Journal of Computers Vol. 35 No. 2, April 2024, pp. 75-89
doi: 10.53106/199115992024043502005

S-MBR: An Efficient Scaling Scheme for PM-MBR-Coded Distributed Storage System

Cong Li[1], Hong-Liang Cai[1*], Wen-Jie Deng[2],
Dan Tang[3], Yuan-Ping Xu[4], Tie-Yuan Hong[5]

School of Software Engineering, Chengdu University of Information Technology,
Chengdu 610225, China

3415847554@qq.com, caihl@cuit.edu.cn, 1437780164@qq.com,
tang-dan@foxmail.com, ghxpy@hotmail.com, 534601123@qq.com

Received 3 August 2023; Revised 12 December 2023; Accepted 3 January 2024

Abstract. With the rapid increase in data storage, distributed storage systems need to add new nodes to reduce storage and computation pressure. However, the existing scaling schemes are not efficient enough. To solve these issues, A scaling scheme called S-MBR is proposed, which has two notable features. Firstly, S-MBR achieves uniform data distribution by using a symmetric data layout to evenly place data blocks on the expanded data nodes. Secondly, S-MBR minimizes data movement during data redistribution and parity update processes by re-locating data blocks to symmetric positions and performing partial parity block calculations after the storage nodes are expanded. S-MBR can reach the lower bound of data migration volume when the number of nodes required for reconstruction is equal to the number of nodes required for repair. According to mathematical analysis and experimental results, compared to many typical scaling schemes, S-MBR can reduce data transmission volume by up to approximately 93% and average response time by around 41%.

Keywords: regeneration code, distribution storage system, scaling scheme

1 Introduction

Currently, the era of big data has arrived. The scale and growth rate of data have become enormous and rapid [1]. This rapid growth in data poses significant challenges for distributed storage systems to provide high-quality and efficient services.

When dealing with massive data storage, some distributed storage systems (such as HDFS [2], Ceph [3], NetApp [4], Windows Azure [5], and Oceanstone [6]) face the risks of data loss and node failures. Distributed storage systems have introduced erasure coding technology to ensure data reliability and availability. However, with technological advancements, the main challenge of erasure coding in distributed storage has shifted from high computational complexity to the consumption of network resources. These render classic erasure codes such as Reed-Solomon (RS) code [7] and early codes used in RAID storage systems like EVENODD [8], RDP [9], X-Code [10], STAR [11], and P-Code [12] unsuitable for distributed scenarios. Therefore, Rashmi et al. [13-14] proposed a Regeneration Code (RC) based on the idea of network coding. RC can be further categorized into Product-Matrix Minimum Bandwidth Regeneration (PM-MBR) codes and Product-Matrix Minimum Storage Regeneration (PM-MSR) codes based on Product-Matrix (PM) construction. RC is more suitable for distributed storage scenarios due to its lower repair overhead. Compared with the PM-MSR code, the PM-MBR code has a lower space utilization rate in order to reduce the repair cost. Therefore, clusters built with PM-MBR codes are more likely to face insufficient storage space. At this time, an efficient and fast scaling scheme is particularly important.

Before the introduction of erasure coding technology, regularly distributed storage systems were used to address the computational and storage pressure by adding more nodes and balancing the workload through data migration [15]. However, with the incorporation of erasure coding technology, while the data reliability is guaranteed, system scalability becomes more challenging. Now, the system needs to consider not only load balancing but also issues related to data layout, parity block updates, and data transmission. Observing that the scaling schemes are closely related to the chosen coding layout, and efficient scaling schemes are designed based on specific types of coding schemes.

* Corresponding Author

Currently, in the field of Redundant Arrays of Inexpensive Disks (RAID) [16], there are well-established scaling schemes. Such as scaling schemes [17-23] based on RAID-0 or RAID-5 coding layouts, [24-28] based on RAID-6 coding layouts. However, in the relatively shorter development time of RC, there are fewer scaling schemes available in this domain. There is an urgent need for efficient and reliable scaling schemes to effectively scale distributed storage systems.

In order to solve the above problems, the S-MBR scaling scheme is proposed. The main contributions of this paper are as follows.

1) An efficient scaling is proposed. In the data migration stage, the idea of completing data migration within the node is used to reduce the amount of data migration. In the verification update phase, the idea of combined calculation within the node is used to share the calculation of the verification block to multiple nodes, thereby easing the computing pressure on the computing nodes and reducing the amount of data transmission.

2) Analyzing and implementing the RR, Scale-RS, and EMBRScale scaling schemes, and comparing their data transmission amounts.

3) Experimental Demonstration: Conducting comparative experiments on the S-MBR, RR, Scale-RS, and EMBRScale scaling schemes. Experimental results show that S-MBR can reduce data transfer volume by up to 83%, 93%, and 84% compared to EMBRScale, RR, and Scale-RS, respectively.

The remaining sections of this paper are organized as follows. Section 2 introduces the relevant scaling schemes. Section 3 presents the preparatory work for S-MBR. The design of S-MBR is detailed in Section 4. Section 5 describes the experimental setup and results. Finally, our work concludes in Section 6.

2 Related Work

The scaling schemes in the RAID domain have been introduced earlier, and this paper focuses more on the RC field. This section will introduce the current scaling schemes (refer to Table 1 for parameter explanations).

The scaling schemes of erasure codes are closely related to the type and application scenario of erasure codes. The scaling scheme on RAID is currently the most studied field by domestic and foreign scholars, and has also achieved rich results. At present, common scaling schemes for optimizing the data migration process include the RR (Round-Robin) scaling scheme proposed in the literature [17]. This scheme requires a re-layout of almost all data blocks, so the amount of data transmission is large. Literature [18] is better than the RR Scaling scheme in terms of response time, but it has the same shortcomings as the RR scaling scheme. The transmission volume is large and it cannot achieve load balancing. Common scaling schemes to reduce the amount of data migration include the scaling scheme in literature [19-20] implemented on RAID-5 and the scaling schemes in literature [24-28] implemented on RAID-6. Literature [19] optimized the RR scaling scheme and reduced the amount of data migration, but the amount of data migration in this scheme is still large. Literature [20] uses the idea of data partitioning to realize the scaling of data nodes. This feature is not available in most scaling schemes, but this scheme is not suitable for storage systems with strong local access. Literature [24] performs data migration based on stripes. Compared with the RR scaling scheme and literature [19], its data migration volume and migration time are greatly reduced, but it is only applicable to specific RAID-6 encoding layouts. Literature [25] based on X-Code encoding minimizes the number of migrated data blocks while maintaining a consistent data layout, which is better than the RR scheme in terms of time and I/O latency. Literature [26] uses RDP-encoded row check chain and skew check chain to migrate data on the chain, thus reducing the data required for check updates. Literature [27] improves on the migration idea of Literature [26]. Based on RDP coding, it uses the layout of rotation deployment and Piggback [28] technology to reduce the amount of data migration and the amount of data that needs to be read when verifying updates. Compared with the literature [26], the literature [27] achieves the optimal amount of data block and check block migration.

