IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 12 March 2024, accepted 26 March 2024, date of publication 29 March 2024, date of current version 9 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3382987

RESEARCH ARTICLE

HSM: A Hybrid Storage Method Based on the Heat of Data and Global Disk Space Utilization

YING SONG[1,2,3], WENXUAN ZHAO[1,2], YINGAI TIAN[1], AND BO WANG[4]

[1]Beijing Information Science and Technology University, Beijing 100101, China
[2]Beijing Advanced Innovation Center for Materials Genome Engineering, Beijing Information Science and Technology University, Beijing 100101, China
[3]State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100086, China
[4]Software Engineering College, Zhengzhou University of Light Industry, Zhengzhou 450002, China

Corresponding author: Ying Song (songying@bistu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872043; in part by the State Key Laboratory of Computer Architecture [Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS)] under Grant CARCHA202103; and in part by the Key Scientific and Technological Projects of Henan Province under Grant 232102211084.

ABSTRACT In distributed systems, the method for data storage is crucial. Previous data storage work use the replication or Erasure Coding method to store data. Such single storage method leads to the excessive storage overheads for cold data with low access frequency or the low reading performance for hot data with high access frequency. Nowadays, the research on the hybrid storage has become a hot topic of concern for many scholars. Existing hybrid storage works take into account data reading performance and the storage overheads, and use the replication and Erasure Coding methods to store the hot data and cold data respectively. However, in the scenarios of sufficient disk space or low disk space, these fixed data storage methods will lead to the relatively low system data reading performance or the excessively low disk space of the system. In this paper, we propose HSM, a hybrid storage method based on the heat of data and global disk space utilization. HSM fully considers the system's requirements for the data reading performance and storage overheads under different global disk space utilization scenarios, and adaptively selects appropriate storage methods for data whose heat is different through data deletion, data reconstruction, and data archiving. The experiment results show that when system disk space is sufficient, HSM reduces data reading time by up to 18%; when system disk space is low, although increasing storage overhead by up to 7%, HSM reduces cross-rack data transfer traffic by up to 20% and cross-rack data transfer time by up to 15% compared with ERP in the process of changing the storage methods.

INDEX TERMS Distributed storage system, hybrid storage, erasure coding, replication.

I. INTRODUCTION

Now we live in a big data age [1]. With the widespread use of distributed systems and the development of information technology, the big data technology is increasingly widely used in various industries [2]. On the other hand, the explosive growth of data volume has also put tremendous pressure on storage systems [3]. Large-scale distributed clusters, such as Google File System [4] and Hadoop [5], are usually composed of many independent low-reliability commercial components, and unexpected failures of components are common. How to improve reliability, availability and reading performance of large-scale data, reduce the storage overheads of data are major challenges for distributed systems in the big data scenario. A typical way is to use three-way replication redundancy technology, which provides fault tolerance by storing multiple replicas of the same data on different nodes. When the amount of data is small, the replication technique is simple to implement. However, in large data centers, Large-scale data makes storing multiple replicas a quite expensive solution, which requires twice the storage overheads.

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

Therefore, as an alternative, Erasure Coding has been introduced by many distributed storage systems, like Ceph [6], Azure [7] and Hadoop Distributed File System [8]. Among multiple Erasure Coding families, Reed-Solomon Code is the most widely utilized coding schemes, providing fault tolerance closer to that of replication technologies but with lower storage requirements. An RS code usually has two parameters, n and m, which represent that an RS code encodes n into m parity chunks, so that n data chunks and m parity chunks form a stripe. Although Erasure Coding can reduce storage costs than replication, they significantly increase disk I/O and network bandwidth occupation when systems reads popular unavailable data than replication.

At present, in a distributed system, a single data storage method has been difficult to meet the storage and reading requirements of data with different heat, and may aggravate the disk space tension of the system. In order to improve the storage efficiency and reading performance, existing works design new hybrid storage methods [9], [10], [11], [12], [13], [14]. For example, EC Fusion [11] is a hybrid storage of the RS Code and the minimum-storage regenerating Code, MSR Code. It dynamically selects the appropriate code based on the workload of the application. For write-intensive workloads, RS code is used to reduce computational overheads and storage costs. For read-intensive application workload, using MSR code can improve the recovery efficiency. Similar to the above the hybrid storage methods, EC Fusion does not fully considering the heat of data, which may use the code with the low parallel reading efficiency to store high access frequency hot data, leading to poor parallel data reading performance. Some works begin to consider this reality. For example, the encoding-oriented replica placement policy [15], ERP, cold data with low access frequency is archived and stored from replication to RS code, which saves the storage overheads of cold data with low access frequency and maintains high parallel reading efficiency of the hot data which is still stored in replication. However, when disk space is abundant, these fixed data storage methods may still use Erasure Coding to store data, reducing the efficiency of parallel data reading. In the other hand, when disk space is tight, they may continue to use multiple replicas to store data, exacerbating the problem of low disk space.

In order to solve the problem of low parallel data reading performance and low disk space, in this paper, we propose HSM, a hybrid storage method based on the heat and global disk space utilization. Under different global disk space utilization scenarios, HSM weighs the requirements of the system for data reading performance and storage overheads, and adaptively selects appropriate storage methods for data with different heat through data deletion, data reconstruction or data archiving, so as to increase the parallel reading performance of hot data, reduce the storage overheads of cold data and cross-rack data transmission time and traffic in the process of changing data storage methods.

Our contributions are summarized as follows:

- We present HSM, a hybrid storage method based on the heat of data and global disk space utilization. It can adaptively select appropriate storage methods for data with different heat under the system with different global disk space utilization, greatly improve parallel data reading performance and reduce the storage overheads for system with low disk space.
- We conduct a group of experiments to evaluate HSM. The results show that compared with ERP and RS Codes, when system disk space is sufficient, HSM reduces data reading time by up to 18%. When system disk space is low although increasing storage overhead by up to 7%, HSM reduces cross-rack data transfer traffic by up to 20% and cross-rack data transfer time by up to 15% in the process of changing the storage methods.

