SDM: A Stripe-based Data Migration Scheme to Improve the Scalability of RAID-6

Chentao Wu[1,2*], Xubin He[1*], Jizhong Han[3†], Huailiang Tan[4‡] and Changsheng Xie[2§]
[1]Department of Electrical & Computer Engineering, Virginia Commonwealth University, Richmond, VA 23284, USA
[2]Wuhan National Lab for Optoelectronics, Huazhong University of Science & Technology, Wuhan, China 430074
[3]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China 100190
[4]College of Information Science & Engineering, Hunan University, Changsha, China 410082
*{wuc4, xhe2}@vcu.edu, †hjz@ict.ac.cn, ‡tanhuailiang@hnu.edu.cn, §cs_xie@hust.edu.cn

Abstract—**In large scale data storage systems, RAID-6 has received more attention due to its capability to tolerate concurrent failures of any two disks, providing a higher level of reliability. However, a challenging issue is its scalability, or how to efficiently expand the disks. The main reason causing this problem is the typical fault tolerant scheme of most RAID-6 systems known as *Maximum Distance Separable* (MDS) codes, which offer data protection against disk failures with optimal storage efficiency but they are difficult to scale.**

To address this issue, we propose a novel *Stripe-based Data Migration* (SDM) scheme for large scale storage systems based on RAID-6 to achieve higher scalability. SDM is a stripe-level scheme, and the basic idea of SDM is optimizing data movements according to the future parity layout, which minimizes the overhead of data migration and parity modification. SDM scheme also provides uniform data distribution, fast data addressing and migration. We have conducted extensive mathematical analysis of applying SDM to various popular RAID-6 coding methods such as RDP, P-Code, H-Code, HDP, X-Code, and EVENODD. The results show that, compared to existing scaling approaches, SDM decreases more than 72.7% migration I/O operations and saves the migration time by up to 96.9%, which speeds up the scaling process by a factor of up to 32.

Index Terms—**RAID-6; MDS Code; Performance Evaluation; Scalability; Reliability**

I. Introduction

Redundant **A**rrays of **I**nexpensive (or **I**ndependent) **D**isks (**RAID**) [18] [5] is a popular choice to supply both high reliability and high performance storage services with acceptable spatial and monetary cost. In recent years, with the development of cloud computing, scalability becomes an important issue [1] which is urgently demanded by RAID systems due to following reasons,

1) By extending more disks, a disk array provides higher I/O throughput and larger capacity [29]. It not only satisfies the sharp increasing on user data in various online applications [7], but also avoids the extremely high downtime cost [17].

2) RAID-based architectures are widely used for clusters and large scale storage systems, where scalability plays a significant role [14] [23].

3) The storage efficiency can be improved by adding more disks into an existing disk array which also decreases the cost of the storage system.

With higher possibility of multiple disk failures [24] [19], RAID-6 has received more attention than ever. There are many implementations of RAID-6 based on various of erasure coding technologies, of which **M**aximum **D**istance **S**eparable (**MDS**) codes are typical. MDS codes offer protection against disk failures with given amount of redundancy [4]. According to the distribution of data and parity, MDS codes can be categorized into horizontal codes [22] [2] [6] [3] [20] and vertical codes [4] [28] [27] [15] [25] [26].

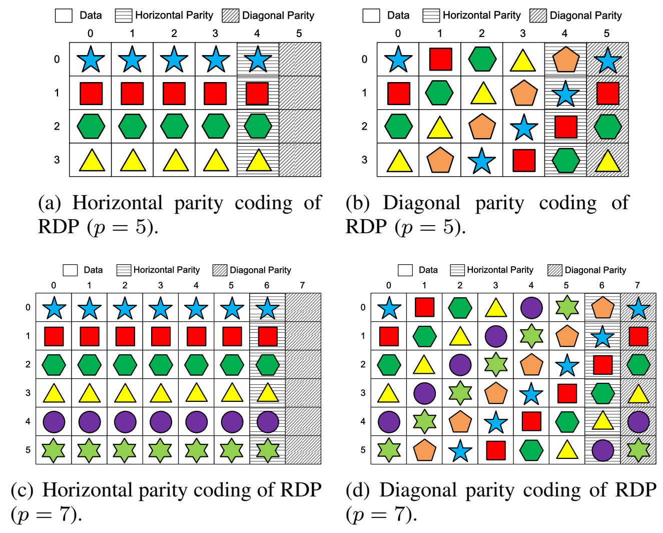
However, existing solutions in disk arrays [32] [31] are not suitable for RAID-6 scaling (a process to add disks to an existing disk array). Researchers face a challenge to find an effective solution to scale RAID-6 systems based on MDS codes efficiently. First, existing approaches are proposed for general case in RAID-0 or RAID-5 [9] [16] [30] [8] [31] [12] [32], which cannot adopt various coding methods in RAID-6. For example, As shown in Figures 1 and 2, RDP and P-Code have different layouts of data and parity. Thus the scaling scheme should be designed according to the characteristic of RDP or P-Code, respectively. Second, typical scaling schemes are based on round-robin order [9] [16] [30] [31], which are not suitable for RAID-6 due to high overhead on parity migration, modification and computation. One of the reasons is that the parity layouts of RAID-6 codes are complex. Another reason is that the stripes are dramatically changed after scaling. For example, a movement on any data element may lead to up to eight additional I/O operations on its corresponding parity elements.

To address the above challenge, we propose a new *Stripe-based Data Migration* (**SDM**) scheme to accelerate RAID-6 scaling. Different from existing approaches, SDM exploits the relationships between the stripe layouts before and after scaling to make scaling process efficiently.

We make the following contributions in this work:

- We propose a new data migration scheme (SDM) to address RAID-6 scalability problem, a significant issue in large scale data storage systems. SDM accelerates RAID-6 scaling process, in terms of the number of modified parities, the number of XOR calculations, the total number of I/O operations and the migration time;
- SDM balances I/O distribution among multiple disks in a disk array, reducing the migration time indirectly;
- SDM provides fast data addressing algorithms.

The rest of this paper continues as follows: Section II discusses the motivation of this paper and details the background of existing scaling methods. SDM scheme is described in detail in Section III. Section IV gives the quantitative analysis on scalability. Finally we conclude the paper in Section V.



