Code 5-6: An Efficient MDS Array Coding Scheme to Accelerate Online RAID Level Migration

Chentao Wu[1*], Xubin He[2†], Jie Li[1*] and Minyi Guo[1*]
[1]Shanghai Key Laboratory of Scalable Computing and Systems, Department of Computer Science & Engineering, Shanghai Jiao Tong University, Shanghai, China 200240
[2]Department of Electrical & Computer Engineering, Virginia Commonwealth University, Richmond, VA 23284, USA
*{wuct, lijie, guo-my}@cs.sjtu.edu.cn, †Corresponding Author: xhe2@vcu.edu

***Abstract*—With the rapid growth of data storage, the demand for high reliability becomes critical in large data centers where RAID-5 is widely used. However, the disk failure rate increases sharply after some usage, and thus concurrent disk failures are not rare, therefore RAID-5 is insufficient to provide high reliability. A solution is to convert an existing RAID-5 to a RAID-6 (a type of "RAID level migration") to tolerate more concurrent disk failures via erasure codes, but existing approaches involve complex conversion process and high transformation cost.**

To address these challenges, we propose a novel MDS code, called "Code 5-6", to combine a new dedicated parity column with the original RAID-5 layout. Code 5-6 not only accelerates online conversion from a RAID-5 to a RAID-6, but also demonstrates several optimal properties of MDS codes. Our mathematical analysis shows that, compared to existing MDS codes, Code 5-6 reduces new parities, decreases the total I/O operations, and speeds up the conversion process by up to 80%, 48.5%, and $3.38\times$, respectively.

***Index Terms*—RAID; RAID Level Migration; MDS Code; Horizontal Parity; Diagonal Parity; Reliability**

I. Introduction

Redundant **A**rrays of **I**nexpensive (or **I**ndependent) **D**isks (**RAID**) [38] [9] especially RAID-5 has become one of the most popular choices to supply high reliability, high performance and balanced I/O for storage services. However, a RAID-5 confronts several potential threats on reliability, which are caused by various unrecoverable errors such as Undetected Disk Errors (UDEs) [23] [45], Latent Sector Errors (LSEs) [2] [47], etc. By summarizing the related results from previous literatures in Table I, we notice that the average Annualized Failure Rates (AFRs), Annualized Repair Rates (ARRs) and Annual Sector Error Rates (ASERs) increase sharply after a disk has been used for a few years. It is insufficient to meet the requirements of users (less than 1% in terms of AFR [53]). So in this paper, we set out to answer the following question:

For a large data center based on RAID-5 arrays which has run a few years, how to maintain its high reliability and tolerate concurrent disk failures?

Although several approaches are proposed to reduce unrecoverable errors and increase the reliability of disk arrays [28] [36] [15] [37] [30], a proper answer to this question is to "***convert a RAID-5 to a RAID-6***", which is a type of ***RAID level migration*** and makes the arrays tolerate more concurrent disk failures.

TABLE I
Annualized Failure Rates (AFRs), Annualized Repair Rates (ARRs) and Annual Sector Error Rates (ASERs) by Age Groups [48] [39] [2] [49] [53]

	1 Year	2 Year	3 Year	4 Year	5 Year
AFR	1.7%	8.1%	8.6%	5.8%	7.2%
ARR	0.7%	1.7%	4.3%	7.6%	6.8%
ASERs	$1.1 * 10^{-9}$	$5.7 * 10^{-9}$	–	–	–
Product Manual	Seagate: AFR is 0.73%				
	Western Digital: AFR is $0.5\% - 0.8\%$				

RAID level migration is a typical method to provide both high reliability and high flexibility for data centers [10] [51]. Many cooperations, such as HP [54], DELL [10], LSI [13], Adaptec [18], denote to supply automatic/adaptive RAID level migration. Right now, the conversion from a RAID-5 to a RAID-6 is not only involved in the road map of RAID driver in the Linux kernel [35], but also appeared in many industrial products of disk arrays [14] [24] [1], where RAID level migration is a significant function for RAID controllers.

Combining with existing erasure codes in a RAID-6 [44] [6] [3] [11] [59], there are three ways to support a conversion from an existing RAID-5[1] to a RAID-6 [34]:

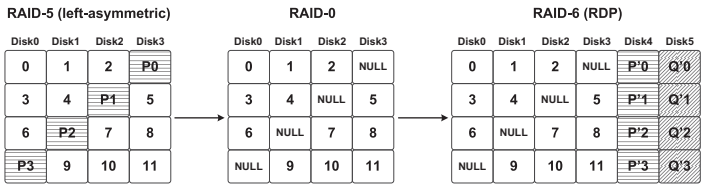
1) ***RAID-5→RAID-0→RAID-6:*** Degrade the RAID-5 to a RAID-0 and then upgrade it to a RAID-6. An example is shown in Figure 1(a), where four stripes[2] in a RAID-5 are integrated into one stripe in a RAID-6 by using the layout of RDP code [11] (shown in Figure 2). In the degrade step (RAID-5→RAID-0), all old parities in original RAID-5 are invalid after conversion and set to "NULL" for incoming requests. Two disks are added in the upgrade step (RAID-0→RAID-6) to store new parities. In the conversion process, all old parities are lost in the degrade step, and all horizontal and diagonal parities (new parities) in RDP need to be generated.

