Hindawi
Scientific Programming
Volume 2020, Article ID 8869237, 11 pages
https://doi.org/10.1155/2020/8869237

Review Article

Research on Multifeature Data Routing Strategy in Deduplication

Qinlu He,[1] **Genqing Bian**,[1] **Bilin Shao**,[2] **and Weiqi Zhang**[1]

[1]*School of Information and Control Engineering, Xi'an University of Architecture and Technology, Xi'an 710043, China*
[2]*School of Management, Xi'an University of Architecture and Technology, Xi'an 710043, China*

Correspondence should be addressed to Qinlu He; luluhe8848@hotmail.com

Received 15 June 2020; Revised 10 August 2020; Accepted 30 September 2020; Published 14 October 2020

Academic Editor: Cristian Mateos

Deduplication is a popular data reduction technology in storage systems which has significant advantages, such as finding and eliminating duplicate data, reducing data storage capacity required, increasing resource utilization, and saving storage costs. The file features are a key factor that is used to calculate the similarity between files, but the similarity calculated by the single feature has some limitations especially for the similar files. The storage node feature reflects the load condition of the node, which is the key factor to be considered in the data routing. This paper introduces a multifeature data routing strategy (DRMF). The routing strategy is made based on the features of the cluster, including routing communication, file similarity calculation, and the determination of the target node. The mutual information exchange is achieved by routing communication, routing servers, and storage nodes. The storage node calculates the similarity between the files stored, and then the file is routed according to the information provided by the routing server. The routing server determines the target node of the route according to the similar results and the node load features. The system prototype is designed and implemented; also, we develop a system to process the feature of cluster and determine the specific parameters of various features of experiments. In the end, we simulate the multifeature data routing and single-feature data routing, respectively, and compare the deduplication rate and data slope between the two strategies. The experimental results show that the proposed data routing strategy using multiple features can improve the deduplication rate of the cluster and maintain a lower data skew rate compared with the single-feature-based routing strategy MCS; DRMF can improve the deduplication rate of the cluster and maintain a lower data skew rate.

1. Introduction

According to the International Data Corporation (IDC) report, global data spheres will grow from 33 ZB in 2018 to 175 ZB in 2025 [1]. With the amount of stored data growing constantly and exponentially, and storage costs are increasing dramatically, economical storage solutions are needed. In recent years, data storage has attracted more and more attention; how to reduce the maintenance cost and improve storage efficiency is a new hot research point in storage system. Deduplication works by representing files and their chunks with hash values and comparing these values against hashes of previously processed files and chunks [2]. If a matching hash entry exists, then the data the hash represents is a duplicate; otherwise, it is considered unique. Duplicates are replaced with links to old data while data belonging to unique hashes are stored [3].

However, these backup systems with data deduplication usually can only delete duplicate data onto a single node, and the storage capacity of a single node is low, and it is difficult to meet large-scale storage requirements [3]. An effective solution to solve the single-node low-throughput and difficult-to-expand defects is to create a cluster of multiple storage nodes. Each storage node in the cluster can independently perform deduplication work [3,4]. Deduplication clusters can meet increasing data backup needs [5]. Since 2010, the international academic community has focused on deduplication clusters [6–8] and applied many popular data routing strategies to deduplication clusters. Although the

deduplication cluster improves the data operation throughput of the storage backup system, it reduces the deduplication rate of a single node in the cluster. If there are 64 nodes in the cluster, the data reduction rate of the cluster is about 50% of the single node. If the backup data is e-mail information, the data reduction rate is reduced to about 20% of the single node. Moreover, as the number of nodes increases, the data reduction rate of the cluster will further decrease [9]. Using a suitable data routing strategy in the cluster aimed to route files containing duplicate data to the same node can ensure a high deduplication rate for the cluster nodes.

Single-node data storage is difficult to meet the needs of large-scale data storage. The deduplication cluster in a distributed environment can manage the huge throughput of massive data storage backup. However, the deduplication rate of each node in the deduplication cluster is much lower than that of a single node. The main reason is that it is difficult to deduplicate data globally in the entire cluster and can only perform deduplication on independent nodes. Therefore, it is important to use appropriate routing strategies to distribute similar or related data to the same nodes as much as possible. This paper studies the multicharacteristic data routing strategy in the deduplication cluster, using the cluster's multicharacteristics to make routing decisions, improving the single-characteristic data routing strategy, and verifying it through experiments. The main research works are shown as follows:

(1) For the problem that the file label dimension is too large, feature dimensionality reduction is performed based on similar hash. Using the node's disk storage utilization as the node's load characteristics: multi-feature-based data routing strategies are used for multifeature fusion. The effects of file characteristics on similarity are compared through experiments to determine the value of each parameter.

(2) According to the multifeature processing results of the cluster, a multifeature data routing strategy is proposed. The routing strategy mainly includes routing communication, file similarity calculation, and routing decision.

(3) The experiment simulated the multicharacteristic data routing strategy and the popular single-characteristic data routing strategy. The experiment showed that the multicharacteristic data routing performed better in cluster deduplication rate and load balancing.

2. Related Work

In single-node deduplication schemes like ALG-Dedupe [10] and ADMAD [11], different types of chunking methods were utilized for files of different application types. In HPDV [12], global shared fingerprint index is divided into subindexes according to the type of operating systems that a virtual machine is hosting. This approach is intended to reduce the scope of fingerprint index search. In AppDedupe [13], the type of files is used to decide where to route its chunks. DRMF creates groups of files based on their application type. Independent index space and its bloom filters are then used for each group throughout all the nodes in the system. Moreover, DRMF uses file size to filter small-sized files and create file segments from these small files. The file segments are used for deduplication instead of the small files. A similar technique of filtering files using their size is utilized in [13] but their intention is only to increase data transfer rate over the network and they completely ignore these small files from the deduplication process. In [11–16], super chunks are created from chunks to minimize disk index-lookup bottleneck and exploit data locality. DRMF also exploits locality and improves disk index-lookup performance by first chunking files into segments using Content-Defined Chunking (CDC) and then further chunking these segments into fine-grained chunks.

