum of the values of each bit of the frequency array, where P follows the route. The progress of the process changes dynamically. Since all storage nodes are empty during the routing startup process, stateless routing must be used for each block, and P is approximately 100 at this time. As the routing process deepens, P gradually decreases and finally stabilizes at about 20. When the value of P is determined, the highest frequency included in this part of the range can be used as the threshold, and the frequency range is the non-hot spot data range. The corresponding part higher than this frequency range is the hotspot data range. Since the threshold value will continue to increase with the number of processed data blocks, it satisfies the formula (3-1). A threshold is a threshold, ChunkNum is the number of superblocks that have been processed, the threshold is proportional to the number of routing data blocks, and x is the proportional coefficient. For each data block to be processed, read the frequency information of the bit hit by the byte bloom filter in which is located, and observe which range it is in to determine whether it is hot data.

$$threshold = \frac{ChunkNum}{x} \tag{3-1}$$

3) FEATURE FINGERPRINT SAMPLING

When performing stateful routing of data, if all the block fingerprints of the superblock are sent to all storage nodes for fingerprint search and matching, then the system overhead will be huge, and it will also bring many storage nodes. The disk I/O read and write overhead seriously affects the efficiency of the system and causes the performance of the deduplication system to decrease. For this problem, the current better way to deal with this problem is to sample the data that needs to be routed, select a set of characteristic fingerprints as the routing standard for routing data, and send it to each storage node, and then use the similar matching situation of the characteristic fingerprints to indicate this The hit condition of the superblock determines the target node of the superblock routing.

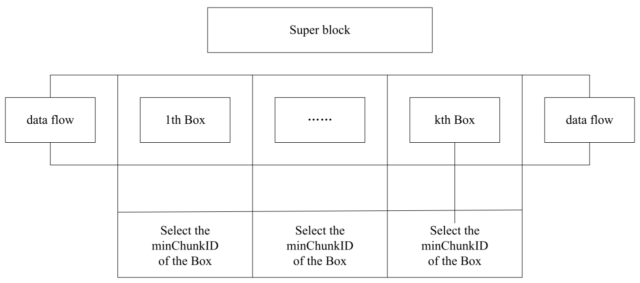
Super block
data flow
1th Box
……
kth Box
data flow
Select the minChunkID of the Box
Select the minChunkID of the Box
Select the minChunkID of the Box

FIGURE 5. Selection of characteristic fingerprint set.

As shown in Figure 5, to select a representative set of characteristic fingerprints, it is necessary to perform fingerprint sampling on the data block fingerprints in the superblock. First, the superblock is further divided into the data stream, and a superblock containing thousands of data blocks is subdivided into multiple continuous area blocks. We call such area blocks Boxes. Each box contains a fixed number of chunks. These chunks also retain the original locality in the superblock. According to Broder's theorem, the smallest block fingerprint in each box is selected as the characteristic fingerprint of the box. The characteristic fingerprints of multiple boxes form a characteristic fingerprint set, and this set is regarded as the characteristic fingerprint set of the superblock. Since each element in the characteristic fingerprint set is the smallest block fingerprint of the Box to which it belongs, the characteristic fingerprint set composed of multiple smallest block fingerprints can represent the entire superblock.

4) DATA ROUTING

The processing flow of the DRDF algorithm is shown in Figures 6. First, after the client collects the user backup data, it divides the data into blocks with a fixed-length block method. It selects thousands of consecutive data blocks to form a superblock, which is further subdivided into multiple boxes. Then the fingerprint is calculated for each data block in the Box, and the characteristic fingerprint is selected for the Box; that is, the smallest one is selected. The smallest block fingerprint set selected from all boxes in the superblock is used as the fingerprint list information of the superblock, and the superblock is sent to the server node at the same time. At this point, the client's data processing work is completed. Then, when the server node processes the received data block, the DRDF algorithm first reads the characteristic fingerprint information in the superblock fingerprint list and selects the smallest value among the multiple characteristic fingerprints as the representative block fingerprint of the entire superblock. The hash function used is mapped to the corresponding byte position of the byte bloom filter to find whether the fingerprint exists in the byte bloom filter and the value of the byte position, that is, the frequency of the fingerprint value.

