IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 5, 2021, accepted December 27, 2021, date of publication December 30, 2021,
date of current version January 27, 2022.

Digital Object Identifier 10.1109/ACCESS.2021.3139757

Research on Data Routing Strategy of Deduplication in Cloud Environment

QINLU HE[1], (Member, IEEE), FAN ZHANG[1], GENQING BIAN[1], (Member, IEEE), WEIQI ZHANG[1], DONGLI DUAN[1], ZHEN LI[2], AND CHEN CHEN[3]

[1]School of Information and Control Engineering, Xi'an University of Architecture and Technology, Xi'an 710054, China
[2]Shaanxi Institute of Metrology Science, Xi'an 710043, China
[3]Network Information Department, The First Affiliated Hospital of Xi'an Jiaotong University, Xi'an 710048, China

Corresponding authors: Qinlu He (luluhe8848@hotmail.com) and Zhen Li (billywin1982@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872284 and Grant 72071153; 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 Shannxi Province, China, under Grant 2014JM2-6127; in part by the Talent Fund Project of the Xi'an University of Architecture and Technology under Grant RC1707; in part by the Youth Fund Project of the 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 The application of data deduplication technology reduces the demand for data storage and improves resource utilization. Compared with limited storage capacity and computing capacity of a single node, cluster data deduplication technology has great advantages. However, the cluster data duplication technology also brings new issues on deduplication rate reduction and load balancing of storage nodes. The application of data routing strategy can well balance the problem of deduplication rate and load balancing. Therefore, this paper proposes a data routing strategy based on distributed Bloom Filter. 1)Superchunk is used as the basic unit of data routing to improve system throughput. According to Broder's theorem, k leastsized fingerprints are selected as the Superchunk features and send to the storage node. The optimal node is selected as the routing node by matching the BloomFilter, and the storage capacity of the node and maintained in the memory of the storage node. 2) Design and implement system prototypes. The specific parameters of all kinds of routing strategies are obtained through experiments, and the routing strategies proposed in this paper are tested. The theoretical analysis and experimental results prove the feasibility of the strategies proposed by this paper. Compared with the other routing strategies, our method improved 3% of the deduplication rate, reduces the communication query overhead by more than 36% and improves the load balancing degree of the storage system.

INDEX TERMS Cloud, deduplication, data routing, load balancing.

I. INTRODUCTION

Data is more important today than ever for enterprises and individuals. With exponential growth of data, it is difficult to manage a huge lump of data. Normally the volume of data reaches the PB level even EB- level in the enterprise data manage center, which increase the cost of data management. Improving the storage efficiency on data backup has become a research hotspot. Almost 75% data in the data world are duplicated based on survey [1], and duplicated data reaches approximately 90% in the backup data and file system [2]. The development of deduplication technology in recent years has provided an effective solution for duplicated data, and deduplication is a data compression strategy applied to storage systems with high data compression rates [51]–[56].

Limited to the computational and storage capacity of a single node, Cloud Deduplication (CD) makes use of the powerful storage capacity and parallelism of clustered systems so that deduplication can be used in large-scale distributed storage systems [57]–[60]. Cloud deduplication is benefit for data routing strategy to send data to each storage node and implement data deduplicate at each storage node, or the data is deduplicated before it is sent to the storage node. Both approaches have their pros and cons, but to ensure the parallelism of the system and low system overhead, the current cluster deduplication system mainly uses the former for

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

data routing [3]–[5]. Implement the routing strategy requires maintaining fingerprint indexes of data blocks in the memory of storage nodes, which the size of data block is 4KB and the fingerprint size of each data block is about 40B. The ratio of the size of the block fingerprint to the size of the data volume is about 1:100 [6]. Considering the memory space limitation of storage nodes and the cost limitation, such a huge amount of fingerprint indexes cannot be stored in the memory of storage nodes, so most of them are stored on disk, and the needed fingerprint indexes are read into the memory only when deduplication is performed. However, too many disk accesses will cause disk bottleneck problems and lead to lower system throughput. At the same time, in the cluster deduplication system is often only the data stored in the node itself for deduplication, which will lead to a reduction in the overall deduplication rate of the entire system, and also lead to a large number of duplicate data routed to a few nodes resulting in load imbalance. Hence, the main problems facing cloud deduplication are query performance, deduplication rate, and load balancing.

With the development of cloud computing, cloud computing services not only unify the deployment and management of data resources but also optimize the utilization of resources. In contrast, cloud storage is a cloud computing service with data storage and management as the core. Applying deduplication technology to cloud storage systems and using the computing and storage capabilities of nodes in cloud storage can improve the performance of the entire storage system. However, managing a large number of storage nodes in the cloud storage system and ensuring node load balancing while avoiding a significant drop in deduplication rate are two key issues that need to be addressed. The main role of the data routing policy is to send the data from the client to the storage nodes. It needs to send as many duplicate data to the same node as possible. However, after a certain time, it will cause the state that some nodes store a large amount of data of the system while other nodes are idle, so how to balance these two problems is the key point to consider in the data routing policy. The routing strategy proposed in this paper, using the data similarity principle and relevant information such as storage node capacity, achieves a balance between the redundancy rate and load balancing, avoiding a significant drop in the redundancy rate while achieving a balanced distribution of data in the storage nodes.

II. RELATED WORK

Cloud deduplication systems are widely used in data centers nowadays. To achieve a cluster deduplication system with high throughput and high storage space, an effective and reasonable routing algorithm is essential, and routing data to the corresponding nodes in the cluster at a more reasonable data granularity is also an effective measure to improve the system performance. Cluster deduplication system has the advantage on improving storage space utilization and decreasing the reduplication rate, so the overall deduplication rate would decrease with the number of nodes in the cluster increases. An effective solution is to send duplicate data to the same node as much as possible to ensure the deduplication rate of the system. Another important issue in clustered systems is that sending a large amount of duplicate data to the same node can cause load imbalance among nodes, which seriously affects the system throughput rate and further limits the system performance. Therefore, the routing algorithm need to ensure the load balance of the system while keeping the deduplication rate.

The main consideration in cluster deduplication is how to send data to the storage nodes. NEC's HYDRAstor system uses a 64KB data block granularity [8] and uses distributed hash tables (DHTs) to determine the routing nodes for the data. Although this strategy is effective in reducing the communication overhead, but taking use of larger data block granularity reduces the data deduplication rate, and it need to update the hash table when the data changes in the cluster. The hash table stores the fingerprints of data blocks, and each block fingerprint occupies only a small amount of storage space. However, the hash table becomes very large when the stored data is huge. It is difficult for the node memory to maintain such a huge hash table, which needs to be transferred to disk storage, and bring about the disk bottleneck problem.

EMCstateful and EMCstateless are stateful and stateless routing policies proposed by EMC, respectively [9]. To ensure the performance of the deduplication system, both policies use superchunk as basic routing unit. Superchunk is a colleetion of data chunks, which its size is from 1M-16M and vary depending on the system. The stateless routing strategy selects the representative fingerprint ID of the Superchunk to select the routing node, usually using a hash algorithm to get the corresponding routing node, and a hash table is used within the node to maintain the data information inside the node. This strategy has a good load balancing effect in small-scale clusters, but the shortcoming is low deduplication rate and poor scalability, and it is difficult to guarantee the load balancing effect when the cluster is too large. Stateful routing is mainly used in large-scale clusters, where the fingerprint of the superblock needs to be sent to all storage nodes before sending the superblock to the storage nodes, which maintain the BloomFilter of the data block of the current node. Then the optimal storage node is calculated by querying the number of hits of the fingerprint in the superblock and the load of the storage nodes. The objective is to route as many duplicate data as possible to the same node while ensuring load balancing. This strategy ensures a good deduplication rate in a clustered environment. However, the system throughput overhead caused by the broadcast data query method and the BloomFilter with many fingerprints maintained within the query node greatly affect the system performance.

Extreme Binning is a stateless data routing strategy based on file similarity [5]. This policy is based on Broder's theorem, which selects the smallest block fingerprint of a file as the representative fingerprint ID of the whole file. If the smallest block fingerprints of two files are the same, then the two files have high similarity. In the same way, a two-level

structure is used for data maintenance inside the storage node. The primary index in the memory of the storage node stores the representative fingerprint ID of the file, and one bin (box) is used to store the data in the disk. Each time deduplication is performed, the corresponding bins are read into the memory for deduplication to avoid bottleneck problem. This strategy has a high deduplication rate, but the stateless-like routing approach and the uneven file size lead to the same load imbalance problem as the stateless routing strategy.