In [11, 13, 14, 17, 18], a stateful routing is utilized for routing data to minimize the effect of deduplication node information island. In [11], a set of representative handprints are selected from each super chunk and the similarity of super chunks is decided by comparing representative handprint chunks. In [14], super chunks are routed as a unit. Counting bloom filters are used in every node to trace the number of times a fingerprint is stored in the nodes. This approach can achieve a considerable duplicate removal performance, but it is susceptible to high communication cost, high memory consumption, and computational overhead. In [13], a two-tiered internode routing and an application-aware intranode deduplication are performed to tackle the issue of deduplication node information island effect. In this system, the director-node first selects a group of storage nodes for each file using file type as criteria; the client node then chooses a representative chunk for each super chunk and broadcasts it to all the selected nodes with the node containing a matching chunk chosen as the destination. The communication cost of this system is high because it floods the network with representative chunk fingerprints for every super chunk. DRMF introduces a drastic reduction of communication overhead because multifeature-based data routing strategy is utilized. The strategy represents a whole data stream and is sent only to network for similarity computation.

3. DRMF: Multifeature-Based Data Routing Strategy

3.1. System Architecture. The system architecture of a deduplication cluster has a significant impact on the system's throughput and deduplication performance. By increasing the parallelism between data processing in the system, the system throughput can be effectively improved. At the same time, by performing multifeature preprocessing on the deduplication cluster, the similar or duplicate data is routed to the same node to improve the deduplication rate. The system needs to consider a load of each storage node while ensuring the deduplication rate. The load imbalance of the storage node will make it a performance bottleneck of the entire system, resulting in a decrease in the overall performance of the system. Therefore, the load characteristic

value of the cluster is considered in the cluster feature preprocessing. The system architecture of the deduplication cluster studied in this paper is shown in Figure 1.

Multiple routing servers in the cluster make routing decisions at the same time. The routing server preprocesses multiple features of the cluster and completes routing decisions based on these characteristics. The master node in the cluster collects operational information such as the load of the storage nodes and sends this information to the routing server. This runtime information is the key factor affecting routing decisions. The routing server feeds back the routing information of the file to the master node, and the master node records the target node of the file to facilitate later file recovery. The master node monitors the running status of the storage node. When the load of the storage node exceeds the threshold, the master node marks the storage node as a high load, and the routing server will not use this node as the destination node of the route in the next routing decision.

Each storage node in the cluster has the function of deduplication. The minimum granularity of deduplication is data blocks. In order to improve the efficiency of deduplication, storage nodes use local caches and Bloom Filter data structures. The local cache stores the data recently stored in the node. When duplicate data is detected, the node first searches for duplicate data in the cache area. If duplicate data is found, there is no need to query disk data, which improves data retrieval efficiency. When the cache misses, the node performs data filtering through the Bloom Filter to determine whether the data is new. If the data is determined to be new data, the node directly stores the new data stream to avoid further disk operations. The structure of the storage node is shown in Figure 2.

The storage node uses the content-based segmentation algorithm to divide the routed file data stream into multiple data blocks and calculates unique fingerprint values for each data block. According to these fingerprint values, it is determined whether the routed file data block and the local data are duplicated. If repeated, the data block does not need to be stored; only relevant metadata is saved. Otherwise, the file data block is new data and needs to be stored on disk. The data flow in deduplication meets the principle of locality: in the same backup data flow, if the previous data block is duplicated, then the next data block is also likely to be duplicated [19].

When finding duplicate data blocks, the storage node uses the principle of localities of duplicate data to avoid a large number of index lookups on the disk on the data onto the cache, which greatly reduces the performance loss caused by I/O bottlenecks and improves the system throughput. When the cache misses, the storage node uses the Bloom Filter to determine whether the data is new. The Bloom Filter can determine that data is not duplicated, but it cannot be determined if data is duplicated. The cache hit ratio has a great impact on search efficiency. In order to improve the cache hit ratio when a storage node loads data from disk to the cache, it selects the most recently stored piece of data at the same time, because these data meet the principle of deduplication localities.

3.2. DRMF Data Routing. DRMF (Data Routing Strategy Using Multiple Features) uses cluster multiple features to make routing decisions. Unlike stateless data routing strategies such as MCS (Minimum Chunk Signature based Routing Strategy), DRMF is a stateful data routing strategy. It needs to interact with candidate target nodes and obtain the runtime information of candidate nodes to determine the final target node. The processes required by DRMF to complete file routing mainly include routing communication, file similarity calculation, and routing decision.

3.2.1. Route Communication. To improve the deduplication rate of the cluster, the routing server needs to route similar files to the same storage node. DRMF uses multiple features of files for similarity detection. Therefore, when determining similar files, it needs to pass the multiple features of the file to be routed to the storage node. When the routing destination node is determined, the routing server needs to consider the load of the storage node, and the storage node needs to pass its own load characteristics to the routing server. The routing communication is mainly the transfer of the characteristics of the file to be routed and the storage node load characteristics. Figure 3 shows the specific routing communication.

In Figure 3, during routing communications, the routing server and storage nodes pass related features to each other. The storage node receives multiple features of the file to be routed. Based on the multiple features of the file, it finds the file data that is most similar to the file to be routed and passes the maximum similarity and load characteristics of the storage node to the routing server. After receiving the information returned by the storage node, the routing server fuses the similarity and the load characteristics of the storage node to determine the destination node of the route.

3.2.2. File Similarity Calculation. File similarity calculation is calculating the similarity between the files to be routed and the file data stored by the node. However, calculating the similarity between the files to be routed and each file stored by the node is time consuming and reduces system efficiency. This article stores the similar files in the node in a "box." During the similarity calculation, only the similarity between the files to be routed and the "box" representative file is calculated, and the calculated similarity is regarded as the file to be routed and each in the "box." The similarity of files. In addition, before similarity calculation, very dissimilar "boxes" are excluded to avoid unnecessary calculation overhead. The file similar calculation is designed as given in Algorithm 1.