2) ***RAID-5→RAID-4→RAID-6:*** Degrade the RAID-5 to a RAID-4 and then upgrade it to a RAID-6. An example is shown in Figure 1(b), where four stripes in a RAID-5 are converted into one stripe in a RAID-6 by using the layout of RDP code [11]. In the degrade step (RAID-5→RAID-4), all

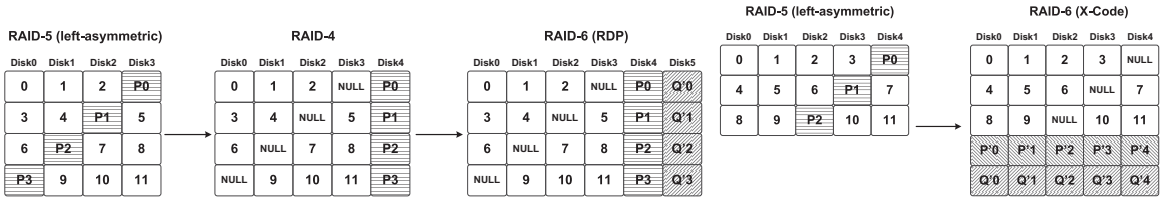
[1]In this paper, the default layout of a RAID-5 is left-asymmetric unless otherwise mentioned.

[2]In this paper, a stripe in a RAID-5 corresponds to a row, while in a RAID-6 one stripe is a matrix.

0190-3918/15 $31.00 © 2015 IEEE
DOI 10.1109/ICPP.2015.54



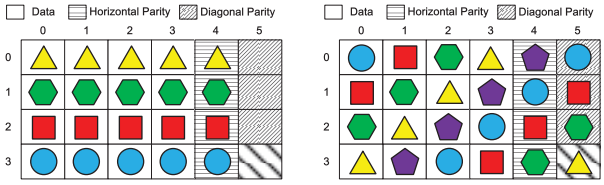
(a) RAID-5→RAID-0→RAID-6 (from 4 to 6 disks using RDP).



(b) RAID-5→RAID-4→RAID-6 (from 4 to 6 disks using RDP). (c) 5 disks RAID-5→RAID-6 Using X-Code.

Fig. 1. Three typical conversions from a RAID-5 to a RAID-6 (Parity layouts of RDP and X-Code are shown in Figures 2 and 3, respectively).

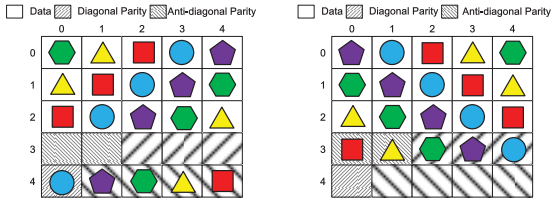
old parities in original RAID-5 (served as horizontal parities in RAID-6) are retained and migrated to a new disk. Another disk is added in the upgrade step (RAID-4→RAID-6) to save diagonal parities. In the transformation process, only diagonal parities in RDP are generated.



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

Fig. 2. RDP code for $p+1$ disks ($p=5$, $n=6$).

*3) **RAID-5→RAID-6:*** Convert the RAID-5 to a RAID-6 directly. An example is shown in Figure 1(c) by using the layout of X-Code [59] (shown in Figure 3), where one stripe in RAID-6 is aggregated by three stripes in RAID-5. All old parities in original RAID-5 are invalidated and set to "NULL", and both diagonal and anti-diagonal parities (new parities) in X-Code are generated in the conversion process.



(a) Diagonal parity coding. (b) Anti-diagonal parity coding.

Fig. 3. X-Code for p disks ($p=5$, $n=5$).

However, existing conversion methods have some problems. The first two approaches need use some intermediate RAID forms (either a RAID-0 or a RAID-4) during the conversion, which increases the complexity and overhead of the conversion process due to the upgrade and degrade steps. An ideal solution is to convert a RAID-5 to a RAID-6 directly, but existing solutions as outlined above have some limitations, such as extra space in the original RAID-5. For example, in Figure 1(c), 40% capacity of each disk is reserved before conversion. Second, the conversion costs of existing approaches are very expensive in terms of I/O operations. These I/Os come from moving/erasing old parities and generating new parities. For example, as shown in Figure 1(a), the total number of data blocks is 12 where 4 blocks are old parities and 8 blocks are served as new parities. All old parities will be set to "NULL" for upcoming requests. Thus the total write I/Os is $8+4=12$, which is the same amount of I/Os as all data blocks are written to a new RAID-0. Third, both availability and reliability of a disk array is decreased for online applications. For a typical conversion from a RAID-5 to a RAID-6, a section of the disk array (including all I/O blocks in this section) is frozen in the conversion process [35]. Until the conversion is complete, these blocks cannot be accessed by any current I/O requests, nor be recovered when a corresponding parity is lost or fails.

To address these problems, in this paper, we propose a novel MDS code called **Code 5-6**. It utilizes the original layout of a RAID-5 by simply adding a dedicated parity column, which supports fast bidirectional transformation between a RAID-5 and a RAID-6. Depending on the number of disks in a disk array, Code 5-6 is a solution for p disks, where p is a prime number (we will discuss how Code 5-6 supports any number of disks in a RAID-5 in Section IV-B).

We make the following contributions in this work:

- We propose a novel XOR-based RAID-6 code (Code 5-6) to offer not only the property provided by typical MDS codes such as optimal storage efficiency, and optimal encoding/decoding computational complexity, but also high efficiency on online bidirectional conversions between a

TABLE II
SYMBOLS

Parameters & Symbols	Description
m (m', m'')	number of disks in a RAID-5 (before conversion)
n	number of disks in a RAID-6 (after conversion)
v	number of virtual disks
p	a prime number
w	word size (can be regarded as "number of rows in each stripe" when an element size is a few bits)
i, r	row ID
j	column ID or disk ID (before transformation)
B	total number of data elements/blocks in a disk array (Typically, an element corresponds to a block)
$C_{i,j}$	element at the ith row and jth column
f_1, f_2 ($f_1 < f_2$)	two random failed columns with IDs f_1 and f_2
$\sum$	XOR operations on elements (e.g., $\sum_{j=0}^{5} C_{i,j} = C_{i,0} \oplus C_{i,1} \oplus \cdots \oplus C_{i,5}$)
$\langle\ \rangle$	modular arithmetic (e.g., $\langle i \rangle_p = i \bmod p$)
T_e	access time of a read/write request to an element

RAID-5 and a RAID-6 due to its special layout.