The rest of this paper is summarized as follows. In Section II, the background and some preliminary work are introduced. Section III introduces the latest works. Section IV describes the design of HSM in detail. Section V evaluates the performance of HSM. Section VI summarizes this work.

II. BACKGROUND

In this section, we briefly review the basic properties of replication and the Erasure Coding, which are the basis of our storage method.

A. REPLICATION

Replication is one of the first fault-tolerant methods used to tolerate hard disk failures. It can copy one data $u + 1$ times and store the data to $u + 1$ different storage node, so as to tolerate data failures caused by up to u storage node failures. When a storage node fails, the complete data can be obtained directly from other nodes where the data replication is stored. Replication storage is simple to implement, and effectively ensure the excellent efficiency of reading data. But the replication storage may bring the additional u times space overheads, reduce the utilization of the system resources and increase power consumption.

B. ERASURE CODING

This paper focuses on a famous Reed-Solomon (RS) code. Compared with replication storage, Erasure Coding storage provides similar fault tolerance, but the storage cost is greatly reduced. An RS code usually has two parameters, n and m, which represent that an RS code encodes n normal chunks into m parity chunks, so that n normal chunks and m parity chunks form a stripe. Any remaining n chunks can be decoded to repair the original chunk, so that at most m data chunk failures can be tolerated. Figure 1 presents the details of the matrix coding process of a RS(5, 3) code. Here, data chunks and parity chunks are organized into fixed-size units called chunks.

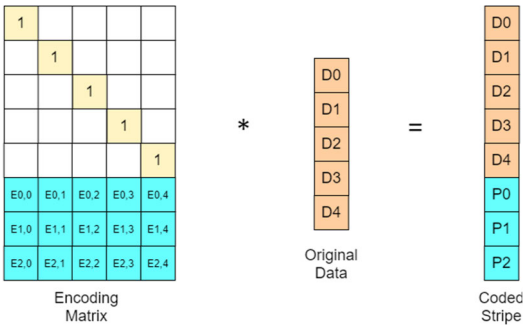


FIGURE 1. Encoding process of an RS (5, 3) code.

III. RELATED WORK

Now there are many works on adaptive hybrid storage methods, including hybrid storage methods based on multiple types or the same type of Erasure Coding and hybrid storage methods based on Erasure Coding and replication. Next, we briefly review the latest scientific research achievements of hybrid storage as follows.

To keep high storage efficiency and reliability with low recovery costs, Xie et al. proposed a encoding and decoding algorithm, AZ Code [12]. Based on the MSR code and LRC codes, AZ Code utilizes a specific MSR code to generate a local parity check chunk, and utilizes a typical RS code to generate a global parity check chunk. However, the data redundancy is slightly larger. In order to reduce the high repair bandwidth and storage overheads of stable nodes, Jin-Ping proposed a LRC-RS [16] hybrid coding method. LRC-RS uses the LRC coding to store data in high trusted nodes, and uses the RC coding with low code rate to store data in low trusted nodes by classifying nodes according to confidence level. But it also adds some of the complexity of encoding and decoding.

Zhang et al. proposed a hybrid cloud-scale chunk storage system called U_{RSA} [17]. U_{RSA} directly stores primary replicas on SSDs and replicates backup replicas on hard disk drives (HDDs). To make up for the performance gap for random writes between SSDs and HDDs, Yiming Zhang et al. design an adaptive journal, through transforming small backup writes into journal appends, which are then asynchronously replayed and merged to backup HDDs. Then, in order to efficiently index the journal, they design a novel range-optimized merge-tree structure. U_{RSA} have slightly higher data redundancy because they do not adequately consider erasure coding methods.

EC-Fusion [11] proposed by Qiu et al. is an efficient hybrid erasure code storage framework in cloud storage systems, which combines RS and MSR codes. According to the application workloads, EC-Fusion dynamically selects the appropriate code. For write-intensive workloads, RS code is used to reduce computational overhead and storage costs. For read-intensive application workload, using MSR code can improve the recovery efficiency, which make it possible for them to solve the problem that front and back office workloads can simultaneously and efficiently process in parallel.

Wei et al. proposed a hybrid fault-tolerance scheme based on potential replicas, called HFPR [18], for solving the problem that mobile distributed systems contend with inconsistent network signals and relatively low bandwidth in weak network environments. In order to disperse network transmission pressure and reduce network transmission, HFPR stores data with the Erasure coding redundancy and then gradually increases data redundancy by reserving potential replicas. However, HFPR is similar to U_{RSA}.

To save the storage overheads of cold data and maintain high parallel reading efficiency for hot data with high access frequency, Xu et al. proposed an encoding-oriented replica storage method, ERP [15]. ERP archives the cold data which makes the storage method of the cold data change from replication to Erasure Coding. At the same time, the hot data is still stored by the replication. In the process of data archiving, for the help of their chunk placement method, ERP reduces the cross-rack data transmission time and traffic.

In the above existing hybrid storage methods, whether based on the hybrid storage methods of Erasure Coding and replication, or the internal hybrid coding of Erasure Coding, or the hybrid storage methods of multiple Erasure Coding, they failed to fully consider the heat of data and global disk space utilization of the system. This makes the data with low access frequency adopt the storage methods with high storage redundancy [19], [20], resulting in the waste of storage resource, or the data with high access frequency use the inappropriate storage methods, resulting in poor parallel read performance [21].

IV. THE DESIGN OF HSM

For improving the storage efficiency and parallel read ability of distributed systems, we first analyze the distribution of the access frequency in this section. Then, we define the expression of heat to better analyze the differences of the access frequency on data in the system. Next, we introduce the design concept of HSM. Finally, we introduce the algorithm implementation of HSM in detail.

A. HEAT CALCULATION AND CLASSIFICATION

Zipf law is an experimental law proposed by G.K.Zipf, and it is used to describe the law of word frequency distribution. According to Zipf law, in a particular natural language corpus, the frequency of words is inversely proportional to its ranking in the frequency table, i. e., the product is a constant. The available formula represents that:

$$f * r = C \tag{1}$$

This f represents the frequency of word occurrence, and r represents the frequency ranking. The product of the two is approximately a constant C, but the constant C is not a fixed constant and floats up and down around a central number. This means that a few words appear very frequently, while most words appear relatively infrequently.