The amount of data stored by a node can reflect the workload of the node, which is suitable as the storage load characteristic of the node. However, in a deduplication cluster, the hardware resources of each storage node are different, and the disk storage capacity may be different. Therefore, the disk storage usage of a node cannot be used to compare workloads between nodes. This article uses the node's disk storage usage as a feature of node loaded to eliminate the impact of node hardware differences.

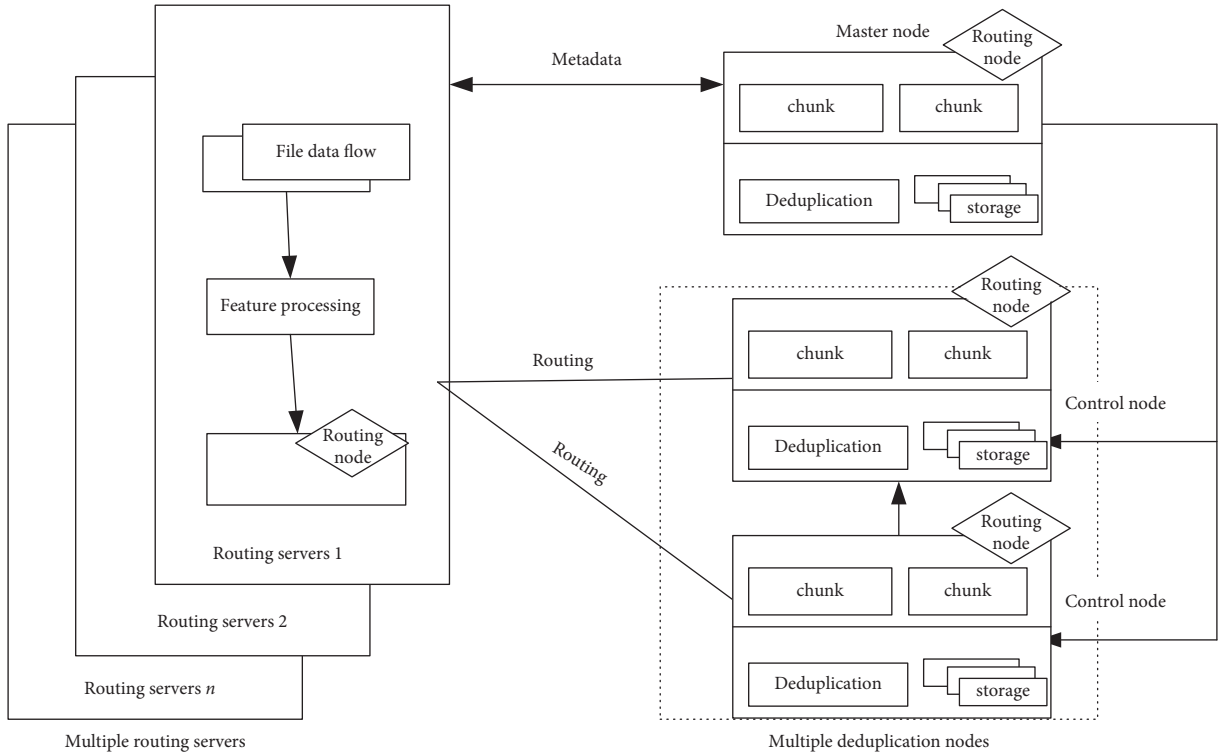

Figure 1: Deduplication cluster system architecture.

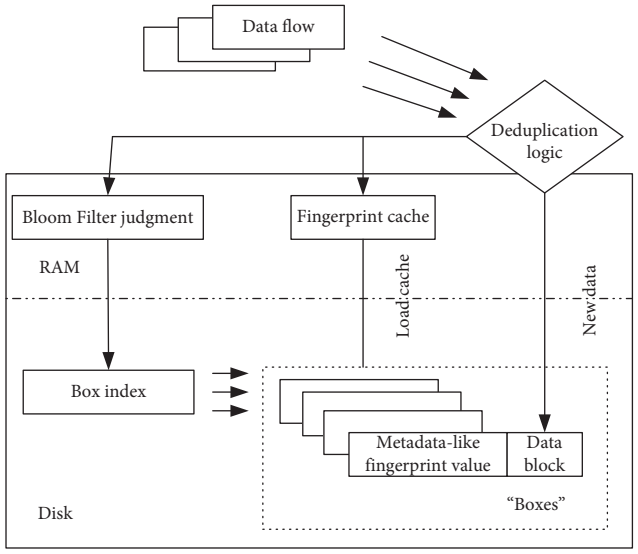

Figure 2: Deduplication storage node structure.

By viewing the runtime information of a storage node, it can obtain the disk storage capacity and current disk usage of the node and then calculate the disk storage utilization of the node by the following formula:

$$\mu = \frac{D_{\text{use}}}{D_{\text{total}}}. \tag{1}$$

In formula (1), μ is node disk storage utilization, D_{use} is the disk storage usage of the node, and D_{total} is the disk storage capacity of the node.

Based on the minimum w data block signatures of the file, there are some limitations of using formula (1) to calculate the file similarity. When the file is small, a slight modification of the file may cause the signatures of the smallest w data blocks to change, and the calculated similarity may be biased. Also, the accuracy of the similarity is related to the size of the data block. The smaller the data block size, the smaller the data granularity calculated by formula (2), and the more accurate the calculated similarity:

$$\text{psim}(A, B) = \frac{\left|\text{MIN}_w(M(A) \cup M(B)) \cap M(A) \cap M(B)\right|}{\left|\text{MIN}_w(M(A) \cup M(B))\right|}. \tag{2}$$

In formula (2), $M(A)$ is the set of the smallest w data block signatures in file A, $M(B)$ is the set of the smallest w data block signatures in file B, MIN_w function is the smallest A set of w elements, and psim(A, B) is the approximate value of the similarity between file A and file B.

Broder and others have shown that psim(A, B) is an unbiased estimate of sim (A, B) [14]. Therefore, this article chooses the smallest w data block signatures as the characteristics of the file. w affects the accuracy of psim(A, B) estimation; it needs to be set according to the size of the set and the actual situation.