Compared with the scaling scheme on RAID, Zhao et al. [29] initially proposed a scaling scheme based on Exact Minimum Bandwidth Storage Regeneration (E-MBR) codes, which set a maximum scaling number t and construct a large coding matrix. This scheme minimizes the data migration when the actual scaling number s is less than or equal to t. However, its major drawback is that it imposes an upper bound on the number of scaling nodes, which limits its practicality. Huang et al. [30] designed a transpose layout based on RS and proposed the Scale-RS scaling scheme to achieve the minimum data migration. However, it fails to achieve the minimum scaling overhead. Zhang et al. [31] improved upon the Scale-RS scheme and applied it to RC, developing the NCScale distributed system based on XOR operations. Compared to Scale-RS, NCScale reduces the scaling time by up to 50%. However, its operation mode restricts its applicability to PM-MBR and PM-MSR codes. Rai

et al. [32] achieved the lowest data download volume during scaling from parameters [n, k] to [$n + s$, $k + s$], but specific implementation details and experimental conclusions were not provided. Additionally, Zhang et al. [33] designed the EMBRScale scheme based on MBR and MSR codes to minimize the data migration during scaling. However, this scheme only implemented MBR code scaling for $n - k = 1$ and $n - k = 2$, and MSR code scaling for $s = 1$. Hu et al. [34] conducted an analysis using information flow graphs and studied scaling parameters. They achieved scaling from [n, k] to [n', k'] ($n' > n$, $k' > k$), but this scheme only achieves the theoretical minimum data migration when $n = n'$ and $k = k'$.

RC scaling schemes often have significant parameter limitations that hinder their practical application in real distributed storage systems. And the amount of data migration in these scaling schemes failed to reach the theoretical optimal value.

3 Background

3.1 Basic Concepts

This paper focuses on the scenario of adding nodes to distributed storage systems. In this section, the symbols used in this paper are explained, as shown in Table 1. The symbol x represents the parameters before scaling, while the symbol x' represents the corresponding parameters after scaling. Additionally, to provide a clearer description of the scaling process of the S-MBR scheme, the following explanation of some related concepts is provided [35-36].

1) Encoding: Encoding refers to the process of storing data by using encoding algorithms to generate encoded blocks. It is commonly denoted by the symbol [n, k, d].

2) Decoding: Decoding refers to the process of obtaining the original data from encoded blocks using decoding algorithms.

3) Stripe: A stripe is a collection of related encoded blocks that can independently perform erasure coding algorithms.

4) Systematic Code: A systematic code is a type of erasure code that allows the original data object to be directly recovered by reading and concatenating the data blocks within a stripe, without the need for decoding operations when no data blocks within the stripe are lost or damaged.

5) Scaling: In this paper, the process of system expansion from the parameter [n, k, d] to [$n + s$, $k + s$, $d + s$], denoted as [n, k, d, s]. The scaling process involves operations on the encoded blocks and is divided into stripe partitioning, data migration, and parity update.

6) Stripe Partitioning: Stripe partitioning is the preparatory step for scaling, which involves grouping the stripes based on their functionality to facilitate subsequent scaling operations.

7) Data Migration: Data migration refers to the process of evenly placing all the data blocks from the original data nodes to the expanded data nodes after scaling.

8) Parity Update: Parity update refers to updating the parity blocks in all the parity nodes to maintain the fault tolerance of the stored data after scaling.

Table 1. Symbols and definitions

Symbols	Definition
n	Total number of nodes
k	Number of nodes required for reconstruction
d	Number of nodes required for exact repair
r	Number of parity nodes or redundancy count (r=n-k)
s	Number of expansion nodes
B	Number of unique data blocks in a stripe
Di	The i-th data node in a stripe
Pj	The j-th parity node in a stripe
Di,j	Data block in the i-th row of data node j
Pi,j	Parity block in the i-th row of parity node j

Si	The i-th stripe
M(d,d)	Message matrix with d rows and d columns
Ψ(n,d)	Encoding matrix with n rows and d columns
I(k,k)	Unit matrix with k rows and k columns
V(r,d)	Vandermonde matrix with r rows and d columns
C(n,d)	Code matrix with n rows and d columns and C=ΨM
T^t	Transpose the matrix T
dm	Data transfer volume during the data migration phase
pu	Data transfer volume during the parity update phase
td	Total data transfer volume and td = dm + pu

3.2 Construction of PM-MBR Codes

PM-MBR codes are commonly represented by a triplet [n, k, d] ($k \le d \le n-1$). Use PM-MBR code with the characteristics of the systematic code for expansion [37]. Compared with the original PM-MBR code, its construction method is different. The construction of the encoding matrix $\psi_{(n,d)}$ for the systematic version of the PM-MBR involves replacing the matrix $V_{(n,d)}$ with a new matrix, as shown in Equation (1).

$$\psi_{(n,d)} = V_{(n,d)} -> \psi_{(n,d)} = \begin{bmatrix} I_{(k,k)} & 0_{(k,d-k)} \\ & V_{(r,d)} \end{bmatrix}. \tag{1}$$

When the parameter is [n, k, d], the total number of data blocks required in a stripe is $B=C_{k+1}^2 + k(d-k)$, specifically $\{b_1, b_2, ..., b_B\}$. The construction of the message matrix $M_{(d,d)}$ and the encoding matrix $\psi_{(n,d)}$ is given by Equation (1) and Equation (2) respectively. The matrix $S_{(k,k)}$ is a $k \times k$ symmetric matrix, where the lower triangular part (including the diagonal elements) is filled with data blocks in a Round-Robin (RR) cyclic manner [17-19], and the remaining positions can be obtained based on symmetry. The matrix $T_{(k,d-k)}$ is formed by the remaining data blocks, $\{b_{C_{k+1}^2+1}, b_{C_{k+1}^2+2}, ..., b_B\}$ placed in a similar manner as before. Finally, by following these steps, the code matrix $C_{(n,d)} = \psi_{(n,d)} M_{(d,d)}$ is obtained, and the encoding process is completed.

$$M_{(d,d)} = \begin{bmatrix} S_{(k,k)} & T_{(k,d-k)} \\ T^t_{(d-k,k)} & 0_{(d-k,d-k)} \end{bmatrix}. \tag{2}$$

$$\psi_{(n,d)} = \begin{bmatrix} I_{(k,k)} & 0_{(k,d-k)} \\ & V_{(n-k,d)} \end{bmatrix}. \tag{3}$$

Taking the [4,2,3] as an example, the code matrix $C_{(4,3)}$ is obtained from Equation (3), and the encoding result is shown in Fig. 1.

$$\begin{aligned} C_{(4,3)} = \psi_{(4,3)} M_{(3,3)} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_4 \\ b_2 & b_3 & b_5 \\ b_4 & b_5 & 0 \end{bmatrix} \\ &= \begin{bmatrix} b_1 & b_2 & b_4 \\ b_2 & b_3 & b_5 \\ b_1+b_2+b_4 & b_2+b_3+b_5 & b_4+b_5 \\ b_1+2b_2+4b_4 & b_2+2b_3+4b_5 & b_4+2b_5 \end{bmatrix}. \end{aligned} \tag{4}$$

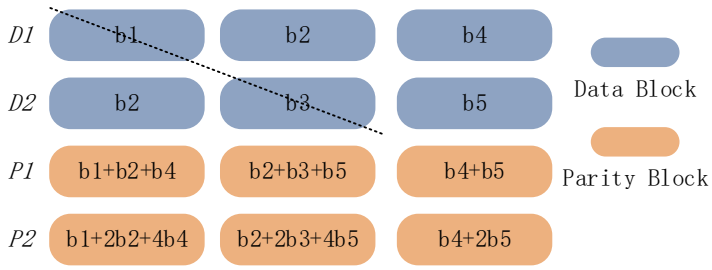

Fig. 1. Encoding result for PM-MBR code with parameter [4,2,3]

3.3 Related Scaling Schemes

Compare three fast scaling schemes: RR, Scale-RS, and EMBRScale. The data migration process for scaling three nodes using the [4,2,3] example is shown in Fig. 2.