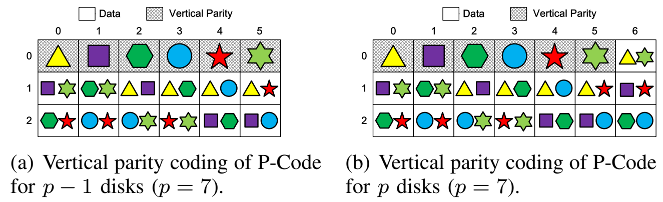
(a) Horizontal parity coding of RDP ($p = 5$).

(b) Diagonal parity coding of RDP ($p = 5$).

(c) Horizontal parity coding of RDP ($p = 7$).

(d) Diagonal parity coding of RDP ($p = 7$).

Fig. 1. RDP Code (for $p + 1$ disks).



(a) Vertical parity coding of P-Code for $p - 1$ disks ($p = 7$).

(b) Vertical parity coding of P-Code for p disks ($p = 7$).

Fig. 2. P-Code.

TABLE I
A LIST OF SYMBOLS IN THIS PAPER

Symbols	Description
n	number of disks in a disk array before scaling
m	number of extended disk(s)
B	total number of data blocks (data elements)
P, Q	parity blocks before scaling
P', Q'	parity blocks after scaling
S, S'	total number of stripes before/after scaling
$n_d, n_{d'}$	number of data disks before/after scaling
S_{id}, S'_{id}	stripe ID before/after scaling
i, i'	row ID in a stripe before/after scaling
j, j'	column ID (disk ID) in a stripe before/after scaling
n_s	number of stripes in each stripe set
n_e	total number of data elements in each stripe before scaling
n_{ec}	total number of data elements in each column per stripe set
R_d	data migration ratio
R_p	parity modification ratio
n_{io}	total number of I/O operations
T_b	access time of a read/write request to a block
T_m	migration time

II. Background and Motivation of Our Work

In this section we discuss the background of the scaling schemes, problems in existing MDS codes and our motivation. To facilitate our discussion, we summarize the symbols used in this paper in Table I.

A. Desired Scaling Features in RAID-6

To extend a disk array, some data need to be migrated to the new disk(s) to achieve a balanced data distribution. During data migration, we prefer to keep an evenly distributed workload and minimize the data/parity movement. Combined with existing scaling approaches [32] and the real cases in RAID-6, the following four features are typically desired,

Feature 1 (*Uniform Data Distribution*): Each disk has the same amount of data blocks to maintain an even workload.

Feature 2 (*Minimal Data & Parity Migration*): The extended disks are served as data disks because the parity disks exist before scaling, so by adding m disks to a RAID-6 system with n_d data disks storing B data blocks, the expected total number of data movements is $m * B/(m + n_d)$.

Feature 3 (*Fast Data Addressing*): The locations of blocks in the array can be efficiently computed.

Feature 4 (*Minimal Parity Computation & Modification*): A movement on data block could bring modification cost on its corresponding original parities and computation cost on new parities, so movements on data blocks should be limited in the original parity chain and thus parity blocks can be retained without any change.

B. Existing Fast Scaling Approaches

Existing approaches to improve the scalability of RAID systems include Round-Robin (RR) [9] [16] [30], Semi-RR [8], ALV [31], MDM [12], FastScale [32], etc. To clearly illustrate various strategies in RAID-6, we use P/Q (e.g., P_1 and Q_1) to delegate various parity blocks before scaling and P'/Q' (e.g., P'_1 and Q'_1) for the parity blocks after scaling. If the parity block is still presented by P/Q after scaling, it means that parity is unchanged.

Traditional RR and Semi-RR approaches can be used in RAID-6 under two restrictions. First, all data blocks are migrated based on round-robin order in the scaling process. Second, all parity blocks are retained without any movement.

For a traditional RR scaling approach (as shown in Figure 3), obviously, all parities need to be modified and recalculated after data migration. Although RR is a simple approach to implement on RAID-6, it brings high overhead.

Based on RR approach, Brown [16] designed a reshape toolkit in the Linux kernel (MD-Reshape), which writes mapping metadata with a fixed-size window. Due to the limitation of RR approach, metadata are frequently updated by calling MD-Reshape function, which is inefficient.

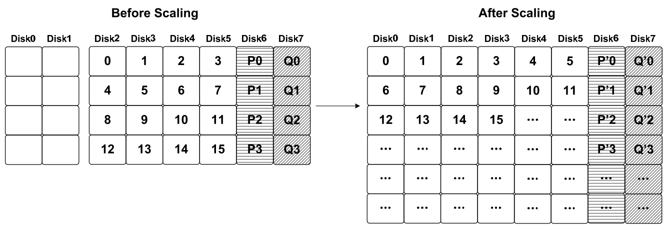


Fig. 3. RAID-6 scaling in RDP from 6 to 8 disks using RR approach (all data blocks are migrated).

Semi-RR [8] (Figure 4) is proposed to decrease high migration cost in RR scaling. Unfortunately, by extending multiple

TABLE II
SUMMARY ON VARIOUS FAST SCALING APPROACHES

Name	Features in Section II-A				Used in RAID-6?
	1	2	3	4	
RR	√	×	√	×	conditionally
Semi-RR	×	×	√	×	conditionally
ALV	√	×	√	×	×
MDM	×	√	√	√	×
FastScale	√	√	√	√	×
SDM	√	√	√	√	√

TABLE III
PRIORITIES OF DATA/PARITY MIGRATION IN SDM (FOR A MOVEMENT ON SINGLE DATA/PARITY ELEMENT)

Priority	Number of modified Parities (for parity consistency)	Overhead of Data/Parity Migration & Modification	Parity Computation Cost
1	none	2 I/Os	none
2	one	4 I/Os	1 XOR
3	two	6 I/Os	2 XORs
4	three or more	$\geq$ 8 I/Os	4 XORs

disks, the data distribution is not uniform after scaling.

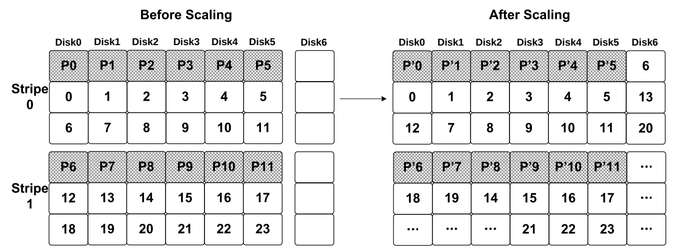


Fig. 4. RAID-6 scaling in P-Code from 6 to 7 disks using Semi-RR approach.