This article converts multiple tag characteristics of a file into a hash value, which is a similar hash. The traditional hash value is used to distinguish files. If the contents of file are slightly modified, the file hash will be different. The main difference between similar hashes and traditional hashes is that the more similar the files are, the more similar the hash

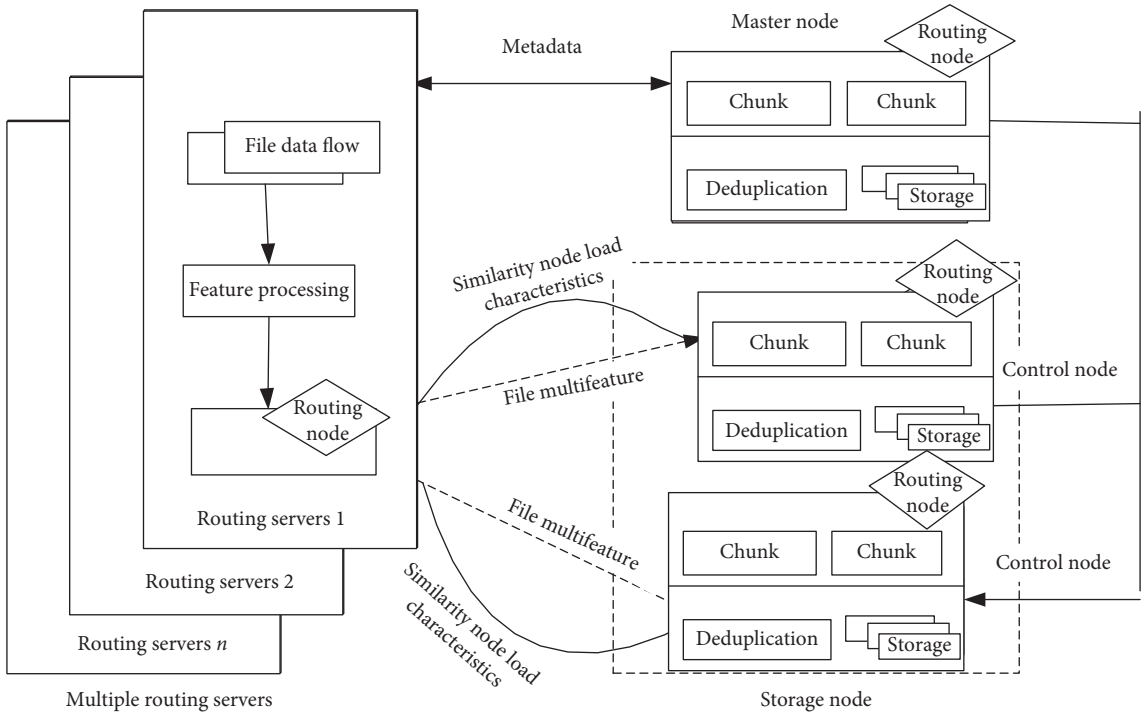

Figure 3: Route communication.

Input: load the "box" indexes of the storage node into memory as the main index of the node. Each index entry of the main index is "boxing" information. Compared to creating an index entry for each file, the main index can reduce the index size.
Output: bsim and the "box" b address.
(1) Reading the file signature chain of the main index. The file signature is the hash value of the entire file. If the same file signature as the file to be routed is found, it means that the file to be routed is a duplicate file with a similarity of 1, and no subsequent operations are required. Only the metadata of the file to be routed is saved. If the same file signature is not found, proceed to the next step.
(2) Comparing the similar hash of the file to be routed with the "box" representing the similar hash of the file. The "box" represents the first file saved to the "box." A similar hash is a binary number. If the number of different corresponding bits of two similar hash reaches η, it means that the two files are very similar, and the similarity of the two is set to 0 directly, and there is no need to calculate the similarity. Otherwise, mark the "box" as "to be compared."
(3) For all "boxes" marked as "to be compared," reading the "box" on the disk at one time represents the file corresponding to the minimum w data block signatures.
(4) Fusing the files with similar hash file and the minimum w data block signatures, calculating the similarity between the "box" representative files and the files to be routed, and obtaining the "box" b and the corresponding similarity bsim that are most similar to the file to be routed.

Algorithm 1: File similarity calculation algorithm.

is. The similarity calculation formula is shown in the following formula:

$$\mathrm{sim}(A, B) = 1 - \frac{\mathrm{diff}(A, B)}{N}. \tag{3}$$

In formula (3), sim(A, B) is the similarity of files A and B, diff(A, B) is the number of different bits corresponding to the similar hash of A and B, and N is the number of bits in a similar hash.

Use the weight function to fuse file data block features and label features to improve the accuracy of file similarity calculation. The weighting function that fuses the features of the file is shown in the following formula:

$$\mathrm{sim}(A, B) = w_1 \bullet \mathrm{sim}_1(A, B) + w_2 \bullet \mathrm{sim}_2(A, B). \tag{4}$$

In formula (4), sim(A, B) is the similarity of files A and B after fusing block features and label features, sim1(A, B) is the file similarity calculated by equation (2), sim2(A, B) is the similarity of the file calculated by equation (3), w_1 is the weight of data block signature, and w_2 is the weight of the label feature.

3.2.3. Target Node Determination. DRMF uses the maximum similarity and the load status of the storage node between the node storage file and the file to be routed to

make routing decisions. Considering only the file similarity will improve the deduplication rate of the cluster as a whole, but it will increase the probability of cluster loaded imbalance and reduce the performance of the cluster. If the routing strategy only makes routing decisions based on the similarity of the files, then if the node storing data is much higher than other nodes, the data is more easily routed to that node, and this causes a node load imbalance and therefore more duplicate data because the node contains more data. The DRMF routing is designed as given in Algorithm 2.

