GSR: A Global Stripe-based Redistribution Approach to Accelerate RAID-5 Scaling

Chentao Wu and Xubin He Department of Electrical & Computer Engineering Virginia Commonwealth University {wuc4, xhe2}@vcu.edu

Abstract—Under the severe energy crisis and the fast development of cloud computing, nowadays sustainability in large data centers receives more attention than ever. Due to its high performance and reliability, RAID, particularly RAID-5, is one of the most popular components and widely used in these data centers. However, a challenge on the sustainability of RAID-5 is its scalability, or how to efficiently expand/reduce the disks. The main reason causing this problem is the special layout of RAID-5 with parity blocks, which is difficult to be extended efficiently.

To address this problem, in this paper, we propose a novel redistribution approach to accelerate RAID-5 scaling, called Global Stripe-based Redistribution (GSR). The basic idea is to maintain the layout of most stripes while sacrificing a small portion of stripes according to a global view of all stripes. GSR has four main advantages: 1) It supports bidirectional RAID-5 scaling (both scale-up and scale-down); 2) GSR minimizes the overhead of scaling process, including the data migration cost, parity modification and computation cost, and the operations of metadata; 3) different from previous approaches, GSR provides high flexibility and high availability for the write requests; 4) A disk array can achieve higher capacity, performance and storage efficiency by extending more disks via GSR. In our mathematical analysis, GSR maintains uniform distribution, saves up to 81.5%I/O operations and reduces the migration time by up to 68.0%, which speeds up the scaling process by a factor of up to 3.13.

Index Terms—RAID-5; Scaling; Reliability; Scalability

I. INTRODUCTION

Redundant Arrays of Inexpensive (or Independent) Disks (RAID) [19] [4] is a popular choice to supply both high reliability and high performance storage services with acceptable spatial and monetary cost. In recent years, scalability in RAID systems is in high demand due to the following reasons,

- 1) To meet the requirements of larger capacity and higher throughput [22]. Adding more disks into an existing disk array is a cost-performance effective solution.
- To fulfill the needs of energy saving. By removing some inefficient disks of a disk array, the power consumption can be reduced to be cost-effective.
- 3) To match the increasing demands of online applications. Typically, RAID is widely used in various online services such as cloud computing [1]. High scalability not only satisfies the sharp increasing on user data in various online applications [7], but also avoids the extremely high downtime cost [18].
- Necessity in data centers. RAID-based architectures are widely used for clusters and large scale storage systems,

where scalability plays a significant role in these systems [15] [20].

Among different RAID layouts, RAID-5 is one of the most significant forms and widely used in large scale data centers. Recently, research on RAID-5 scaling¹ receives much attention and many approaches are proposed in this area, including **R**ound-**R**obin (RR) [9] [17] [23], Semi-RR [8], ALV [24], MDM [12], etc.

However, there are two challenging issues on RAID-5 scaling. The first challenge is the high overhead of the scaling process. In traditional RR-based approaches [9] [17] [23], almost all data are migrated and thus all parities should be recalculated and modified. It also causes additional updates on metadata. Semi-RR [8] suffers from unbalanced data distribution. ALV [24] aggregates the migration I/O and decreases the total number of redistribution I/Os, but it cannot decrease the total number of access to data blocks. Although MDM [12] can decrease the data movements and the number of parity modification, it causes some new problems. Compared to RR and Semi-RR approaches, the storage efficiency and the performance are not improved after scaling using MDM. Furthermore, MDM adds another parity into the original RAID-5 layout, which makes the data mapping more complicated when read and write requests are processed.

The second challenge is the support on both scale-up (adding disks) and scale-down (removing disks). Except RR, other approaches only support scale-up.

To address the above challenging issues, in this paper we propose Global Stripe-based Redistribution (GSR), a new approach to RAID-5 scaling. Based on a global view on all stripes, a proper number of stripes are retained in GSR while others are selected to fill the empty blocks in the extending disk(s). GSR has the following advantages:

- GSR provides bidirectional scaling by adding or removing any number of disks to/from a RAID5.
- GSR not only minimizes the total number of migration and modification I/Os, but also reduces the parity computation cost and the operations of metadata. It dramatically accelerates the scaling process of RAID-5.

¹In this paper, scaling is a process to add disks (scale-up) to or remove disks (scale-down) from an existing disk array.

TABLE I Symbols in This Paper

Parameters & Symbols	Description
n	number of disks in a disk array (before scaling)
m	scaled number of disk(s)
	(<i>m</i> is negative when scale-down)
В	total number of data blocks
S, S'	total number of stripes (before/after scaling)
i, X, Y	stripe ID (row ID) before scaling
j	disk ID (column ID) before scaling
i'	stripe ID (row ID) after scaling
j'	disk ID (column ID) after scaling
P_i	parity block in Stripe <i>i</i> before scaling
$Q_{i'}$	parity block in Stripe i' after scaling
D_k	data block with ID is k before scaling
$D_{k'}$	data block with ID is k' after scaling
S_s	stripe set ID
N_s	total number of stripe sets
S_r	total number of retained OUS/NUS
S_m	total number of remapped OUS/NUS
S_d	total number of destructed OUS/NUS
N_d	total number of migrated data blocks
N_p	total number of modified parity blocks
R_d	data migration ratio
R_p	parity modification ratio
R_m	metadata modification ratio
T_b	access time of a read/write request to a block
T_m	migration time

• By efficiently adding more disks to a disk array, the performance and storage efficiency are improved.