- We design bidirectional conversion algorithms based on Code 5-6, which provides concurrent RAID level migration between a RAID-5 and a RAID-6.
- We conduct a series of quantitative analysis on various conversion methods based on different MDS codes, and show that Code 5-6 achieves higher conversion efficiency compared to other existing approaches.

The rest of this paper continues as follows: Section II discusses the related work of existing RAID-6 codes and the motivation of our work. Code 5-6 is described in detail in Section III. The online conversion algorithms are discussed in Section IV. Section V gives the quantitative analysis on the conversion cost among various approaches. Finally we conclude our work in Section VI.

II. RELATED WORK AND MOTIVATION

In this section we discuss existing erasure codes and our motivation. To facilitate the discussion, we summarize the symbols used in this paper in Table II.

A. Existing Erasure Codes

RAID-6 implementations are based on various erasure coding technologies and can be divided into two categories: MDS codes and non-MDS codes. **M**aximum **D**istance **S**eparable (**MDS**) codes in RAID-6 can be further divided into two subclasses: horizontal codes and vertical codes. Horizontal codes include Reed-Solomon codes [44], Cauchy Reed-Solomon codes [6], EVENODD code [3], RDP code [11], Blaum-Roth code [5], Liberation code [41], Liber8tion code [40], etc. Vertical codes contain B-Code [60], X-Code [59], P-Code [29], Cyclic code [8], H-Code [55], HDP code [56] and HV Code [50]. Typically, non-MDS codes involve LDPC codes [19] [32] [42], WEAVER code [22], HoVer codes [21], MEL code [57], Pyramid code [26], Flat XOR-Code [20] and Code-M [52]. Besides these codes in RAID-6, many erasure codes such as RSL-Code [16], RL-Code [17], STAR code [27], HoVer codes [21], GRID Codes [31], Local Reconstruction Codes [25], partial-MDS codes [4], Locally Repairable Codes[46], SD codes[43], can tolerate concurrent failures of three or more disks and be applied in cloud computing systems.

TABLE III
COMPARISON AMONG DIFFERENT MDS CODES ON CONVERSION FROM A RAID-5 TO A RAID-6

MDS Codes	Single Write Performance	Conversion Complexity	Conversion Efficiency
EVENODD	Low	High	Low
RDP	Medium	High	Low
X-Code	High	Medium	Medium
P-Code	High	Medium	Medium
H-Code	High	High	Low
HDP	Medium	Medium	Medium
Code 5-6	High	Low	High

B. Motivation

RAID-5 systems are widely used in today's large scale data centers, where reliability is a big challenge. First, the failure rate of disks increases sharply after a few years of usage as shown in Table I. Second, when a disk array scales up, concurrent disk failures are not uncommon. Therefore it's critical to tolerate multiple failures in large data centers. One possible solution is to use RAID-6. However, for large data centers, the cost is very high to build a RAID-6 from the scratch. An alternative way is to convert an existing RAID-5 based storage system to a RAID-6 on the fly and thus to improve the reliability and still provide service without affecting the availability of the system.

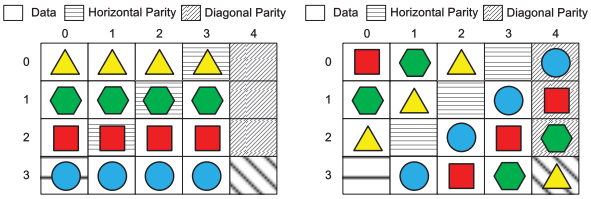
As summarized in Table III, for a conversion from a RAID-5 to a RAID-6, the conversion efficiency is decided by: 1) single write performance, which indicates the I/O cost to move a data element/block; 2) conversion complexity (the total number of I/Os in a conversion), which is decided by the conversion type (i.e., RAID-5→RAID-4→RAID-6 or RAID-5→RAID-6) and the differences between the layouts of RAID-5 and RAID-6. According to Table III, existing coding schemes have high conversion complexity, which motivates us to propose a novel code to achieve efficient RAID level migration.

III. CODE 5-6

Code 5-6 accelerates online transformation from a RAID-5 to a RAID-6 by adding one column to store diagonal parities. Code 5-6 is a solution for p disks, where p is a prime number.

A. Data/Parity Layout and Encoding of Code 5-6

Code 5-6 is composed of a $(p-1)$-row-p-column matrix with a total number of $(p-1)*p$ elements. There are three types of elements in the matrix: ***data elements***, ***horizontal parity elements***, and ***diagonal parity elements***. $C_{i,j}$ ($0 \le i \le p-1$, $0 \le j \le p-2$) denotes the element at the ith row and the jth column. The last column (column $p-1$) is used for diagonal parity. Excluding the last column, the remaining matrix is a



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

Fig. 4. Code 5-6 for p disks ($n = p = 5$).

$(p-1)$-row-$(p-1)$-column square matrix. The anti-diagonal part of this matrix is used to represent horizontal parity.