According to the Zipf law, the top few words in the corpus appear much more frequently than other words. For example,

the first word is about twice as frequent as the second word, while the second word is about twice as frequent as the fourth word. This distribution is also vividly known as the two-eight law, that 20% of the words in the corpus account for 80% of the occurrence frequency, and there is a serious polarization.

Zipf law not only applies to the field of linguistics, but also similar frequency distribution laws can be observed in other fields. For example, in distributed storage systems, the access frequency of clusters' files also follows this priori rule [22]. These files are divided into several chunks during the storage and stored in distributed systems in the form of chunks. Therefore, the data mentioned in this paper refers to chunks of the data, including the heat of data, which also refers to the heat of chunks. Since the access frequency of files is different and each file contains chunks which have the same access characteristics as the file, the chunks also have different access characteristics. According to this rule, most of the data access is concentrated on a few popular chunks, while most chunks have relatively low access frequency. Typically, about 80% of the access is concentrated on 20% of the data in the entire data set. Therefore, based on this access rule, we can define the heat of the data in the cluster. Define the top 20% of the most popular data as hot data, and the remaining 80% of the data as cold data. Hot data occupy most of the data access relative to cold data. In order to speed up reading performance in distributed systems, HSM selects the storage method with high parallel data reading performance for hot data.

B. DESIGN CONCEPT

In order to solve the above-mentioned problems of low data reading performance and low disk space, this section first introduces the design of HSM for the data heat, then analyzes the design of HSM for the global disk space utilization, and finally introduces a data layout which reduces the cross-rack transmission traffic when the storage method of data need to be changed.

1) THE DESIGN FOR THE HEAT

In the replication method scenario, when a large number of read requests are made on the same chunk, the requests can be directed to multiple replica nodes to increase the parallel read efficiency and thus alleviate hot-spot issues. When applying an access data chunk, we can select the nearest replica for it to reduce the data transfer time. In the case of the Erasure Coding, when data loss occurs, data reading will bring additional decoding calculation overheads. Therefore, for the parallel data reading performance, the replication storage method is often better than the Erasure Coding storage method. On the other hand, compared to the replication, the Erasure Coding has less storage cost because it only needs to encode a few chunks to ensure its reliability. In a distributed system, only a few hot data can be frequently accessed in a short time, while other cold data may have only a small amount of access in a long time. If we only use the common Erasure Coding method to store data, it may affect the parallel reading performance of the hot data. On the contrary, if only the replication is used to store the data, this may greatly increase the storage overheads of the cold data. Therefore, we use the replication storage method to store the hot data, which can improve the parallel reading performance of the hot data, while the cold data only needs to be stored at a lower storage cost to reduce data storage overheads. At the same time, the heat of data will change. When the heat of the data decreases or increases, the data storage method should also be able to change with low overheads for the system read performance and data storage overheads.

2) THE DESIGN FOR THE GLOBAL DISK SPACE UTILIZATION

When the global disk space utilization of the distributed system is low and the system has sufficient storage space, we give priority to data reading performance. When the global space utilization is moderate, we weigh read performance and storage space utilization, giving priority to improving the read performance of the hot data and reducing the storage overheads of the cold data. When the global space utilization is high, we give priority to alleviating storage space constraints. By archiving cold data, the data stored in the replication is converted into the Erasure Coding method to reduce storage overheads.

3) THE DESIGN IN THE LAYOUT OF DATA

When the heat of data changes from hot to cold or vice versa and the global disk space utilization changes in intervals, the data storage method would change accordingly. In the process of changing data storage methods, operations such as data archiving and data reconstruction will generate intra-rack and cross-rack data transmission traffic. Since cross-rack traffic is more scarce than intra-rack traffic [3], we focus on cross-rack traffic. In order to reduce the cross-rack transmission traffic required to change the data storage method, we adopt a specific data layout method to store the chunks, which makes the chunks that may need to be encoded to store into a main rack during the data archiving process, thus avoiding the situation that transferring chunks to this main rack cross racks and then encoding data. At the same time, in order to reduce the cross-rack traffic generated by subsequent data recovery, the Erasure Coding layout after data archiving will maximize the advantages of partial decoding, which is beneficial to subsequent aggregate recovery.

C. ALGORITHM

We divide the rack into a main rack and some secondary racks, as shown in Figure 2. First of all, considering that the main rack subsequently performs operations such as data encoding and data reconstruction, and these operations require network bandwidth, so the rack with the best network bandwidth performance is selected as the main rack, and the 1^{st} replica of the data is stored in the main rack. Then, the 2^{nd} and 3^{rd} replica chunks of the data are stored together on each secondary rack in the order of the replica numbers.