The traditional multifeature-based data routing strategy usually uses the node ε where the file most similar to the file to be routed is the target node. This approach will cause more and more files to be routed to node ε, resulting in node load imbalance. DRMF does not take the similarity between the node and the file to be routed as the sole criterion for routing decisions when making target node decisions and also considers the load characteristics of the nodes. Through the fusion of node load characteristics and similar characteristics of files, DRMF not only ensures a high cluster deduplication rate, but also reduces the probability of cluster load imbalance. DRMF does not necessarily select the node where the file most similar to the file to be routed is the target node, because the node may have an unbalanced load. A specific DRMF routing decision example is shown in Figure 4.

Figure 4 illustrates an example of a routing decision for DRMF. The maximum similarity between the file stored in node 1 and the file to be routed is 0.5, and the maximum similarity corresponding to nodes 2, 3, and 4 is 0.6, 0, and 0.3, respectively. Because node 2 has a larger load, DRMF multiplies its similarity by a factor less than 1 to reduce its routing benefit. Therefore, although the node 2 storage file is the most similar to the file to be routed, DRMF has not made it the final target node. The route benefit value of node 1 is 0.5, which is the maximum route benefit value. DRMF uses node 1 as the destination node of the route.

4. Experimental Evaluation

4.1. System Prototype. The system prototype simulates a deduplication cluster consisting of many nodes. The cluster contains multiple routing servers. Multiple routing servers can distribute data onto the nodes on the entire cluster in parallel. The routing server program executes the data routing algorithm, and the program implements the MCS algorithm and the DRMF algorithm. The data routing algorithm analyses the characteristics of the input file and storage nodes to determine the target node for the final routing of the file. At the same time, the simulated deduplication cluster also contains many storage nodes. Each storage node performs deduplication tasks on its node. Each deduplication storage node in the cluster is no different from deduplication in a stand-alone environment. The storage node stores a master index containing "box" information [20]. When deduplication of files is performed, in order to reduce the performance loss caused by frequent I/O operations, the data block signatures are first filtered in memory by Bloom Filter to determine the new data blocks of the routing file. Then look up the data block in the cache. If the cache hits, it means that the data blocks are duplicating data. If neither the Bloom Filter nor the cache can confirm whether the routing file is new data or duplicate data, it is needed to be retrieved on the local disk of the storage node. The simulator contains a file feature analysis module. The file features analysis module extracts and analyses multiple features of the file data to obtain the minimum w data block signatures of the file and the label features of the file.

4.2. Data Set and Evaluation Indicator Test Environment. The data set of this experiment is a Linux source code kernel archive, which contains a series of versions from 3.0.1 to 4.1.6, with a total of 723 versions of data [21]. The deduplication metrics for subsequent experiments are based on the deduplication effect of a single node. Therefore, the deduplication situation of a single node is listed here. Related information is shown in Table 1.

4.3. Test Results and Analysis

4.3.1. Analysis of DRMF Data Routing Strategy Effectiveness. In this paper, the similarity offset is used to determine the values of w and n, and the evaluation function is used to determine the weight of the data block signature and label when calculating the similarity.

The calculation of the similarity offset is shown in the following formula:

$$\rho = \frac{\left|\text{sim} - \text{sim}_{\text{part}}\right|}{\text{sim}}. \tag{5}$$

In formula (5), ρ is the similar deviation degree, sim_{part} is the file similarity calculated using partial features, sim is the similarity calculated using all eigenvalues, and $|\text{sim} - \text{sim}_{\text{part}}|$ is the absolute value of the difference between sim and sim_{part}.

sim_{part} can be the similarity calculated using the smallest w data block signatures of the file, or the similarity calculated using the key tags; sim can be calculated using the entire data block signature of the file or all tags' similarity.

When the minimum w data block signatures are used as the characteristics of the file, w needs to be determined. When w is small, there are too few features to accurately calculate the similarity of the file; when w is large, although the similarity can be calculated more accurately, it will increase the load of the deduplication cluster. This article randomly selects the files in the 50 versions of the Linux source code archives from the data set as the training set. The file blocker divides the file data in training set; the data block signature calculator calculates the signature of each data block, obtains the data block signature set of each training file, and selects the smallest w data block signatures as training characteristics of the file. For files in training set, set w to 1/2560, 1/1280, 1/640, 1/320, 1/160, 1/80, 1/40, and 1/20 of the total number of data blocks in the file, 1/10, 1/2, calculate the

Input: when DRMF routing, avoid routing file data to storage nodes with high load. Before routing, the master node sends the disk storage utilization of each node to the routing server, and the routing server calculates the average disk storage utilization of the node μ.
Output: *Indexid*.

(1) Candidate nodes are determined. When the disk storage utilization of a node does not exceed a specific threshold σ, the node is used as a candidate target node. Σ is usually set to 0.05; that is, when the node disk storage utilization does not reach 1.05 of μ, this node is used as a candidate node.
(2) Data routing in the initial state of the cluster. In the initial state, no storage node in the deduplication cluster stores any data. At this time, MCS is used for routing decisions.
(3) Multifeature-based data routing. The routing server performs routing communication with multiple candidate nodes to obtain the maximum similarity between the candidate node storage file and the file to be routed, the disk storage utilization of the candidate node, and the most similar "box" address of the node. According to the similarity, node load characteristics are fused based on the benefit function, and the route benefit value of each candidate node is calculated. The node corresponding to the maximum benefit value is selected as the final target node.
(4) File storage. Route the file to the target node. If there is a specific "box" on the node, the "box" stores the file the most similar to the routing file; then the routing file is stored in the "box." Otherwise, create a new "box," save the routing file in the "box," use the routing file as the representative file of the "box," and add an index entry to the main index.

Algorithm 2: The DRMF routing algorithm.