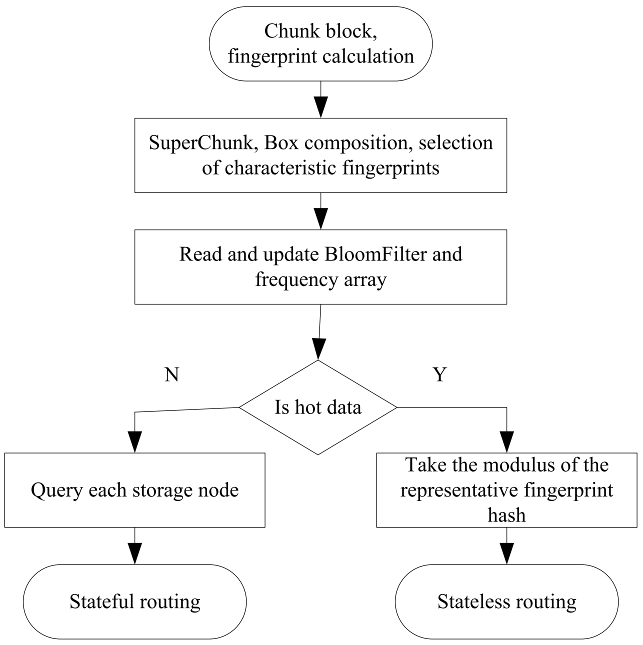
Chunk block, fingerprint calculation
SuperChunk, Box composition, selection of characteristic fingerprints
Read and update BloomFilter and frequency array
N
Y
Is hot data
Query each storage node
Take the modulus of the representative fingerprint hash
Stateful routing
Stateless routing

FIGURE 6. DRDF algorithm processing flow chart.

The byte bloom filter still has the problem of misjudgment because the possibility of each element being misjudged is very slight, as the number of bits of the byte bloom filter increases, this possibility will It is further reduced, and the occurrence of hash collisions is random, so the impact of this misjudgment is ignored in the DRDF scheme. When the Bloom filter uses multiple hash functions, the frequency information of the byte bits mapped to each of the hash functions is different, the following formula (3-2). To determine the frequency value corresponding to the representative block fingerprint. *BF.h(x)* represents the frequency value of the corresponding bit of the Bloom filter mapped to each data block fingerprint x using the hash function *h(x)*. The subscript n represents the use of the nth hash function, n is generally 3 to 5, which can be calculated by the number of bits of the Bloom filter and the control misjudgment rate. *BF.hreal(x)* represents the true frequency of the fingerprint of the data block, that is, the result of selecting multiple hash function mappings. The smallest value among them is regarded as the true frequency value. At the same time, the status information of the Bloom filter and the frequency array must be updated.

$$BF.h_{real}(x)=\min(BF.h_1(x), BF.h_2(x), \ldots BF.h_n(x)) \quad \text{(3-2)}$$

After obtaining the fingerprint value frequency of the representative block of the superblock, which determined whether the data block is hot data according to the hot data judging method in the previous section. For data blocks that have been determined to be cold data, the server node adopts a query-style stateful routing strategy. By sending the characteristic fingerprint set of the superblock to the storage node cluster, the node with the highest similarity to the fingerprint set is obtained. Then the superblock to be routed is sent to the node for deduplication and data storage. For a data block determined to be hot data, the server node performs a hash modulus on its representative fingerprint. The result of the modulus is the node serial number of the storage node. According to the modulus result, the superblock is directly sent to the corresponding. Storage nodes can be deduplicated and stored by the node.

The specific algorithm pseudo-code of the classification routing algorithm based on data frequency is as follows.

Classification Routing Algorithm Based on Data Frequency(DRDF)

```
Input: SuperChunks, NumOfNodes, StorageNodes
Output: aimNode
  1: chunkFrequency=ByteBF.get(SuperChunk);
  2: thresholdFrequency=ByteBF.get(SuperChunks);
  3: if chunkFrequency=thresholdFrequency then
  4: minChunkID=findMininumChunkID(SuperChunks);
  5: aimNode=minChunkID%numofNodes;
  6: else
  7: aimNode=maxHitRatioNode(SuperChunks.FeartrureChunk,
StroageNodes);
  8: end if
  9: return aimNode;
```

The SuperChunks to be processed, the number of storage nodes and the load information of each storage node are the input of the algorithm, and the target node is the output after the algorithm is processed.