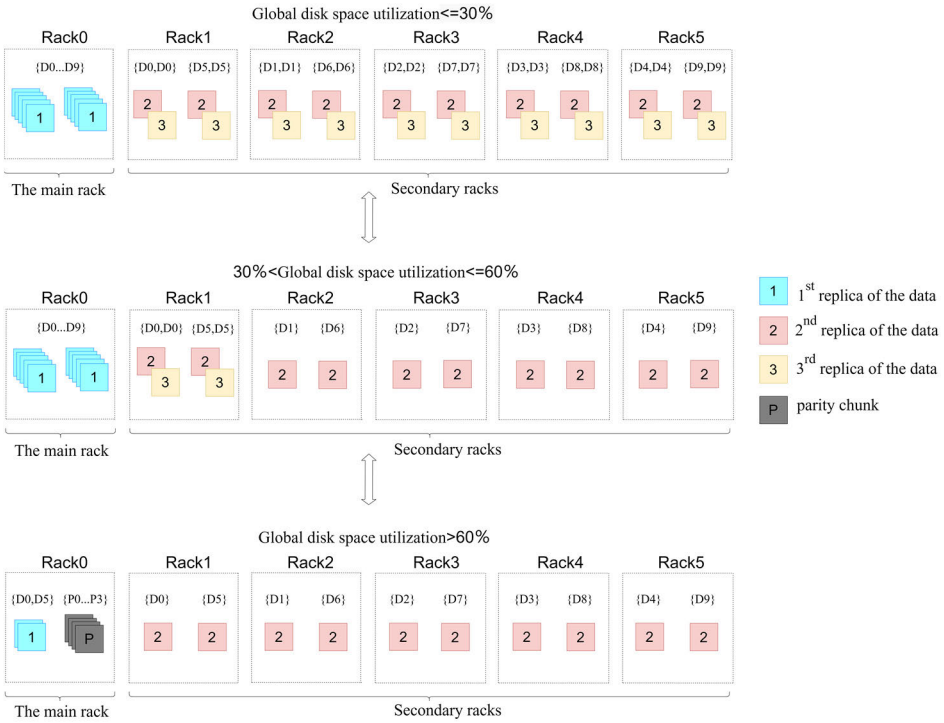


FIGURE 2. Examples of the data storage with HSM.

In order to maintain node-level fault tolerance, the 2nd and 3rd replica chunks of the same data are stored in the same secondary rack on different nodes. In the process of storing replicas on the main rack and the secondary rack, in order to balance the storage load, the replicas are preferentially stored on data nodes with low storage load in the rack.

We define data popularity as the number of times data is accessed by users per unit time. The available formula is simply expressed as

$$H = \frac{N}{T} \tag{2}$$

Among them, H represents the popularity value of the data, N represents the number of accesses of the data, and T represents the time. According to Zipf law, 20% of the data has 80% of the visits, so the data with the top 20% of user visits are defined as hot data. Define the data with the bottom 80% of user access times as cold data.

In the process of data storage, the global disk space utilization and the popularity of the chunks will change. At this time, we need to weigh the system's requirements for parallel data reading performance and storage overheads under different global disk space utilization scenarios, and adaptively choose data storage methods.

As shown in Algorithm 1, when the global disk space utilization is less than or equal to 30%, the system storage space is sufficient. Between data storage overheads and parallel reading performance, we give priority to parallel data reading performance. At this time, whether it is hot data or cold data, we all use the three-way replication method with good parallel data reading performance to store the data.

In terms of experimental conclusion that new data is often hot data [23], when storing new data, the new data stored in the three-way replication method.

When the amount of data stored in the system begins to increase and the global disk space utilization is greater than 30% and less than or equal to 60%, the storage space is no longer sufficient. Therefore, while ensuring that the system has high data reading performance, we must also consider reducing the data storage overheads. Firstly, we rank the popularity of data and define hot data and cold data. Because hot data has a very high frequency of data access compared to cold data. In order to achieve high read performance of hot data, we continue to use the three-way replication method to store hot data. On the other hand, in order to save storage overheads, we use the two-way replication method to store cold data. In the process of changing the storage method from the three-way replication method to the two-way replication method, data deletion is performed, and the 3rd replica chunks of the cold data in the secondary rack is deleted. If there is a change from cold data to hot data, we reconstruct the data, copy 2nd replica chunks of the data as the 3rd replica chunks, and store it on other low-load nodes in the same rack. When storing new data, the new data stored in the three-way replication method.

When the global disk space utilization is greater than 30% and less than or equal to 60%, if we delete data to save storage overheads, there is a possibility that the global disk space

utilization will experience a large interval change, causing the global disk space utilization to be less than or equal to 30%. In order to prevent the data storage method from continuously oscillating, a "P" mark is set for the data stored in two-way replication method after data deletion. When the data marked with "P" is stored in a system where the global disk space utilization is less than or equal to 30%, its data storage method is not change to the three-way replication method due to the global disk space utilization of the system or the change from cold data to hot data. It still uses the two-way replication method for storage.

When the global disk space utilization is higher than 60%, the storage space is extremely low. Between the storage overhead and parallel data reading performance, we give priority to storage overhead. We firstly rank the popularity of the data, use the two-way replication method to store the hot data, and use the Erasure coding method to store the cold data. When the hot data is stored from the three-way replication method to the two-way replication method, data deletion will be performed to delete the 3rd replica chunks of the hot data in each rack. In the storage process of cold data changing from the two-way replication method to the Erasure Coding method, the data will be archived to form an Erasure Coding stripe, which is applied to RS(n, m). If there is a situation where hot data stored by the three-way replication method becomes cold data, data deletion will be performed firstly, and then data archiving will be performed. Next, if there is the cold data stored by the Erasure coding method becomes hot data, the data is changed from the Erasure coding method to the two-way replication method. This process requires data reconstruction. We copy all 2nd replica chunks of the data as 1st replica chunks, and stores them on the different low-load nodes of the original main rack. In this process of data reconstruction of a RS(n, m) stripe which changes to two-way replication method, T_{re} is the cross-rack data transfer traffic and n is from RS(n, m).

$$T_{re} = n \tag{3}$$

Because new data is often hot data, it is stored in two-way replication method. Similarly, in order to prevent the data storage method from constantly oscillating, we set an "E" to mark the data that changes the data storage method after data deletion and data archiving. When the data marked with "E" is stored in a system where the global disk space utilization is greater than 30% and less than or equal to 60%, its data storage method will not change due to the global disk space utilization of the system or the change from cold data to hot data.

In the process of data archiving, we need chunks to form the parity chunks and normal chunks part of the stripe. In order to reduce the cross-rack traffic caused by encoding, we have chosen a specific data layout method to store all the 1st replica chunks that may participate in encoding in the main rack. This avoids the need to transmit data to the machine and then encoding. Therefore, the parity chunks encoded by 1st replica chunks constitute the parity chunks part of the Erasure Coding stripe. These parity chunks need to be saved on different nodes of the same main rack. After the encoding is completed, we delete these 1st replica chunks of the main rack which participates in encoding. The 2nd replica chunks of these data whose 1st replica chunks participates in encoding then forms the normal chunks part of the stripe.