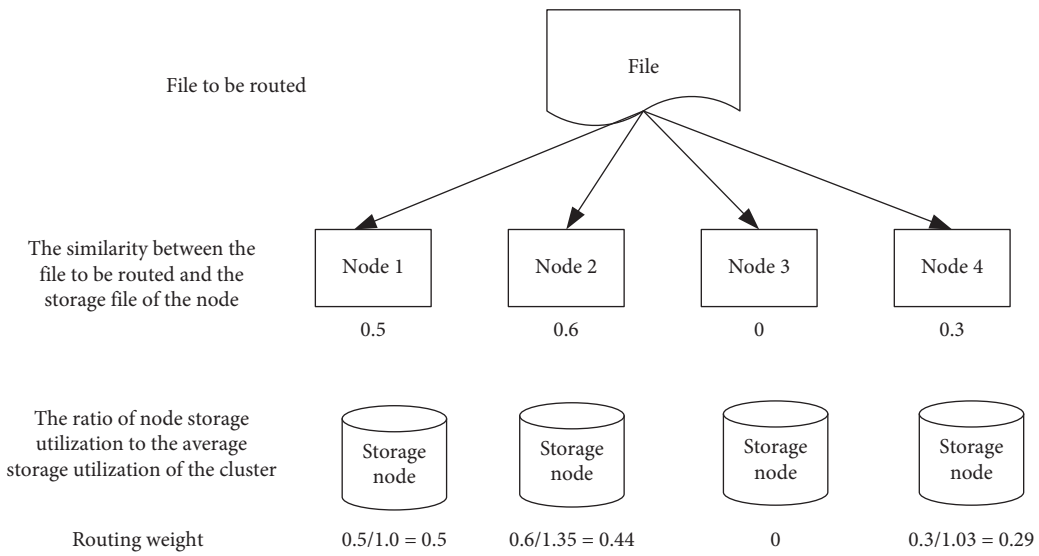

Figure 4: Example of DRMF routing decision.

Table 1: Data set and related information.

Data set	Number of files	Data size (GB)	Single-node deduplication rate
Linux source code	3186361	71.678	1.3296

corresponding file similarity offset. The similarity in the similarity offset between training files has a similar trend. Figure 5 shows the similarity between different versions of memory-barriers files in training set as w changes.

It can be seen from Figure 5 that when the ratio of w to the total number of file data blocks is greater than 1/320, the similarity offset is small. Moreover, the degree of similarity shift is relatively stable, and the change is not obvious. When the ratio is less than 1/320, the degree of similarity shift is large, and the variation range is large and unstable. The value of w selected in this paper is 1/20 of the total number of data blocks. At this time, the similarity offset is small, and the number of data blocks that needs to be transmitted for routing communication is small, which will not increase the load on the system.

When n tags with higher frequencies are used as features of the file, n needs to be determined. When n is small, there are too few tags to accurately calculate the similarity of the file; when n is large, the complexity of the similarity calculation increases. In this paper, the files in 50 versions of Linux archives are selected as the training set. The feature label extraction module analyses the files

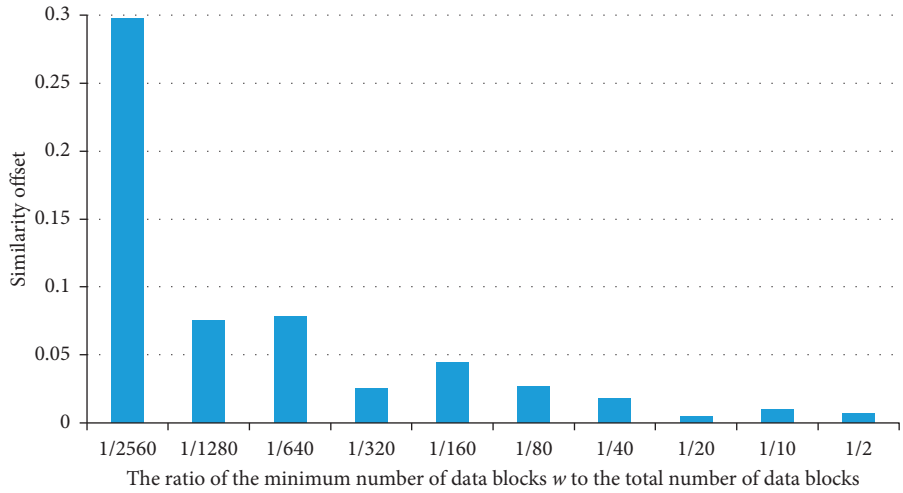

Figure 5: Similarity offset caused by minimum w data block signatures.

in training set, extracts the label features of the file, calculates the weight of each label of the file, and selects the n labels that appear most frequently Feature as training files. Set n to 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, and 200 in turn. Use a similar hash to calculate the similarity between files. The similarity varies with n. The similarity between files varies with the number of selected high-frequency tags. Figure 6 shows the similarity of different versions of memory-barriers files with the number of high-frequency words.

It can be seen from Figure 6 that when n is changed, the similarity between files is constantly changing. The maximum value of similarity is 0.8593, and the minimum value is 0.7813. The difference between them is 0.078, and the difference is small. In this paper, n is set to 20 by default.

The experimental results show that when w is 1/20 of the number of signatures of the entire data block, the similarity offset is small and the dimensions are small; when 20 key tags are selected, the similarity calculation result is more accurate; the file has multiple features. In fusion, the effect is better when the weights of the data block signature and file label are 0.58 and 0.42, respectively.

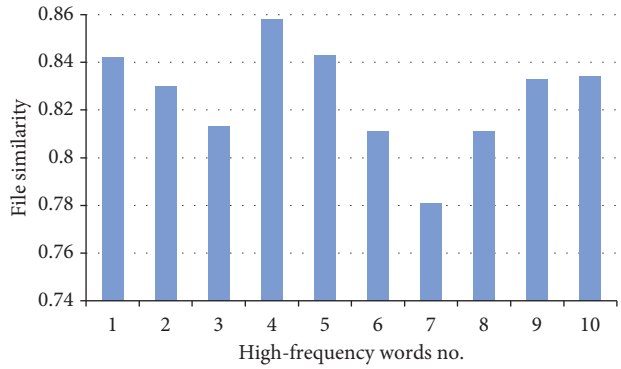

Figure 6: The effect of the number of high-frequency words on similarity.

4.3.2. Performance. The metrics involved in the experiment are as follows:

(1) Deduplication Rate (DR): the ratio of the size of the original data to the size of the data after deduplication.