The rest of this paper continues as follows: Section II discusses the motivation of this work and details the background of existing scaling methods. Global Stripe-based Redistribution (GSR) approach is described in detail in Section III. Section IV provides quantitative analysis on performance and scalability. Finally we conclude the paper in Section V.

II. BACKGROUND AND MOTIVATION

To improve the efficiency of the RAID-5 scaling, different approaches have been proposed. In this section we discuss the background of the scaling schemes, problems in existing schemes and the motivations of our work. To facilitate our discussion, we summarize the symbols used in this paper in Table I.

A. Desired Features to Scale RAID-5

To scale a disk array, some data need to be migrated to achieve a balanced data distribution. During the data migration process, we need to keep an approximate evenly distributed workload and minimize the data/parity movement. Combined with existing scaling approaches in RAID-5, the following six features are desired for efficient scaling,

- Feature 1 (*Uniform Data & Parity Distribution*): Each disk has the same amount of data and parity blocks to maintain an evenly distributed workload.
- Feature 2 (*Minimal Data & Parity Migration*): By increasing/decreasing m disks to a RAID-5 system with n disks storing B data blocks, the expected total number of

data movement is $\frac{mB}{m+n}$ (scale-up) or $\frac{B}{|m|}$ (scale-down). Parity movement should also be minimized.

- Feature 3 (*Fast Data Addressing*): The locations of blocks in the array can be easily computed at low cost.
- Feature 4 (*Minimal Parity Computation & Modification*): A movement on data block could cause modification cost on its corresponding original parity and computation cost on the new parity, so the original parity chain should be reserved as much as possible.
- Feature 5 (*High Flexibility on Scaling Process*): Flexible schemes should be provided for scaling process with various numbers of *m* and *n*.
- Feature 6 (*Better Storage Efficiency and Performance by Extending More Disks*): In RAID-5, the storage efficiency is $\frac{n-1}{n}$. By adding *m* disks (m > 0), the storage efficiency is improved $(\frac{n+m-1}{n+m} > \frac{n-1}{n})$. The write performance and throughput should also be increased after scaling [22].

B. Existing Fast Scaling Approaches

Existing approaches to improve the scalability of RAID-5 system include Round-Robin (RR) [9] [17] [23], Semi-RR [8], ALV [24], MDM [12], FastScale [25], etc. To clearly illustrate various strategies in RAID-5, the default data and parity distribution is right-asymmetric².

1) Round-Robin (RR): As shown in Figure 1, traditional RR scaling approach is based on round-robin order where nearly all data are migrated except the first stripe (nearly 100% data migration). Obviously, all parities need to be regenerated after data migration. RR is simple to implement on RAID-5 and has been used in some products [5] [13]. However, the overhead is high due to the large data migration.

Gonzalez *et al.* [9] found that RR achieves better performance in left-symmetric or right-symmetric distribution, where Gradual Assimilation (GA) algorithm is used on RAID-5 scaling (as shown in Figure 2). A little more data blocks can be reserved without any change, but all parities still need to be modified and recalculated after data migration.

Based on RR approach, Brown [17] designed a reshape toolkit in a Linux MD driver (MD-Reshape), which writes mapping metadata using a fixed-size window. Due to the limitation of RR approach, metadata are updated frequently by calling a MD-Reshape function, which is inefficient.

2) Semi-RR: Semi-RR [8] is proposed to decrease high migration cost in RR scaling as shown in Figure 3. Unfortunately, by extending multiple disks, the data distribution is not uniform after scaling [8]. It can easily lead to load balancing problem, which is an important issue in disk arrays [14] [10].

3) ALV: ALV [24] is shown in Figure 4. Different from RR-based approaches, ALV changes the movement order of migrated data and aggregates these small I/Os. However, ALV is essentially based on round-robin order and thus cannot

²There are many layouts of RAID-5 based on the placement of parity blocks. Typically four types of data and parity distribution are preferred, left-symmetric, left-asymmetric, right-symmetric and right-asymmetric [16].

	Bef	ore Sc	aling		After Scaling						
Disk0	Disk1	Disk2	Disk3	Disk4	Disk0	Disk1	Disk2	Disk3	Disk4		
PO	0	1	2		Q0	0	1	2	3		
3	P1	4	5		4	Q1	5	6	7		
6	7	P2	8	\square	 8	9	Q2	10	11		
9	10	11	P3		12	13	14	Q3			
P4	12	13	14					•••	Q4		

(a) RAID-5 scaling from 4 disks to 5 disks (all data blocks need to be migrated except blocks 0, 1 and 2).

	Bef	ore Sca	aling	After Scaling				
Disk0	Disk1	Disk2	Disk3 (removed)	Disk0	Disk1	Disk2		
PO	0	1	2	Q0	0	1		
3	P1	4	5	2	Q1	3		
6	7	P2	8	4	5	Q2		
9	10	11	P3	Q3	6	7		
			J	8	Q4	9		
				10	11	Q5		

(b) RAID-5 scaling down from 4 to 3 disks (all data blocks need to be migrated except blocks 0 and 1).

Fig. 1. Round-Robin approach.