In order to maximize the advantages of partial decoding and reduce the cross-rack data transmission traffic generated by subsequent data recovery, the layout of stripes after data archiving should follow the layout principles: In order to maximize the advantages of partial decoding, we should place as many chunks of the same stripe as possible on the same rack. But for node fault tolerance, any two chunks of the same stripe are not distributed to the same node. At the same time, in order to obtain fault tolerance of m nodes, at most m chunks of the same stripe are placed on the same rack. Finally, for the uniform distribution of chunks, the numbers of chunks of a same stripe in any two racks differ by at most 1. $N(n, m) = \lceil \frac{n+m}{m} \rceil$ is the number of racks which store chunks of the same stripe.

$$N(n, m) = \lceil \frac{n+m}{m} \rceil = k \tag{4}$$

For short, we call the ith stripe Si, the ith rack Ri and the ith data Di. As shown in Figure 2, the six racks of $\{R0 \ldots R5\}$ need to store the ten chunks $\{D0 \ldots D9\}$. At this time, the global disk space utilization is less than or equal to 30%, and the disk space is sufficient. For the parallel data reading performance, the three-way replication method with good reading performance is selected for data storage. First, we compare the network bandwidth performance of the six racks, and select $R0$ with the best network bandwidth performance as the main rack to store the 1st replica chunk. Then we store the 2nd replica chunks and 3rd replica chunks of $\{D0 \ldots D9\}$ in $R1$ to $R5$ in order.

When the number of stored data begins to increase and the global disk space utilization is greater than 30% and less than or equal to 60%, we rank the popularity of the data and find that $\{D0, D5\}$ is hot data and the remaining data is cold data. we use a three-way replication method to store $\{D0, D5\}$, a two-way replication method to store the remaining cold data, and performs data deletion, deleting 3rd replica chunks of the cold data in $\{R2 \ldots R5\}$.

When the global disk space utilization is greater than 60%, storage space is extremely low. we rank the popularity of data and find that $\{D0, D5\}$ is hot data and the rest of the data is cold data. For storage overheads, we use two-way replication method to store hot data $\{D0, D5\}$, and the 3rd replica chunks of $\{D0, D5\}$ in $R1$ is deleted. For other cold data, we archive data and use the Erasure coding method to store it. In order to maximize the advantages of partial decoding, a specific Erasure coding layout idea is followed. For RS(4,2), $\{D1, D6, D2, D7\}$ and $\{D3, D8, D4, D9\}$ as the normal chunks parts of the two stripes. Next, we encode the data in $\{D1 \ldots D9\}$ in the $R0$, and generate the parity chunks parts $\{P0, P1\}$ and $\{P2,$

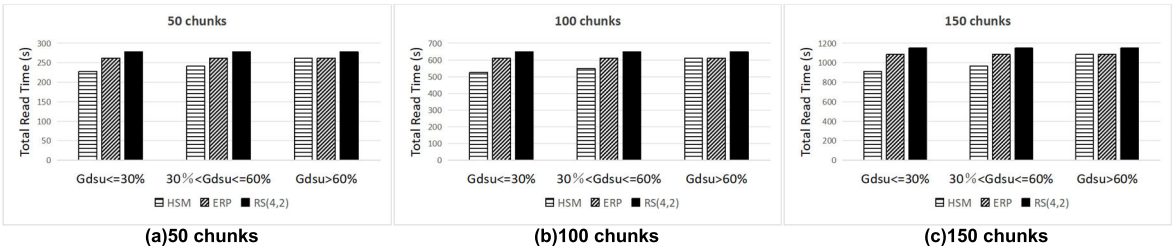


FIGURE 3. Total read time for HSM, ERP and RS(4,2) under the different global disk space utilization scenarios.

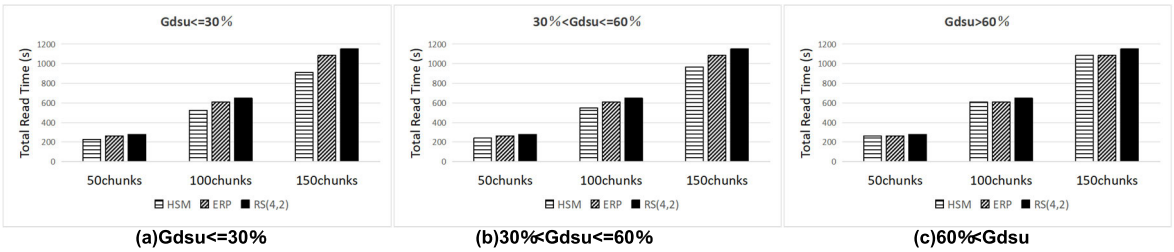


FIGURE 4. Total read time for HSM, ERP and RS(4,2) under the different amount of the data.

Algorithm 1 A Hybrid Storage Method Based on the Heat and Global Disk Space Utilization

Input: the data of the system
Output: storage_mode

```
if global disk space utilization <= 30%
    if the data has flag 'P '
        return;
    else
        Apply 3Replica();
if 30% < global disk space utilization <= 60%
    if the data has flag 'E '
        return;
    if the data changes from hot to cold
        Apply 3to2Replica(); Add flag 'P';
    else if the data changes from cold to hot
        Apply 2to3Replica();
if global disk space utilization −60%
    if the data changes from hot to cold
        if the data is 3Replica
            Apply 3to2Replica(); Add flag 'E ';
        else
            Apply 2toECO; Add flag 'E';
    else if the data changes from cold to hot
            Apply ECto2Replica();
```

P3}. And then when the encoding is completed, we delete {*D1*...*D9*} in the *R0*.

V. PERFORMANCE EVALUATION

In this section, we present extensive experiment results to evaluate the performance of the storage method HSM.