IV. EXPERIMENTAL EVALUATION

A. SYSTEM SIMULATION

In the system simulation experiment, several Java classes are written to simulate the entire deduplication system cluster. Specifically, it contains a server master node for distributing data. The implementation class of this node is mainly used to execute data routing algorithms. The program implements the DRDF algorithm and the Stateful Stateless algorithm based on the smallest block in the EMC solution. The data routing algorithm processes the input data stream and routes each data block to the corresponding storage node. The other part of the system simulation experiment contains multiple storage nodes. Each Java class simulates a node. Each Java class implements the *doQuery()* and *doDedup()* methods, which are responsible for querying the stored data fingerprints and routing the node. The data to this node is deduplicated. During node deduplication, the existing duplicate data blocks are deleted, and only the data that appears for the first time is stored. There is also a program about data block preprocessing in the simulation experiment. This program is mainly responsible for the preprocessing of the input data stream, such as the fixed-length block of the file, etc. The size of the divided data block is 4KB, and the data is divided into 1000 pieces. Blocks are assembled into a superblock as a unit, and a box of the superblock is composed of 100 consecutive data blocks. At the same time, the data block preprocessing program calculates the fingerprint value for each data block and selects the smallest block in this part of the data block as the representative block of the superblock, and selects the smallest block fingerprint of each Box to form the characteristic fingerprint set of the superblock. And then hand over the superblock to the server master node for processing.

B. DATA SET AND EVALUATION INDEX

During the experiment, to evaluate the effect of the classification routing algorithm based on data frequency, the experiment used the source code documents of different Linux kernel versions as the test data set. The specific content is shown in Table 2.

TABLE 2. Data set.

data set	Number of files	Data set size
Linux source code	3186361	35.678GB

The experiment uses the following evaluation indicators:

(1) Deduplication rate: When the system completes the deduplication operation, the ratio of the deleted part to the total part before being deleted. The calculation formula (4-1) is as follows.

$$Rdr = \frac{Rd - Ad}{Rd} \tag{4-1}$$

where:

Rdr—deduplication rate;

Rd—the total amount of raw data;

Ad—the total amount of data remaining after deduplication.

(2) The running logic time of the algorithm: the time required from the start of the routing algorithm program of the server node to the completion of data block fingerprint allocation and duplicate fingerprint deletion.

(3) System routing query communication overhead: the number of times the data block fingerprints are queried from the start to the end of the server node routing algorithm program. Here, the number of data block fingerprints used for the query is used as the evaluation index.

C. EXPERIMENTAL RESULTS

The experiment first uses a single-node deduplication system to perform global deduplication, and the repetition rate of the data set used is about 88.39%. Then tested the deduplication rate of deduplication clusters in the environment with a different number of nodes without performing feature fingerprint sampling on the superblock and compared the classification routing algorithm based on data frequency (DRDF), the stateful routing of EMC scheme, and Stateless routing, and other methods use the deduplication rate when the number of nodes in the deduplication cluster is 1, 3, 7, 15, 31, 63, and 127. Choosing these nodes is because they are multiples of 2 minuses 1, which is good for hashing the routing data.

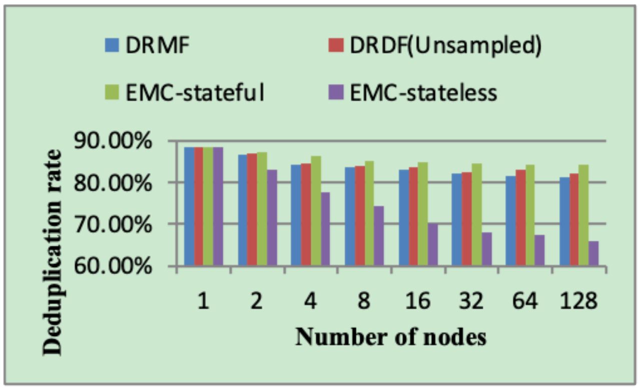

FIGURE 7. Comparison of algorithm deduplication rate.

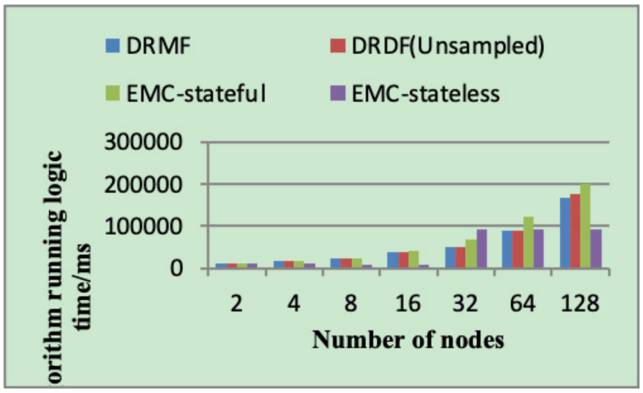

FIGURE 8. Comparison of the algorithm running logic time.

Figure 7 shows the deduplication rate between the stateful EMC-stateful, stateless EMC-stateless, and DRMF [19] algorithms based on the EMC scheme and the data frequency-based classification routing algorithm DRDF.