	Befo	re Sca	ling		After Scaling				
Disk0	Disk1	Disk2	Disk3	Disk4	Disk0	Disk1	Disk2	Disk3	Disk4
	PO	2	1	0	Q0	3	2	1	0
	3	P1	5	4	7	Q1	6	5	4
	7	6	P2	8	11	10	Q2	9	8
	11	10	9	P3		14	13	Q3	12
	P4	14	13	12				•••	Q4

Fig. 2. RAID-5 scaling from 4 to 5 disks using GA algorithm (nearly all data blocks need to be migrated except several special blocks 0, 1, 2, 4, etc.).

	Bef	ore Sca	aling			After Scaling					
Disk0	Disk1	Disk2	Disk3	Disk4	Disl	k0	Disk1	Disk2	Disk3	Disk4	
PO	0	1	2		Q	9	0	1	2	3	
3	P1	4	5		6		Q1	4	5	7	
6	7	P2	8		9		10	Q2	8	11	
9	10	11	P3				12	13	Q3	14	
P4	12	13	14			•				Q4	

Fig. 3. RAID-5 scaling from 4 to 5 disks using Semi-RR approach (many blocks remain in the original disks by changing the metadata, e.g., blocks 6, 10 and 13).

decrease the total I/Os caused by data migration and parity modification.

	Befo	re Sca	ling			After Scaling					
Disk0	Disk1	Disk2	Disk3	Disk4	Dis	0 Disk1	Disk2	Disk3	Disk4		
	PO	0	1	2	Q	0 0	1	2	3		
	3	P1	4	5	4	Q1	5	6	7		
	6	7	P2	8	8	9	Q2	10	11		
	9	10	11	P3	1:	2 13	14	Q3			
	P4	12	13	14				•••	Q4		

Fig. 4. RAID-5 scaling from 4 to 5 disks using ALV approach (all data blocks need to be migrated).

4) *MDM*: MDM [12] eliminates the parity modification/computation cost and decreases the migration cost, however it causes new problems. For example, as shown in Figure 5, blocks 0, 4 and 8 are moved to the new disk and their original positions are served as a new parity (P4), which leads to an uneven data and parity distribution. In MDM approach, all parity blocks are maintained but it cannot improve the storage efficiency by adding more disks. The layout after scaling becomes much more complex than a typical RAID-5. Because the number of data blocks in a parity chain remains unchanged, the performance is limited.

	Bef	ore Sc	aling			After Scaling					
Disk0	Disk1	Disk2	Disk3	Disk4		Disk0	Disk1	Disk2	Disk3	Disk4	
PO	0	1	2			PO	P4	1	2	0	
3	P1	4	5			3	P1	12	5	4	
6	7	P2	8		ĺ→	6	7	P2	13	8	
9	10	11	P3		ĺ	9	10	11	P3	14	
P4	12	13	14		ĺ				•••		

Fig. 5. RAID-5 scaling from 4 to 5 disks using MDM approach.

5) *FastScale:* FastScale [25] is the latest RAID-0 scaling approach with low overhead and high performance. However, as shown in Figure 6, it cannot be used in RAID-5.

	Befe	ore Sca	aling		After Scaling						
isk0	Disk1	Disk2	Disk3	Disk4	Disk0	Disk1	Disk2	Disk3	Disk4		
0	1	2				1	2	0			
3	4	5					5	3	4		
6	7	8			6			7	8		
9	10	11			9	10			11		
12	13	14			12	13	14				

Fig. 6. RAID-0 scaling from 3 disks to 5 disks using FastScale approach.

Except for the above scaling approaches, some RAID-based systems focus on the scalability issue. In 1990s, HP AutoRAID [21] permits an online expansion of disk array. Later, several RAID-based architectures [15] [20] are proposed for large scale storage systems, and scalability is one of the most significant impacts in these systems. Brinkmann *et al.* [3] gives mathematical analysis on a storage system by adding several disks. Franklin *et al.* [6] introduces a feasible method to support extension of RAID systems, but it needs an additional disk as spare space. Recently, with the support of different file systems, RAID-Z [2] and HDFS RAID [11] achieve acceptable scalability in distributed storage systems.

C. Our motivation

We summarize the existing scaling approaches in Table II. Although existing scaling approaches offer some advantages, they have some drawbacks. First, previous approaches cause high overhead on the scaling process, including high overhead on data migration, high parity modification, XOR calculations and updates on metadata. Second, MDM has low migration cost, but it cannot improve the performance and storage

 TABLE II

 Summary on Various Fast Scaling Approaches in RAID-5 (Features 1-6 Come From Section II-A)

Name	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Down-scale support?	Others
RR		X	\checkmark	X	X	\checkmark	\checkmark	none
Semi-RR	×	×	\checkmark	×	×	\checkmark	×	none
ALV		×	\checkmark	×	×	\checkmark	×	aggregate small I/Os
MDM	×	\checkmark	\checkmark	\checkmark	×	×	X	low storage efficiency
GSR	\checkmark	high availability and flexibility						

efficiency via scaling. The last problem is the reliability issue, particularly on moving data during the scaling process.

In summary, existing scaling approaches are insufficient to scale a RAID5 efficiently, which motivates us to present a new approach, GSR, to acieve efficient RAID scaling.