1) RR: As depicted in Fig. 2(a), extract all unique data blocks from the nodes $\{D_1, D_2\}$ and place them sequentially using the Round-Robin scheme. Then, to replicate the blue blocks based on the diagonal and complete the data migration.

2) EMBRScale: Fig. 2(b) demonstrates the EMBRScale scheme. Initially, select three orange blocks within nodes $\{D_1, D_2\}$ and perform intra-node migration. Then, migrate the green blocks to the lower triangular portion of the newly added nodes $\{D_3, D_4, D_5\}$. Finally, replicate symmetrically the data to obtain the blue blocks, completing the data migration.

3) Scale-RS: Fig. 2(c) shows the Scale-RS scheme. Place the orange blocks $\{b_6, ..., b_{20}\}$ according to the arrows in the figure and then perform data symmetry to obtain the blue blocks, completing the data migration.

Due to the encoding and decoding process of the PM-MBR code, the update strategies of the above scaling schemes are not applicable. Thus, parity blocks that need to be updated can only be obtained by reconstruction. For all scaling schemes, the number of data blocks that need to be migrated is $(r + k')d'$. Please refer to section 3.2 for the reconstruction process.

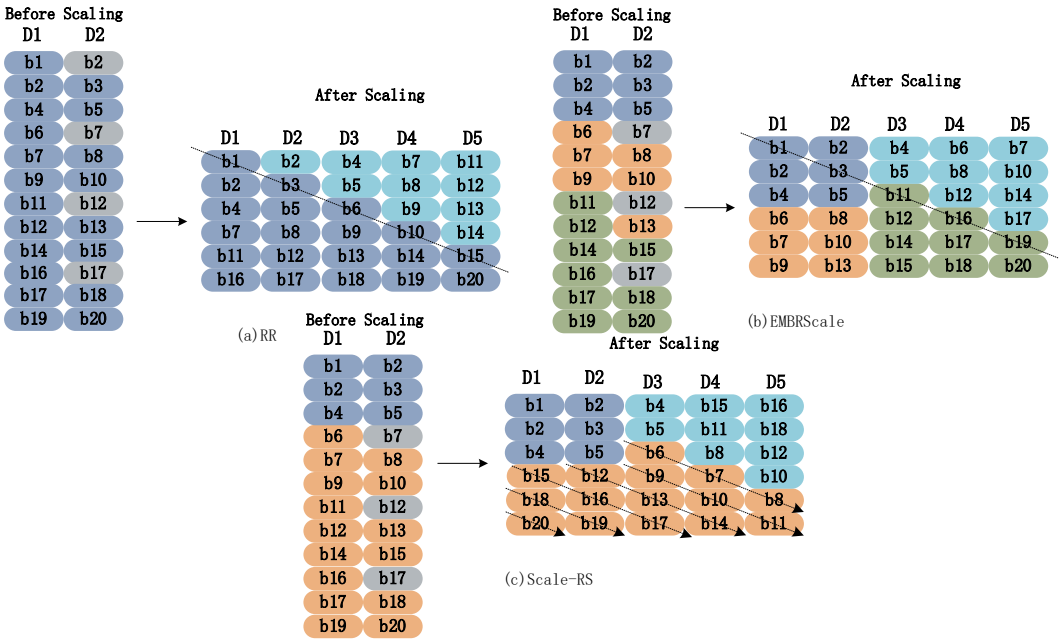

Fig. 2. Data migration process for RR, EMBRScale, and Scale-RS scaling schemes when scaling from [4,2,3] to [7,5,6]

4 S-MBR: An Efficient Scaling Scheme

4.1 Overview

The scaling process of data nodes from k to $k + s$ in a cluster is generally accomplished through the following three steps:

1) Partitioning: Based on the symmetric encoding layout of the PM-MBR code shown in Fig. 1, all stripes are divided into sacrificial stripes (Fig. 3(a) and Fig. 3(b)) and non-migrating stripes (Fig. 3(c)). The areas that need to be filled are divided into V1, V2, and Symmetry.

2) Filling: The data blocks from sacrificial are filled into the V1 and V2 areas. The Symmetry area is obtained by symmetric replication.

3) Discarding: The secondary diagonal indicates that the upper and lower parts are symmetrically duplicated. Therefore, some data blocks from the sacrificial stripes need to be discarded after scaling.

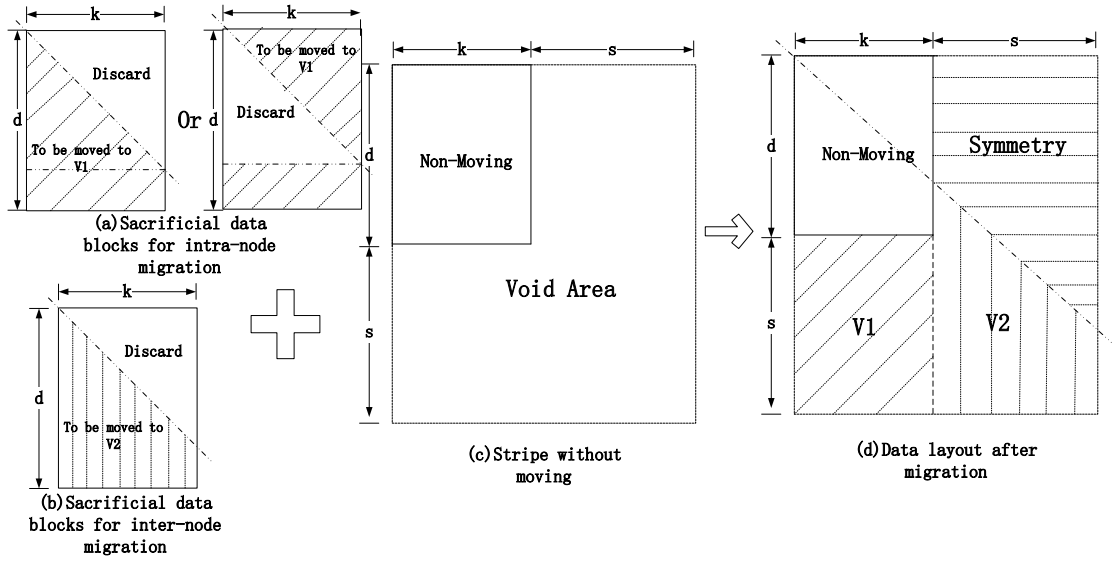

Fig. 3. The scaling process of data nodes from k to $k + s$

4.2 Stripe Partitioning

The S-MBR scheme refers to GSR (Global Striped-based Redistribution) [20] and EMBRScale scheme. Classify and group all the stripes. S-MBR groups $B'=C_{k'+1}^{2}+k'(d'-k')$ old stripes together. Each group contains $B'B$ non-repetitive data blocks, which can exactly form B new stripes. Therefore, the first B stripes in each group are classified as non-moving stripes, while the remaining $B' - B$ old stripes are classified as sacrificial stripes. The specific implementation is shown in Fig. 4, where $\{s_1, ..., s_5\}$ represents the non-moving stripes and $\{s_6, ..., s_{20}\}$ represents the sacrificial stripes.

1) Non-moving Strips: All the data blocks in this category remain unchanged, as shown in Fig. 4(a).