Because single-node deduplication can find all duplicate data blocks, other solutions cannot do this and can only detect duplicate data in each storage node. For this reason, there will be duplicate data between nodes, so the deduplication rate of the DRDF and EMC scheme algorithms will be lower than the actual duplication rate of the data set, that is, lower than the deduplication rate of the single-node deduplication system.

The data in Figure 7 shows that the deduplication rate of the EMC scheme's stateful routing algorithm has the highest deduplication rate compared to other distributed deduplication systems in different cluster sizes. With the gradual increase in nodes, the deduplication rate of DRDF and DRMF algorithm and EMC-stateful is always close and remains above 80%. EMC-stateful has the highest deduplication rate, DRDF; compared with the DRMF algorithm, the deduplication rate of the former is slightly lower, but the difference does not exceed 0.6%. To control the load and storage status of some overloaded nodes during data routing; each superblock cannot be sent to the storage node that maximizes the deduplication effect, resulting in a decrease in the deduplication rate. Therefore, the classified routing algorithm based on data frequency can better route similar data to the same node and can almost achieve the effect of a stateful routing algorithm.

Figure 8 shows the logic time comparison of the DRDF and DRMF [19] and EMC stateful and stateless routing algorithms for processing about 11 million fingerprint data. EMC-stateful has to query storage nodes every time routing data, so it needs more time overhead. With the increase of the cluster size, the number of nodes to be queried is increasing, and the logic time of the algorithm running is basically about the nodes. The logarithm of the number increases exponentially. EMC-stateless uses stateless routing, so the time required is the least. Stateless routing does not incur query costs as the number of nodes increases; the running time of the entire algorithm is maintained at a stable level. The DRDF algorithm uses stateful routing for most non-hotspot data and only uses stateless routing for some of the hotspot data. Hence, the required system running time is between the two, which is closer but slightly lower than the stateful routing algorithm.

With the increase of the number of nodes, when the DRMF routing algorithm is used, the load data tilt rate of the system is getting larger and larger, which means that some heavy-loaded nodes need to process more and more data, It is easy to cause the node or even the entire The data of the congested system. When using the DRDF algorithm, the load data tilt rate of the system also increases with the number of nodes, but it is greatly improved compared to the DRMF algorithm, which reduces the load pressure on the heavily loaded nodes as shown in Figure 8.

The experiment further tested the deduplication rate of the deduplication cluster in the environment with a different number of nodes after using the DRDF algorithm to sample the characteristic fingerprint of the superblock, and the comparison with that before sampling is shown in Figure 9.

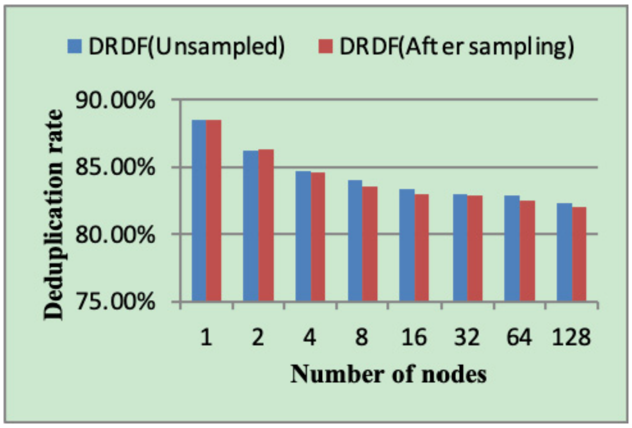

FIGURE 9. Comparison of deduplication rate before and after sampling by DRDF algorithm.

It can be seen from the figure that when the number of nodes is small, the deduplication rate of the DRDF algorithm before and after sampling is similar. As the number of nodes increases, the DRDF algorithm only performs similarity matching on the feature fingerprints after sampling.

Unable to obtain the optimal matching node of the entire superblock, resulting in a slight drop in the system's deduplication rate, maintained at a level within 0.5%.

Similarly, the experiment also tested the logic time of the algorithm in the environment with a different number of nodes after using the DRDF algorithm to sample the characteristic fingerprint of the superblock. The comparison with that before the sampling is shown in Figure 10.

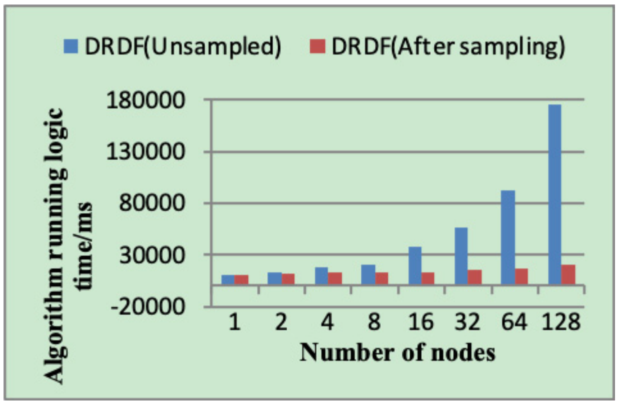

FIGURE 10. Comparison of running logic time before and after DRDF algorithm sampling.