A. EXPERIMENTAL ENVIRONMENT

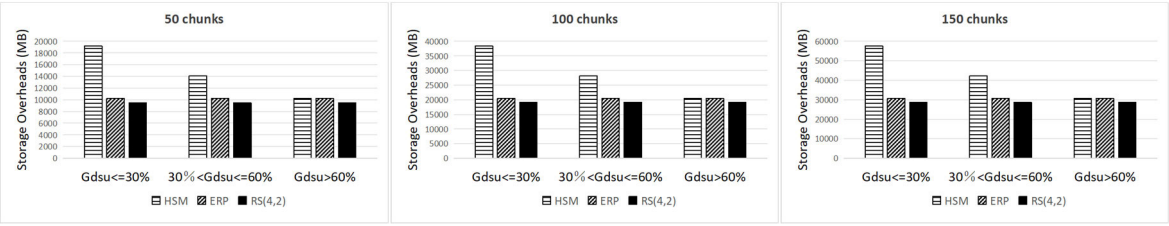
System configuration: we use one server with ten-core 2.40 GHz Intel(R) Xeon(R) Gold 5115 CPU, 256G memory, Dell PERC H730P integrated RAID hard disk, total storage space 1T, running Ubuntu 16.04.1. We configure KVM on this server and create seven new virtual machines, each with 8G memory, 5GB RAM, and 50GB SCSI hard disk and running Ubuntu 16.04.1. We use one virtual machine as a control terminal and use six virtual machines to simulate six racks. Each virtual machine has four processes to simulate four nodes. The bandwidth of the switch is 1Gbps. And we use the 128MB chunks in this experiment.

In order to verify various performance indicators of HSM, we select ERP (Encoding-oriented Replica Placement Policy) and RS(4,2) for comparative experiments. Among them, ERP uses different storage methods for data based on data popularity, which is similar to HSM, and RS(4,2) is widely used in the paper experiments.

B. EXPERIMENTAL DESIGN

In this section, we compare the parallel read performance and the storage overheads of HSM, ERP and RS(4,2) through the time of data reading and the storage overheads. Then we use cross-rack transmission traffic and transmission time during the data reconstruction and data archiving to compare the transmission overheads of HSM and ERP in the process of changing data storage methods.

In order to test the performance of HSM in different global disk space utilization (Gdsu) scenarios, we store chunks in virtual machines with the fixed disk space size to make the global disk space utilization of the system reach 10%, 40%, and 70%. In this way, we simulate three different interval scenarios of global disk space utilization (Gdsu) <= 30%, 30% < global disk space utilization (Gdsu) <= 60%, global disk space utilization (Gdsu) > 60%. And three groups of control experiments are set up under these three scenarios. At the



(a)50 chunks (b)100 chunks (c)150 chunks

FIGURE 5. Total storage overheads for HSM, ERP and RS(4,2) under the different global disk space utilization scenarios.

same time, in order to test the impact of the amount of stored data on the experiment, we store 50, 100, and 150 chunks in a rack composed of 6 virtual machines. Because the maximum amount of data in these three groups of control experiments is 150 chunks and the size of a chunk is 128MB, the global disk space utilization occupied by it when stored using the three-way replication method with the maximum storage overhead is about 18%. In this way, the global disk space will not change when storing the data, so that the range of disk space utilization does not affect the data storage method. Because 80% of accesses focus on 20% of the entire data set, based on this access rule, we define 20% of the data as hot data and the remaining 80% of the data as cold data. Since RS(4,2) is widely used in experiments, when archiving data, we set the Erasure coding stripe to RS(4,2).

C. ANALYSIS OF EXPERIMENTAL RESULTS

1) PARALLEL READ PERFORMANCE ANALYSIS

We test the parallel reading performance of HSM, ERP and RS(4,2) based on data reading time. Because HSM has different data storage methods under different global disk space utilization scenarios, we firstly test the data reading performance of HSM, ERP, and RS(4,2) under different global disk space utilization scenarios. As shown in Figure 3(a), Figure 3(b) and Figure 3(c), when the global disk space utilization is less than or equal to 30%, compared with ERP and RS(4,2), HSM reduces by up to 16 % and 18% data reading time. When the global disk space utilization is greater than 30% and less than or equal to 60%, the read performance advantage of HSM decreases. Compared with ERP and RS(4,2), HSM reduces the data read time by up to 11% and 13% respectively. When the global disk space utilization is greater than 60%, HSM no longer has a read performance advantage compared with ERP. Compared with RS(4,2), HSM reduces data read time by up to 6%. The reason is that the access parallelism of the replication method is higher than the Erasure Coding method. HSM uses a three-way replication method to store data when the global disk space utilization is less than or equal to 30%, so HSM is better than ERP and RS in data reading performance (4,2). However, with the increase of global disk space utilization, HSM needs to consider storage overhead and change the data storage method. The data is stored from a three-way replication method to a two-way replication or Erasure Coding method, so that HSM reduces the parallel read performance.

In addition, we also verify the impact of storing different amounts of data on parallel read performance under the same global disk space utilization scenario. As shown in Figure 4(a) and Figure 4(b), in two scenarios where the global disk space utilization is less than or equal to 30% and the global disk space utilization is greater than 30% and less than or equal to 60%, with the increase of the amount of data, compared with ERP and RS(4,2), the advantage of read performance in HSM becomes larger and larger. Because the access hot-spot problems of ERP and RS(4,2) will become more serious when the amount of data increases and data read requests are excessively concentrated on the racks that store hot data, which may create more network and disk I/O blocking. On the contrary, HSM distributes data read requests to many racks that store replicas of data for reducing congestion. As shown in Figure 4(c), when the global disk space utilization is higher than or equal to 60%, HSM changes the data storage method, which also causes a lot of network and disk I/O blocking when reading data, so that the parallel read performance advantage is not obvious.

2) STORAGE OVERHEAD ANALYSIS

Then, we compare HSM with ERP and RS(4,2) based on the data storage overheads. Because HSM has different data storage methods under different global disk space utilization scenarios, we also test the storage overheads of HSM, ERP, and RS(4,2) under different global disk space utilization scenarios. As shown in Figure 5(a), Figure 5(b) and Figure 5(c), when the global disk space utilization is less than or equal to 30%, due to sufficient disk space, HSM gives priority to parallel data reading performance and uses the three-way replication method to store data. Therefore, compared to ERP and RS(4,2), HSM increases storage overhead by up to 88% and 100% respectively. However, when the global disk space utilization is greater than 30% and less than or equal to 60%, the disk space is no longer sufficient. For ensuring high data reading performance, HSM should consider reducing the data storage overheads. HSM changes the storage method from the three-way replication method to two-way replication method for cold data, so the storage overhead is greatly reduced. Compared with ERP and RS(4,2), HSM only increases the storage overhead by up to 38% and 47% respectively. When

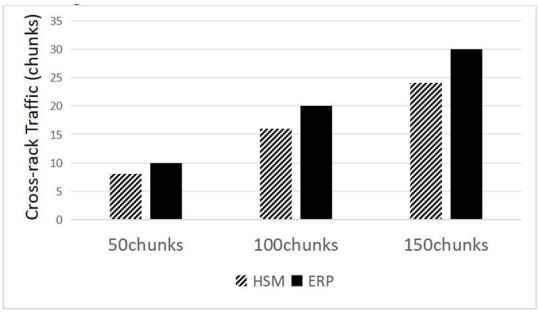


FIGURE 6. Total cross-rack transmission traffic for HSM and ERP.