2) Sacrificial Strips: All the data blocks in this category need to be migrated to the non-moving stripes to help them form new stripes. The sacrificial stripes can be further divided into sacrificial stripes for internode migration and sacrificial stripes for internode migration, as shown in Fig. 4(b) and Fig. 4(c). It is worth noting that although the diagonal lines are consistent, the discarded data block regions are divided into two cases that need to be alternately selected at intervals, as shown in gray-colored data blocks in Fig. 4(b) and Fig. 4(c).

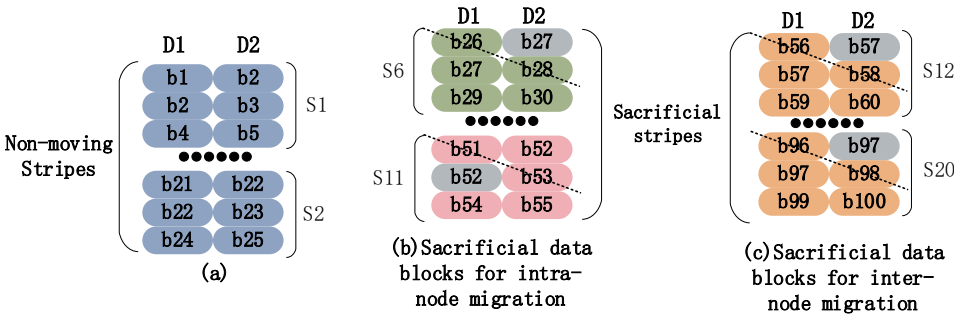

Fig. 4. The process of stripe partitioning in the E-MSR scheme when scaling from [4,2,3] to [7,5,6]

4.3 Data Migration

Data migration is the second process in the scaling, and the S-MBR scheme is designed to minimize the number of data blocks. It involves three steps: intra-node data migration, inter-node data migration, and data symmetric filling. The following example illustrates the scaling operation of a set of stripes.

1) Intra-node data migration: Data blocks from the upper and lower triangles of the sacrificial stripes $\{S_i \mid i \in [B+1, B']\}$ are selected with an interval and migrated to the V1 region of the non-moving stripes $\{S_i \mid i \in [1, B]\}$, as shown in Fig. 3(d).

2) Inter-node data migration: The remaining lower triangle data blocks of the sacrificial stripes are migrated across nodes to the V2 region of the non-moving stripes.

3) Data Symmetric Filling: Fill the Symmetry region of the non-moving stripes based on the secondary diagonal. With this step, the new stripes $\{S'_1, ..., S'_B\}$ are completed.

4.4 Parity Update

During the study of parity block generation in PM-MBR code, discovered that some parity blocks are essentially linear combinations of the data blocks within the current node. Since the calculations occur within the storage nodes, they only consume the computational resources of the storage nodes. This aligns with the distributed computing principles and alleviates some computational burden on the compute nodes. Therefore, designed the update algorithm to update the parity nodes in each stripe. Divide $P_{(d', r)}$ into two parts, $P_{(k', r)}$ and $P_{(d'-k', r)}$, for the update process.

Let $T_{(k', k')}$ be the matrix composed of data blocks in the data node $\{D_1, ..., D_{k'}\}$, as shown in Equation (5), where the i-th column is denoted as T_i. Let $H_{(d'-k', k')}$ denotes the matrix composed of the last $d' - k'$ rows of data blocks in data nodes $\{D_1, ..., D_{k'}\}$. The entire update process is completed in three steps.

$$T_{(k',k')} = \begin{bmatrix} D_{1,1} & \cdots & D_{1,k} & \cdots & D_{1,k'} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ D_{k,1} & \cdots & D_{k,k} & \cdots & D_{k,k'} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ D_{k',1} & \cdots & D_{k',k} & \cdots & D_{k',k'} \end{bmatrix}. \tag{5}$$

1) Updating $P_{(k', r)}$, where the parity blocks are derived from $P_{i,j} = V'_j T_i$, and then replacing the first k' rows of parity blocks in each new stripe.

2) Downloading the data blocks from the last $d' - k'$ rows of data nodes $\{D_1, ..., D_{k'}\}$ to obtain the matrix $H_{(d'-k', k')}$, as shown in Equation (6).

$$H_{(d'-k',k')} = \begin{bmatrix} D_{k'+1,1} & \cdots & D_{k'+1,k'} \\ \vdots & \ddots & \vdots \\ D_{d',1} & \cdots & D_{d',k'} \end{bmatrix}. \tag{6}$$

3) $P_{(d'-k',r)}$ are calculated using Equation (7). Then replacing the last $d'-k'$ rows of parity blocks in each new stripe $\{S'_1,...,S'_B\}$. The parity update is completed.

$$P_{(d'-k',r)} = (V'_{(r,d')} \begin{bmatrix} H^t{}_{(d'-k',k')} \\ 0_{(d'-k',d'-k')} \end{bmatrix})^t. \tag{7}$$

4.5 An Example for S-MBR Scaling Scheme

In Sections 4.1-4.4, a detailed explanation of the entire S-MBR algorithm is provided. This subsection aims to illustrate the practical operation process of the S-MBR scaling algorithm using a specific example [4,2,3,3], to help readers better understand it.

1) Stripe Partitioning: By calculation, to obtain $B' = 20$ and $B = 5$. Therefore, divide the 20 old stripes into one group, where the first 5 stripes are non-moving stripes, and the remaining 15 stripes are sacrificial. For specific examples, please refer to Fig. 4.

2) Intra-node Data Migration: Within nodes $\{D_1, D_2\}$, select three data blocks from each stripe at intervals and migrate them to stripes $\{S_1, ..., S_5\}$. For example, blocks $\{b_{26}, ..., b_{30}, b_{32}\}$ from Fig. 5(b) are migrated to Fig. 5(a). The result is shown in Fig. 5(d).

3) Inter-node Data Migration: The non-repeating data blocks $\{b_{56}, ..., b_{64}\}$ in the lower triangular part of Fig. 5(c) need to be migrated to the newly added data node $\{D_3, D_4, D_5\}$. The resulting configuration is shown in Fig. 5(e).

4) Symmetric Filling: The data blocks in Fig. 5(f) are symmetrically obtained based on the secondary diagonal, forming the new stripes as shown in Fig. 5(g). At this point, the data migration process is completed.