Boafft is a routing policy that uses Broder's theorem and partial sampling of the Superchunk to select the representative ID of the Superchunk [10]. This strategy divides the Superchunk into several same size regions, selecting the smallest block ID in each region, and then sending these representative IDs to all storage nodes, maintaining the metadata information of the data blocks in the memory of the storage nodes. and also uses the structure of container in the disk to save the data blocks, reducing the disk I/O by this method. While using Jaccard distance [11], the similarity of each node is calculated, and then the optimal routing node is obtained by the corresponding calculation. This strategy selects the Superchunk representative ID by sampling the Superchunk, which greatly improves the similarity determination of the Superchunk. In addition, this strategy also utilizes the broadcast type of query, which causes excessive system communication overhead.

$\sum$-Dedupe is a stateful routing policy based on Superchunks [12], which is different from EMC's stateful routing policy in that in the memory of the storage node, this policy uses a similarity index table to keep the representative fingerprint IDs of all Superchunks stored in that node, similar to the primary index of Extreme Binning, where each representative fingerprint ID corresponds to an Instead of sending fingerprints to all storage nodes to select the best node as the routing node, the strategy selects the k smallest block fingerprints of the superblock, uses these k fingerprints to model the number of nodes to select k alternative storage nodes, then sends k fingerprints to all alternative nodes, calculates the similarity using Jaccard distance, and then uses the capacity of the storage nodes The matching values of each node are calculated to select the optimal node. This strategy achieves global load balancing by implementing local load balancing by selecting a small number of representative IDs to send to a small number of storage nodes, which reduces the communication overhead within the system.

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. Big data has brought new challenges to the existing storage systems on capacity, maintainability, throughput performance, scalability, and reliability. For the great demand on the storage space for big data, it is necessary to eliminate redundancy on which saving data and optimizing 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 data size, it has relatively strict space limitations. Using data compression for two files with the same content will still get two identical compressed encoded files, and then complete the reduction of stored data. 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 very important for optimized storage of big data.

III. DISTRIBUTED BLOOMFILTER-BASED DATA ROUTING STRATEGY

This section designs a distributed BloomFilter-based data routing strategy that enables fast data routing queries using BloomFitler and a few superblock representative fingerprint IDs. Unlike EMC's stateful routing strategy, this routing strategy uses BloomFilter to maintain the superblock representative fingerprint IDs in the memory of the storage node without all the fingerprint indexes stored in that storage node. Only a few superblock representative fingerprint IDs are sent. A few routing nodes are queried during the query process to achieve global load balancing using local load balancing.

A. SIMILARITY OF SUPER BLOCK

In cluster data deduplication, to ensure system throughput, normally the routing units is a super block. Before sending data, the destination node of these superblocks needs to be determined. It is necessary to determine which storage node stores the most duplicate data in the super block. To reduce the communication overhead of the system, it is impossible to send the data block to all storage nodes for one-by-one comparison. Methods such as MD5 or SHA-1 are often used to calculate the fingerprint information of the data block, send the data fingerprint information to the storage node, and then use the relevant data. The similarity algorithm determines the similarity of the data. This paper uses BloomFilter to judge the similarity of superblock data. It uses BloomFilter to query the advantages of high efficiency and low storage

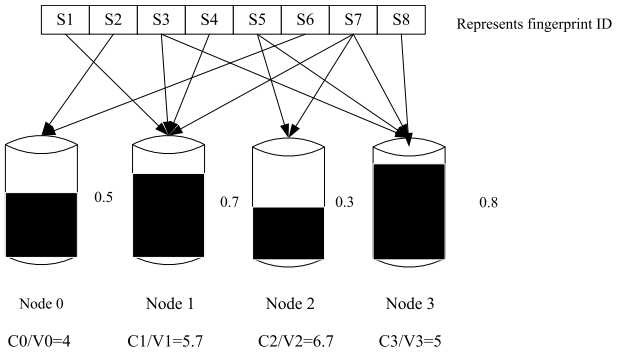

FIGURE 1. Similarity matching diagram.

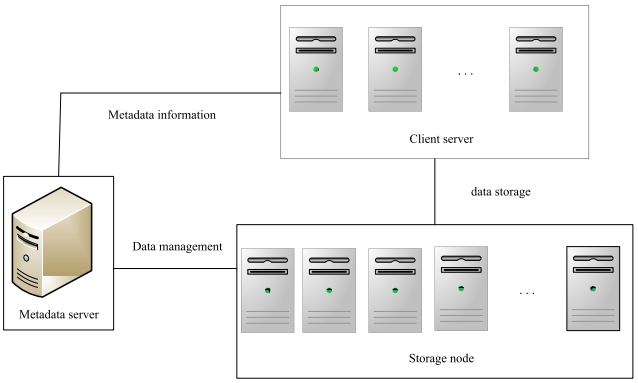

FIGURE 2. Cloud deduplication system architecture diagram.

occupancy while avoiding the influence of misjudgment rate on data judgment. In addition, to ensure load balance in a cluster storage system, it is not only necessary to send as much duplicate data as possible to the same storage node, but also to combine the storage load of the storage node and comprehensively select the storage node as the destination node for data routing. As shown in Figure 1, suppose that the number of k representative fingerprint IDs is 8, and the number of storage nodes in the cluster deduplication system is 4. As shown in the figure, for Node0, 2 of the 8 representative fingerprint IDs have BloomFilter in the storage node, and the storage capacity of the storage node is 0.5. For Node1, 4 fingerprints already exist, 2 in Node2, and 4 in Node3. Intuitively speaking, the superblock should be sent to Node1 or Node3, but from the figure 1, It shows that a large amount of data has been stored in these two nodes. Continue sending data to the two storage nodes will lead to no more storage capacity to receive incoming data on those two storage nodes for a short while. Therefore, the method of selecting nodes in this article uses Ci/Vi (Ci represents the number of data block fingerprints that have been stored in BloomFilter in storage node i, and Vi represents the data block capacity of the storage node has been stored) to calculate the optimal node for data routing. It can be seen from the figure that the value of the calculation result of the Node2 node is the largest, so Node2 is selected as the final node of the data routing.

Node2 node only has two data block fingerprints matching successfully, and sending the super block to this node is not the optimal choice in terms of deduplication rate. In actual application, the system starts from the data routing of the first super block and considers the optimal storage node; instead of considering the deduplication rate first, the data based on load balancing starts after the entire system has stored a certain number of data blocks. Routing strategy. Therefore, afterload balancing is adopted in the entire deduplication system, the storage capacity of all storage nodes is increased simultaneously. There will be no storage capacity of some storage nodes that is much larger than the storage capacity of other nodes. Therefore, when the similarity matching of the super block is used, the storage capacity of the storage node is relatively balanced, and the situation shown in the figure will not occur. Some storage nodes still have free space even it stores a large amount of data. Therefore, practical applications, when the storage node stores more balanced data, the main factor determining the routing node of the superblock is the number of matching fingerprint IDs. Only using max (Ci/Vi) may not be a good choice, so using the set threshold or the average storage capacity of the system to select the optimal storage node strategy are proposed. Use the set threshold or use the average storage capacity of the system to select the optimal storage node strategy. Literature [9] sets a threshold, which specifies the minimum value of Ci. When the value is lower than this value, the storage node will not be considered. This can solve certain problems. In addition, the storage capacity of the node exceeds the system. The percentage of the average capacity of all storage nodes and the product of Ci/Vi is used as the judgment basis for selecting the optimal routing node. These methods have achieved certain results.

B. SYSTEM DESIGN

The routing algorithm proposed in this section is based on the cluster deduplication system architecture as in Figure 2. It is not possible to send all the data of the superblock to the storage node for querying to determine the routing node when data routing is performed. If the storage node uses indexed tables to store the data block fingerprints, then the storage node does not have such a large memory space to keep all the fingerprint data. It is also possible to maintain in memory only a representative fingerprint ID index of the superblock. This approach can reduce a large amount of memory space, but the sequential query mode of the index table increases the communication overhead of the system. BloomFilter is used to manage fingerprint indexes in memory, and stateful routing policies use this approach, but this causes a large amount of system overhead.