ALV [31] and MDM [12] are RAID-5 scaling approaches, Fastscale [32] accelerates the scaling process of RAID-0. They take advantages of both RR and Semi-RR approaches, and improve the migration efficiency. However, they cannot be applied in RAID-6.

C. Our Motivation

We summarize the existing fast scaling approaches in Table II. Although these fast scaling approaches offer some nice features for RAID-0 or RAID-5, it is not suitable for RAID-6 due to the special coding methods, of which MDS codes are popular. Figures 1 and 2 show the parity layout of RDP [6] and P-Code [15], where there are several problems regarding scalability. First, existing scaling approaches are difficult to be applied in these codes. If we extend a RAID-6 disk array based on RDP from 6 to 8 disks, any new data cannot be migrated into the dedicated parity disks (columns 6 and 7). On the other hand, if we select P-Code, we needn't care about the dedicated parity disks. Second, few strategies on reducing parity modification are involved in existing methods. Actually, for keeping parity consistency, too many parity elements have to be recalculated and modified because of the movements on their corresponding data elements in RAID-6. It causes high computation cost, modification overhead, and an unbalanced migration I/O.

In summary, existing scaling approaches are insufficient to satisfy the desired four features listed in Section II-A, which motivates us to propose a new approach on RAID-6 scaling.

III. STRIPE-BASED DATA MIGRATION SCHEME

In this section, a **S**tripe-based **D**ata **M**igration (SDM) scheme is designed to accelerate the RAID-6 scaling. The purpose of SDM is to minimize the parity migration, modification and computation according to a global point view on single/multiple stripe(s), not limited to a migration on single data/parity element as Round-Robin [9] [16] [30].

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.

In our SDM scheme, an OUS/OES corresponds to a NUS/NES with the same stripe ID, respectively. SDM scheme takes the following steps,

1. Priority Definition: Define the different priorities for the movements on single data/parity.

2. Layout Comparison: Compare the layouts before and after scaling and find a cost-effective way to change the layout of a stripe. The data/parity movements in this step should have the highest priority.

3. Load Balancing Check: According to the data distribution in the stripes (after Step 2), select a small portion of stripes to balance the workload. The data/parity movements in this step can have acceptable overhead.

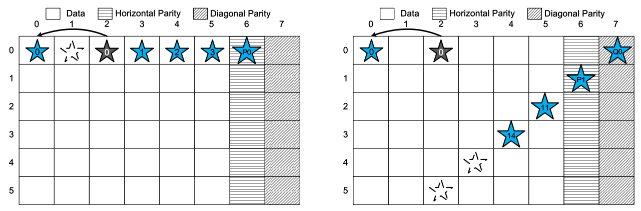
4. Data Migration: Based on the migration schemes in Steps 2 and 3, start data migration for each stripe.

Typically, without any special instructions, a data/parity block (in logical address view) corresponds to a data/parity element (in parity layout view) in a stripe. Due to the difference between horizontal and vertical codes, we design various scaling schemes and discuss them separately. In this section, we use RDP [6] (a typical horizontal code) and P-Code [15] (a typical vertical code) as examples to show how SDM works in RAID-6, scaling from 6 to 8 disks and from 6 to 7 disks, respectively. Their corresponding parity layouts are shown in Figures 1 and 2.

A. Priority Definition of Data/Parity Movements

Due to complex parity layouts in RAID-6, several scenarios on data movements should be considered. Due to this reason, we define the priorities of data/parity migration in SDM according to the number of modified parities as summarized in Table III. Higher priority means lower overhead on total IOs and parity computation.

According to the various priorities in Table III, we can measure whether a movement is efficient or not. For example, as shown in Figure 5, Block 0 is migrated from disk 2 to disk 0, and other blocks are retained without any movement. From horizontal parity's point of view (shown in Figure 5(a)), Block 0 also stays in the original horizontal parity chain and the corresponding parity $P0$ needn't be changed. On the other hand, considering the aspect of diagonal parity (shown in Figure 5(b)), Block 0 also shares the same parity chain with other blocks (e.g., Blocks 11, 14, $P1$ and $Q0$), and the corresponding parity element could be retained. Therefore, the movement of Block 0 doesn't change any parity and has the highest priority (Priority 1) in the scaling process.



(a) Horizontal parity view. (b) Diagonal parity view.

Fig. 5. Data movement of Block 0 in Figure 3.

B. Layout Comparison

Compared to the current and future parity layouts, we propose different rules for horizontal and vertical codes,

*1) **Rules for Horizontal Codes:***

- *(Parity Disks Labeling)* The parity disks are retained and the extending disk(s) are used for data disk(s).
- *(Extended Disk(s) Labeling)* If m disk(s) are added into a disk array, the new disk(s) are labeled as the m middle column(s) (in the middle of all data columns).
- *(Rows Labeling)* If an Old Used Stripe (OUS) contains n_r rows, these rows are labeled as the first n_r rows in the New Used Stripe (NUS).
- *(Special parity handling)* If horizontal parities take part in the calculation of diagonal/anti-diagonal parities, the priority of data/parity movement in the horizontal chain is higher than that in diagonal/anti-diagonal parity chain. Conversely, if diagonal/anti-diagonal parities take part in the calculation of horizontal parities, the priority of data/parity movement in the diagonal/anti-diagonal chain is higher than that in horizontal chain.
- *(Data/Parity Migration)* Select proper data/parity movements with the highest priority.

For example, if we want to scale a RAID-6 array using RDP from 6 to 8 disks, compared to the layouts in Figure 1, we have the following strategies according to the above rules,

1.(Parity Disks Labeling) Columns 4 and 5 in Figures 1(a) and 1(b) are retained as parity columns, and the column IDs are changed to columns 6 and 7 in Figures 1(c) and 1(d).

2.(Extended Disk(s) Labeling) Two extended disks are regarded as data disks and labeled as columns 2 and 3, causing the lowest overhead as shown in Table IV.