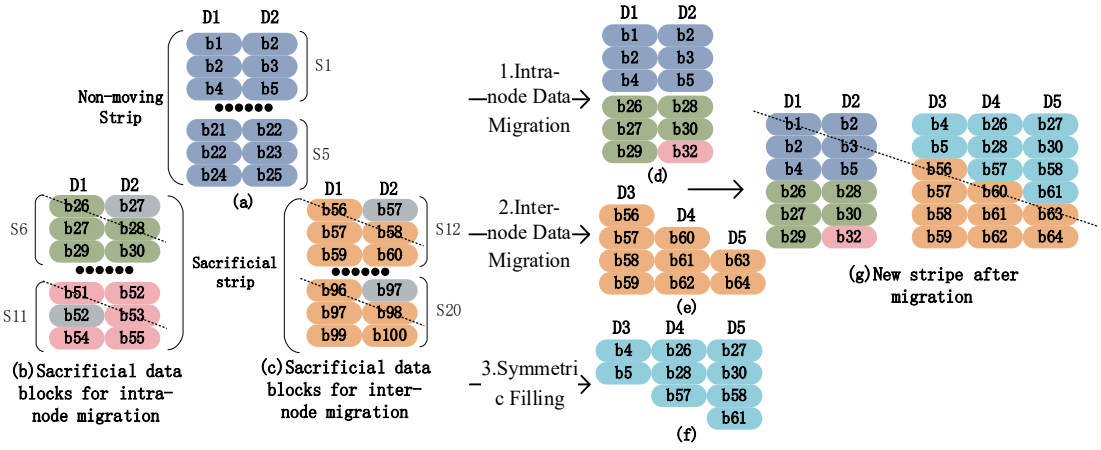

Fig. 5. The process of migrating data blocks in the E-MSR scheme when scaling from [4,2,3] to [7,5,6]

1) Within each data node $\{D_1, ..., D_5\}$, $\{T_1, ..., T_5\}$ are obtained by linear combination. T_1 is obtained using Equation (8). $P_{1,1}$ and $P_{1,2}$ are calculated by using Equation (9) and Equation (10). The remaining data blocks $\{p_{i,j} \mid 1 \le i \le 5, 1 \le j \le 2\}$ in the parity node $\{P_1, P_2\}$ are derived from this process. As shown in Fig. 6.

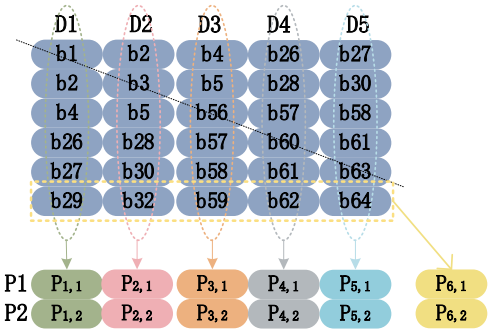

Fig. 6. The process of updating parity blocks in the E-MSR scheme when scaling from [4,2,3] to [7,5,6]

$$T_1 = [b_1 \quad b_2 \quad b_4 \quad b_{26} \quad b_{27} \quad b_{29}]^t . \tag{8}$$

$$P_{1,1} = V_1' T_1 = b_1 + b_2 + b_4 + b_{26} + b_{27} + b_{29} . \tag{9}$$

$$P_{1,2} = V_2' T_1 = b_1 + 2b_2 + 4b_4 + 8b_{26} + 16b_{27} + 32b_{29} . \tag{10}$$

2) This step is executed only if $k \neq d$. First, get all data blocks from the last row of the data nodes. $H_{(1,5)}$ as shown in Equation (11). $\{P_{6,1}, P_{6,2}\}$ are calculated from Equation (12). At this point, the scaling process of S-MBR is completed.

$$H_{(1,5)} = \begin{bmatrix} b_{29} & b_{32} & b_{59} & b_{62} & b_{64} \end{bmatrix} . \tag{11}$$

$$\begin{bmatrix} P_{6,1} & P_{6,2} \end{bmatrix} = (V^{`}_{(2,6)} \begin{bmatrix} H^t_{(1,5)} \\ 0_{(1,1)} \end{bmatrix})^t = \begin{bmatrix} b_{29} + b_{32} + b_{59} + b_{62} + b_{64} & b_{29} + 2b_{32} + 4b_{59} + 8b_{62} + 16b_{64} \end{bmatrix} . \tag{12}$$

4.6 Data Transfer Volume Analysis

Data transfer volume is a key focus of this paper. In this section, analyze the data transfer volume using the scaling process of a single stripe as an example. During the data migration phase of the S-MBR scheme, data transfer occurs only during inter-node migration and symmetric filling, resulting in a total transfer volume of $2sd'$. In the parity update phase, the transfer volume is calculated in two parts: the transfer volume for part $P_{(k',r)}$ is $k'r$, and the transfer volume for part $P_{(d'-k',r)}$ is $(d'-k')(k'+r)$. Therefore, the total data transfer volume of the S-MBR scheme can be expressed using Equation (13).

$$td_{S-MBR} = 2sd' + k'r + (d'-k')(k'+r) . \tag{13}$$

The lower bounds of the transfer volume during the data migration and parity update phases are $2sd'$ and sd'. The lower bound of the transfer volume can be expressed as Equation (14).

$$td_{Optimal} = 2sd' + d'r . \tag{14}$$

By observing Equation (13) and Equation (14), see that the difference between S-MBR and the theoretically optimal transfer volume lies in the consumption of transfer volume during the parity update phase. However, in certain cases when $d' = k'$ or $d = k$, we have $td_{S-MBR} = td_{Optimal}$, indicating that S-MBR achieves the theoretically optimal transfer volume. The transmission volume of all scaling schemes can be shown in Table 2 below.

Table 2. The theoretical number of data blocks transmitted for one stripe expansion process in the Optimal, S-MBR, RR, Scale-RS, EMBRScale scaling scheme

Scaling scheme	Data migration phase	Parity update phase	Total
Optimal	$2sd'$	$d'r$	$2sd'+d'r$
S-MBR	$2sd'$	$k'r+(d'-k')(k'+r)$	$2sd'+k'r+(d'-k')(k'+r)$
RR	$2k'd'$	$(r+k')d'$	$3k'd'+rd'$
Scale-RS	$(d'+k)s$	$(r+k')d'$	$(s+r+k')d'+ks$
EMBRScale	$2sd'$	$(r+k')d'$	$(2s+r+k')d'$

Compare the number of data blocks that need to be transferred for one stripe scaling under different parameter conditions for various scaling schemes. The results shown in Fig. 7, indicate that compared to RR, S-MBR can reduce the number of transferred data blocks by approximately 76% at its highest.

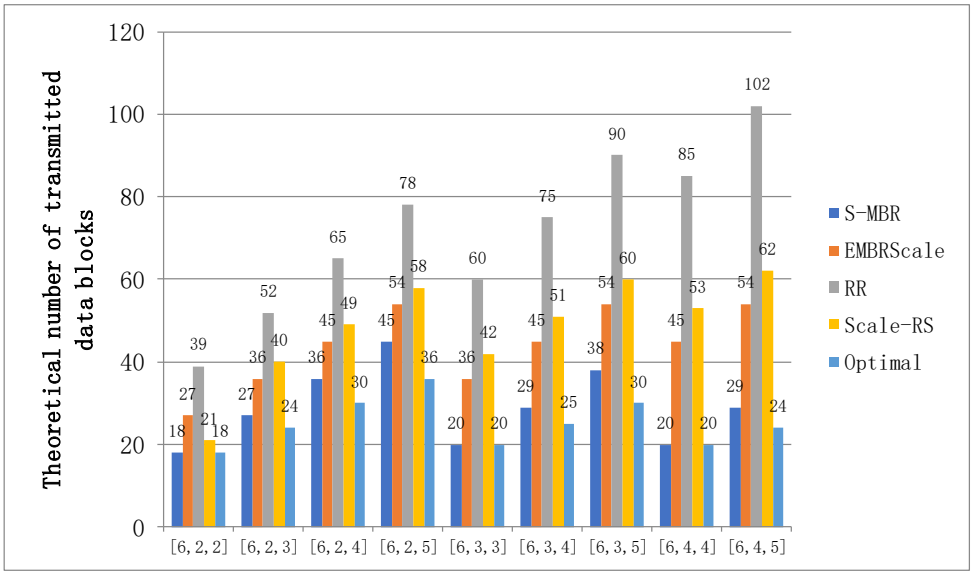

Fig. 7. The theoretical transmission volume required for one stripe scaling in the RR, EMBRScale, Scale-RS, S-MBR, and Optimal scaling schemes for all cases from [6,2] to [6,4]

5 Experimental Evaluation

5.1 Experimental Environment

The experiments in this paper were conducted on the same erasure coding test platform, which consists of 20 nodes. Except for one client and two monitor nodes, all other nodes functioned as storage nodes for Ceph OSD. Each node was equipped with an Intel Core i5-5200U 2.2GHz processor, 32GB of RAM, a 500GB solid-state drive, and a 1Gbps Ethernet card. All nodes ran the CentOS 7.5 operating system and were installed with Python 3.0 and Ceph 12 distributed storage system.