Horizontal parity and diagonal parity elements of Code 5-6 are constructed according to the following encoding equations,

Horizontal parity:

$$C_{i,p-2-i} = \sum_{j=0}^{p-2} C_{i,j} \quad (j \neq p-2-i) \tag{1}$$

Diagonal parity:

$$C_{i,p-1} = \sum_{j=0}^{p-2} C_{\langle 4-p+i-j \rangle_p, j} \quad (j \neq i) \tag{2}$$

The encoding of the horizontal parity is shown in Figure 4(a). We use different icon shapes to denote different sets of horizontal parity chains (including a horizontal parity element and its corresponding data elements). The horizontal parity elements are calculated according to Equation 1. For example, $C_{0,3}$ can be calculated by $C_{0,0} \oplus C_{0,1} \oplus C_{0,2}$. The encoding of the diagonal parity is given in Figure 4(b). According to Equation 2, the diagonal parity elements can be calculated through modular arithmetic and XOR operations. For example, to calculate the diagonal parity element $C_{1,4}$ ($i = 1$), first we fetch the proper data elements ($C_{\langle 4-p+i-j \rangle_p, j}$). When $j = 0$, $\langle 4-p+i-j \rangle_p = \langle 0 \rangle_p = 0$, so the first data element is $C_{0,0}$. The following data elements can be calculated in the same way, which are $C_{3,2}$ and $C_{2,3}$. Second, the diagonal parity element ($C_{1,4}$) is constructed via XOR operations on these data elements (i.e., $C_{1,4} = C_{0,0} \oplus C_{3,2} \oplus C_{2,3}$).

B. Construction Process

According to the above encoding scheme, the construction process of Code 5-6 is straightforward:

- Label all data elements and calculate both horizontal and diagonal parity elements according to Equations 1 and 2.

C. Proof of Correctness

To prove the correctness of Code 5-6, we take a case of one stripe. The reconstruction of multiple stripes is just a matter of scale and similar to the reconstruction of one stripe. In a stripe, we have the following lemma and theorem,

Lemma 1: We can find a sequence of a two-integer tuple (T_k, T'_k) where

$$T_k = \left\langle p-2+\frac{k+1+\frac{1+(-1)^k}{2}}{2}(f_2-f_1) \right\rangle_{p-1},$$
$$T'_k = \frac{1+(-1)^k}{2} f_1 + \frac{1+(-1)^{k+1}}{2} f_2 \quad (k = 0, 1, \cdots, 2p-3)$$

with $0 < f_2 - f_1 < p-1$, all two-integer tuples $(0, f_1)$, $(0, f_2)$, $\cdots$, $(p-2, f_1)$, $(p-2, f_2)$ occur exactly once in the sequence.

Similar proofs of the lemma can be found in previous papers on RAID-6 codes [11] [29] [55] [56].

Theorem 1: A $(p-1)$-row-p-column stripe constructed according to the formal description of Code 5-6 can be reconstructed under concurrent failures of any two columns.

Proof: There are two cases of double failures, depending on whether diagonal parity column fails or not.

Case I: diagonal parity column fails (column $p-1$), and the other failed column is a data column. From the construction of Code 5-6, when one data column fails, all data and parity elements can be recovered through the horizontal parity chains. After the lost data elements are recovered, the diagonal parity elements can be reconstructed by Equation 2.

Case II: diagonal parity column doesn't fail. Both two failed columns are data columns (denoted by f_1 and f_2, where $0 < f_1 < f_2 < p-1$).

In the construction of Code 5-6, any two horizontal parity elements cannot be placed in a same row, as well for any two diagonal parity elements. When two columns fail (f_1 and f_2), based on the layout of Code 5-6, two data elements $C_{f_2-f_1-1,f_1}$ and $C_{p-1-f_2+f_1,f_2}$ can be recovered since only one lost element in their corresponding diagonal parity chains.

For the failed columns f_1 and f_2, if a data element C_{i,f_2} on column f_2 can be reconstructed, another missing element can be reconstructed C_{i,f_1} by the same horizontal parity chain if the corresponding parity exists. Similarly, a data element C_{r,f_1} in column f_1 can be reconstructed, we can reconstruct the missing data element C_{r,f_2}. Then according to C_{i,f_1} and C_{r,f_2}, more elements can be restored until reaching the endpoints of the reconstruction sequence (C_{p-2-f_1,f_1} and C_{p-2-f_2,f_2}). This sequence is based on the sequence of the two-integer tuple in Lemma 1, which traverses all lost elements just once. Therefore all lost elements can be recovered.

In conclusion, Code 5-6 can be reconstructed under concurrent failures from any two columns. ■

D. Reconstruction Process

We first consider how to recover a lost data element since any missing parity element can be recovered based on Equations 1 and 2. If a horizontal parity element and the related $p-3$ data elements are retained, we can recover the missing data element (assuming it's C_{i,f_1} in column f_1 and $0 \leq f_1 \leq p-2$) using the following equation,

$$C_{i,f_1} = \sum_{j=0}^{p-2} C_{i,j} \quad (j \neq f_1) \tag{3}$$

If there exists a diagonal parity element and its $p-3$ data elements, to recover the lost data element (C_{i,f_1}), first we should obtain the expression of the diagonal parity element. Assume the diagonal parity element is in row r and represented by $C_{r,p-1}$ (based on Equation 2), we have,

$$r = \langle i + f_1 - 4 \rangle_p \tag{4}$$

Based on Equation 2, the lost data element can be recovered,

$$C_{i,f_1} = C_{r,p-1} \oplus \sum_{j=0}^{p-2} C_{\langle i+f_1-j \rangle_p, j} \quad (j \neq f_1 \quad and \quad j \neq \langle i + f_1 + 6 \rangle_p) \tag{5}$$