(2) Data Skew (DS): the ratio of the largest node storage usage in the cluster to the average node storage usage. DS is used to measure the load balancing status of the cluster. Through the DS value, one can observe the impact of routing policies on cluster load balancing.

(3) Normalised Deduplication (ND): the ratio of the deduplication rate of the entire cluster after deduplication to the deduplication rate in a single-node environment. ND is used to measure the gap between the deduplication effect achieved by the routing strategy and the optimal situation.

The amount of data stored in the backup system is large. The amount of storage space used is one of the important factors determining the cost of the system. The deduplication rate is an important indicator of the storage space efficiency of the backup system. The routing policy distributes the files across the nodes of the cluster, and there is duplicate data between the storage nodes. Therefore, the deduplication rate of the system in a multinode environment is lower than the deduplication rate of a single node. This article uses ND to measure the impact of routing policies on the deduplication rate. The larger the ND value, the closer the deduplication effect of routing policy is to a single node, the better the deduplication effected is. The optimal ND value is 1. At this time, the cluster deduplication rate is the same as the value in the single-node environment.

This article simulates MCS routing strategy and DRMF routing strategy in the simulator and uses the deduplication tool to perform deduplication rate statistics. The number of nodes in the deduplication cluster is set to 1, 2, 4, 8, 16, 32, and 64, respectively. Among them, the experiment with a node number of 1 detects the

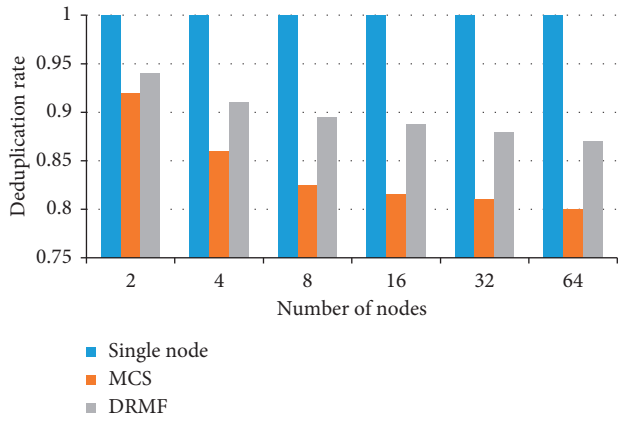

Figure 7: Normalised deduplication rates for different strategies.

deduplication rate in a single-node environment, which does not involve data routing. The deduplication rate is the maximum value that the cluster can achieve in the same environment, which is called "optimal deduplication Rate"; as shown in Table 1, the deduplication rate is 1.3296, and 17.7 GB of duplicate data is deleted. Figure 7 shows the normalised deduplication rates of clusters of different routing strategies.

Figure 7 uses a normalised deduplication rate to measure the impact of a routing strategy on the deduplication rate on the cluster. Compared to the deduplication rate, the normalised deduplication rate more intuitively reflects the routing strategy compared to a single node on the cluster deduplication. Remove performance impact. As shown in Figure 7, when the number of cluster nodes increases, the deduplication rate of MCS decreases rapidly. When the number of nodes is large, MCS is not suitable as a data routing strategy. DRMF's deduplication rates decrease slowly with the number of nodes and maintain a high value, indicating that DRMF can better route similar files to the same node.

In the above experimental environment, this article simulates the effects of MCS and DRMF on the cluster load. The experimental results are shown in Figure 8.

In this article, the threshold of load imbalance is set to 0.05. When the data slope of the cluster is greater than 1.05, the cluster will have an unbalanced load. As can be seen from Figure 8 as the number of nodes increases, the data slope of both the MCS routing strategy and the DRMF routing strategy becomes larger, and the increasing rate becomes largely high. When the number of nodes reaches 8, MCS has a load imbalance. However, DRMF remains to be load balanced. Therefore, DRMF can improve the deduplication rate while reducing the probability of load imbalance.

This article proposes a clustered multifeature stateful data routing strategy DRMF algorithm. DRMF not only considers the similarity in files, but also considers the load characteristics of nodes to make routing decisions. The experimental results show that compared with the single-feature-based routing strategy MCS, DRMF can improve the deduplication rate of the cluster and maintain a lower data skew rate.

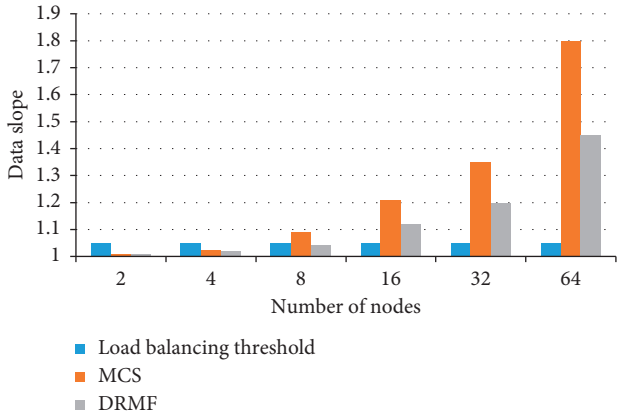

Figure 8: Algorithm data slope.

5. Conclusions

This paper focuses on the data routing strategy in clusters. According to the multiple characteristics of the cluster, a DRMF routing strategy is proposed. DRMF makes routing decisions based on multiple features of the cluster. It not only considers the similarity between the node storage file and the file to be routed, but also takes into account the load balancing of the storage node. Experimental results show that, compared with the traditional single-feature data routing strategy MCS, DRMF can improve the deduplication rate of the cluster and reduce the probability of cluster load imbalance.

Data routing strategy is a research hotspot and difficulty in deduplication clusters. In the past few years, academia has studied the impact of data routing strategies on deduplication rates and cluster load balancing and explored the performance bottlenecks of data routing strategies. However, there are still many problems that have not been completely solved. Research work on multicharacteristic data routing strategies will be carried out in the following areas:

(1) Research on stateless data routing based on multiple features: multifeature-based data routing strategy DRMF can take into account the cluster's deduplication rate and load balancing, but it is a stateful routing

strategy and has a large system overhead. In future research, a stateless data routing strategy based on multiple features may be considered to reduce the system overhead while guaranteeing a certain deduplication rate.