Figure 10 shows that the running logic time of the DRDF algorithm after sampling has been greatly reduced. When the number of nodes is small, the fingerprint query time overhead of the system itself is very small. Due to the logic running time of the overall system algorithm Less, the reduced fingerprint query time overhead as sampling is not obvious. With the increase of the number of nodes, the logic time of the algorithm before and after sampling of the DRDF algorithm shows a linear increase. However, the gap between the required running time before and after sampling is getting bigger and bigger.

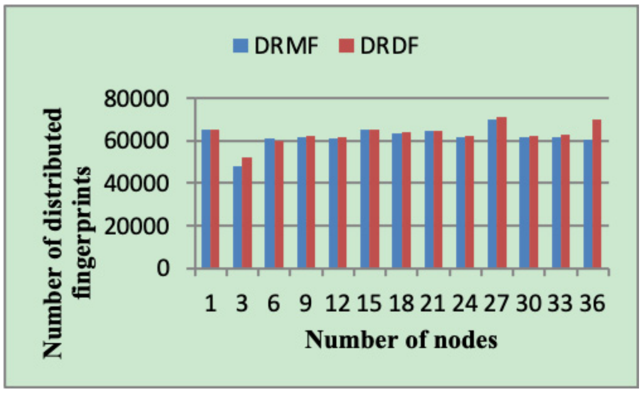

FIGURE 11. Load distribution after 31 node deduplication.

The experiment also tested the data distribution of the cluster under different node scales. From Figure 11, we can see that the 31-node cluster deduplication system uses the DRMF algorithm and the DRDF algorithm to deduplicate the fingerprints of each storage node, that is, the load distribution. It can be seen from the figure that when we use the DRMF algorithm when the deduplication process is completed, nodes 17 and 19 are respectively distributed with the maximum load and minimum load data; and the difference is nearly doubled. After using the DRDF algorithm, this difference has been reduced. It can be seen that the DRDF algorithm can make the load distribution more even than the DRMF algorithm, and the standard deviation of a load of each node of the entire cluster deduplication system is smaller, indicating that the algorithm reduces the excessive load or excessive load after adopting the load balancing guarantee strategy. The generation of lightly loaded nodes.

Since the running time of the algorithm is not only related to the time it takes to query the storage node or the memory index lookup of each storage node, but also other factors such as the response of the storage node itself or some of the logic of the algorithm itself, so the algorithm running logic time is used It is not accurate enough to judge the cost between different routing algorithms. Therefore, we use the system's routing query communication overhead as another indicator to evaluate the system overhead. The specific performance is how many data block fingerprints have been sent to the node query operation before routing during the routing process. The comparison is shown in Table 3.

TABLE 3. Comparison of routing query communication overhead of different algorithms under num nodes.

Routing scheme	Query the number of fingerprints of the data block (data block * number of nodes)
Stateless routing	0
Stateful routing	11000*1000*num
DRDF(Unsampled)	8234*1000*num
DRDF(After sampling)	8234*10*num

In the same data set, using stateless routing does not require querying storage nodes but directly routing to the corresponding node based on the feature of fingerprint hash, so the solution does not require query overhead. Stateful routing uses a query method for each superblock. The number of data block fingerprints queried is the product of the number of superblocks, the number of data blocks in each superblock, and the number of nodes. In contrast, the DRDF scheme only queries data blocks with a low frequency of data concentration. It uses a method of routing data blocks with a high frequency to the storage node according to the superblock feature fingerprint hash, so the whole process is only Among them, 8,234 queries of storage nodes were performed. The cost was reduced by about 25% compared with stateful routing. After sampling the characteristic fingerprints, the DRDF algorithm only performs fingerprint query operations on the ten characteristic fingerprints in each superblock, so the overhead is only 1% of the pre-sampling cost.

Experimental results show that as the node size of the deduplication cluster increases from 1 to 127, the DRDF algorithm runs logic when the system's deduplication rate is only less than 2% lower than EMC's stateful routing algorithm. The time cost is slightly lower than that of the stateful routing algorithm. Only less than 75% of the number of superblocks in the complete stateful routing scheme is used in the query routing of the superblock, which also improves the throughput rate to a certain extent. Thereby improving the real-time processing capability of the deduplication system. Achieved the goal of reducing system overhead without reducing the deduplication rate too much.