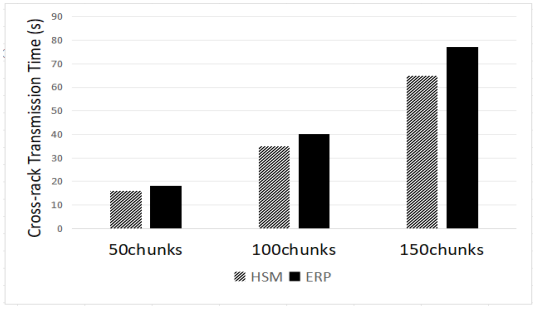


FIGURE 7. Total cross-rack transmission time for HSM and ERP.

the global disk space utilization is higher than 60%, HSM gives priority to storage overhead and changes the storage method from the two-way replication method to the Erasure Coding method for the cold data, and changes the hot data from the three-way replication method to the two-way replication method, which greatly reduces the storage overheads. Compared with ERP, HSM no longer has a storage overhead disadvantage. Compared with RS(4,2), HSM only increases storage overhead by up to 7%.

3) TRANSMISSION OVERHEAD BY CHANGING THE DATA STORAGE METHODS ANALYSIS

In this experiment, we test the cross-rack data transmission overheads of HSM and ERP based on the cross-rack data transmission traffic and data transmission time in the process of changing the storage methods. When the global disk space utilization is greater than 60%, we set 20% of the cold data stored in the Erasure Coding method as the data which requires data reconstruction based on Zipf law. Then we count the data reconstruction cross-rack transmission traffic and cross-rack data transmission time of the HSM and ERP in the process of changing storage methods. During the experiment, RS(4,2) isn't involved in operations such as data archiving and data reconstruction. The storage method remains unchanged in RS(4,2) throughout the entire storage process and RS(4,2) isn't involve cross-rack data transmission. Therefore, no relevant experiment for RS(4,2) in this section.The experimental results are shown in Figure 6 and Figure 7. Compared with ERP, HSM reduces cross-rack data transmission traffic by up to 20% and cross-rack data transmission time by 15%. With the amount of data increases, HSM improves cross-rack data transmission. The advantage of rack data transfer time will also be more obvious. In order to maximize the advantage of partial decoding, HSM stores the two parity chunks replicas in the same main rack in the RS(4,2) stripe, which generated by the data encoding in the main rack. Therefore, compared with ERP, HSM increases the cross-rack traffic during data reconstruction, but reduces cross-rack traffic for transferring parity chunks replicas to storage in other racks. As shown in Figure 7, in the process of cross-rack data transmission, with the increase of the cross-rack transmission traffic, the blocking of cross-rack data transmission will become more obvious. At the same time, because HSM has less cross-rack data transmission traffic than ERP, the cross-rack data transmission congestion is lighter in HSM. Due to the limitations of the experimental equipment and the time constraints, this work did not experiment with systems with different disk storage/RAM configurations and measure their impact on the final results which is a limitation of the current work and a goal for future work.

VI. CONCLUSION

In order to solve the problems of existing data storage methods such as low parallel data reading performance when disk space is sufficient and the excessive data storage overheads when disk space is low, in this paper we propose a hybrid storage method based on the heat of data and global disk space utilization, called HSM. HSM weighs the system's requirements for parallel data reading performance and the storage overheads under different global disk space utilization scenarios, and adaptively selects appropriate storage methods for the data which have different heat through operations such as data deletion, data reconstruction, or data archiving. It can increase the parallel reading performance for the hot data and reduce the storage overheads for the cold data. Firstly, we introduce Zipf law and define the popularity of the data based on Zipf law. Secondly, we introduce in detail the principles and algorithms of the hybrid storage method based on the heat and global disk space utilization. It is designed to effectively improve the parallel reading efficiency of distributed systems, alleviate global disk space constraints, and reduce cross-rack data transmission traffic and time in the process of changing the storage methods. Experimental results show that compared with ERP and RS(4,2), when disk space is sufficient, HSM reduces data reading time by up to 18%. When disk space is low, HSM increases storage overhead by up to 7%, but in the process of changing the storage methods, HSM reduced cross-rack transfer traffic by 20% and cross-rack transfer time by 15%.

REFERENCES

[1] L. Qian, "Cloud computing: An overview," in *Proc. Cloud Comput., 1st Int. Conf. (Cloudcom)*, Beijing, China, Dec. 2009, pp. 626–631.
[2] K. Michael, "Securing the cloud: Cloud computer security techniques and tactics," *Comput. Secur.*, vol. 31, no. 4, p. 633, Jun. 2012.