This algorithm improves the query speed of fingerprints by maintaining BloomFilter in the memory of storage nodes representing fingerprint ID indexes, while selecting the k smallest block of fingerprints, by querying the number of fingerprints already maintained in BloomFilter for these k fingerprint indexes and the capacity of storage nodes and then using the corresponding calculation to get the optimal

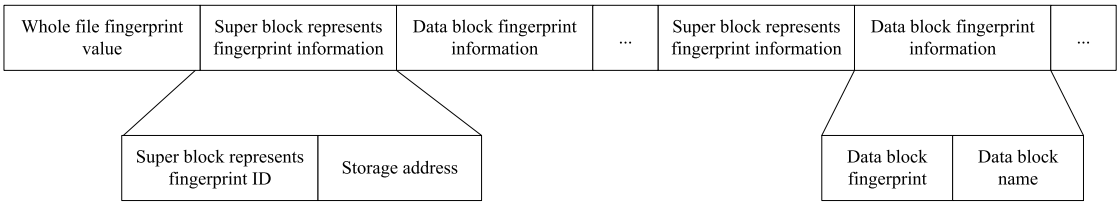

FIGURE 3. Backup metadata information.

storage node as the data routing node. Selecting the optimal routing node among a small number of storage nodes leads to a decrease in the overall deduplication rate of the system compared to the globally optimal routing strategy because the storage node is a locally optimal node rather than a globally optimal node. Therefore, considering the storage capacity and deduplication rate, this policy selects n ($\leq$ k) minimum block fingerprints to add to BloomFilter as the maintained superblock fingerprint index. The role of each part in this algorithm is analyzed below according to the cloud storage system structure diagram in Figure 2.

1) CLIENT-SERVER

The main function of this part is to process the data stream. In this algorithm, when the client-server receives the uploaded data stream, it performs data chunking, calculates the fingerprint of the data block, and the combination and routing of the superblock.

Also, after determining the routing node of the data, the metadata information of the data needs to be sent to the metadata server for data recovery. When determining the routing node, k minimum block fingerprints are sent to the storage node for querying instead of sending the data blocks to the storage node, which can reduce the communication overhead of the system.

2) METADATA SERVER

The format of metadata backed up by the metadata server is shown in Figure 3. There are multiple Superchunks representing fingerprint information in the backup metadata information, including the fingerprint ID of the Superchunk and the specific address of the Superchunk in the storage node. The multiple superblock representative fingerprint information is because the data blocks may be assigned to different superblocks. Hence, the data block information after a superblock representative fingerprint ID is all data blocks in the same superblock, and the data block fingerprint information includes the data block fingerprint and the specific name of the data block. When data recovery is performed, the superblock representative fingerprint and data block fingerprint are sent to the corresponding storage node. The superblock representative fingerprint is used to determine the container in the disk where the data block fingerprint is located without searching the whole disk. Then the data recovery algorithm is used to recover the required data.

ChunkID	ChunkName
B9645...	Chunk1...
. . .	. . .
C682B...	Name3...

FIGURE 4. Fingerprint data block correspondence table.

3) STORAGE NODE

BloomFilter is used in the storage node to maintain the n representative fingerprint IDs of all the superblocks stored in the node, that is, the n minimum block fingerprints of the superblocks, instead of maintaining all the data block index information, because storing all the fingerprint information requires considerable storage space, in addition to the query operation also requires much time. The data is stored on disk in containers one by one, and each container has the same superblock representative ID. Container stores the fingerprint information, including real data blocks and all data blocks, and the correspondence between fingerprint and data blocks. As shown in Figure 4, the table is stored in the container and needs to be utilized for de-duplication within the node. ChunkID indicates the fingerprint of the data block, and ChunkName indicates the name of the corresponding data block.

To improve the efficiency of deduplicated data queries and reduce memory occupation and disk I/O, the structure shown in Figure 5 is used for optimization at the storage node. As shown in the figure, the BloomFilter, including the node and the fingerprint cache, are stored in the memory of the storage node. The structure of the fingerprint cache includes the ID of the container and the fingerprint of the data block saved in the container. The fingerprint information of the data block and the ChunkName are saved to the index table maintained in the container only when the non-duplicate data is written to disk. After the superblock is sent to this node, the method of Min(ID) mod N (Min(ID) indicates the minimum chunk fingerprint of the superblock, and N indicates the number of containers in the disk) is used to select the container corresponding to the superblock. If the container

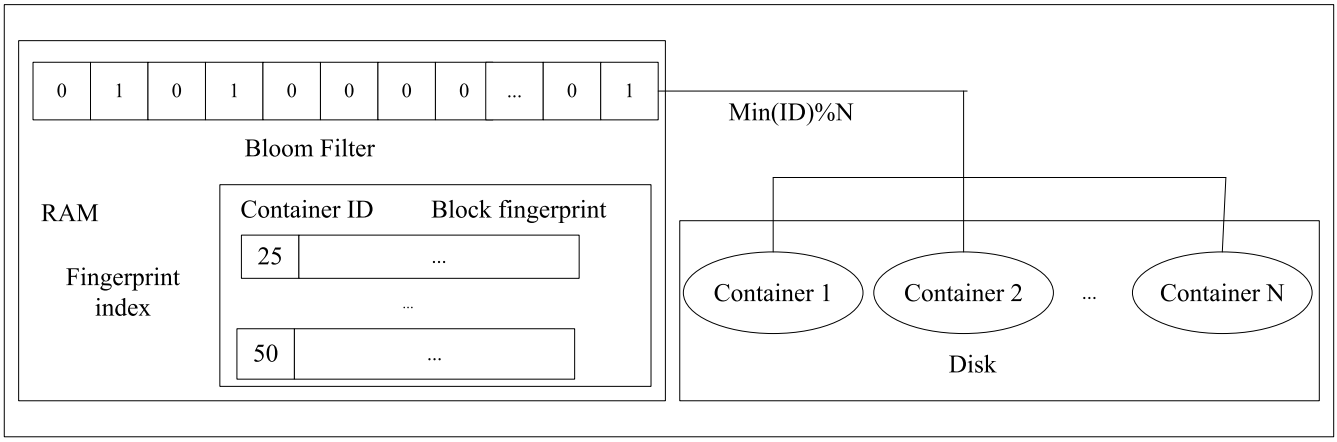

FIGURE 5. Logical structure of storage node.

exists in the cache, the duplicate data is directly deleted. Then the non-duplicate data is written to the cache and container, if the container is not in the cache, the container is read for deduplication, and the cache is updated using LRU. Since using only the minimum block, fingerprint ID may lead to a low deduplication rate, the wWrR method [33] can be used to read the container and perform deduplication.

C. ALGORITHM DESCRIPTION

BloomFilter has faster query speed and lower memory consumption than the traditional indexing method of hash table and similar index table, so it can further reduce disk I/O by applying a larger cache structure. Although BloomFilter has the problem of false-positive rate, the distributed BloomFilter-based data routing strategy proposed in this chapter does not utilize The BloomFilter-based data routing strategy proposed in this chapter does not utilize BloomFilter for deduplication, thus avoiding the impact of misclassification rate on deduplication. The in-memory index structure of the storage nodes is designed to achieve fast query and data deduplication while ensuring load balancing of the storage nodes.

The minimum k block fingerprints of the superblock are selected based on Broder's minimum independent replacement theorem. These k fingerprints determine p ($p \leq k$, because the same node number may be obtained) storage nodes. The optimal node is selected as the data routing node after calculating these p nodes using the corresponding algorithm. Finally, the n ($n \leq k$) minimum block fingerprints are added to the BloomFilter, achieving global load balancing through local load balancing. Also, Broder's theorem it does not significantly reduce the deduplication rate compared to the stateful routing strategy but also reduces the query time and decreases the memory usage.

To reduce disk I/O, the data is stored in each container (container), and each container stores the superblock with the same value after the representative fingerprint ID is calculated by hash. When deduplication is performed, the container number is obtained by hash calculation, and the corresponding container is read into memory for deduplication. The new data is stored in the container and then written to disk. The fingerprint cache is the fingerprints in the recently used containers in memory to reduce the disk I/O. The specific flow chart is shown in Figure 6.

1) The client-server divides the data into blocks and combines them into superblocks.

2) Choose k minimum block fingerprints (C_1, $C_2 \ldots C_k$), and determine p nodes ($p \leqslant k$) through C_i%N (N is the number of nodes).