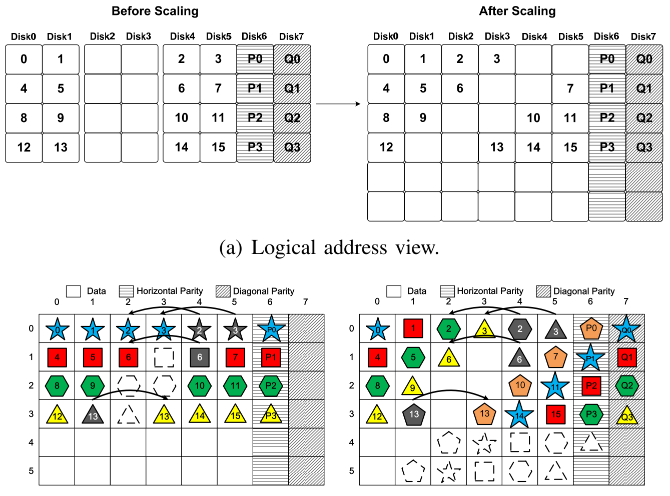
TABLE IV
DIFFERENT OVERHEAD OF EXTENDED DISK(S) LABELLING IN SDM

Extended disk(s) labeling	Minimal number of moved data/parity	Migration Cost
columns 0 and 1	10	20 I/Os
columns 1 and 2	6	12 I/Os
columns 2 and 3	4	8 I/Os
columns 3 and 4	4	8 I/Os
columns 4 and 5	6	12 I/Os
other cases	≥ 6	≥ 12 I/Os

3.(Rows Labeling) The row IDs in each Old Used Stripe (OUS) are retained. By extending more disks, the size of stripes becomes larger and contains several new rows, which are known as "phantom" rows [21] in the scaling process.

4.(Special parity handling) The priority of data/parity movement in the horizontal chain is higher than that in diagonal/anti-diagonal parity chain.

5.(Data/Parity Migration) Blocks 2, 6, 3 and 13 are selected to migrate as shown in Figure 6, and all data blocks also share the same parities with the new parity layout. Therefore, all parity blocks are retained and these movements have the highest priority.



(a) Logical address view.

(b) Horizontal parity view. (c) Diagonal parity view.

Fig. 6. Data/parity migration in Layout Comparison Step (RDP code, scaling from 6 to 8 disks).

*2) **Rules for Vertical Codes:***

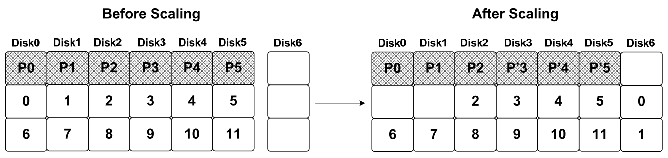
- *(Original Disks Labeling)* Original disk IDs are retained and the extended disks are used for data disks.
- *(Extended Disk(s) Labeling)* By extending m disk(s) into a disk array, the new disks are labeled as the last m columns.
- *(Rows Labeling)* If an Old Used Stripe (OUS) contains n_r data rows, which are labeled as the same row ID in the New Used Stripe (NUS). The dedicated parity rows are labeled according to the new layout.
- *(Data/Parity Migration)* Select proper data/parity movements which have the highest priority.

For example, if we want to scale a RAID-6 array using P-Code from 6 to 7 disks, compared to the layouts in Figure 2, we have the following strategies according to the above rules,

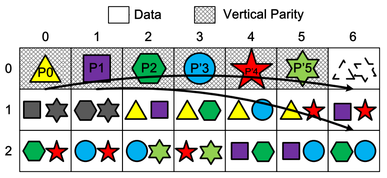
1.(Disks Labeling) As shown in Figure 7, original column IDs are retained. The new disk is labeled as Column 6.

2.(Rows Labeling) The row IDs in each Old Used Stripes (OUSs) are retained.

3.(Data/Parity Migration) Blocks 0 and 1 are selected to migrate as shown in Figure 7, which have the highest priority and three parities ($P3$, $P4$ and $P5$) are modified in each stripe.



(a) Logical address view.



(b) Vertical parity view.

Fig. 7. Data/parity migration in Layout Comparison Step (P-Code, scaling from 6 to 7 disks).

C. Load Balancing Check in SDM

Although the layout comparison can minimize the overhead of scaling process due to the highest priority data/parity movements in this step, it suffers unbalanced I/O which is a serious issue in RAID systems [26] [13] [10]. To address this problem, a load balancing check is applied to achieve an even workload.

In the load balancing check step, first we get the statistics of data distribution after layout comparison. For example, the data distributions in RDP and P-Code are shown in the second row (Stripe ID is 0) in the Tables V and VI. We notice that they are unbalanced distributions, where different columns have various number of data elements.

For horizontal and vertical codes, the following methods are used to achieve a uniform data distribution,

(For horizontal codes) Most horizontal codes have asymmetrical data and parity distribution, so a small portion of stripes are sacrificed with a little higher migration cost (called "sacrificed stripes"). In the load balancing check step, the portion of sacrificed stripes are calculated, and proper blocks in these stripes are migrated to achieve an even workload.

(For vertical codes) Most vertical codes have symmetrical data and parity distribution, the data elements in each stripes are migrated alternately, and a small portion of stripes are chosen without any movement (called "retained stripes").

To calculate the percentage of sacrificed/retained stripes, we define "**Stripe Set**" as n_s stripes with uniform data

TABLE V
DATA DISTRIBUTION OF RAID-6 SCALING USING SDM SCHEME (RDP CODE, SCALING FROM 6 TO 8 DISKS)

Stripe ID	Total Data Elements	Column ID					
		0	1	2	3	4	5
0	16	4	3	2	2	2	3
1	16	4	3	2	2	2	3
2	16	0	2	4	4	4	2
stripe set (three stripes)	48	8	8	8	8	8	8

TABLE VI
DATA DISTRIBUTION OF RAID-6 SCALING BY USING SDM SCHEME (P-CODE, SCALING FROM 6 TO 7 DISKS)

Stripe ID	Total Data Elements	Column ID						
		0	1	2	3	4	5	6
0	12	1	1	2	2	2	2	2
1	12	2	2	1	1	2	2	2
2	12	2	2	2	2	1	1	2
3	12	1	1	2	2	2	2	2
4	12	2	2	1	1	2	2	2
5	12	2	2	2	2	1	1	2
6	12	2	2	2	2	2	2	0
stripe set (seven stripes)	84	12	12	12	12	12	12	12

distribution. Suppose $n_{d'}$ is the total number of data disks after scaling, and n_e is the total number of data elements in a stripe before scaling. n_s can be computed by $n_s = \frac{lcm\{n_{d'}, n_e\}}{n_e}$. Thus in a stripe set, the total number of data elements in each column (denoted by n_{ec}) is $n_{ec} = \frac{lcm\{n_{d'}, n_e\}}{n_{d'}}$.