5.2 Experimental Description

To conduct comprehensive tests on various aspects of the S-MBR scaling scheme, the data in each storage node is cleared before the start of each experiment. A process of encoding first and then scaling is adopted. Since

S-MBR cannot be used for node reduction, focus on the scaling results of RR, Scale-RS, EMBRScale, and S-MBR when adding nodes.

The experiment is divided into three parts. The first part discusses the impact of fault tolerance on the scaling schemes. The second part discusses the impact of the number of nodes on the scaling schemes. The third part compares and analyzes the response time of each scaling scheme.

1) Impact of fault tolerance on scaling schemes: The file size for experiments in Fig. 8 and Fig. 9 is 1TB, with a default data block size of 24 MB. Fig. 8 shows the data transfer volume for scaling one node with different values of r (redundancy). Fig. 9 shows the data transfer volume for scaling two nodes with different values of r. From Fig. 8 and Fig. 9, it can be observed that the S-MBR scaling scheme has reduced data transfer volume by up to 83%, 93%, and 84% compared to EMBRScale, RR, and Scale-RS, respectively. As shown in Fig. 10, as the level of fault tolerance increases, all scaling schemes exhibit an upward trend in data transfer volume. However, the S-MBR scheme shows an increase in data transfer volume of only 5% and 35% compared to EMBRScale and RR, respectively. In comparison to Scale-RS, it exhibits a decrease of 3%.

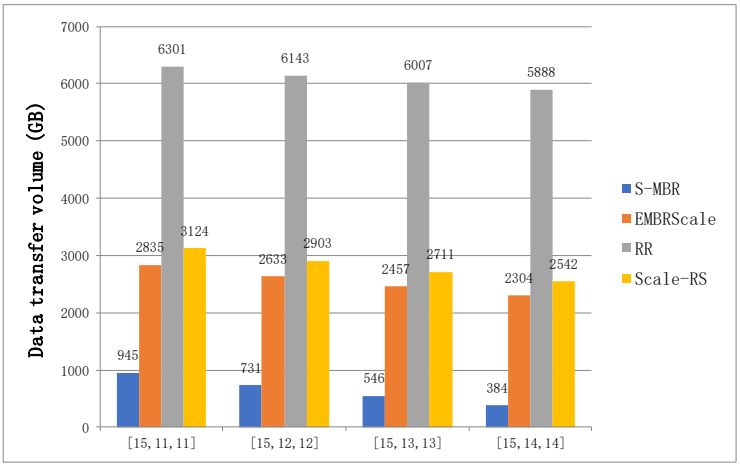

Fig. 8. The experimental results for the data transfer volume when scaling one node with RR, EMBRScale, Scale-RS, and S-MBR scaling schemes are shown for r = 1, 2, 3, and 4

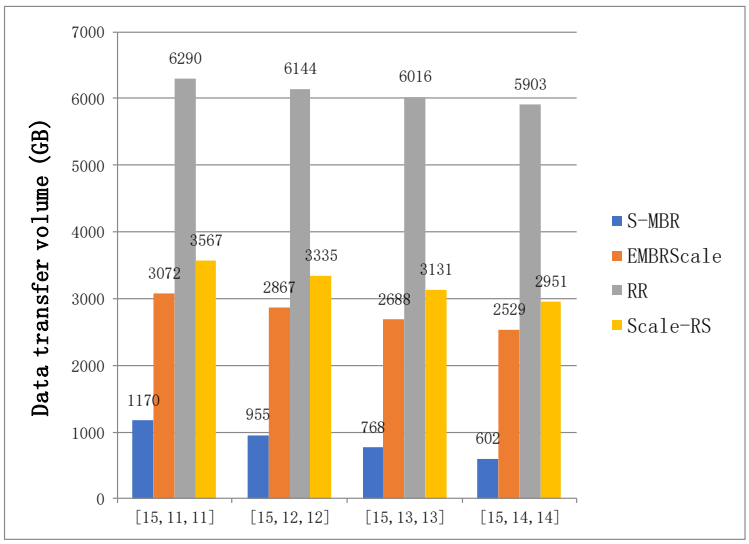

Fig. 9. The experimental results for the data transfer volume when scaling two node with RR, EMBRScale, Scale-RS, and S-MBR scaling schemes are shown for r = 1, 2, 3, and 4

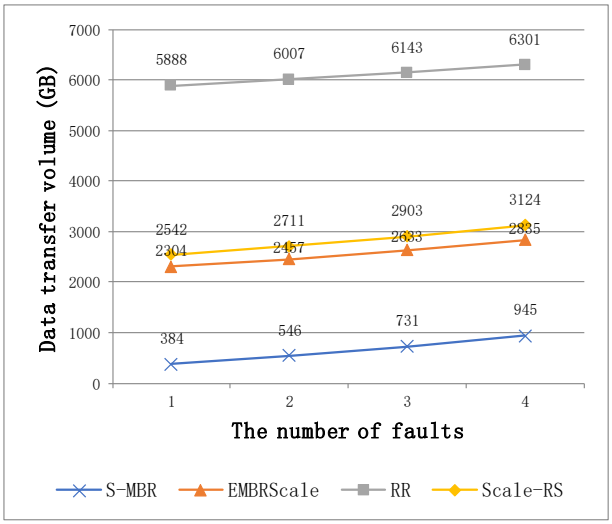

Fig. 10. When s = 1 and the number of faults (r) increases, the data transfer volume varies for the RR, EMBRScale, Scale-RS, and S-MBR scaling schemes

2) Impact of node count on scaling schemes: Fig. 11 and Fig. 12 present the results of experiments with a test file size of 50GB and a default data block size of 2MB. The encoding scheme used in Fig. 11 is [4,2,3], while Fig. 12 uses [12,10,10] as the coding scheme. Fig. 11 illustrates the data transfer volume that the cluster is scaling by adding 1, 2, and 3 nodes respectively, starting from 4 nodes. Compared to EMBRScale, RR, and Scale-RS, the S-MBR scaling scheme achieves a reduction in data transfer volume of up to 32%, 56%, and 41% respectively. Fig. 12 shows the data transfer volume that the cluster is scaling by adding 1, 2, and 3 nodes respectively, starting from 12 nodes. In comparison to EMBRScale, RR, and Scale-RS, the S-MBR scaling scheme achieves a reduction in data transfer volume of up to 73%, 88%, and 76% respectively. It can be observed that as the number of nodes increases, the advantages of S-MBR in scaling become more prominent.