Algorithm 1: Reconstruction Algorithm of Code 5-6

Step 1: Identify the double failure columns: f_1 and f_2 ($f_1 < f_2$).
Step 2: Start reconstruction process and recover the lost data and parity elements.
switch $0 \leq f_1 < f_2 \leq p-2$ **do**
  case *I:* $f_2 = p-2$ *(diagonal parity column is lost)*
    Step 2-IA: Recover the lost data and horizontal parity elements in column f_1 based on Equations 1 and 3;
    Step 2-IB: Recover the lost diagonal parity elements in column f_2 based on Equation 2.
  endsw
  case *II:* $f_2 \neq p-2$ *(diagonal parity column is saved)*
    Step 2-IIA: Compute two starting points ($C_{f_2-f_1-1,f_1}$ and $C_{p-1-f_2+f_1,f_2}$) of the recovery chains based on Equation 5.
    Step 2-IIB: Recover the lost data elements in the two recovery chains.
    Two cases start synchronously:
    case *starting point is* $C_{f_2-f_1-1,f_1}$ **repeat**
      (1) Compute the next lost element (in colomn f_2) in the recovery chain based on Equations 1 and 3;
      (2) Then compute the next lost data element (in colomn f_1) in the recovery chain based on Equation 5.
    until *at the endpoint of the recovery chain* (C_{p-2-f_2,f_2}).
    case *starting point is* $C_{p-1-f_2+f_1,f_2}$ **repeat**
      (1) Compute the next lost element (in colomn f_2) in the recovery chain based on Equations 1 and 3;
      (2) Then compute the next lost data element (in colomn f_1) in the recovery chain based on Equation 5.
    until *at the endpoint of the recovery chain* (C_{p-2-f_1,f_1}).
  endsw
endsw

Based on Equations 1 to 5, we can easily recover the lost elements upon a single disk failure. If two disks fail (for example, column f_1 and column f_2, $0 \leq f_1 < f_2 \leq p-2$), based on Theorem 1, we have our reconstruction algorithm of Code 5-6 as shown in Algorithm 1. A reconstruction case in Algorithm 1 (recovered by two chains) is shown in Figure 5.

E. Properties of Code 5-6

1) ***Optimal Storage Efficiency:*** From the proof of Code 5-6's correctness, Code 5-6 is essentially an MDS code, which has optimal storage efficiency [59] [8] [29].

2) ***Optimal Encoding/Decoding Computational Complexity:*** From the encoding and decoding equations (Equations 1 to 5) of Code 5-6, each equation has $(p-3)$ XOR operations. To generate all $2*(p-1)$ parities in a stripe, $2*(p-1)*(p-3)$ XOR operations should be applied which is $\frac{2*(p-1)*(p-3)}{(p-1)*(p-2)} = \frac{2p-6}{p-2}$ XOR operations per data element. On the other hand, to recover any element, $(p-3)$ XOR calculations should be taken.

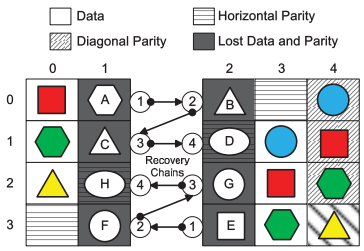


Fig. 5. Reconstruction by two recovery chains (Assume that double failures are in columns 1 and 2): First we identify the two starting points of recovery chain: data elements A and E. Second we reconstruct data elements according to the corresponding recovery chains until they reach the endpoints (parity elements D and H).The orders to recover data and parity elements are: one is A→B→C→D, the other is E→F→G→H.

Suppose n_d is the number of data elements and n_e is the total number of elements in each stripe. It is demonstrated that the optimal encoding and decoding computational complexity are $\frac{3*n_d-n_e}{n_d}$, $\frac{3*n_d-n_e}{n_e-n_d}$, respectively [29]. For Code 5-6, $n_d = (p-2)*(p-1)$ and $n_e = p*(p-1)$. Thus the optimal computational complexity are $\frac{3*n_d-n_e}{n_d} = \frac{2p-6}{p-2}$, $\frac{3*n_d-n_e}{n_e-n_d} = p-3$, which are the same as the results based on encoding and decoding equations. Therefore Code 5-6 has optimal encoding/decoding computational complexity.

3) ***Optimal Single Write Performance (also known as "Optimal Update Complexity"):*** From the construction of Code 5-6, each data element takes part into the generation of two and only two parity elements. Therefore, a single write on one data element in Code 5-6 only causes two modifications on parity elements, which has been proved to be optimal [55].

4) ***High Reliability on Single Disk Recovery:*** Xiang *et al.* [58] proposes a hybrid recovery approach based on RDP code, which uses both horizontal and diagonal parities to recover single disk failure. It can minimize I/O cost and reduce the recovery time up to 12.60%, which can be used in many MDS codes to provide higher reliability [56].

This approach can also be applied in Code 5-6 to achieve high reliability. For example, as shown in Figure 6, when column 1 fails, not all elements need to be read for recovery. Because some elements such as $C_{2,2}$ are shared to recover two failed elements in different parities. By this approach, when $p = 5$, Code 5-6 reduces up to 33% read operations (9 read I/Os vs. 12 read I/Os by typical recovery approach) per stripe to recover single disk, which decreases the recovery time (MTTR) and thus increase the reliability of the disk array.

IV. Online Conversion Algorithms

A. Bidirectional Conversions between RAID-5 and RAID-6

Bidirectional conversions between a RAID-5 and a RAID-6 (a type of RAID level migration) are straightforward using Code 5-6, adding one disk to a RAID-5 as a dedicated parity disk to get a RAID-6, or removing the last column from a RAID-6 to restore RAID-5. Although Code 5-6 supports

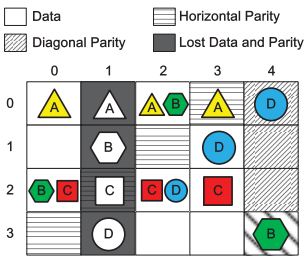


Fig. 6. Single disk recovery in Code 5-6 when column 1 fails (the corresponding data or parity elements to recover the failed elements are marked with the same character, e.g., element $C_{2,0}$ marked with "BC" is used to recover the elements B and C).