III. GSR APPROACH

In this section, Global Stripe-based Redistribution (GSR) approach is designed to accelerate RAID-5 Scaling. The purpose of GSR is to minimize the data migration, parity modification and computation cost from a global view on all stripes, not limited to operations on any single data/parity element as Round-Robin [9] [17] [23].

Except for reducing the overhead of scaling process, GSR retains the original data and parity layout of the RAID-5 (unlike the MDM approach [12]), which achieves better performance after scaling.

To clearly illustrate the stripes before/after scaling, we define four types of stripes as follows,

- Old Used Stripe (OUS): A used stripe before scaling.
- Old Empty Stripe (OES): An empty stripe before scaling.
- New Used Stripe (NUS): A used stripe after scaling.
- New Empty Stripe (NES): An empty stripe after scaling.

A. Overview of GSR

GSR is shown in Figure 7, which is a stripe-level scaling approach. The data movements in scale-down (removing disks) is in an opposite direction of scale-up (adding disks). According to the difference on parity chains before/after scaling, some stripes with shorter parity chains are retained in the original disks, while the others are destructed for migration. Based on different functions, the stripes with shorter parity chains are further divided into three categories in GSR:

- Retained OUS/NUS (Stripes 0-2 with shorter parity chains in Figure 7): all data and parity blocks are retained in a same disk. The parity blocks will be modified if data blocks are migrated into (or removed from) the corresponding parity.
- **Remapped OUS/NUS** (Stripes 3-5 with shorter parity chains in Figure 7): all data blocks are retained in a same disk by remapping to a new stripe.
- **Destructed OUS/NUS** (Stripes 6-9 with shorter parity chains in Figure 7): all data blocks are migrated to another disk(s). In each destructed OUS/NUS, the blocks are



(b) Scale-down (removing disks).

Fig. 7. GSR approach for RAID-5 scaling.

migrated to the new disk(s) for scale-up or the remaining disk(s) for scale-down.

GSR abides by the following four steps,

Step 1 (Identification): Identify the disk array before scaling. Check the free space of each disk (including new disk(s)) and acquire the related parameters, such as m and n.

Step 2 (Stripe Distribution): Calculate the amount of the retained, the remapped and the destructed OUS/NUS.

Step 3 (Stripe Processing): Handle the retained, the remapped and the destructed OUS/NUS concurrently. Reliability and availability schemes are provided.

(For retained OUS/NUS): Update the stripe ID;

(For remapped OUS/NUS): Remap all data blocks and distribute new stripe IDs;

(For destructed OUS/NUS): Migrate all data blocks and distribute new stripe IDs.

Step 4 (Parity Processing): Modify all parities.

According to these four steps, in Figure 7(a), we take RAID-5 scaling from 3 to 5 disks as an example (n = 3, m = 2) and the total number of stripes is 10. After identification, we calculate the amount of retained, remapped and destructed OUS which are 3, 3, and 4, respectively. In the stripe processing step, blocks 6-11 are remapped and the metadata information are updated, blocks 12-19 are migrated to the new disks. The corresponding stripe IDs are updated accordingly. Finally, we modify the parities Q0-Q4. For example, $Q_0 = P_0 \oplus D_{14} \oplus D_{17}$.

As shown in Figure 7(b), scale-down is the reverse process of scale-up. In this paper, we only present the theorems, equations, algorithms and schemes on scale-up (adding disks), the related theorems and equations on scale-down (removing disks) can be easily derived by similar methods or through mathematic transformations, and are not presented here due to the page limit.

B. Scaling Process in GSR

Section III-A describes the process to scale a RAID-5 of n disks by m disks. In this section, we give detailed description of the scaling process. To simplify the description, the default data and parity distribution in RAID-5 is right-symmetric or right-asymmetric, similar equations can be derived for the left-symmetric or left-asymmetric distribution.

Figure 7(a) shows a simple scale-up example. Actually, for large amount of stripes, a detailed scaling process is shown in Figure 8, which presents multiple stripe sets after scaling and each stripe set consists of m + n stripes.

1) Stripe Distribution: The portion of various types of stripes are based on the following theorem,

Theorem 1: In GSR approach, the ratio among the retained, remapped and destructed OUS is

$$\frac{n}{m+n-1}: \frac{mn}{(n-1)(m+n-1)}: \frac{m}{n-1}$$

Proof: Based on the layout of RAID-5, each stripe has (n - 1) data blocks before scaling and (n + m - 1) data blocks after scaling. The total number of data blocks remains unchanged,

$$\begin{cases} B = (n-1)S\\ B = (m+n-1)S' \end{cases}$$
(1)

The total number of stripe set is,

$$N_s = \frac{S'}{m+n} = \frac{B}{(m+n)(m+n-1)}$$
(2)

Each stripe set contains n retained OUS and $m*\frac{n}{n-1}$ remapped OUS, obviously,

$$S_r = n * N_s = \frac{nB}{(m+n)(m+n-1)}$$
(3)



Fig. 8. RAID-5 scaling from 3 to 5 disks using GSR approach (multiple stripe sets after scaling with n = 3 and m = 2).

$$S_m = |m| * \frac{n}{n-1} * N_s = \frac{mnB}{(m+n)(n-1)(m+n-1)}$$
(4)

The remaining stripes are destructed OUS,

$$S_d = S - S_r - S_m = \frac{mB}{(m+n)(n-1)}$$
(5)

According to Equations 3, 4 and 5, the ratio among the retained, remapped and destructed OUS is $\frac{n}{m+n-1}$: $\frac{mn}{(n-1)(m+n-1)}:\frac{m}{n-1}$. \blacksquare Obviously, in Figure 8, for the stripe ID of the remapped OUS and destructed OUS, $X = S_r, Y = S_r + S_m$.