[3] H. Zhou, D. Feng, and Y. Hu, "Bandwidth-aware scheduling repair techniques in erasure-coded clusters: Design and analysis," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 3333–3348, Dec. 2022, doi: 10.1109/TPDS.2022.3153061.
[4] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," *ACM SIGOPS Operating Syst. Rev.*, vol. 37, no. 5, pp. 29–43, Dec. 2003, doi: 10.1145/1165389.945450.
[5] R. G. Masur and S. K. Mcintosh, "Preliminary performance analysis of Hadoop 3.0.0-Alpha3," in *Proc. New York Sci. Data Summit (NYSDS)*, Aug. 2017, pp. 1–3.
[6] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proc. 7th USENIX Symp. Operating Syst. Design Implement.*, 2006, pp. 307–320.
[7] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in windows Azure storage," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)* Boston, MA, USA: USENIX Association, 2012, pp. 15–26.
[8] *Andrew Wang. 2017. Apache Hadoop 3.0.0*, Apache Software Foundation, Forest Hill, MD, USA, Mar. 2020.
[9] M. Xia, "A tale of two erasure codes in HDFS," in *Proc. Usenix Conf. File Storage Technol.*, 2015.
[10] A. Chiniah and A. Mungur, "Dynamic erasure coding policy allocation (DECPA) in Hadoop 3.0," in *Proc. 6th IEEE Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/ 5th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, Jun. 2019, pp. 29–33.
[11] H. Qiu, C. Wu, J. Li, M. Guo, T. Liu, X. He, Y. Dong, and Y. Zhao, "EC-fusion: An efficient hybrid erasure coding framework to improve both application and recovery performance in cloud storage systems," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2020, pp. 191–201.
[12] X. Xie, C. Wu, J. Gu, H. Qiu, J. Li, M. Guo, X. He, Y. Dong, and Y. Zhao, "AZ-code: An efficient availability zone level erasure code to provide high fault tolerance in cloud storage systems," in *Proc. 35th Symp. Mass Storage Syst. Technol. (MSST)*, May 2019, pp. 230–243.
[13] J. Jeong, J. Kwak, D. Lee, S. Choi, J. Lee, J. Choi, and Y. H. Song, "Level aware data placement technique for hybrid NAND flash storage of log-structured merge-tree based key-value store system," *IEEE Access*, vol. 8, pp. 188256–188268, 2020, doi: 10.1109/ACCESS.2020.3031322.
[14] Y. Yuan, J. Zhang, and W. Xu, "Dynamic multiple-replica provable data possession in cloud storage system," *IEEE Access*, vol. 8, pp. 120778–120784, 2020, doi: 10.1109/ACCESS.2020.3006278.
[15] B. Xu, J. Huang, X. Qin, and Q. Cao, "Traffic-aware erasure-coded archival schemes for in-memory stores," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 12, pp. 2938–2953, Dec. 2020, doi: 10.1109/TPDS.2020.3009092.
[16] H. Jin-Ping, "Hybrid coding LRC-RS in heterogeneous decentralized storage," *Comput. Eng. Des.*, vol. 42, no. 2, pp. 301–308, 2021.
[17] Y. Zhang, H. Li, S. Liu, and P. Huang, "Hybrid block storage for efficient cloud volume service," *ACM Trans. Storage*, vol. 19, no. 4, pp. 1–25, Nov. 2023.
[18] X. Wei, Y. Tan, D. Liu, D. Wu, Y. Wu, X. Chen, and J. Li, "Weak network oriented mobile distributed storage: A hybrid fault-tolerance scheme based on potential replicas," in *Proc. IEEE 24th Int. Conf. High Perform. Comput. Commun., 8th Int. Conf. Data Sci. Syst., 20th Int. Conf. Smart City; 8th Int. Conf. Dependability Sensor, Cloud Big Data Syst. Appl. (HPCC/DSS/SmartCity/DependSys)*, Hainan, China, Dec. 2022, pp. 372–379.
[19] Y. Hu, Y. Wang, B. Liu, D. Niu, and C. Huang, "Latency reduction and load balancing in coded storage systems," in *Proc. Symp. Cloud Comput.*, Sep. 2017, pp. 365–377.
[20] S. Mitra, R. Panta, M.-R. Ra, and S. Bagchi, "Partial-parallel-repair (PPR): A distributed technique for repairing erasure coded storage," in *Proc. 11th Eur. Conf. Comput. Syst.*, Apr. 2016, p. 30.
[21] Y. Chen, S. Alspaugh, and R. Katz, "Interactive analytical processing in big data systems: A cross-industry study of MapReduce workloads," *Proc. VLDB Endowment*, vol. 5, no. 12, pp. 1802–1813, Aug. 2012, doi: 10.14778/2367502.2367519.
[22] Z. Wang, H. Wang, A. Shao, and D. Wang, "An adaptive erasure-coded storage scheme with an efficient code-switching algorithm," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Nov. 2020, pp. 1177–1178.
[23] W. Yi-Jie, X. Fang-Liang, and P. X. Qiang, "Research on erasure code-based fault-tolerant technology for distributed storage," *Chin. J. Comput.*, vol. 40, no. 1, pp. 236–255, 2017.

YING SONG received the Ph.D. degree in computer engineering from the Institute of Computing Technology (ICT), Chinese Academy of Sciences. She is currently an Associate Professor with the Computer School, Beijing Information Science and Technology University. Her work has covered topics, such as performance modeling, resource management, cloud computing, and big data computing platform. She has been authored or coauthored more than 30 publications in these areas, since 2007. Her main research interests include computer architecture, parallel and distributed computing, and virtualization technology. She served in various academic conferences.

WENXUAN ZHAO received the B.S. degree from Guangzhou University, Guangdong, China, in 2021. He is currently pursuing the master's degree with the Computer School, Beijing Information Science and Technology University, Beijing, China. His main research interest includes distributed storage.

YINGAI TIAN received the B.S. degree in computer science and technology from Beijing University of Science and Technology, Beijing, China, in 2006. She is currently an Associate Professor with the Computer School, Beijing Information Science and Technology University. Her main research interests include big data analysis methods, machine learning algorithm, document information pressing technology, and document format standard.

BO WANG received the B.S. degree in computer science from Northeast Forest University (NEFU), Harbin, China, in 2010, and the Ph.D. degree in computer science from Xi'an Jiaotong University (XJTU), Xi'an, China, in 2017. He was the Guest Student of the State Key Laboratory of Computer Architecture, Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), from 2012 to 2016. He is currently a Lecturer with the Software Engineering College, Zhengzhou University of Light Industry (ZZULI). He has published more than ten research articles in these areas. His research interests include distributed systems, cloud computing, edge computing, resource management, and task scheduling. He served in various academic journals and conferences.