V. CONCLUSION

This paper proposes a new classification routing algorithm based on data frequency. This algorithm greatly reduces system query and communication overhead at the expense of a low deduplication rate and improves system throughput and real-time processing capabilities. The experimental results show that the DRDF algorithm based on data frequency for routing compared to the stateful routing algorithm proposed by the EMC scheme has a difference in the deduplication rate within two percentages. The algorithm logic time of the system is slightly reduced. The corresponding query communication overhead is also reduced by more than 25%, which improves the real-time processing capability of the storage system. As the cluster size increases, the deduplication rate of the DRDF algorithm also tends to stabilize, indicating that the algorithm has certain scalability. Therefore, the DRDF algorithm is an effective data routing algorithm in a data deduplication cluster.

In the improved classification routing scheme in this article, a certain load balance is ensured by querying the load status information of the storage nodes. However, the load balancing here mainly refers to controlling the actual storage of data by each storage node to be equivalent, that is, the actual physical load. Rather than instantaneous processing traffic load. In further research, the real-time traffic of each node can be counted, and the storage node with larger traffic can be diverted to the storage node with smaller traffic, thereby improving the performance of the system.

REFERENCES

[1] J. Gantz and D. Reinsel, *Extracting Value From Chaos, An IDC White Paper Sponsored by EMC*. Framingham, MA, USA: IDC, 2019.

[2] P. Strzelczak, E. Adamczyk, U. Herman-Izycka, J. Sakowicz, L. Slusarczyk, J. Wrona, and C. Dubnicki, "Concurrent deletion in a distributed content-addressable storage system with global deduplication," in *Proc. 11th USENIX File Storate Technol.*, Feb. 2013, pp. 161–174.

[3] *Intel Intelligent Storage Acceleration Library Crypto*. [Online]. Available: https://github.com/intel/isa-l_crypto

[4] Y. Allu, F. Douglis, M. Kamat, R. Prabhakar, P. Shilane, and R. Ugale, "Can't we all get along? Redesigning protection storage for modern workloads," in *Proc. USENIX Conf. USENIX Annu. Tech. Conf. (ATC)*, Boston, MA, USA, Jul. 2018, pp. 705–718.

[5] Q He, B. Shao, and W. Zhang, "Data deduplication technology for cloud storage," *Tehnički Vjesnik*, vol. 27, no. 5, pp. 1445–1446, 2020.

[6] N. Zhao, H. Albahar, S. Abraham, K. Chen, V. Tarasov, D. Skourtis, L. Rupprecht, A. Anwar, and A. R. Butt, "Duphunter: Flexible high-performance deduplication for Docker registries," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, Jul. 2020, pp. 769–783.

[7] C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W. Kilian, P. Strzelczak, J. Szczepkowski, C. Ungureanu, and M. Welnicki, "HYDRAstor: A scalable secondary storage," in *Proc. FAST*, vol. 9, 2009, pp. 197–210.

[8] 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. Model., Anal. Simul. Comput. Telecommun. Syst.*, Sep. 2009, pp. 1–9.

[9] T. Yang, H. Jiang, D. Feng, Z. Niu, K. Zhou, and Y. Wan, "DEBAR: A scalable high-performance de-duplication storage system for backup and archiving," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. (IPDPS)*, Apr. 2010, pp. 1–12.

[10] C. Ungureanu, B. Atkin, A. Aranya, S. Gokhale, S. Rago, C. Dubnicki, and A. Bohra, "HydraFS: A high-throughput file system for the HYDRAstor content-addressable storage system," in *Proc. FAST*, vol. 10, 2010, pp. 225–239.

[11] A. Z. Broder, "On the resemblance and containment of documents," in *Proc. Compress. Complex. SEQUENCES*, Jun. 1997, pp. 21–29.

[12] B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file system," in *Proc. 6th USENIX Conf. File Storage Technol. (FAST)*, Berkeley, CA, USA, 2008, pp. 269–282.

[13] 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.

[14] (2011). *IBM ProtecTIER Deduplication Gateway*. [Online]. Available: https://www03.ibm.com/systems/storage/ tape/ts7650g/index.html

[15] W. Dong, F. Douglis, K. Li, R. H. Patterson, S. Reddy, and P. Shilane, "Tradeoffs in scalable data routing for deduplication clusters," in *Proc. FAST*, vol. 11, 2011, pp. 15–29.

[16] R. J. Honicky and E. L. Miller, "Replication under scalable hashing: A family of algorithms for scalable decentralized data distribution," in *Proc. 18th Int. Parallel Distrib. Process. Symp.*, Apr. 2004, p. 96.