(2) Diverse selection of features: the multifeatures selected in this paper are content-based file data block signatures, file labels, and disk storage utilization of nodes. These features can be combined to better guide data routing. However, there are many other cluster multifeatures that can be used to extract, merge, and route decisions from other categories of features.

Data Availability

The experimental data set used is the source code of Linux from 3.0.1 to 4.1.6, with a total of 723 versions. Data source address is https://www.kernel.org/.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61872284), Natural Science Research in Shaanxi (2020GY-012), Special Scientific Research Project of Shaanxi Provincial Department of Education (18JK0466), "Thirteenth Five-Year" National Key R&D Program Project (2019YFD1100901), Talent Fund Project of Xi'an University of Architecture and Technology (RC1707), Youth Fund Project of Xi'an University of Architecture and Technology (QN1726), and Natural Science Project of Xi'an University of Architecture and Technology (ZR18050).

References

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

[2] D. Harnik, M. Hershcovitch, Y. Shatsky, A. Epstein, and R. Kat, "Sketching volume capacities in deduplicated storage," *ACM Transactions on Storage (TOS)*, vol. 15, no. 4, pp. 1–23, 2019.

[3] A. Duggal, F. Jenkins, P. Shilane, R. Chinthekindi, R. Shah, and M. Kamat, "Data domain cloud tier: backup here, backup there, deduplicated everywhere! 2019," *USENIX Annual Technical Conference (ATC)*, vol. 19, pp. 647–660, 2019.

[4] D. Harnik, E. Khaitzin, and D. Sotnikov, "Estimating unseen deduplication—from theory to practice," in *Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST)*, pp. 277–290, Santa Clara, CA, USA, February 2016.

[5] K. Matsuzawa, M. Hayasaka, and T. Shinagawa, "The quick migration of file servers," in *Proceedings of the 11th ACM International Systems and Storage Conference*, pp. 65–75, Haifa, Israel., January 2018.

[6] N. Xia, C. Tian, Y. Luo, H. Liu, and X. Wang, "UKSM: swift memory deduplication via hierarchical and adaptive memory region distilling," in *Proceedings of the 16th USENIX Conference on File and Storage Technologies (FAST)*, pp. 325–340, San Jose, CA, USA, September 2018.

[7] 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 *Proceedings of the 16th USENIX Conference on File and Storage Technologies (FAST)*, pp. 309–324, San Jose, CA, USA, September 2018.

[8] 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 *Proceedings of the 2018 USENIX Annual Technical Conference (ATC)*, pp. 705–718, Boston, MA, USA, November 2018.

[9] F. Chen, T. Luo, and X. Zhang, "CAFTL: a content-aware flash translation layer enhancing the lifespan of flash memory based solid state drives," in *Proceedings of the 9th USENIX Conference on File and Stroage Technologies (FAST)*, pp. 77–90, San Jos, CA, USA, February 2011.

[10] Y. Fu, H. Jiang, N. Xiao, L. Tian, F. Liu, and L. Xu, "Application-aware local-global source deduplication for cloud backup services of personal storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 5, pp. 1155–1165, 2014.

[11] J. Wang, Z. Zhao, Z. Xu, H. Zhang, L. Li, and Y. Guo, "I-sieve: an inline high performance deduplication system used in cloud storage," *Tsinghua Science and Technology*, vol. 20, pp. 17–27, 2015.

[12] C. Lin, Q. Cao, J. Huang, J. Yao, X. Li, and C. Xie, "HPDV: a highly parallel deduplication cluster for virtual machine images," in *Proceedings of the 18th IEEE/ACM International Symposium on Cluster , Cloud and Grid Computing*, pp. 472–481, Washington DC, USA, May 2018.

[13] Y. Fu, N. Xiao, H. Jiang, G. Hu, and W. Chen, "Application-aware big data deduplication in cloud environment," *IEEE Transactions on Cloud Computing*, vol. 1, 2017.

[14] F. Douglis, A. Duggal, P. Shilane, T. Wong, S. Yan, and F. Botelho, "The logic of physical garbage collection in deduplicating storage," in *Proceedings of the 15th USENIX Conference on File and Storage Technologies (FAST)*, pp. 29–44, Santa Clara, CA, USA, March 2017.

[15] P. Zhang, P. Huang, X. He, H. Wang, L. Yan, and K. Zhou, "RMD: a resemblance and mergence based approach for high performance deduplication," in *Proceedings of the 45th International Conference on Parallel Processing*, pp. 536–541, Philadelphia, PA, USA, August 2016.

[16] W. Xia, H. Jiang, D. Feng, and L. Tian, "DARE: a deduplication-aware resemblance detection and elimination scheme for data reduction with low overheads," *IEEE Transactions on Computers*, vol. 65, no. 6, pp. 1692–1705, 2016.

[17] P. Zhang, X. P. Huang, and K. H. ZhouWang, "Resemblance and mergence based indexing for high performance data deduplication," *Journal of Systems and Software*, vol. 128, pp. 11–24, 2017.

[18] L. Aronovich, D. R. Asher, and S. T. M. KleinHirsch, "Similarity based deduplication with small data chunks," *Discrete Applied Mathematics*, vol. 212, pp. 10–22, 2016.

[19] F. Douglis, D. Bhardwaj, H. Qian, and P. Shilane, "Content-aware load balancing for distributed backup," in *Proceedings of the 25th International Conference on Large Installation System Administration (LISA)*, pp. 1–18, San Jose, CA, USA, December 2011.

[20] C. Wang, J. Wang, X. Sun, and F. Wang, "A novel soil nutrient classification method based on hadoop platform," *Revue d'Intelligence Artificielle*, vol. 32, no. 1, pp. 25–40, 2018.

[21] Y. Li, "Design and implementation of intelligent travel recommendation system based on internet of things," *Ingénierie des systèmes d'information*, vol. 23, no. 5, pp. 159–173, 2018.