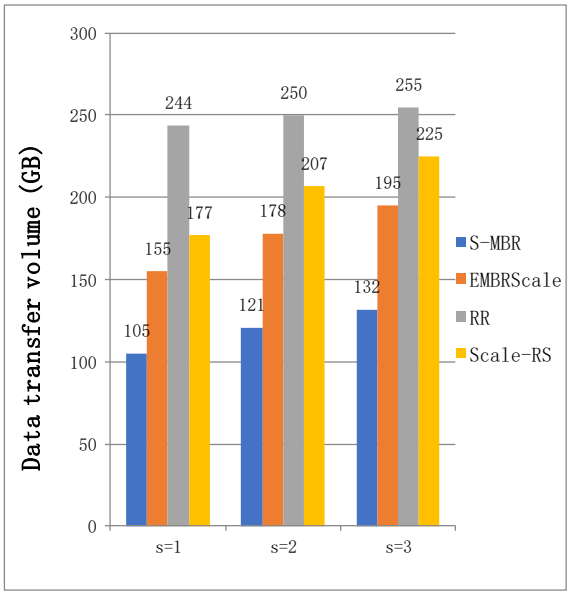

Fig. 11. The experimental results of the transmission volume that the cluster is scaling by adding 1, 2, and 3 nodes respectively, starting from 4 nodes when the encoding parameter is [4, 2, 3]

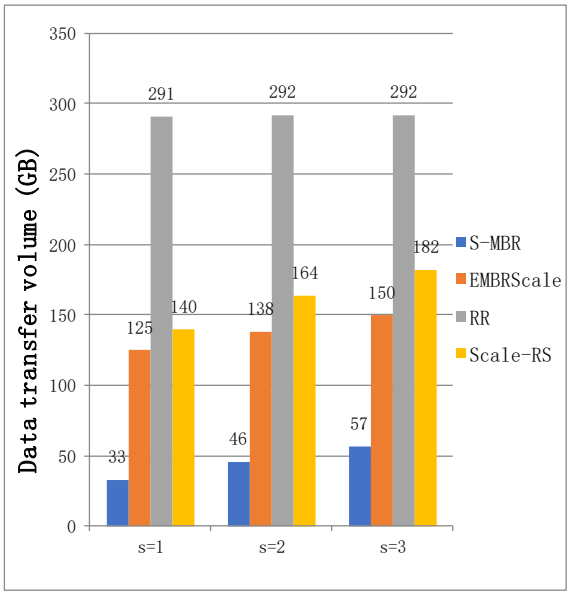

Fig. 12. The experimental results of the transmission volume that the cluster is scaling by adding 1, 2, and 3 nodes respectively, starting from 4 nodes when the encoding parameter is [12, 10, 10]

3) Response time comparison: In the experiment shown in Fig. 13, the test file size was 1GB, the data block size was set to 128KB, and the encoding parameter was [3, 2, 2]. The number of scaling nodes was set to s=2. To compare the response time of various scaling schemes, simulated real user access by running an automated script on the client side. The results showed that during the period from 0s to 400s, the response times of the different scaling schemes were similar because this phase involved data migration. From the 400s until the end, the systems entered the verification and update phase. During this phase, S-MBR performed the parity update calculations at the storage nodes, which consumed their computational resources but relieved the pressure on the client and monitor nodes. The results showed that the S-MBR scheme was significantly better than other scaling schemes, and compared to Scale-RS, it achieved an average reduction of approximately 41% in response time, as shown in Fig. 14.

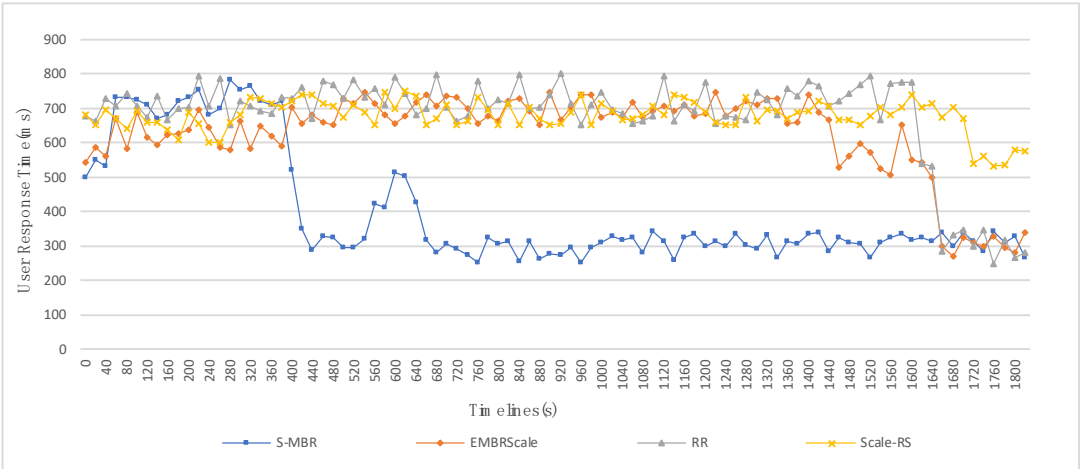

Fig. 13. Experimental results of user response time for RR, EMBRScale, Scale-RS, and S-MBR scaling schemes when the encoding parameter is [3, 2, 2] and the cluster scaled by 2 nodes

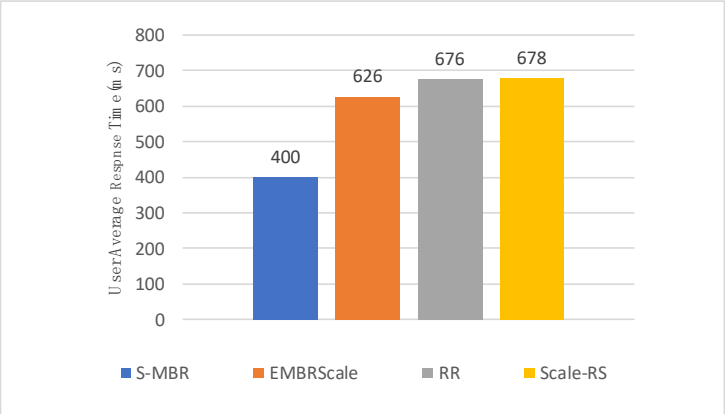

Fig. 14. Experimental results of user average response time for RR, EMBRScale, Scale-RS, and S-MBR scaling schemes when the encoding parameter is [3, 2, 2] and the cluster scaled by 2 nodes

6 Conclusion

S-MBR scheme demonstrates high practicality and applicability. Compared to other scaling schemes, S-MBR significantly reduces data transfer volume. This conclusion is drawn from a comprehensive mathematical analysis and extensive experimental data. Discuss the impact of fault tolerance and the number of nodes on scaling schemes. The results indicate that for a larger number of nodes and larger data volumes, S-MBR can reduce data transfer volume by up to 83%, 93%, and 84% compared to EMBRScale, RR, and Scale-RS, respectively. This highlights the greater advantage of S-MBR in scenarios with a larger number of nodes. With increasing fault tolerance, the additional data transfer volume in S-MBR compared to EMBRScale and RR increases by 5% and 35%, respectively, while it decreases by 3% compared to Scale-RS. Evaluate the performance of S-MBR and other scaling schemes by simulating user access patterns. The experimental results show that S-MBR achieves an average response time reduction of approximately 41% compared to Scale-RS. These advantages make S-MBR a highly practical distributed scaling scheme. However, there are some limitations to our scheme. Currently, it is only applicable to PM-MBR codes and incurs higher computational overhead on storage nodes during scaling. In the future, we will continue to optimize our approach and plan to extend its application to the entire regenerating code domain.