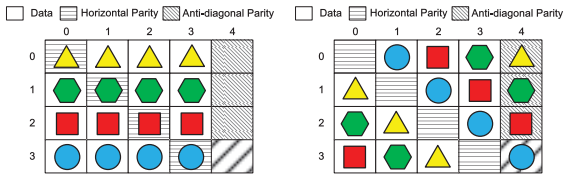
bidirectional conversion, our discussion and analysis focus on conversion from a RAID-5 to a RAID-6 simply because there is not much demand to convert a RAID-6 to a RAID-5.

To implement online conversions, both conversion and online application I/Os should be considered to maintain data consistency. From the layout of Code 5-6, during the conversion from a RAID-5 to a RAID-6, only last column (column $p-1$) has write I/Os and other columns have read I/Os, so there is no conflict between an online read request and the conversion process. However, for online write request to a data block, it will modify the corresponding parity blocks and thus causing write I/Os on the last column. To avoid the data consistency problems, conversion process is interrupted until the response of this request is done. In a word, we treat various online requests differently in our conversion algorithm.

The conversion algorithms are shown in Algorithm 2.

B. Further Discussion

1) ***Support on Different RAID-5 Layouts:*** So far, only left-asymmetric RAID-5 is considered in the conversion. For the left-symmetric layout, Code 5-6 can also be used because it has the same parity distribution as left-asymmetric. If the layout of RAID-5 is right-symmetric or right-asymmetric, similar conversion can be easily developed as shown in Figure 7 based on the symmetry of horizontal codes.



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

Fig. 7. Code 5-6 for right-symmetric/asymmetric RAID-5 ($n = p = 5$).

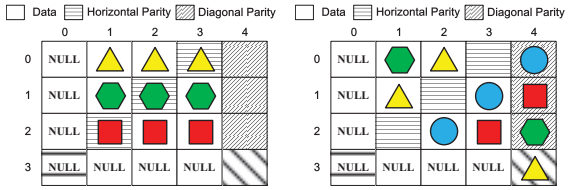
2) ***Support on Any Number of Disks in a RAID-5:*** By default, Code 5-6 is limited to convert a RAID-5 of $p-1$ disks, where p is a prime number. To convert a RAID-5 with any number of disks(> 2), we introduce virtual disks. Each virtual disk has infinite number of "NULL" data blocks and doesn't physically exist. To convert a RAID-5 of m disks to a RAID-6 of p disks (where p is a prime number closest to but greater than m), we add v virtual disks where $v = p - m - 1$ to the original RAID-5, so that the RAID-5 has a layout of $p-1$ disks, where the first v disks are labeled as virtual disks.

Algorithm 2: Bidirectional Conversions between a RAID-5 and a RAID-6

Conversion from a RAID-5 to a RAID-6
Step 1: Check the status of the RAID-5 (e.g., $m == p-1$).
Step 2: Add a new disk to the array and labeled as column $p-1$.
Step 3: Start two threads simultaneously. One handles online application, the other one is for conversion process.
/*Online application thread*/
if *read request to a data block* **then**
  read the data block.
end
if *write request to a data block (including new write and update)* **then**
  1) interrupt the conversion thread;
  2) read the original data block (if exist) and the corresponding parity blocks (if exist);
  3) calculate and modify their corresponding parity blocks, write the data block;
  4) resume the conversion thread.
end
/*Conversion thread*/
if *a diagonal parity block has not been generated in column* $p-1$ **then**
  1) read the corresponding data blocks;
  2) calculate the diagonal parity based on Equation 2;
  3) write the diagonal parity block.
end
Conversion from a RAID-6 to a RAID-5
Step 1: Check the status of the RAID-6 (e.g., $n == p$).
Step 2: Delete the last disk (column $p-1$).

To distinguish the existing elements and elements in a virtual disk, we define ***virtual elements***, which include, (1) all data and parity elements in virtual disk(s); (2) the data elements whose corresponding parity element is in virtual disk(s). Typically, all virtual elements are treated as "NULL" data blocks and don't physically exist.

We give an example to show how virtual disks work in the conversion from a RAID-5 to a RAID-6 (shown in Figure 8). The RAID-5 has 3 disks ($m = 3$), so the RAID-6 will have 5 disks ($p = 5$, which is the next prime number which is greater than m). We need add 1 virtual disk ($v = p - m - 1 = 1$) as the first column. According to the definition of virtual elements, $C_{0,0}$, $C_{1,0}$, $C_{2,0}$, $C_{3,0}$, $C_{3,1}$, $C_{3,2}$ and $C_{3,3}$ are virtual elements. By this way, three stripes in original RAID-5 can be constructed to one stripe in the new RAID-6.



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

Fig. 8. Conversion from a RAID-5 to a RAID-6 when $m = 3$ and $n = 4$ (the elements marked as "NULL" are virtual elements).

By the introduction of virtual disks, Code 5-6 can covert

a RAID-5 of any disks to a RAID-6. It slightly sacrifices the storage efficiency, which is calculated by,

$$\frac{m*(m-1)}{m*(m+1)+v} = \frac{(n-1)*(n-2)}{(n-1)*n+v} < \frac{n-2}{n} \quad (6)$$

As shown in Figure 8, when $m = 3$ and $n = 4$, the storage efficiency is $\frac{(4-1)*(4-2)}{(4-1)^4+1} = 6/13$, which is smaller than MDS codes in RAID-6 ($\frac{n-2}{n} = \frac{4-2}{4} = 1/2$). Our analysis in next section also shows that the penalty on storage efficiency is marginal as shown in Figure 18.