3) Send k fingerprints to the corresponding p nodes, and query the BloomFilter in the memory to obtain the matching number of each node (H_1, $H_2 \ldots H_p$).

4) If the values of H_1, $H_2 \ldots H_p$ is all 0, the node with the smallest storage capacity is randomly selected as the routing node.

5) If the values of H_1, $H_2 \ldots H_p$ is not all 0, divide the number of matches by the node capacity Hi/Vi, and select Max (Hi/Vi) as the data routing node.

6) Send the superblock to the storage node, add the n minimum block fingerprints of the super block to BloomFilter, use the minimum block fingerprint ID of the super block to select the corresponding container (or use the wWrR method, which can increase a certain deduplication rate but increase the disk I/O).

7) If the container is already in the cache, the deduplication will be performed directly; if it is not in the cache, the container will be read into the memory.

8) If the container is not read, after deduplication, the non-duplicated data is directly added to the cache and then written to the disk. If the container is read, the LRU updates the cache after deduplication, and the data is written to the disk.

The meaning of distributed BloomFilter is that each storage node maintains a BloomFilter in its memory, where only n representative fingerprint IDs of each superblock are stored. This results in a high redaction rate and, at the same time, causes too much communication overhead in the system

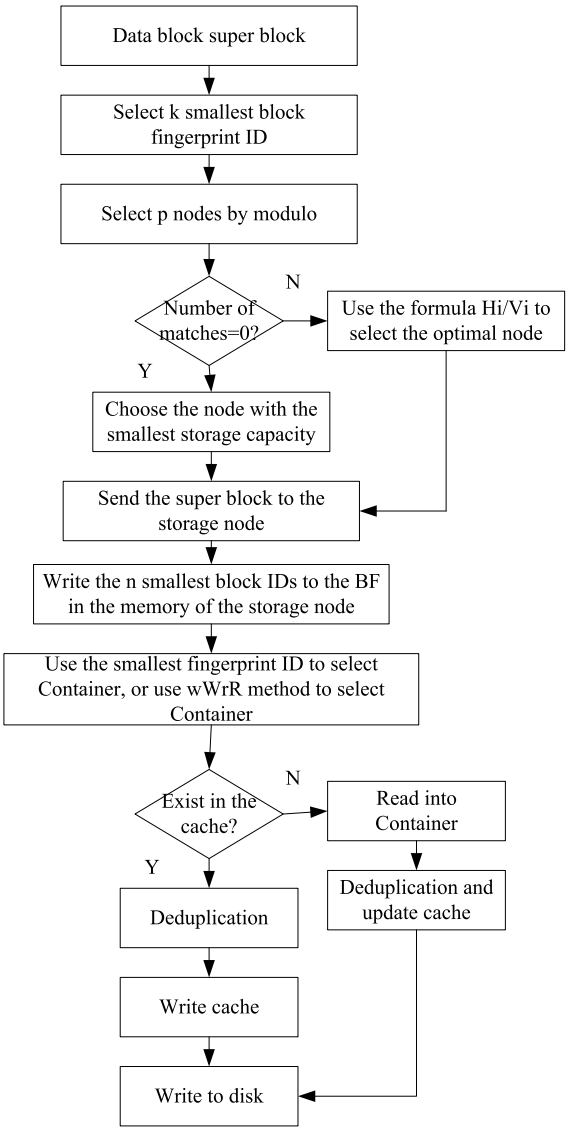

FIGURE 6. Flowchart of distributed BloomFilter-based routing policy.

because the system needs to send the fingerprint information to all nodes for the query. In contrast, this method does not need to use a broadcast-like approach to send the data to all nodes, but only to some nodes, and also ensures load balancing of the entire storage node and does not cause a significant decrease in the deduplication rate. The overall system architecture of the algorithm is shown in Figures 7.

As shown in Figure 7, the algorithm uses the logical structure shown in Figure 5 at each storage node. BloomFilter maintains the n minimum block fingerprints of all the superblocks at that storage node, which is then representative fingerprint ID of the superblock. This avoids the problem of false positive rate caused by Bloom Filter, because we only use Bloom Filter to determine whether the super block is sent to the storage node, while deduplication within the node is still performed using the fingerprint index table stored in the disk in Figure 4. Because the algorithm uses a sampling of the superblock to select a representative fingerprint, the worst that can happen when matching the BloomFilter (although the probability of this is extremely low) is that a storage node will have one or two more matches, but this may not be the final basis for routing the superblock to that node. The BloomFilter is also designed to match the maximum storage capacity of the storage node to minimize the misclassification rate and the impact of the misclassification rate.

Since there are multiple data blocks in the superblock, reading the container by the minimum block fingerprint of the superblock will cause a certain decrease in the deduplication rate. So that the wWrR method can improve the deduplication rate, but it will cause an increase in disk I/O, so in practice, it can choose a specific algorithm according to the specific needs. Extreme Binning takes the file as the unit, the data block signatures of all files with the same minimum block fingerprint are stored in a box, and the box is indexed by the minimum data block fingerprint of this box. These boxes contain the same minimum block fingerprint. So that these boxes may contain many of the same data blocks. When processing a new file, Extreme Binning selects a box by its minimal block fingerprint and updates it. wWrR is an extension and generalization of Extreme Binning, which we selects r minimal block fingerprints of superblocks. It then uses these r minimal block fingerprints to determine r containers, and after deduplication to write non-duplicated. In this paper, by selecting the superblock r minimum block fingerprint and then using this r minimum block fingerprint to determine the r containers, after performing deduplication the non-duplicated data is written to the w containers, which are determined by the w minimum block fingerprint. $w \leq r$ in practice. Extreme Binning reads a box and then updates this box (a total of two I/O operations are required). In contrast, wWrR requires $w + r$ I/O operations.

D. ALGORITHM SCALABILITY

Due to the gradual increase in storage demand, data centers need to increase the storage capacity of the system according to the storage situation, while one of the advantages of clustered storage architecture is to expand the storage capacity of the system by adding storage nodes according to the needs of users. Therefore, whatever the system can continue to meet the load balancing requirements when the storage nodes are increased is a situation that must be considered in a data routing strategy. In the case of increased storage nodes, some of the data stored in the storage nodes may need to be transferred to the newly added storage nodes, and this is an important research point for cluster deduplication. However, this aspect is not in the scope of this paper. Therefore, the definition of scalability in this paper refers to whether the data routing policy still adapts well to the deduplication system when both the storage size and the storage node size increase accordingly. There is still a good system deduplication rate, a good load balancing situation, and a low load.

The distributed BloomFilter-based data routing policy simply selects the k smallest block fingerprints as the

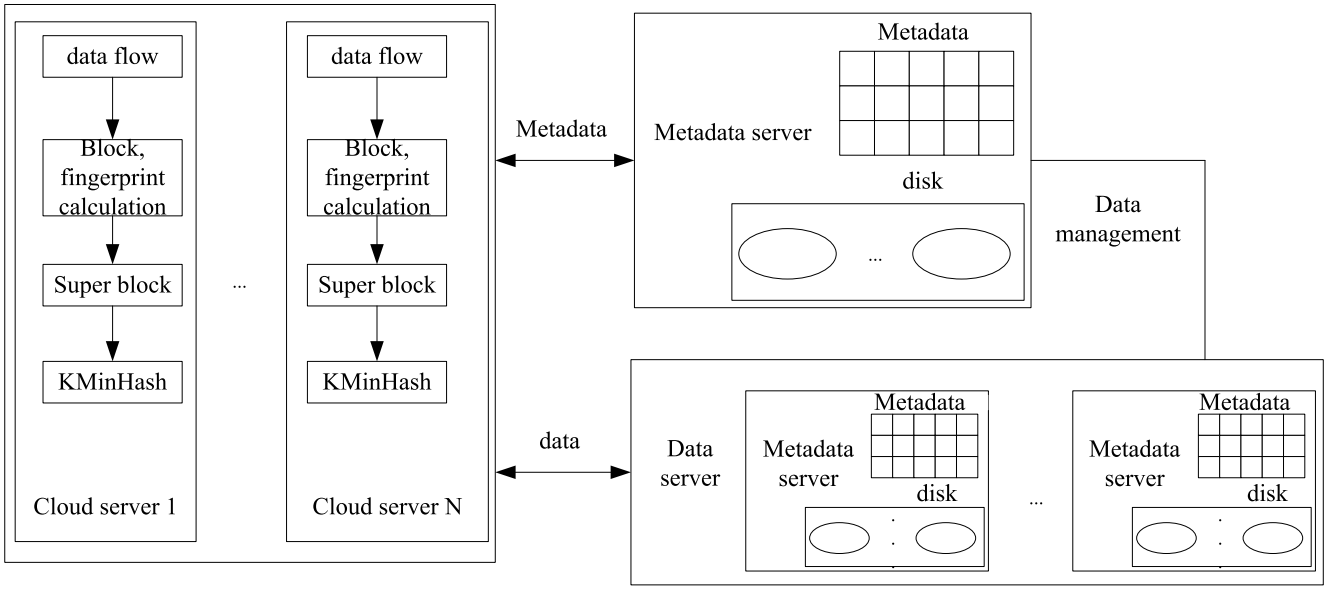

FIGURE 7. System structure of distributed BloomFilter-based routing algorithm.

representative fingerprints of the superblock to determine p routing nodes. It then selects the optimal node among these p as the final routing node, whether the method implemented can achieve load balancing. The following theorem can be obtained here [34].