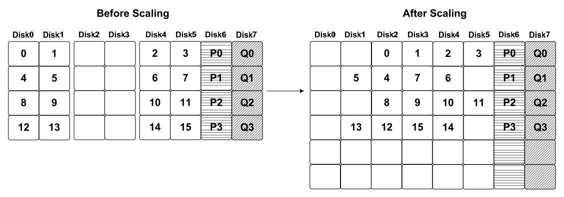
A small portion of stripes in each stripe set (one stripe in typical) are selected as sacrificed or retained stripes. For example, as shown in Table V, each stripe set contains three stripes ($n_s = \frac{lcm\{16,6\}}{16} = 48/16 = 3$), where the last one is selected as sacrificed stripe. In each stripe set, each column should contain 8 data elements ($n_{ec} = \frac{lcm\{16,6\}}{6} = 48/6 = 8$), so the data distribution in the sacrificed stripe (Stripe ID is 2) is $0, 2, 4, 4, 4, 2$ (e.g., the number of data elements in column 0 is: $n_{ec} - 2 * 4 = 8 - 8 = 0$). Similarly, as shown in Table VI, the stripe set size for P-Code is 7 ($n_s = \frac{lcm\{12,7\}}{12} = 84/12 = 7$) and the last stripe is used as retained stripe.

We summarize the migration process in the load balancing check step based on the priority level, shown in Figures 8 and 9. Due to the space limit, in Figure 9, we only give the migration process of the third stripe (Stripe ID is 2 in Table VI) in each stripe set. Obviously, in Tables V and VI, each stripe set has an even data distribution. Because the data selections and operations are the same in each stripe set, the corresponding calculations are performed only once. Thus the overhead of load balancing check step is very small.

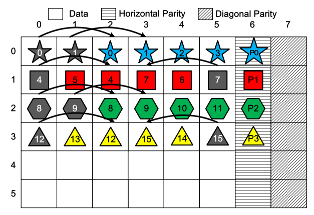
D. Data Migration

After the steps of layout comparison and load balancing check, a comprehensive migration strategy can be derived and then the system can start the data migration process. We notice that Feature 2 can be satisfied and have the following theorem,

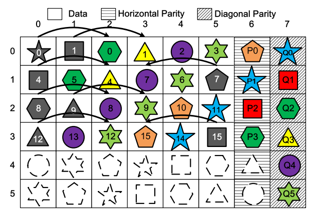
Theorem 1: For any MDS code scaling from n disks to $n+m$ disks by using SDM scheme, the total number of migrated



(a) Logical address view.

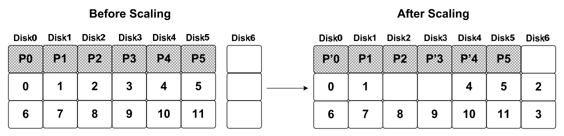


(b) Horizontal parity view.

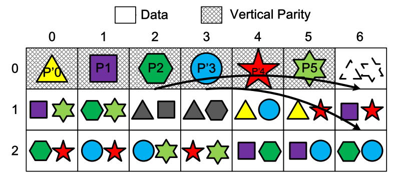


(c) Diagonal parity view.

Fig. 8. Data/parity migration in load balancing check step (RDP code, scaling from 6 to 8 disks).



(a) Logical address view.



(b) Vertical parity view.

Fig. 9. Data/parity migration in load balancing check step (P-code, scaling from 6 to 7 disks).

data blocks is $m * B/(m + n_d)$, where n_d is the number of data disks before scaling.

Due to space limit, we only demonstrate this theorem is correct for RDP scaling from 6 to 8 disks and P-Code scaling from 6 to 7 disks. The total number of migrated data blocks in the two examples are $B/3$ and $B/7$, respectively.

E. Data Addressing Algorithm

The data addressing algorithms of the two examples on RDP and P-Code are shown in Algorithms 1 and 2. SDM scheme satisfies fast addressing feature in Section II-A. For a disk array scaling from 6 to 8 then to 12 disks by using RDP code, our algorithms can be used multiple times by saving the initialization information.

F. Property of SDM

From the above discussion, we can see that SDM satisfies the desired features 1-3 of RAID-6 scaling defined in Section II-A. SDM also satisfies the Feature 4: minimal modification and computation cost of the parity elements, which is discussed in detail in Sections IV.

IV. Scalability Analysis

In this section, we evaluate the scalability of various MDS codes by using different approaches.

A. Evaluation Methodology

We compare SDM scheme to **Round-Robin (RR)** [9] [16] [30] and **Semi-RR** [8] approaches. ALV [31], MDM [12] and FastScale [32] cannot be used in RAID-6, so they are not evaluated.

We also propose an ideal fast scaling method as a baseline. The ideal case is based on the Feature 2 (Section II-A) with minimal data movements to maintain a uniform workload in the enlarged new used stripe. We assume this case doesn't involve any parity migration, modification and computation as in RAID-0. Because no movement in dedicate parity disks (e.g., for RDP code), actually the number of ideal movements is $m * B/(m + n_d)$, where n_d is the number of data disks.

Several popular MDS codes in RAID-6 are selected for comparison,

Algorithm 1: Data Addressing Algorithm of RDP Scaling from n to $n+m$ disks Using SDM Scheme (where $n = p_1 + 1$, $n + m = p_2 + 1$, $p_1 < p_2$, p_1 and p_2 are prime numbers)

```
Get or calculate the $S_{id}$, $i$, $j$, $n_s$ and $n_{ss}$ value, then label the new disks
with column IDs from $n - \frac{m}{2} - 3$ to $n + \frac{m}{2} - 4$
$S'_{id} = S_{id}$;    /*Stripe ID unchanged*/
$k = S_{id} \bmod n_s$;
if $0 \le k \le n_s - n_{ss} - 1$ (migrated stripes in layout comparison step) then
    if $i + j \le p_1 - 1$ then
        $i' = i, j' = j$;
    end
    else
        $i' = i, j' = j + m$.
    end
end
if $n_s - n_{ss} \le k \le n_s - 1$ (sacrificed stripes in load balancing checking step)
then
    if $(i = 1, 3, 5, \cdots, p_1 - 1)$ && $(j = 1, 3, 5, \cdots, n + m - 3)$ then
        $i' = i, j' = j$;
    end
    else
        $i' = i, j' = j + m$.
    end
end
```