[17] B. K. Debnath, S. Sengupta, and J. Li, "ChunkStash: Speeding up inline storage deduplication using flash memory," in *Proc. USENIX Annu. Tech. Conf.*, 2010, pp. 1–16.

[18] Y. J. Nam, D. Park, and D. H. C. Du, "Assuring demanded read performance of data deduplication storage with backup datasets," in *Proc. IEEE 20th Int. Symp. Model., Anal. Simul. Comput. Telecommun. Syst.*, Aug. 2012, pp. 201–208.

[19] Q. He, G. Bian, B. Shao, and W. Zhang, "Research on multifeature data routing strategy in deduplication," *Sci. Program.*, vol. 2020, pp. 1–11, Oct. 2020.

[20] F. Douglis, A. Duggal, P. Shilane, T. Wong, S. Yan, and F. Botelho, "The logic of physical garbage collection in deduplicating storage," in *Proc. 15th Usenix Conf. File Storage Technol. (FAST)*, Santa Clara, CA, USA, Feb. 2017, pp. 29–44.

[21] Z. Sun, N. Xiao, F. Liu, and Y. Fu, "DS-dedupe: A scalable, low network overhead data routing algorithm for inline cluster deduplication system," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2014, pp. 895–899.

[22] Z. Yan, H. Jiang, Y. Tan, S. Skelton, and H. Luo, "Z-dedup: A case for deduplicating compressed contents in cloud," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2019, pp. 386–395.

[23] X. Lin, F Douglis, J. Li, X. Li, R. Ricci, S. Smaldone, and G. Wallace, "Metadata considered harmful to deduplication," in *Proc. 7th USENIX Workshop Hot Topics Storage File Syst.*, 2015, pp. 1–5.

[24] D. Meister, J. Kaiser, A. Brinkmann, T. Cortes, M. Kuhn, and J. Kunkel, "A study on data deduplication in HPC storage systems," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2012, pp. 1–11.

[25] C. Policroniades and I. Pratt, "Alternatives for detecting redundancy in storage systems data," in *Proc. USENIX Conf. USENIX Annu. Tech. Conf.*, 2004, pp. 73–86.

[26] D. Meyer and W. Bolosky, "A study of practical deduplication," in *Proc. 9th USENIX Conf. File Storage Technol.*, vol. 2011, pp. 229–241.

[27] N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch, "A fiveyear study of file-system metadata," in *Proc. 5th USENIX Conf. File Storage Technol.*, vol. 2007, pp. 31–45.

[28] (2014). *Linux Archives*. [Online]. Available: https://www.kernel.org

[29] V. Tarasov, A. Mudrankit, W. Buik, P. Shilane, G. Kuenning, and E. Zadok, "Generating realistic datasets for deduplication analysis," in *Proc. Conf. USENIX Annu. Tech. Conf.*, 2012, pp. 24–34.

[30] (2016). *Destor: An Experimental Platform for Chunk-Level Data Deduplication*. [Online]. Available: https://github.com/fomy/destor

[31] 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. 13th USENIX Conf. File Storage Technol.*, 2015, pp. 331–345.

[32] M. O. Rabin, "Fingerprinting by random polynomials," Center Res. Comput. Techn., Aiken Comput. Lab., Univ., 1981.