Theorem 1: If the k representative fingerprints in the superblock obtain the local load balancing achieved by k storage nodes under the modal operation, all storage nodes under the whole clustered system can approach the global load balancing.

It is proved by using the converse method. Suppose there are N storage nodes in the cluster when N is much larger than k. If the original proposition is false, then there exist two load cases for storage nodes, high-load nodes $\{H_1,\ldots,H_i\}$ and low-load nodes $\{L_1,\ldots,L_j\}$, where there are $i + j = N$. For any superblock representative fingerprints, either map to high-load nodes or mapped to low-load nodes. If the representative fingerprints of superblock A are mapped to high load, and the fingerprints of superblock B are mapped to low load, then a superblock C is constructed with half from superblock A and a half from superblock B, so half are mapped to high load, and half are mapped to low load. Therefore, superblock C is neither high load nor low load, forming a paradox, so the original proposition is true. We can obtain global load balancing by implementing local load balancing.

Although the distributed BloomFilter-based data routing strategy can achieve better load balancing as the system scales, there are certain drawbacks. If the k smallest block fingerprints are chosen to remain the same, then the number of p storage nodes determined by these k smallest block fingerprints also remains the same. Then the probability of the optimal routing node of the system being selected for the superblock will decrease, and the probability decreases as the system grows larger. Although the deduplication rate decreases gradually, the system performance also increases with the increase of system nodes, compared with the broadcast query. Therefore, in practice, this data routing strategy is suitable for deduplication systems with high system performance requirements. If the system has a high deduplication rate requirement, it is necessary to use the relevant strategy to improve it.

This section proposes a data routing strategy based on distributed BloomFilter, which uses deduplication at storage nodes, selects a small number of data block fingerprints according to Broder's theorem, and uses these fingerprints to determine a small number of storage nodes, which is implemented through partial load balancing Global load balancing, this strategy can effectively reduce the communication overhead of the system, and is suitable for large-scale cluster environments.

IV. EXPERIMENTAL EVALUATION

A. DATA SET AND EXPERIMENTAL ENVIRONMENT

In order to verify the distributed BloomFilter-based data routing strategy, a total of 124.3G of kernel file data from Linux 3.0.1 to Linux 4.6 was selected. Meanwhile, this experiment is based on the simulation experimental environment, using a single machine to simulate the cluster environment. All data files are divided into data blocks according to the size of 4KB by using the fixed-length chunking method. The fingerprints of all data blocks are calculated using MD5, and then all data block fingerprints are processed. The experimental test environment is shown in Table 1.

Due to the performance limitations of the stand-alone environment and the need to simulate clusters, this experiment

TABLE 1. Test host configuration.

Software and hardware environment	Configuration information
CPU	Inter® Core™ i7-10700K CPU @3.60GHz
RAM	16GB
Disk	2TB HDD+500GB SSD
operating system	Win10
Operating environment	JDK1.8

uses a simulation cluster for experimental testing, using a new folder in the disk as the storage node for simulation, each storage node stores a txt file with fingerprint data, each txt file represents the container stored in the disk, each time the deduplication type will read the txt file into memory After data processing, the non-duplicate data is written to the txt file. Then the number of fingerprints in the txt file is calculated with the original number of fingerprints to obtain the deduplication rate of the system.

B. EXPERIMENTAL RESULTS AND ANALYSIS

In this experiment, the Superchunk is used as the basic unit of data routing. The distributed BloomFilter-based data routing strategy select the k smallest block fingerprints in the Superchunk as the representative fingerprint ID of the Superchunk and how to determine the value of k is also an important condition that affects the deduplication rate of the system, and in the literature [30] by, In addition, the size of the superblock also has an impact on the deduplication rate and system overhead, the stateful data routing strategy uses a 1MB size superblock. In contrast, the literature [30] uses a 16MB superblock size. 16MB superblock size, due to the simulation-based experimental validation approach used in this experiment, a 4MB size superblock is used as the basic unit of data routing by balancing the deduplication rate and system performance. In addition, due to the distributed BloomFilter-based, as the system nodes, increase there will be a lower probability of the global optimal storage node being selected, the storage nodes need to maintain more representative fingerprints, while considering the storage occupancy and performance aspects, this paper selects the four smallest block fingerprints in the superblock to add to the BloomFilter maintained by the storage nodes.

Firstly, we test the comparison of the deduplication rate of the algorithms proposed in this paper under different storage nodes. As shown in Figure 8, Stateful denotes the stateful data routing policy, DRMF [19] algorithms and Σ-Dedupe [12] algorithms, Stateless denotes the stateless data routing policy proposed in this paper, and DBF denotes the global BloomFilter-based data routing policy proposed in this paper. In the experiments, five storage nodes, 8 nodes, 16 nodes, 32 nodes, 64 nodes, 128 nodes, 256 nodes, and 512 nodes, are selected to verify the number of nodes.

From Figure 8, we can see that as the storage nodes increase, the stateful routing policy selects the optimal

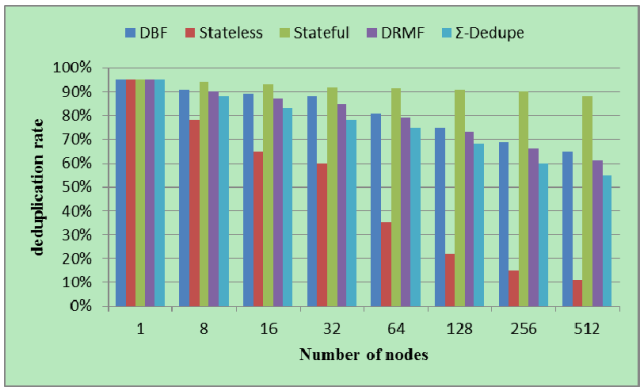

FIGURE 8. Deduplication rate of routing strategy.

storage node based on all the data blocks in the superblock, so the overall deduplication rate of the system does not decrease significantly as the nodes increase and still maintains a very high deduplication rate. However, the stateless data routing strategy decreases quite severely with the increase of nodes. The deduplication rate decreases to about 20% in the case of 128 nodes, mainly because the stateless routing selects the first 64 bits of the first data block fingerprint in the superblock as the representative fingerprint ID of the superblock. Because DRMF uses data characteristics for routing, the effect is slightly lower than the strategy in this article by 3%. This strategy cannot be used to determine the similarity of the superblocks essentially. Mainstream data routing strategies are mainly used to improve the deduplication rate of the system and exploit the principle of data similarity and locality. The distributed BloomFilter-based data routing policy proposed in this paper has a correspondingly higher deduplication rate than the stateless data routing policy, which is lower than Σ-Dedupe. With the change of the number of nodes, the performance improvement of the algorithm in this paper relative to Σ-Dedupe can reach up to 10%. However, this routing strategy achieves the idea of global load balancing through the local load balancing strategy. only a few optimal nodes are selected among the nodes, which mean that the selected nodes could be local optimal nodes instead of global optimal nodes.

which means that it is likely that the optimal nodes in the global are not in these selected storage nodes. The probability of being able to obtain the global optimal storage nodes decreases as the number of nodes in the system increases. Only local merit selection is possible, so as the storage nodes, Therefore, as the number of storage nodes increases, the deduplication rate decreases to a certain extent.