2) Stripe Processing: Different strategies are applied to various types of stripes in the stripe processing step. Assuming that the stripe ID and disk ID of an OUS before scaling are i and j, the corresponding stripe ID and disk ID after scaling are i' and j'.

2.1) For Retained OUS: The stripe ID will be changed for retained OUS. Based on Theorem 1, the following equation can be derived,

$$i' = (m+n) * \left\lfloor \frac{i}{n} \right\rfloor + (i \mod n) \tag{6}$$

For example, as shown in Figure 8, if we need remap Stripe 5 before scaling, first we should calculate the stripe set ID $(\lfloor \frac{i}{n} \rfloor = \lfloor \frac{5}{3} \rfloor = 1)$. Second we have the corresponding stripe ID after scaling which is 7 $(1 * 5 + 5 \mod 3 = 7)$.

 TABLE III

 DIAGONAL ORDER OF DATA MIGRATION USING GSR IN FIGURE 8

Einst Stains Sat	2Y+2	2Y+5	2Y+1	2Y+4
First Surpe Set	2Y	2Y+3	2Y+8	2Y+11
Cocond String Sat	2Y+7	2Y+10	2Y+6	2Y+9
Second Stripe Set	2Y+14	2Y+13	2Y+12	2Y+15

It is also clear that the corresponding disk ID for all data blocks in the retained OUS remains unchanged (j' = j).

2.2) For Remapped OUS: For data blocks in remapped OUS, the key problem is to determine their corresponding positions. Assuming that the corresponding stripe set ID of a data block is denoted by S_s , it can be calculated by,

$$S_s = \left\lfloor \frac{i - S_r}{m} * \frac{n - 1}{n} \right\rfloor \tag{7}$$

Suppose the related data block after scaling (in Stripe i' with disk ID j') and we have the following equation,

$$\begin{cases} i' = (m+n) * S_s + n + \left(\left\lfloor \frac{i*(n-1)+j-S_r*(n-1)}{n} \right\rfloor \mod m \right) \\ j' = j \end{cases}$$
(8)

For example, as shown in Figure 8, if we want to remap the Block (2X+9) before scaling, first we should calculate the Stripe Set ID $\left(\left\lfloor \frac{i-S_r}{m} * \frac{n-1}{n} \right\rfloor = \left\lfloor \frac{4}{2} * \frac{2}{3} \right\rfloor = 1\right)$. Second we have the corresponding Stripe ID after scaling which is 9 (1 * 5 + 3 + 3 mod 2 = 9).

2.3) For Destructed OUS: Typically, GSR processes the blocks in destructed OUS for every n stripes to ensure high reliability. As shown in Figure 8, Stripe Y to Stripe (Y+2) are distributed to the new disks simultaneously (6 data blocks are processed together). Thus for a data block in the destructed OUS, the range of a stripe set ID S_s after scaling is,

$$\left\lfloor \frac{(i-S_r-S_m)*(n-1)+j}{m*(m+n-1)} \right\rfloor - 1 \le S_s
\le \left\lfloor \frac{(i-S_r-S_m)*(n-1)+j}{m*(m+n-1)} \right\rfloor + 1$$
(9)

GSR first calculates the ranges of stripe ID and disk ID blocks in the destructed OUS,

$$\begin{cases} S_s * (m+n) \le i' \le (S_s+1) * (m+n) \\ n \le j' \le n+m-1 \end{cases}$$
(10)

Then GSR migrates the data blocks in diagonal order as shown in Table III and Algorithm 1.

Regrading the stripe processing, we have the following theorem on the total number of data movements,

Theorem 2: In GSR approach, the total number of migrated data blocks is $\frac{mB}{m+n}$.

Proof: For each stripe set, m * (m + n) - m data blocks are migrated (as shown in Figure 8), so the total number of data blocks to be moved is,

$$N_d = N_s * [m * (m+n) - m] = \frac{mB}{m+n}$$
(11)

Algorithm 1: Get the Blocks in Diagonal Ordering

/*Get the data blocks in destructed OUS for every *n* stripes*/ *k*: a random integer *i*: stripe ID in *n* stripes, $0 \le i \le n - 1$ *j*: disk ID, $0 \le j \le n - 1$ forall the $k = 1; k \le n - 1; k + 4$ do forall the $j = 0; j \le n - 1; j + 4$ do $i = (j + k) \mod n;$ if i! = j (/*right-symmetric or right-asymmetric*/) then | get the block in stripe *i* and disk *j*; end else | break; end end end

3) **Parity Processing:** In our scaling process, each parity is modified only once, saving the modification and computation cost of parity blocks. The total number of modified parities is,

$$N_p = 1 * S' = \frac{B}{m+n-1}$$
(12)

According to the examples in the last subsection, by extending *m* disks, the length of parity chains is increased by a value of *m*. Thus *m* XOR calculations are taken for each modified parity and the total number of XOR calculations is $\frac{mB}{m+n-1}$.