V. Conversion Analysis

In this section, we give our analysis to demonstrate the effectiveness of our Code 5-6 to convert a RAID-5 to RAID-6.

A. Methodology

We select three conversion approaches discussed in Section I and consider various RAID-6 codes:

1) RAID-5→RAID-0→RAID-6 and RAID-5→RAID-4→RAID-6: These two approaches are suitable for several codes and we choose EVENODD [3], RDP [11] and H-Code [55] for comparison.

2) RAID-5→RAID-6: This approach is appropriate for most vertical codes, and we pick X-Code [59], P-Code [29], HDP [56] and Code 5-6 for comparison.

In order to ensure fairness of various codes, we also select the proper layout of RAID-5 and the proper number of disks in our comparison. For example, H-Code has special anti-diagonal parity layout, which is suitable for conversion from right-asymmetric RAID-5 to RAID-6, where the best case is that an original RAID-5 of $p-1$ disks is converted to a RAID-6 by adding 2 disks. Although conversion algorithms based on Code 5-6 can handle online I/O requests, only conversion workloads are considered in our analysis.

In our comparison, "RAID-5→RAID-0→RAID-6 (RDP,4,6)" denotes to convert a RAID-5 of 4 disks to a RAID-6 of 6 disks via RAID-0 using RDP. Similarly, RAID-5→RAID-6 (Code 5-6,4,5) means the conversion from a RAID-5 of 4 disks to a RAID-6 of 5 disks directly using Code 5-6.

In the conversion process, there are three types of operations for parities: 1) set invalid parities to "NULL"; 2) migrate old parities to a new disk; 3) generate new parities. According to various types of operations on parities, we use the following metrics to evaluate the efficiency of various conversions.

1) Invalid Parity Ratio: The ratio between the number of invalid parity blocks (in original RAID-5 and set to "NULL" in the conversion process) and the total number of data blocks. In our analysis, an element corresponds to a data block and we assume the total number of data blocks is B.

2) Old Parity Migration Ratio: The ratio between the sum of migrated parity blocks and the total number of data blocks.

3) New Parity Generation Ratio: The ratio between the number of newly generated parity blocks and the total number of data blocks.

4) Extra Space Ratio: The ratio between the required extra space[3] and the total capacity of each disk.

5) Computation Cost: Total number of XOR operations in the conversion process.

6) Write I/Os: Total number of write operations in the conversion process.

7) Total I/Os: Total number of I/O operations in the conversion process.

8) Conversion Time: Time cost of the conversion process. We assume the same time on a read or write request to a data/parity element and ignore the computation time which is much smaller compared to the disk I/O time.

For example, for a conversion RAID-5→RAID-6(Code 5-6,4,5), the invalid parity ratio, old parity migration & modification ratio and free space ratio are zero. The new parity generation ratio is $\frac{4}{3*4} = 1/3$ and the write I/Os are $\frac{B}{3}$, thus the total I/Os are $B + \frac{B}{3} = \frac{4B}{3}$. The computation cost is $\frac{B}{12} * 2 * 4 = \frac{2B}{3}$ and the conversion time is $\frac{B*T_e}{3}$.

B. Numerical Results

As mentioned in previous section, different implementation methods of RAID-6 [12] [33] are proposed to address load balancing problem for MDS codes. One is implemented with load balancing support, which distributes dedicated parity alternatively for every a few stripes. The other is without load balancing support. It is reasonable because in some cases reliability is the most significant aspect for disk arrays, especially converting to RAID-6 by sacrificing good performance of original RAID-5. The results on under load balancing support are similar to without load balancing support, and in this section we only give the results for conversion time.

First, we calculate the invalid parity ratio, old parity migration ratio, new parity generation ratio and extra space ratio under various approaches as shown in Figures 9 to 12. It is clear that direct conversion from a RAID-5 to a RAID-6 has the smallest ratio among these metrics, which means the lowest cost on old parity migration (decrease by up to 100.0%) and new parity generation (decrease by up to 80.0%).

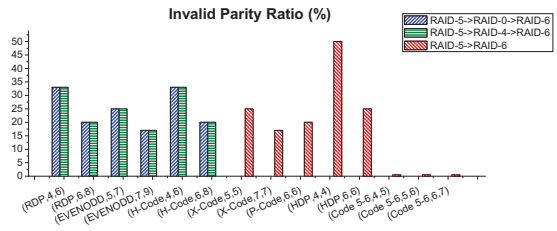


Fig. 9. Comparison on invalid parity ratio under different conversion approaches using various RAID-6 codes.

Second, using various approaches, we compare the computation cost, write I/O operations, total number of I/O operations and conversation time as shown in Figures 13 to 16. With

[3]e.g., extra space is needed for RAID-5→RAID-6(X-Code) as shown in Figure 1(c).

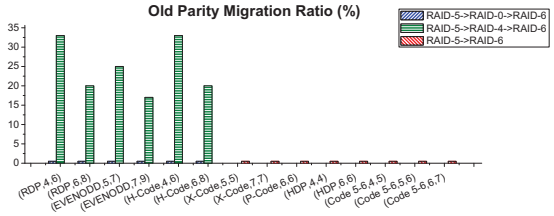


Fig. 10. Comparison on old parity migration ratio under different conversion approaches using various RAID-6 codes.

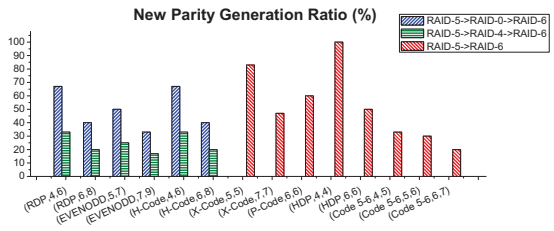


Fig. 11. Comparison on new parity generation ratio under different conversion approaches using various RAID-6 codes.