Algorithm 2: Data Addressing Algorithm of P-Code Scaling from n to $n+m$ disks Using SDM Scheme (where $n = p_1 - 1$, $n + m = p_2$, $p_1 \le p_2$, p_1 and p_2 are prime numbers)

```
Get or calculate the $S_{id}$, $i$, $j$, $n_s$ and $n_{rs}$ value, then label the new disks
with column IDs from $n$ to $n + m - 1$
$S'_{id} = S_{id}$;    /*Stripe ID unchanged*/
$k = S_{id} \bmod n_s$;
if $0 \le k \le n_s - n_{rs} - 1$ (migrated stripes in layout comparison and load
balancing checking step) then
    if migrated data elements then
        distribute $i'$ and $j'$ based on round-robin order, where
        $0 \le i' \le \frac{p_1 - 1}{2}$, $n \le j' \le n + m - 1$;
    end
    else
        $i' = i, j' = j$.
    end
end
if $n_s - n_{rs} \le k \le n_s - 1$ (retained stripes in load balancing checking step)
then
    $i' = i, j' = j$.
end
```

TABLE VII
OVERHEAD OF RAID-6 SCALING USING SDM SCHEME (RDP CODE, SCALING FROM 6 TO 8 DISKS)

Stripe ID	Number of Data & Parity Movements	Number of Modified Parities	Total I/Os	Number of XOR Calculations
0 and 1	4	none	8 I/Os	none
2	8	16	48 I/Os	32 XORs
stripe set (three stripes)	16	16	64 I/Os	32 XORs

TABLE VIII
OVERHEAD OF RAID-6 SCALING USING SDM SCHEME (P-CODE, SCALING FROM 6 TO 7 DISKS)

Stripe ID	Number of Data & Parity Movements	Number of Modified Parities	Total I/Os	Number of XOR Calculations
0,1,2,3,4 and 5	2	4	12 I/Os	8 XORs
6	none	none	none	none
stripe set (seven stripes)	12	24	72 I/Os	48 XORs

1) Codes for $p-1$ disks: P-Code[1] [15] and HDP [26];
2) Codes for p disks: X-Code [27] and P-Code;
3) Codes for $p+1$ disks: RDP code [6] and H-Code [25];
4) Codes for $p+2$ disks: EVENODD code [2].

Suppose the total number of data blocks in a disk array is B, the total number of stripes in a disk array before scaling is S, we can derive the relationship between these two parameters. For example, for RDP code when $p = 5$, $B = 16S$; when $p = 7$, $B = 36S$.

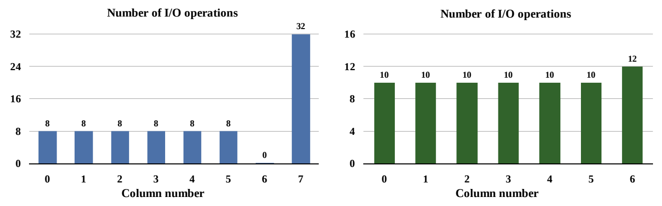
We define **Data Migration Ratio** (R_d) as the ratio between the number of migrated data/parity blocks and the total number of data blocks. **Parity Modification Ratio** (R_p) delegates the ratio between the number of modified parity blocks and the total number of data blocks. For the examples of RDP and P-Code shown in Section III, according to the results presented in Table VII, $R_d = \frac{16S}{16S*3} = 33.3\%$ and $R_p = \frac{16S}{16S*3} = 33.3\%$. For P-Code, based on the numbers in Table VIII, $R_d = \frac{12S}{12S*7} = 14.3\%$ and $R_p = \frac{24S}{12S*7} = 28.6\%$.

In RAID-6 scaling, each data or parity migration only costs two I/O operations, and the modification of each parity also has two I/Os. Based on the data migration ratio (R_d) and parity modification ratio (R_p), the total number of I/O operations is $n_{io} = 2*R_d*B + 2*R_p*B$. According to this equation, the total number of I/O operations for RDP example in Section III is $2*B*33.3\% + 2*B*33.3\% = 1.33B$, and for P-Code example it is $2*B*14.3\% + 2*B*28.6\% = 0.86B$.

If we ignore the computation time and assume the same time on a read or write request to a block (denoted by T_b), and suppose the migration I/O can be processed in parallel on each disk. Based on the I/O distribution shown in Figure 10(a) (column 7 has the highest I/O and longest migration time cost), the migration time T_m for RDP example in Section III is $T_m = 32ST_b/3 = 2BT_b/3$.

Similarly, based on Figure 10(b) (column 6 has the highest I/O and longest migration time cost), the migration time for P-Code example is $T_m = 12ST_b/7 = BT_b/7$.

[1]P-Code has two variations as shown in Figure 2.



(a) RDP Code (every three stripes, scaling from 6 to 8 disks).
(b) P-Code (every seven stripes, scaling from 6 to 7 disks).

Fig. 10. I/O distribution in multiple stripes using SDM scheme.

B. Numerical Results

In this section, we give the numerical results of scalability using different scaling approaches and various coding methods. In the following Figures 11 to 16, a two-integer tuple (n, m) denotes the original number of disks and the extended number of disks. For example, RDP $(6, 2)$ means a RAID-6 scaling from 6 to $6 + 2$ disks using RDP code.

*1) **Data Distribution:*** Regarding to the data distribution, we use the coefficient of variation as a metric to examine whether the distribution is even or not as other approaches [8] [32]. The small value of the coefficient of variation means highly uniform distribution. The results are shown in Figure 11. We notice that load balancing check is necessary for SDM.

*2) **Data Migration Ratio:*** Second, we calculate the data migration ratio (R_d) among various fast scaling approaches under different cases as shown in Figure 12. Our SDM scheme has the approximate migration ratio compared to Semi-RR and the ideal case in RAID-0.

*3) **Parity Modification Ratio:*** Third, parity modification ratio (R_p) among various RAID-6 scaling approaches under different cases is presented in Figure 13. Compared to other schemes with the same p and m, SDM sharply decreases the number of modified parities by up to 96.2%.