[33] A. Muthitacharoen, B. Chen, and D. Mazières, "A low-bandwidth network file system," in *Proc. 18th ACM Symp. Operating Syst. Princ.*, Oct. 2001, pp. 1–14.
[34] H. Cui, H. Duan, Z. Qin, C. Wang, and Y. Zhou, "SPEED: Accelerating enclave applications via secure deduplication," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019.
[35] H. Dang and E.-C. Chang, "Privacy-preserving data deduplication on trusted processors," in *Proc. IEEE 10th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2017, pp. 66–73.
[36] J. B. Djoko, J. Lange, and A. J. Lee, "NeXUS: Practical and secure access control on untrusted storage platforms using client-side SGX," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2019, pp. 401–413.
[37] V. Eduardo, L. C. E. de Bona, and W. M. N. Zola, "Speculative encryption on GPU applied to cryptographic file systems," in *Proc. USENIX Conf. File Storage Technol. (FAST)*, 2019, pp. 93–105.
[38] S. Eskandarian and M. Zaharia, "ObliDB: Oblivious query processing for secure databases," in *Proc. ACM VLDB*, 2019, pp. 1–24.
[39] B. Fuhry, L. Hirschoff, S. Koesnadi, and F. Kerschbaum, "SeGShare: Secure group file sharing in the cloud using enclaves," in *Proc. 50th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2020, pp. 476–488.
[40] D. Harnik, E. Tsfadia, D. Chen, and R. Kat, "Securing the storage data path with SGX enclaves," 2018, *arXiv:1806.10883*. [Online]. Available: https://arxiv.org/abs/1806.10883
[41] T. Kim, J. Park, J. Woo, S. Jeon, and J. Huh, "ShieldStore: Shielded in-memory key-value storage with SGX," in *Proc. 14th EuroSys Conf.*, Mar. 2019, pp. 1–15.
[42] R. Krahn, B. Trach, A. Vahldiek-Oberwagner, T. Knauth, P. Bhatotia, and C. Fetzer, "Pesos: Policy enhanced secure object store," in *Proc. 13th EuroSys Conf.*, Apr. 2018, pp. 1–17.
[43] J. Li, P. P. C. Lee, C. Tan, C. Qin, and X. Zhang, "Information leakage in encrypted deduplication via frequency analysis: Attacks and defenses," *ACM Trans. Storage*, vol. 16, no. 1, pp. 1–30, Apr. 2020.
[44] J. Li, Z. Yang, Y. Ren, P. Lee, and X. Zhang, "Balancing storage efficiency and data confidentiality with tunable encrypted deduplication," in *Proc. ACM Eurosys*, 2020, pp. 1–15.
[45] S. Mofrad, F. Zhang, S. Lu, and W. Shi, "A comparison study of Intel SGX and AMD memory encryption technology," in *Proc. 7th Int. Workshop Hardw. Architectural Support Secur. Privacy*, Jun. 2018, pp. 1–8.
[46] O. Oleksenko, B. Trach, R. Krahn, A. Martin, C. Fetzer, and M. Silberstein, "Varys: Protecting SGX enclaves from practical side-channel attacks," in *Proc. USENIX ATC*, 2018, pp. 227–240.
[47] C. Priebe, K. Vaswani, and M. Costa, "EnclaveDB: A secure database using SGX," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 264–278.
[48] Y. Ren, J. Li, Z. Yang, P. P. C. Lee, and X. Zhang. (2021). *Accelerating Encrypted Deduplication Via SGX*. CUHK. [Online]. Available: http://www.cse.cuhk.edu.hk/ pclee/www/pubs/tech_sgxdedup.pdf
[49] W. You and B. Chen, "Proofs of ownership on encrypted cloud datavia Intel SGX," in *Proc. ACNS*, 2020, pp. 400–416.
[50] Y. Zhang, W. Xia, D. Feng, H. Jiang, Y. Hua, and Q. Wang, "Finesse: Fine-grained feature locality based fast resemblance detection for postdeduplication delta compression," in *Proc. 17th USENIX Conf. File Storage Technol. (FAST)*, Santa Clara, CA, USA, Feb. 2019, pp. 121–128.

QINLU HE (Member, IEEE) received the Ph.D. degree in computer science from Northwestern Polytechnical University. He is currently an Associate Professor with Xi'an University of Architecture and Technology. He has more than 20 publications in journals and international conferences. His current research interests include data deduplication, cloud storage, and distributed file systems. He is a member of China Computer Federation.

GENQING BIAN (Member, IEEE) is currently pursuing the Ph.D. degree with the School of Management, Xi'an University of Architecture and Technology (XAUAT), Shaanxi, China. He is currently an Associate Professor with XAUAT. His research interests include information security, cloud computing security, massive information storage technology, and VANETS security. He is a member of China Computer Federation and the Association for Computing Machinery.

WEIQI ZHANG received the master's degree from Xi'an University of Architecture and Technology (XAUAT). He is currently an Associate Professor with XAUAT. His current research interests include computer storage and cloud computing. He has more than 30 publications in journals and conferences in these areas. He is a member of China Computer Federation.

FAN ZHANG received the B.S. degree from Northwestern Polytechnical University, in 2001, and the Ph.D. degree from the Department of Computing, University of Surrey, U.K., in 2008. She is currently with the Department of Computer Sciences, Xi'an University of Architecture and Technology, Xi'an. Her current research interests include machine learning, data mining, and brain–computer interface.

SHENGQIANG DUAN was born in 1981. He is currently pursuing the Ph.D. degree with XAUAT. His research interests include data cleaning, data repairing, and education informatization.

FENGLANG WU was born in Xi'an, China. He is currently an Engineer with the Network Information Department, The First Affiliated Hospital of Xi'an Jiaotong University, Xi'an. His research interests include hospital informatization construction, artificial intelligence, and medical data analysis.