The distributed BloomFilter-based data routing strategy only keeps the fingerprint index of the n smallest blocks of the superblock in the BloomFilter in the memory of the storage node In order to minimize the memory occupation, it can be seen from Figure 8 that only the n smallest block fingerprint index is maintained In the case of constant data volume, the deduplication rate of the system decreases with the increase of storage nodes, which This is mainly because the distributed

BloomFilter-based data routing strategy selects only p storage nodes using k minimum block fingerprints in order to save the communication overhead of the system, and selects the optimal storage node from these p storage nodes as the node for data routing, so the optimal node of these p storage nodes is not necessarily the global optimal storage node, so with the increase of storage nodes, the global The probability of the optimal node being selected further decreases as the number of storage nodes increases, so the deduplication rate decreases somewhat with the increase of storage nodes in the system, but still improves to a large extent compared to the stateless data routing strategy. Therefore, when designing the experimental system, the number of storage nodes needs to be designed according to the stored data to balance the performance and deduplication rate issues as much as possible.

In order to verify the load balancing of the two data routing strategies proposed in this paper, we use the number of data blocks after deduplication of storage nodes as the basis for judgment. The fluctuation of the number of connected lines of all storage nodes storing is data blocks can indicate the load balancing of storage nodes because the stateless data routing strategy has been The stateless data routing policy has been shown to create load imbalance in case of too many system nodes [9] and the data routing policy does not consider load balancing, so it is not compared with the stateless data routing policy in this experiment. This experiment tests the load balancing of the three policies in 16 storage nodes and 64 storage nodes, which are two sizes of storage nodes, to verify the load balancing of the two policies at different sizes of storage nodes.

First, we tested the load balancing of the system storage nodes based on the distributed BloomFilter data routing policy at 16 storage nodes and 64 storage nodes, and the experimental results are shown in Figures 9 and Figure 10. Here, the load balancing of the system is verified by using the number of non-duplicated data remaining in the storage nodes after deduplication, and the load balancing of the storage nodes is reflected by the connection of the storage situation of these nodes. The vertical coordinate in the figure indicates the number of remaining non-duplicate data in the storage nodes. We can judge the load balance of the system storage nodes by the fluctuation of the data volume in the nodes. The smaller the fluctuation, the better the load balancing of the storage nodes, and the larger the fluctuation, the greater the load balancing of the storage nodes. In the figure, ulb means unload balance. As shown in the figure, without considering load balancing, the data volume of each storage node fluctuates greatly, and this phenomenon will become more obvious with the increase of storage nodes and data volume. However, it is necessary to adopt a load balancing strategy.

As shown in Figure 9, Figure 10, the figures indicate the remaining non-duplicated data blocks in the storage nodes under the capacity of 16 storage nodes and 64 storage nodes with the distributed BloomFilter-based data routing policy with and without considering load balancing. The number of

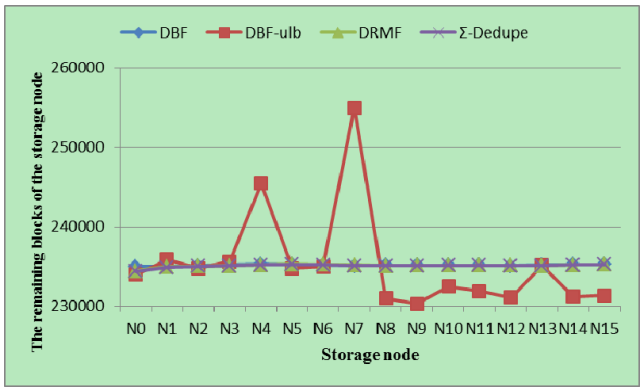

FIGURE 9. 16-node DBF load balancing situation.

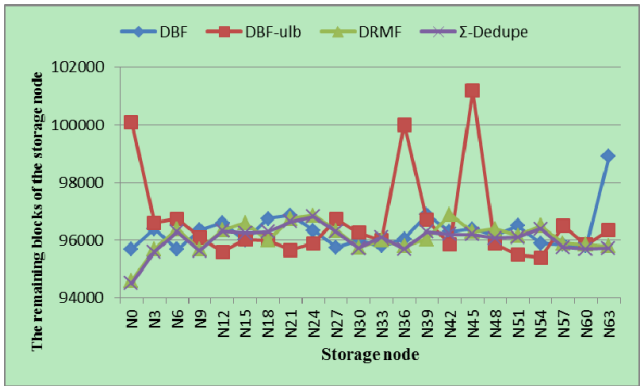

FIGURE 10. 64-node DBF load balancing situation.

non-duplicated data blocks remaining in the storage nodes in both cases with and without load balancing. The load balancing of the system can be known by the connected situation. From the figure, it can be seen that the number of non-duplicate data blocks in the storage nodes without considering load balancing is unevenly distributed, and the gap between their extremely and extreme small values is large. After using the load balancing policy, the number of non-duplicated data blocks stored in the storage nodes fluctuates within a certain range. There is no large difference between the maximum and minimum values. If there is a large difference between the extreme values, then it means that some storage nodes in the system store a large amount of data, and some storage nodes store a small amount. When the number of nodes is small, the clustering phenomenon of users becomes more obvious. When calculating the weighted similarity, the similarity difference between the superblock and each node is much higher than the load difference. At this time, the weighted similarity effect is not obvious. However, as the number of nodes increases, the physical load balance of the algorithm is more reasonable than Σ-Dedupe.

In addition, in the case of two sizes of storage nodes, the overall deletion rate of the system with the added load balancing constraint decreases by less than 1% compared to that without the load balancing constraint. However, the

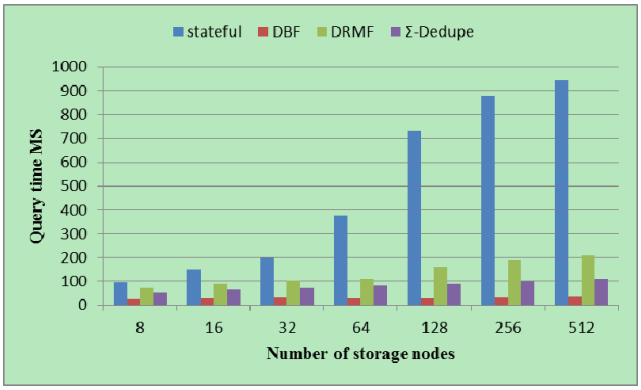

FIGURE 11. Comparison of routing query times.

load on the storage nodes is well balanced. Compared to the decrease in the deduplication rate of the system, the gap in the amount of data stored in the storage nodes will become larger as the amount of data in the system increases without the load balancing strategy. Then those storage nodes that store a large amount of data will become high-frequency access nodes of the whole system, which will seriously limit the performance of the whole deduplication system, so for the deduplication system, load balancing is very necessary, the reduction of the deduplication rate is inevitable. The communication cost of the strategy in this paper is second only to Stateful. Although Σ-Dedupe uses fingerprints to avoid broadcast routing, it also uses a smaller super block and belongs to a stateful routing algorithm. Therefore, compared with the stateless routing algorithm, its communication The overhead is still large. However, palmprint technology makes the communication overhead of Σ-Dedupe no longer increase with the increase of storage nodes, so when there are multiple storage nodes, its routing overhead is far less than Stateful.

Another important factor of the deduplication system is the communication overhead of the system. This experiment tests the query time of data routing with 8,16,32,64,128 and 512 storage nodes. This experiment uses data block fingerprints for simulation experiments, so the whole running experiment starts from combining superblocks until all data fingerprints are sent. Since both the stateful data routing strategy and the distributed BloomFilter-based data routing strategy proposed in this paper require querying to the storage nodes, this experiment first tests the query time from the completion of the combination of superblocks to the determination of storage nodes for both the stateful data routing and the distributed BloomFilter-based data routing strategies, so the time for this data routing policy is shown as a dashed line, and the results are shown in Figure 11.

From the Figure, we can see that the query time of the stateful data routing policy increases rapidly with the increase of storage nodes. In contrast, the routing query time of the distributed BloomFilter-based data routing policy proposed in this paper is always maintained in a stable and low state. Therefore, it can be concluded that the distributed BloomFilter-based data routing strategy proposed in this paper can greatly reduce the communication overhead of data routing, which is a great improvement over the stateful data routing strategy. In addition, experiments tested the time required to send the actual data in the network, the communication time to send the entire superblock of 1024 fingerprints to a single storage node for querying averaged 300ms, while the communication time to a single storage node required to select the 8 smallest block fingerprints averaged 15ms, while the distributed BloomFilter-based data routing strategy only sends to a few storage nodes. Thus It can save the system communication overhead significantly.