*4) **Computation Cost:*** We calculate the total number of XOR operations under various cases as shown in Figure 14. By using RR-based approaches, various codes have similar computation cost. SDM scheme decreases more than 80% computation cost compared to other approaches.

*5) **Total Number of I/O Operations:*** Next, total number of I/O operations are calculated in these cases. If we use B as the baseline, the results of total I/Os are shown in Figure 15. By using SDM scheme, $72.7\% - 91.1\%$ I/Os are reduced.

*6) **Migration Time:*** Migration time is evaluated as shown in Figure 16 and summarized in Table IX. Compared to other approaches, SDM performs well in multiple disks extension and decreases the migration time by up to 96.9%, which speeds up the scaling process to a factor of 32.

C. Analysis

From the results in Section IV-B, compared to RR, Semi-RR and ALV, SDM has great advantages. There are several

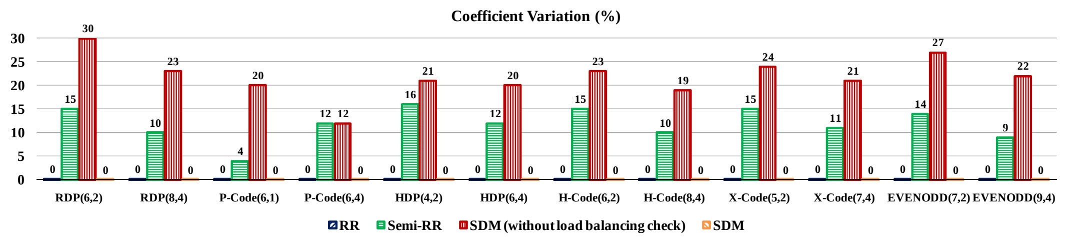


Fig. 11. Comparison on data distribution under various RAID-6 scaling approaches.

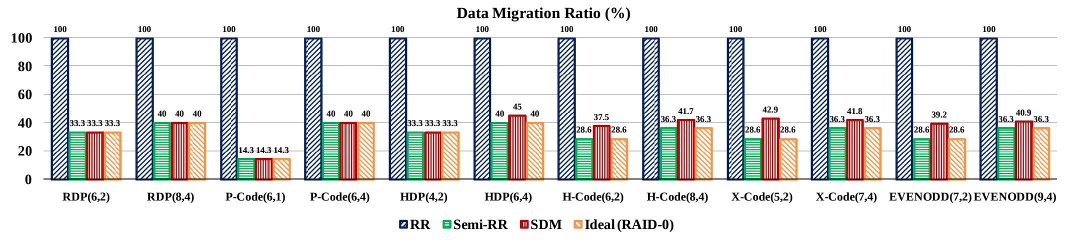


Fig. 12. Comparison on data migration ratio under various RAID-6 scaling approaches.

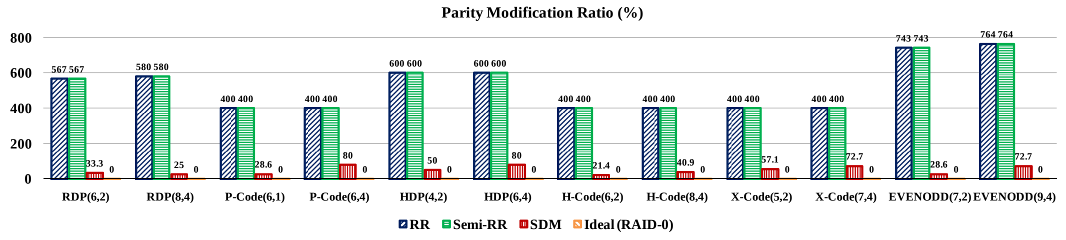


Fig. 13. Comparison on parity modification ratio under various RAID-6 scaling approaches.

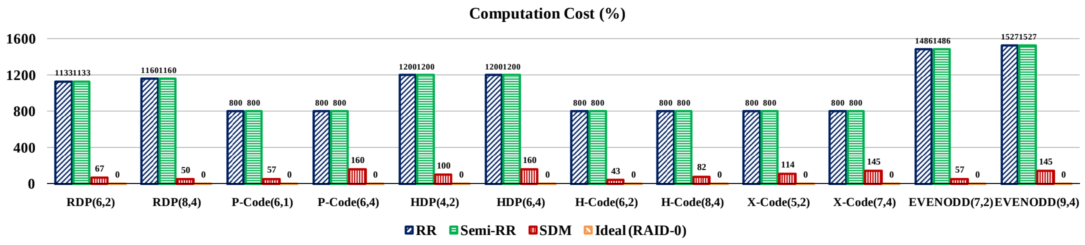


Fig. 14. Comparison on computation cost under various RAID-6 scaling approaches (The number of B XORs is normalized to 100%).

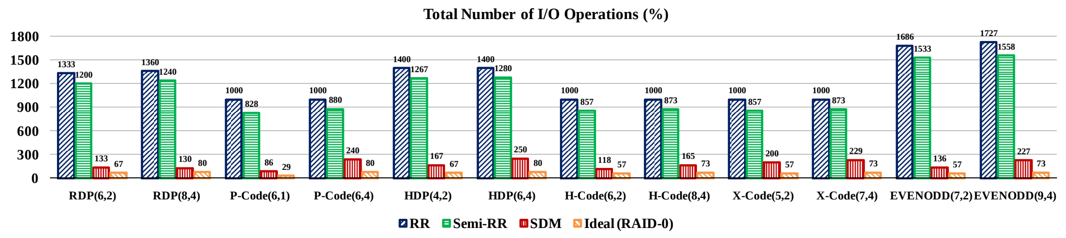


Fig. 15. Comparison on total I/Os under various RAID-6 scaling approaches (The number of B I/O operations is normalized to 100%).

reasons to achieve these gains. First, SDM scheme is a global management on multiple stripes according to the priorities of data movements, which reduces the parity modification cost, computation cost, total I/Os. Second, compared to other

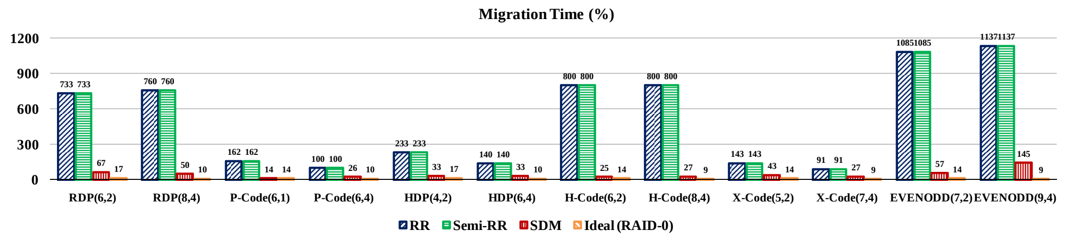


Fig. 16. Comparison on migration time under various RAID-6 scaling approaches (The time $B * T_b$ is normalized to 100%).