C. Data Addressing Algorithm

In RAID scaling, a critical issue is to map the address of a block before scaling to its address after scaling. We propose the filowing data addressing algorithm in Algorithm 2 to calculate the addresses, which is a fast addressing method and can be easily implemented.

Algorithm 2: Data Addressing Algorithm of GSR
Calculate the amount of the retained OUS (S_r) , the remapped OUS (S_m) and the destructed OUS (S_d) .
if data or parity block is in retained OUS $(0 \le i < S_r)$ then calculate i' based on Equation 6, $j' = j$.
if data block is in remapped OUS $(S_r \le i < S_r + S_m)$ then data block is remapped according to Equation 8.
end if data block is in destructed OUS $(S_r + S_m \le i < S_r + S_m + S_d)$
then (1) specify the address range based on Equations 9 and 10; (2) retrieve the data blocks in diagonal order (similar to Algorithm
(3) distribute new addresses sequentially. forall the $i' = 0; i' < (S_r + S_m + S_d) * \frac{n-1}{1-1}; i' + d0$
forall the $j' = n; j' \le n + m - 1; j' + do$ if $j'! = i'$ (/*right-symmetric or right-asymmetric*/) then distribute the address in stripe i' and disk j' ; end
end end

D. Properties of GSR

Section II-A and Table II list six desired features on RAID-5 scaling. Our GSR satisfies all these features. From the discussions in Section III-B and III-C, GSR satisfies the features 1-3, which guarantee uniform data and parity distribution, minimal the movements of data/parity elements and fast data addressing. Features 4 and 6 are discussed in detail in Section IV. GSR also satisfies Feature 5 (felxible) as explained below.

1) High Flexibility (Feature 5): Most previous approaches are not flexible to adapt RAID-5. Round-Robin approach has various effects on different data and parity distribution of RAID-5 [9]. FastScale should consider different cases according to the number of extending disk(s) (value of m) [25]. From the examples shown in Figures 7 and 8, our GSR performs well in any data and parity layouts of RAID-5 and any value of m. Therefore GSR demonstrates higher flexibility than other approaches due to the global view of all stripes.

In addition to satisfy all these desired features of RAID-5 scaling, our GSR also demonstrates high avaiability. **2) High Availability:** GSR approach provides an availability scheme when no space is available for new write requests,

- If a new empty stripe (NES) is available with the corresponding Stripe ID i' ≥ S_r + S_m + S_d, write the NES sequentially.
- In the scaling process, if no NES is available and a stripe set is available with empty blocks, GSR first completes the stripe and parity processing in this stripe set, and then writes the empty blocks for the new requests.

Let's take an example. Assume all disks have the same capacity in a disk array based on RAID-5 (including the extended disks), before scaling, 20% space is available and 80% space are used for storing data. If we expand the disk array according to GSR in Figure 8, we can provide more than 69% free space available for write requests³.

IV. SCALABILITY ANALYSIS

In this section, we evaluate the scalability of GSR compared to other approaches to show its advantages on scalability.

A. Evaluation Methodology

We compare GSR approach to Round-Robin (RR) [9] [17] [23], Semi-RR [8], ALV [24] and MDM [12] approaches. FastScale [25] is not compared because it cannot support RAID-5.

In our comparison, a two-integer tuple (n, m) denotes scaling a RAID5 of n disks by m disks. A negative number of m means to remove |m| disks from the array (scale-down). Our comparisons include:

1) Scale-up (adding disks) among various approaches: comparisons among RR, Semi-RR, ALV, MDM and GSR, several representative values of n and m are chosen;

$$3\frac{20\%*\frac{3+2}{3}+80\%*\frac{3-1}{3}*\frac{2}{3+2-1}}{20\%+1*\frac{2}{3}}\approx 69\%.$$

Bidirectional RAID-5 scaling (both scale-up and scale-down): comparisons between GSR and RR. An original RAID-5 array with six disks (n = 6) by adding or reducing disks whithin a range from −3 to 3 (m = 0,±1,±2,±3).

We define **Data Migration Ratio** (R_d) as the ratio of the number of migrated data/parity blocks to the total number of data blocks. **Parity Modification Ratio** (R_p) denotes the ratio of the number of modified parity blocks to the total number of data blocks, which is caused by the data/parity migration. **Metadata Modification Ratio** (R_m) is used to denote the ratio of the number of modified metadata to the total number of data blocks.

For example, for scale-up, we have the data migration ratio of GSR based on Equation 11 (m > 0),

$$R_d = \frac{N_d}{B} = \frac{m}{m+n} \tag{13}$$

According to Equation 12, all parity blocks need to be modified using GSR and the parity modification ratio is,

$$R_p = \frac{N_p}{B} = \frac{1}{m+n-1}$$
(14)

From the stripe processing in GSR, all data blocks in the retained OUS keep their original metadata information, and only the metadata of the blocks in the remapped or destructed OUS are changed. Therefore, the total number of modified metadata is (the total number of data and parity blocks minus data and parity blocks in retained OUS),

$$B * \frac{n+m}{n+m-1} - S_r * n = \frac{(m^2 + 2mn) * B}{(m+n)(m+n-1)}$$

The metadata modification ratio is,

$$R_m = \frac{m^2 + 2mn}{(m+n)(m+n-1)}$$
(15)

In RAID-5 scaling, each data migration only costs two I/O operations, and the modification cost of each parity also causes two I/Os. According to the data migration ratio (R_d) and parity modification ratio (R_p) , the total number of I/O operations is $2 * N_d + 2 * N_p = 2 * (R_d + R_p) * B$.