Through previous experiments, we conclude that the data routing strategy based on the distributed BloomFilter has a faster decrease in the deduplication rate compared to the other two data routing strategies, mainly because the data routing strategy based on the distributed BloomFilter only stores the n minimum block fingerprint IDs of the superblocks in the BloomFilter maintained by the storage nodes. There are multiple data in the superblocks of data blocks, so the representation of fewer superblock minimum block fingerprint IDs is insufficient, and in addition, as the number of storage nodes increases, the use of k minimum block fingerprint IDs to determine p storage nodes makes the probability of the globally optimal node being selected decreasing, leading to the problem of decreasing deduplication rate. Therefore, this experiment further tests the addition of all 8 representative fingerprint IDs of the superblock to the BloomFilter maintained by the storage nodes. In addition, to ensure further improvement of the deduplication rate of the system, the strategy of using wWrR to select the container is also tested, and the experimental results are shown in Figure 12. In the figure, DBF indicates that the BloomFilter of the storage node maintains the 4 smallest block fingerprint IDs of the superblock, DBF-e indicates that the BloomFilter of the storage node maintains the 8 smallest block fingerprint IDs in the superblock. DBF-w indicates that the container is selected using the wWrR data routing policy, and since the wWrR policy is to select the r smallest block fingerprints by and then update the w containers, with the increase of w and r the corresponding disk I/O will also increase, so this experiment only uses the 1W2R method for deduplication testing, DBF-we indicates that the deduplication rate is calculated by using the combination of wWrR and the 8 smallest block fingerprints at the same time.

From Figure 12, it show that experiment tests the comparison of the deduplication rates of the four methods with 128 storage nodes. DBF-e will achieve better deduplication rates after increasing the number of representative fingerprints maintained compared to maintaining 4 minimal block fingerprint IDs, but maintaining more fingerprint indexes requires more storage space. According to Section 3, maintaining all representative fingerprint IDs, i.e., 8 minimal block fingerprint IDs, requires an increase in memory space. However, the cost is still acceptable compared to maintaining all fingerprint indexes. By increasing the number of fingerprints maintained and using wWrR for deduplication, the

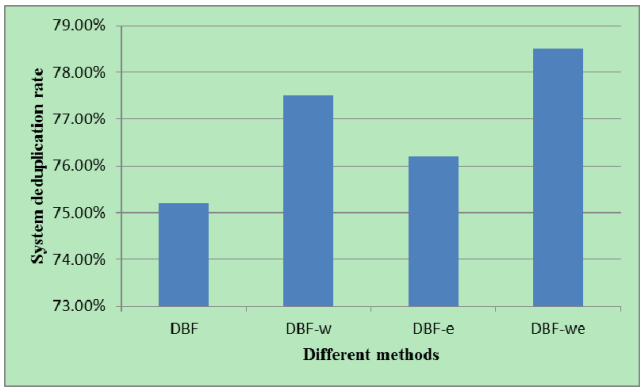

FIGURE 12. DBF, DBF-e, DBF-w, DBF-we deduplication rate comparison.

deduplication rate can be increased more than the method of increasing representative fingerprints. However, this method will increase disk I/O. In general, only two I/O operations are needed to read the container and write the data to the container, while using wWrR requires w + r times. The number of disk I/Os is (w + r)/2 times higher than the normal case. Still, with 128, 256 and 512 storage nodes overall compared to using a simple distributed BloomFilter-based data routing strategy, the deduplication rate can be improved by about 3%. The process is carried out in the storage nodes, which does not affect system performance, so the price paid is worth it. In practical use, this data routing strategy needs to ensure a balance between storage nodes and data volume to prevent the use of too many storage nodes with fewer data resulting in poor deduplication. In contrast, the minimum number of block fingerprints in the BloomFilter maintained by storage nodes needs to be selected reasonably based on the performance of the system storage nodes.

V. CONCLUSION

In order to reduce the communication overhead of the clustered deduplication system, this paper proposes a distributed BloomFilter-based data routing strategy, which takes the superblock as the basic data routing unit, selects the k smallest block fingerprints of the superblock as the representative fingerprint IDs of the superblock, determines p storage nodes by these fingerprint IDs, and then uses BloomFilter to determine these The number of occurrences of these k smallest block fingerprint IDs in the corresponding nodes is then determined using BloomFilter. The optimal node is calculated as the final data routing node using the number of matches and the capacity of the storage node. This strategy can obtain a better deduplication rate and load balancing with lower communication overhead.

At present, the research on cluster deduplication routing strategy mainly focuses on two aspects of deduplication rate and load balancing, and many research results have been obtained. However, there are still many problems that have not been solved or have become the main research points. The following aspects deserve more attention and research in the future:

(1) The deletion and addition of storage nodes in the cluster deduplication system. The storage nodes in the cluster deduplication system may stop working due to some problems, or the storage capacity cannot meet the data growth. At this time, storage nodes need to be added. However, Ruhe will transfer the data to the new storage node and how to recover the lost data. Although the current consistency hash and erasure code-based strategies can have certain effects, they can still be better.

(2) Selection of the representative fingerprint ID of the super block. At present, the Border theorem is mainly used select the representative fingerprint ID of the super block. The effect of the data routing strategy for the data routing unit is not obvious. Therefore, the selection of a good data routing unit or the selection of a representative fingerprint ID can effectively improve the deduplication rate of the system.

REFERENCES

[1] J. Gantz and D. Reinsel, "The digital universe decadelare you ready," IDC, Needham, MA, USA, White Paper, May 2010.
[2] H. Bigger, "Experencing data de-duplication: Improving efficiency and reducing capacity requirements," Enterprise Strategy Group, Silicon Valley, CA, USA, White Paper, Feb. 2007.
[3] 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.
[4] N. Mandagere, P. Zhou, M. A. Smith, and S. Uttamchandani, "Demystifying data deduplication," in *Proc. ACM/IFIP/USENIX Int. Middleware Conf. Companion Middleware Companion (Companion)*, 2008, pp. 12–17.
[5] 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.
[6] 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. 7th Conf. File Storage Technol.*, Berkeley, CA, USA: USENIX Assoc., 2009, pp. 111–123.
[7] D. Meister and A. Brinkmann, "Multi-level comparison of data deduplication in a backup scenario," in *Proc. SYSTOR Israeli Experim. Syst. Conf (SYSTOR)*, 2009, p. 8.
[8] 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. 7th USENIX Conf. File Storage Technol. (FAST)*, Berkeley, CA, USA: USENIX, 2009, pp. 197–210.
[9] W. Dong, F. Douglis, K. Li, H. Patterson, S. Reddy, and P. Shilane, "Tradeoffs in scalable data routing for deduplication clusters," in *Proc. 9th USENIX Conf. File Storage Technol. (FAST)*, Berkeley, CA, USA: USENIX, 2011, pp. 15–29.
[10] S. Luo, G. Zhang, C. Wu, S. U. Khan, and K. Li, "Boafft: Distributed deduplication for big data storage in the cloud," *IEEE Trans. Cloud Comput.*, vol. 8, no. 4, pp. 1199–1211, Oct. 2020.
[11] R. Real and J. M. Vargas, "The probablilistic basis of Jaccard's index of similarity," *Systematic Biol.*, vol. 45, no. 3, pp. 380–385, 1996.
[12] Y. Fu, H. Jiang, and N. Xiao, "A scalable inline cluster deduplication framework for big data protection," in *Proc. ACM/IFIP/USENIX 13th Int. Conf. Middleware (Middleware)*, Montreal, QC, Canada, Dec. 2012, pp. 354–373.
[13] Q. He, G. Bian, B. Shao, and W. Zhang, "Data deduplication technology for cloud storage," *Tehnički Vjesnik*, vol. 27, no. 5, pp. 1445–1446, 2020.
[14] G. Lu, Y. Jin, and D. H. C. Du, "Frequency based chunking for data de-duplication," in *Proc. IEEE Int. Symp. Modeling, Anal. Simulation Comput. Telecommun. Syst.*, Aug. 2010, pp. 287–296.
[15] S. Brin, J. Davis, and H. García-Molina, "Copy detection mechanisms for digital documents," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 1995, pp. 398–409.