TABLE IX
SPEED UP OF SDM SCHEME OVER OTHER RAID-6 SCALING SCHEMES IN TERMS OF MIGRATION TIME

m & p	RDP	P-Code	HDP	H-Code	X-Code	EVENODD
$m = 2$ $p = 5$	10.9×	–	7.1×	32.0×	3.3×	19.0×
$m = 4$ $p = 7$	15.2×	3.8×	4.2×	29.6×	3.4×	7.8×

approaches, SDM scheme distributes the migration I/Os more evenly among data and parity disks, which accelerates the scaling process in parallel. That is why SDM has better effects in horizontal codes which suffer from unbalanced I/Os [26]. Third, although SDM sacrifices a small portion of stripes in each stripe set, it helps SDM to maintain a uniform workload, which creates favorable conditions for the storage system after scaling. SDM also has potential to have positive impact on migration by aggregating small I/Os as FastScale [32].

V. CONCLUSIONS

In this paper, we have proposed a Stripe-based Data Migration (SDM) scheme to achieve high scalability for RAID-6. Our comprehensive mathematic analysis shows that SDM achieves better scalability compared to other approaches in the following aspects: 1) lower computation cost by reducing more than 80% XOR calculations; 2) less I/O operations by 72.7%-91.1%; and 3) shorter migration time and faster scaling process by a factor of up to 32.

VI. ACKNOWLEDGEMENTS

We thank anonymous reviewers for their insightful comments. This research is partially sponsored by the U.S. National Science Foundation (NSF) Grants CCF-1102605 and CCF-1102624, the National Basic Research 973 Program of China under Grant No. 2011CB302303, the National Natural Science Foundation of China under Grant No. 60933002, and the National 863 Program of China under Grant No.2009AA01A402. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF.

REFERENCES

[1] M. Armbrust et al. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, UC Berkeley, 2009.
[2] M. Blaum et al. EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures. *IEEE Trans. on Computers*, 44(2):192–202, Feb. 1995.
[3] M. Blaum and R. Roth. On lowest density MDS codes. *IEEE Transactions on Information Theory*, 45(1):46–59, Jan. 1999.
[4] Y. Cassuto and J. Bruck. Cyclic lowest density MDS array codes. *IEEE Transactions on Information Theory*, 55(4):1721–1729, Apr. 2009.
[5] P. Chen, E. Lee, et al. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, Jun. 1994.
[6] P. Corbett et al. Row-Diagonal Parity for double disk failure correction. In *Proc. of the FAST'04*, 2004.
[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*, e, 2004.
[10] A. Gulati et al. BASIL: Automated I/O load balancing across storage devices. In *Proc. of the FAST'10*, 2010.
[11] J. Hafner. HoVer erasure codes for disk arrays. In *Proc. of the DSN'06*, Philadelphia, PA, June 2006.
[12] S. Hetzler. Storage array scaling method and system with minimal data movement. US Patent 20080276057, June 2008.
[13] M. Holland and G. Gibson. Parity declustering for continuous operation in redundant disk arrays. In *Proc. of the ASPLOS'92*, 1992.
[14] K. Hwang et al. RAID-x: A new distributed disk array for I/O-Centric cluster computing. In *Proc. of the HPDC'00*, 2000.
[15] C. Jin et al. P-Code: A new RAID-6 code with optimal properties. In *Proc. of the ICS'09*, 2009.
[16] 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.
[17] D. Patterson. A simple way to estimate the cost of down-time. In *Proc. of the LISA'02*, 2002.
[18] D. Patterson, G. Gibson, and R. Katz. A case for Redundant Arrays of Inexpensive Disks (RAID). In *Proc. of the SIGMOD'88*, 1988.
[19] E. Pinheiro et al. Failure trends in a large disk drive population. In *Proc. of the FAST'07*, 2007.
[20] J. Plank. The RAID-6 liberation codes. In *Proc. of the FAST'08*, 2008.
[21] J. Plank et al. Minimum density RAID-6 codes. *ACM Transactions on Storage*, 6(4):Article 16, May 2011.
[22] I. Reed and G.Solomon. Polynomial codes over certain finite fields. *J. of the Society for Indus. and Applied Math.*, pages 300–304, 1960.
[23] Y. Saito et al. FAB: Building distributed enterprise disk arrays from commodity components. In *Proc. of the ASPLOS'04*, 2004.
[24] B. Schroeder et al. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In *Proc. of the FAST'07*, 2007.
[25] C. Wu et al. H-Code: A hybrid MDS array code to optimize partial stripe writes in RAID-6. In *Proc. of the IPDPS'11*, 2011.
[26] C. Wu et al. HDP code: A Horizontal-Diagonal parity code to optimize I/O load balancing in RAID-6. In *Proc. of the DSN'11*, 2011.
[27] L. Xu and J. Bruck. X-Code: MDS array codes with optimal encoding. *IEEE Trans. on Information Theory*, 45(1):272–276, Jan. 1999.
[28] L. Xu et al. Low-density MDS codes and factors of complete graphs. *IEEE Trans. on Information Theory*, 45(6):1817–1826, Sep. 1999.
[29] X. Yu et al. Trading capacity for performance in a disk array. In *Proc. of the OSDI'00*, 2000.
[30] G. Zhang et al. SLAS: An efficient approach to scaling round-robin striped volumes. *ACM Trans. on Storage*, 3(1):1–39, Mar. 2007.
[31] G. Zhang et al. ALV: A new data redistribution approach to RAID-5 scaling. *IEEE Trans. on Computers*, 59(3):345–357, Mar. 2010.
[32] W. Zheng and G. Zhang. FastScale: Accelerate RAID scaling by minimizing data migration. In *Proc. of the FAST'11*, 2011.