7 Acknowledgement

This work is supported by Sichuan Science and Technology Program, China Grant No. 24YSZH0019 and 24NSFSC0087.

References

[1] Y.-J. Wang, F.-L. Xu, X.-Q. Pei, Research on erasure code-based fault-tolerant technology for distributed storage, Chinese Journal of Computers 40(1)(2017) 236-255.
[2] K. Shvachko, H. Kuang, S. Radia, R. Chansler, The hadoop distributed file system, in: Proc. 2010 IEEE 26th symposium on mass storage systems and technologies, 2010.
[3] S.-A. Weil, S.-A. Brandt, E.-L. Miller, D.-D.-E. Long, C. Maltzahn, Ceph: A scalable, high-performance distributed file system, in: Proc. 2006 Proceedings of the 7th symposium on Operating systems design and implementation, 2006.
[4] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, S. Sankar, Row-diagonal parity for double disk failure correction, in: Proc. 2004 Proceedings of the 3rd USENIX Conference on File and Storage Technologies, 2004.
[5] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, S. Yekhanin, Erasure coding in windows azure stor-

age, in: Proc. 2012 USENIX Annual Technical Conference, 2012.
[6] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, J. Kubiatowicz, Maintenance-free global data storage, IEEE Internet Computing 5(5)(2001) 40-49.
[7] I.-S. Reed, G. Solomon, Polynomial codes over certain finite fields, Journal of the society for industrial and applied mathematics 8(2)(1960) 300-304.
[8] M. Blaum, J. Brady, J. Bruck, J. Menon, EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures, IEEE Transactions on computers 44(2)(1995) 192-202.
[9] L. Xiang, Y. Xu, J.-C.-S Lui, Q. Chang, Optimal recovery of single disk failure in RDP code storage systems, ACM SIGMETRICS Performance Evaluation Review 38(1)(2010) 119-130.
[10] L. Xu, J. Bruck, X-code: MDS array codes with optimal encoding 45(1)(1999) 272-276.
[11] C. Huang, L. Xu, STAR: An efficient coding scheme for correcting triple storage node failures, IEEE Transactions on Computers 57(7)(2008) 889-901.
[12] C. Jin, H. Jiang, D. Feng, L. Tian, P-Code: A new RAID-6 code with optimal properties, in: Proc. 2009 Proceedings of the 23rd international conference on Supercomputing, 2009.
[13] K.-V. Rashmi, N.-B. Shah, P.-V. Kumar, Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction, IEEE Transactions on Information Theory 57(8)(2011) 5227-5239.
[14] K.-V. Rashmi, N.-B. Shah, P.-V. Kumar, K. Ramchandran, Explicit construction of optimal exact regenerating codes for distributed storage, in: Proc. 2009 47th Annual Allerton Conference on Communication, Control, and Computing, 2009.
[15] J. Xue, Z.-Z. Li, Y.-L. Wang, Development of Technology of load balancing, MINI-Micro systems 24(12)(2003) 2100-2103.
[16] D.-A. Patterson, G. Gibson, R.-H. Katz, A case for redundant arrays of inexpensive disks (RAID), in: Proc. 1988 Proceedings of the 1988 ACM SIGMOD international conference on Management of data, 1988.
[17] J.-L Gonzalez, T. Cortes, Increasing the capacity of RAID5 by online gradual assimilation, in: Proc. 2004 Proceedings of the International Workshop on Storage Network Architecture and Parallel I/O, 2004.
[18] G. Zhang, W. Zheng, J. Shu, ALV: A new data redistribution approach to RAID-5 scaling, IEEE Transactions on Computers 59(3)(2010) 345-357.
[19] A. Goel, C. Shahabi, S.Y.D. Yao, R. Zimmermann, SCADDAR: An efficient randomized technique to reorganize continuous media blocks, in: Proc. 2002 Proceedings 18th International Conference on Data Engineering, 2002.
[20] C. Wu, X. He, GSR: A global stripe-based redistribution approach to accelerate RAID-5 scaling, in: Proc. 2012 41st International Conference on Parallel Processing, 2012.
[21] S.-R. Hetzler, Data storage array scaling method and system with minimal data movement, U.S. Patent 8,239,622, 2012.
[22] G. Zhang, J. Shu, W. Xue, W. Zheng, SLAS: An efficient approach to scaling round-robin striped volumes, ACM Transactions on Storage 3(1)(2007) 3-es.
[23] W. Zheng, G. Zhang, FastScale: Accelerate RAID Scaling by Minimizing Data Migration, in: Proc. 2011 9th USENIX Conference on File and Storage Technologies, 2011.
[24] C. Wu, X. He, J. Han, H. Tan, C. Xie, SDM: A stripe-based data migration scheme to improve the scalability of RAID-6, in: Proc. 2012 IEEE International Conference on Cluster Computing, 2012.
[25] G. Zhang, G. Wu, Y. Lu, J. Wu, W. Zheng, Xscale: online X-code RAID-6 scaling using lightweight data reorganization, IEEE Transactions on Parallel and Distributed Systems 27(12)(2016) 3687-3700.
[26] G. Zhang, K. Li, J. Wang, W. Zheng, Accelerate rdp raid-6 scaling by reducing disk i/os and xor operations, IEEE Transactions on Computers 64(1)(2015) 32-44.
[27] S. Gao, J. Liang, S. Wu, Y.L. Xu, A rotated deployment-based expansion scheme for raid6 distributed storage system, Computer Applications and Software 33(8)(2016) 121-125+189.
[28] L. Zhang, R. Sun, J.W. Liu, Cloned piggybacking framework for distributed storage, Journal of Xidian University 47(6)(2020) 139-147. DOI: 10.19665/j.issn1001-2400.2020.06.020.
[29] H. Zhao Y. Xu, L. Xiang, Scaling up of e-msr codes based distributed storage systems with fixed number of redundancy nodes, International Journal of Distributed and Parallel Systems 3(5)(2012) 1-12.
[30] J. Huang, X. Liang, X. Qin, P. Xie, C. Xie, Scale-RS: An efficient scaling scheme for RS-coded storage clusters, IEEE Transactions on Parallel and Distributed Systems 26(2014) 1704-1717.
[31] X. Zhang, Y. Hu, P.-P.-C Lee, P. Zhou, Toward optimal storage scaling via network coding: From theory to practice, in: Proc. 2018-IEEE Conference on Computer Communications, 2018.
[32] B.-K. Rai, V. Dhoorjati, L. Saini, A.-K. Jha, On adaptive distributed storage systems, in: Proc. 2015 IEEE interna-tional symposium on information theory, 2015.
[33] X. Zhang, Y. Hu, Efficient storage scaling for MBR and MSR codes, IEEE Access 8(2020) 78992-79002.
[34] Y. Hu, X. Zhang, P.-P.-C Lee, P. Zhou, Generalized optimal storage scaling via network coding, in: Proc. 2018 IEEE International Symposium on Information Theory, 2018.
[35] J. Hao, Y. Lu, X. Liu, S. Xia, Survey for regenerating codes for distributed storage, Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition) 25(1)(2013) 30-38.
[36] X.-H. Luo, J.-W. Shu, Summary of research for erasure code in storage system, Journal of computer research and development 49(1)(2012) 1-11.
[37] D.-S. Wei, J. Li, X. Wang, Performance Study of Exact Minimum Bandwidth Regenerating Codes in Distributed Storage, Journal of Computer Research and Development 51(8)(2014) 1671-1680.