If we ignore the computation time and assume the same access time on a read or write request to a block using various RAID-5 scaling approaches (denoted by T_b), suppose the migration I/O can be processed in parallel on each disk, the migration time T_m using GSR approach for scale-up is (Assume the migration time of each original disk is T_1 and the migration time per extended disk is T_2),

$$T_m = \max(T_1, T_2), where \begin{cases} T_1 = (N_d + \frac{n}{m+n} * N_p) * T_b/n \\ T_2 = (N_d + \frac{m}{m+n} * N_p) * T_b/m \\ (16) \end{cases}$$

In our analysis, the default data and parity distribution of RAID-5 is right-asymmetric. Similar results can be derived for other distributions.

B. Numerical Results

In this section, we give the numerical results of scalability using different scaling approaches.

1) Data Distribution: Regarding data distribution, we use the coefficient of variation as a metric to examine whether the distribution is even or not as other approaches [8] [25]. A small value of the coefficient of variation means highly uniform distribution. From the introduction in Section II, Semi-RR and MDM suffer from I/O load balancing problem, which are chosen to be compared with GSR.

The results are shown in Figure 9. We notice that semi-RR and MDM cause excessive oscillation by up to 46.8%, which fail to satisfy Feature 1 (uniform distribution).



Fig. 9. Data distribution under various numbers of extended disk(s) (0 $\leq m \leq 7, n = 3$).

2) Storage Efficiency: Second, we compare the storage efficiency between GSR and MDM as shown in Figure 10. Compared to MDM, it clearly shows that GSR saves the disk space by up to 23.3%.



Fig. 10. Storage efficiency under various numbers of extended disk(s) ($0 \le m \le 7, n = 3$).

In the following Figures 11-16, the numbers (n, m) in X-axis denote to scale a disk array of n disks by m disks. To the right of each figure, we also briefly list the results of scale-down when m is a negative number.

3) Data Migration Ratio: Third, we calculate the data migration ratio (R_d) among various fast scaling approaches as shown in Figure 11. It is obvious that GSR has the minimal data migration ratio as Semi-RR and MDM.

4) Parity Modification Ratio: Fourth, parity modification ratio (R_p) among various fast scaling approaches is presented in Figure 12. Compared to RR, Semi-RR and ALV, GSR reduces the parity modification ratio by up to 87.5%.

5) Metadata Modification Ratio: Fifth, Figure 13 shows the metadata modification ratio (R_m) under various scenarios. Compared to other fast scaling approaches (excludes MDM), GSR reduces the parity modification ratio by up to 69.2%.

6) Computation Cost: Next, we calculate the computation cost in terms of the total number of XOR operations under various cases as shown in Figure 14. RR-based approaches have similar computation cost. Except for MDM, we notice that GSR scheme sharply decreases more than 66.7% computation cost compared to other approaches. Figure 14(b) shows that GSR performs better for scale-up (adding disks), which is reasonable because the effects on the optimization of XOR calculations are dropped under the the fewer number of disks and the shorter parity chains.

7) Total number of I/O Operations: The results are shown in Figure 15. Compared to RR, Semi-RR and ALV, GSR reduces up to 81.5% I/Os during the scaling process.

8) Migration Time: Next, we evaluate migration time which is shown in Figure 16 (the migration time of GSR is based on Equation 16). Due to the uneven data distribution, the migration time of Semi-RR and MDM cannot be calculated by our methodology. Compared to other approaches, GSR performs well in multiple disks extension and decreases the migration time by up to 68.0%, which can speed up the scaling process by a factor of up to 3.13. Compared to RR, GSR is also efficient on scale-down as shown in Figure 16(b).

9) Throughput: Finally, we use the maximum throughput of RAID-5 (n = 3) as the baseline (100%), the expected maximum I/O throughput after scaling can be calculated as shown in Figure 17. We can see a clear performance gap between GSR and MDM approach. Compared to MDM approach, GSR can improve the write performance of storage system up to 15.2%.



Fig. 17. Expected maximum I/O throughput after scaling under various numbers of extended disk(s) $(0 \le m \le 7, n = 3, 100\%)$ write mode with uniform data access).

C. Analysis

From the results in Section IV-B, compared to RR, Semi-RR and ALV, GSR has great advantages. There are several reasons to achieve these gains. First, GSR is a global management scheme considering all stripes, which saves most stripes by retaining their data and parity blocks. It plays an important role to decrease the migration cost. Second, by using a parallel method, GSR optimizes the XOR computations in the scaling process, which decreases the computation cost. Third, GSR



Fig. 11. Data migration ratio under different RAID-5 scaling approaches.





(b) Scale-down and scale-up.

2

Fig. 12. Parity modification ratio under different RAID-5 scaling approaches.



Fig. 13. Metadata modification ratio under different RAID-5 scaling approaches.



Fig. 14. Computation cost under different RAID-5 scaling approaches (the number of B XOR operations is normalized to 100%).

sacrifices a small amount of destructed old used stripes (OUS), which helps keep the original data and parity layout of RAID-5. This maintains a uniform workload and achieves high storage efficiency. GSR also has potential to have positive impact on migration by aggregating small I/Os as ALV [24] and FastScale [25].