[16] S. Wu, J. Zhou, W. Zhu, H. Jiang, Z. Huang, Z. Shen, and B. Mao, "EaD: A collision-free and high performance deduplication scheme for flash storage systems," in *Proc. IEEE 38th Int. Conf. Comput. Design (ICCD)*, Oct. 2020, pp. 155–162.

[17] P. Yang, N. Xue, Y. Zhang, Y. Zhou, L. Sun, W. Chen, Z. Chen, W. Xia, J. Li, and K. Kwon, "Reducing garbage collection overhead in SSD based on workload prediction," in *Proc. 11th USENIX Workshop Hot Topics Storage File Syst. (HotStorage)*, Renton, WA, USA, Jul. 2019, p. 20.

[18] B. Harris and N. Altiparmak, "Ultra-low latency SSDs' impact on overall energy efficiency," in *Proc. 12th USENIX Workshop Hot Topics Storage File Syst. (HotStorage)*, Jul. 2020, pp. 1–25.

[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] 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.

[21] M. O. Rabin, "Fingerprinting by random polynomials," Harvard Univ., Cambridge, MA, USA, Tech. Rep. (TR-15-81), 1981.

[22] A. Z. Broder, "Some applications of Rabin's fingerprinting method," in *Sequences II: Methods in Communications, Security, and Computer Science*, R. Capocelli, A. D. Santis, and U. Vaccaro, Eds. Cham, Switzerland: Springer, 1993, pp. 143–152.

[23] 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 2005-30, 2009.

[24] R. Rivest, *The MD5 Message-Digest Algorithm*, document RFC Editor, 1992.

[25] *Secure Hash Standard*, document FIPS-180, NIST, Federal Information Processing Standard, May 1993.

[26] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[27] [Online]. Available: http://www.google.co.hk/ggblog/googlechinablog/2007/07/bloomfilter_7469.html

[28] C. Liu, Y. Lu, C. Shi, G. Lu, D. H. C. Du, and D.-S. Wang, "ADMAD: Application-driven metadata aware de-duplication archival storage system," in *Proc. 5th IEEE Int. Workshop Storage Netw. Archit. Parallel (I/Os)*, Sep. 2008, pp. 29–35.

[29] Q. He, G. Bian, W. Zhang, F. Wu, and Z. Li, "TCFTL: Improved real-time flash memory two cache flash translation layer algorithm," *J. Nanoelectron. Optoelectron.*, vol. 16, no. 3, pp. 403–413, Mar. 2021.

[30] Y.-J. Fu, N. Xiao, X.-K. Liao, and F. Liu, "Application-aware client-side data reduction and encryption of personal data in cloud backup services," *J. Comput. Sci. Technol.*, vol. 28, no. 6, pp. 1012–1024, Nov. 2013.

[31] R. Real and J. M. Vargas, "The probabilistic basis of Jaccard's index of similarity," *Systematic Biol.*, vol. 45, no. 3, pp. 380–385, Sep. 1996.

[32] Q. He, G. Bian, W. Zhang, F. Zhang, S. Duan, and F. Wu, "Research on routing strategy in cluster deduplication system," *IEEE Access*, vol. 9, pp. 135485–135495, 2021.

[33] Z. Zhang, D. Bhagwat, W. Litwin, D. Long, and S. J. T. Schwarz, "Improved deduplication through parallel binning," in *Proc. IEEE 31st Int. Perform. Comput. Commun. Conf. (IPCCC)*, Dec. 2012, pp. 130–141.

[34] M. Ajdari, P. Park, J. Kim, D. Kwon, and J. Kim, "CIDR: A cost-effective in-line data reduction system for terabit-per-second scale SSD arrays," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2019, pp. 28–41.

[35] Q. He, Z. Yu, G. Bian, W. Zhang, K. Liu, and Z. Li, "Research on key technologies of NBD storage service system based on load classification," *AIP Adv.*, vol. 11, no. 12, Dec. 2021, Art. no. 125124, doi: 10.1063/5.0071929.

[36] 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, pp. 1072–1082.

[37] K. Han, H. Kim, and D. Shin, "WAL-SSD: Address remapping-based write-ahead-logging solid-state disks," *IEEE Trans. Comput.*, vol. 69, no. 2, pp. 260–273, Feb. 2020.

[38] 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.

[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*.

[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, Feb. 2020.

[44] J. Li, Z. Yang, Y. Ren, P. P. C. Lee, and X. Zhang, "Balancing storage efficiency and data confidentiality with tunable encrypted deduplication," in *Proc. 15th Eur. Conf. Comput. Syst.*, Apr. 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, "Accelerating encrypted deduplication via SGX," CUHK, Hong Kong, Tech. Rep., 2021. [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.

[51] W. Xia, X. Zou, H. Jiang, Y. Zhou, C. Liu, D. Feng, Y. Hua, Y. Hu, and Y. Zhang, "The design of fast content-defined chunking for data deduplication based storage systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 9, pp. 2017–2031, Sep. 2020.

[52] S. Park, I. Kang, Y. Moon, J. H. Ahn, and G. E. Suh, "BCD deduplication: Effective memory compression using partial cache-line deduplication," in *Proc. 26th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Apr. 2021, pp. 52–64.

[53] Z. Cao, H. Wen, F. Wu, and D. H. Du, "ALACC: Accelerating restore performance of data deduplication systems using adaptive look-ahead window assisted chunk caching," in *Proc. 16th USENIX Conf. File Storage Technol. (FAST)*, 2018, pp. 309–324.

[54] Y. Tan, C. Xu, J. Xie, Z. Yan, H. Jiang, W. Srisa-An, X. Chen, and D. Liu, "Improving the performance of deduplication-based storage cache via content-driven cache management methods," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 214–228, Jan. 2021.

[55] Z. Cao, S. Liu, F. Wu, G. Wang, B. Li, and D. H. Du, "Sliding look-back window assisted data chunk rewriting for improving deduplication restore performance," in *Proc. 17th USENIX Conf. File Storage Technol. (FAST)*, 2019, pp. 129–142.

[56] Y. Zhang, Y. Yuan, D. Feng, C. Wang, X. Wu, L. Yan, D. Pan, and S. Wang, "Improving restore performance for in-line backup system combining deduplication and delta compression," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 10, pp. 2302–2314, Oct. 2020.

[57] M. Lu, F. Wang, D. Feng, and Y. Hu, "A read-leveling data distribution scheme for promoting read performance in SSDs with deduplication," in *Proc. 48th Int. Conf. Parallel Process.*, Aug. 2019, p. 22.

[58] X. Zou, J. Yuan, P. Shilane, W. Xia, H. Zhang, and X. Wang, "The dilemma between deduplication and locality: Can both be achieved," in *Proc. 19th USENIX Conf. File Storage Technol. (FAST)*, 2021, pp. 171–185.

[59] Q. Wang, J. Li, W. Xia, E. Kruus, B. Debnath, and P. P. C. Lee, "Austere flash caching with deduplication and compression," in *Proc. USENIX Annu. Tech. Conf. (ATC)*, Jul. 2020, pp. 713–726.

[60] Y. Zhang, Y. Yuan, D. Feng, C. Wang, X. Wu, L. Yan, D. Pan, and S. Wang, "Improving restore performance for in-line backup system combining deduplication and delta compression," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 10, pp. 2302–2314, Oct. 2020.

QINLU HE (Member, IEEE) received the Ph.D. degree in computer science from Northwestern Polytechnical University. He is currently an Associate Professor with the 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 the 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.

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 also 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 the China Computer Federation and the Association for Computing Machinery.

WEIQI ZHANG received the master's degree from the 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 the China Computer Federation.

DONGLI DUAN currently works as an Associate Professor and a Master's Supervisor with the Xi'an University of Architecture and Technology. His main research interests include network resilience, complex system reliability, and big data technology.

ZHEN LI is currently a Senior Engineer with the Shaanxi Institute of Metrology. He have been published more than ten papers of various types in journals and international conferences. His research interests include reliability test of electronic and electrical products, EMC electromagnetic compatibility, and computer information network inspection and testing. He is a member of the Intelligent Building and Building Automation Professional Committee of the Shaanxi Institute of Automation.

CHEN CHEN 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.