Compared to MDM approach, GSR has a little higher

cost on parity/matadata modification and computation. This is reasonable because MDM approach keeps the whole parity chains well, which saves the parity modification cost as much as possible. However, as shown in Figure 5, MDM changes the original layout of RAID-5, which causes several problems, such as extremely uneven data distribution, low storage efficiency and poor write performance.



Fig. 15. Total number of I/O operations under different RAID-5 scaling approaches (the number of B I/O operations is normalized to 100%).



Fig. 16. Migration time under different RAID-5 scaling approaches (the migration time of $B * T_b$ is normalized to 100%).

V. CONCLUSIONS

In this paper, we propose a Global Stripe-based Redistribution (GSR) approach for bidirectional RAID-5 Scaling (both scale-up and scale-down). Our comprehensive mathematic analysis shows that GSR achieves better scalability in RAID-5 compared to other schemes in the following aspects: 1) uniform data distribution; 2) fewer operations on data migration, parity/metadata modification and XOR calculation; 3) reduced migration cost by up to 68.0% and faster scaling process by a factor of up to 3.13; 4) high reliability and availability during the migration process; and 5) improved storage efficiency and performance after scaling.

VI. ACKNOWLEDGEMENTS

We thank anonymous reviewers for their insightful comments. This research is sponsored by the U.S. National Science Foundation (NSF) Grants CCF-1102605, CCF-1102624, and CNS-1102629. Any opinions, ndings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reect the views of the funding agencies.

REFERENCES

- M. Armbrust et al. Above the clouds: A berkeley view of cloud computing. Technical Report EECS-2009-28, UC Berkeley, Feb. 2009.
- [2] J. Bonwick. RAID-Z. http://blogs.sun.com/bonwick/entry/raidz, 2010.[3] A. Brinkmann et al. Efficient, distributed data placement strategies for
- storage area networks. In *Proc. of the ACM SPAA'00*, 2000.
 [4] P. Chen, E. Lee, et al. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, June 1994.
- [5] E. Corporation. Leveraging EMC CLARitON CX4 fully automated storage tiering (FAST) for enterprise application deployments. Technical Report H-6951, EMC Corporation, February 2010.
- [6] C. Franklin and J. Wong. Expansion of RAID subsystems using spare space with immediate access to new space. US Patent 2003/0115412 A1, June 2003.

- [7] S. Ghandeharizadeh and D. Kim. On-line reorganization of data in scalable continuous media servers. In Proc. of the DEXA'96, 1996.
- [8] A. Goel et al. SCADDAR: An efficient randomized technique to reorganize continuous media blocks. In Proc. of the ICDE'02, 2002.
- [9] J. Gonzalez and T. Cortes. Increasing the capacity of RAID5 by online gradual assimilation. In *Proc. of the SNAPI'04*, 2004.
- [10] A. Gulati et al. BASIL: Automated I/O load balancing across storage devices. In Proc. of the USENIX FAST'10, 2010.
- [11] Hadoop Wiki. HDFS RAID. http://wiki.apache.org/hadoop/ HDFS-RAID, 2011.
- [12] S. Hetzler. Storage array scaling method and system with minimal data movement. US Patent 20080276057, June 2008.
- [13] Hitachi Data Systems. Hitachi Virtual Storage Platform Architecture Guide. http://www.hds.com/assets/pdf/ hitachi-architecture-guide-virtual-storage-platform.pdf?WT.ac=uk_ hp_sp2r1, March 2011.
- [14] M. Holland and G. Gibson. Parity declustering for continuous operation in redundant disk arrays. In *Proc. of the ASPLOS'92*, 1992.
- [15] K. Hwang et al. RAID-x: A new distributed disk array for I/O-Centric cluster computing. In Proc. of the HPDC'00, 2000.
- [16] E. Lee and R. Katz. Performance consequences of parity placement in disk arrays. In Proc. of the ASPLOS'91, 1991.
- [17] N. Brown. Online RAID-5 Resizing. drivers/md/raid5.c in the source code of Linux Kernel 2.6.18. http://www.kernel.org/, September 2006.
- [18] D. Patterson. A simple way to estimate the cost of down-time. In Proc. of the USENIX LISA'02, Philadelphia, PA, October 2002.
- [19] D. Patterson et al. A case for Redundant Arrays of Inexpensive Disks (RAID). In Proc. of the ACM SIGMOD'88, 1988.
- [20] Y. Saito et al. FAB: Building distributed enterprise disk arrays from commodity components. In *Proc. of the ASPLOS'04*, 2004.
- [21] J. Wilkes et al. The HP AutoRAID hierarchical storage system. ACM Transactions on Comp. Sys., 14(1):108–136, February 1996.
- [22] X. Yu et al. Trading capacity for performance in a disk array. In Proc. of the USENIX OSDI'00, 2000.
- [23] G. Zhang et al. SLAS: An efficient approach to scaling round-robin striped volumes. ACM Trans. on Storage, 3(1):1–39, March 2007.
- [24] G. Zhang et al. ALV: A new data redistribution approach to RAID-5 scaling. *IEEE Trans. on Computers*, 59(3):345–357, March 2010.
- [25] W. Zheng and G. Zhang. FastScale: Accelerate RAID scaling by minimizing data migration. In Proc. of the USENIX FAST'11, 2011.