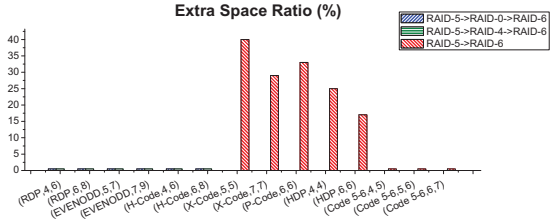


Fig. 12. Comparison on extra space ratio under different conversion approaches using various RAID-6 codes.

increasing number of disks, the computation cost rises due to longer parity chains, while the write I/Os and total I/Os reduce because fewer portions of parities. The conversion time also reduces because more disks run in parallel. Even if several codes have larger number of disks in some cases, Code 5-6 which directly converts a RAID-5 to RAID-6 has the lowest computation cost, fewest write I/Os and total I/Os. It decreases the computation cost, write I/Os and total I/Os by up to 76.4%, 80.0%, 48.5%, respectively.

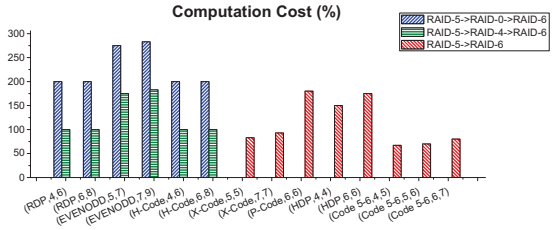


Fig. 13. Comparison on computation cost under different conversion approaches using various codes (B XOR operations is normalized to 100%).

Third, we calculate the conversion time under two scenarios,

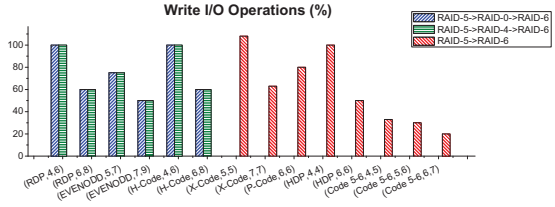


Fig. 14. Comparison on write I/Os under different conversion approaches using various RAID-6 codes (B I/O operations is normalized to 100%).

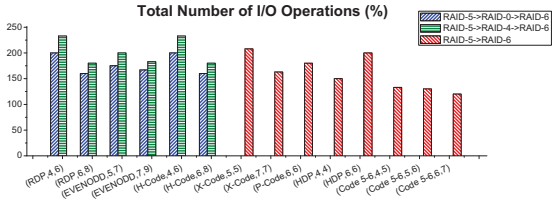


Fig. 15. Comparison on total I/Os under different conversion approaches using various RAID-6 codes (B I/O operations is normalized to 100%).

with load balancing support (LB) and without load balancing support (NLB). The results are shown in Figures 16 and 17. We summarize various codes with their best conversion approaches under the same values of n as shown in Table IV. We notice that Code 5-6 can accelerate the conversion process compared to other codes by up to 150%.

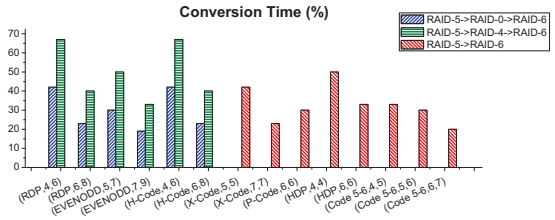


Fig. 16. Comparison on conversion time under different conversion approaches by using various RAID-6 codes without load balancing support (Time $B * T_e$ is normalized to 100%).

Finally, we calculate the storage efficiency with various numbers of disks (m) in a RAID-5, which is discussed in Section IV-B2. The storage efficiency of a typical RAID-6 with $m+1$ disks is $\frac{m+1-2}{m+1} = \frac{m-1}{m+1}$. According to Equation

TABLE IV
SPEEDUP OF CODE 5-6 USING DIRECT CONVERSION FROM A RAID-5 TO A RAID-6 OVER OTHER CODES USING THEIR BEST APPROACHES IN TERMS OF CONVERSION TIME

n	RDP	EVENODD	H-Code	X-Code	P-Code	HDP
$n = 5$ (NLB)	–	–	–	1.27×	–	–
$n = 6$ (NLB)	1.40×	–	1.40×	–	1.00×	1.15×
$n = 7$ (NLB)	–	1.50×	–	1.15×	–	–
$n = 5$ (LB)	–	–	–	1.27×	–	–
$n = 6$ (LB)	1.22×	–	1.22×	–	1.11×	1.22×
$n = 7$ (LB)	–	1.25×	–	1.15×	–	–

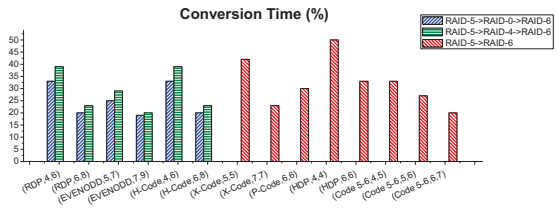


Fig. 17. Comparison on conversion time under different conversion approaches by using various RAID-6 codes with load balancing support (Time $B * T_e$ is normalized to 100%).

TABLE V
SPEEDUP OF CODE 5-6 USING DIRECT CONVERSION FROM A RAID-5 TO A RAID-6 OVER OTHER CODES USING THEIR BEST APPROACHES IN TERMS OF CONVERSION TIME

p	RDP	EVENODD	H-Code	X-Code
$p = 5$ (LB)	2.24×	1.48×	3.00×	2.71×
$p = 7$ (LB)	2.40×	1.83×	3.38×	2.76×

6, the storage efficiency of a RAID-6 using Code 5-6-NLB with $m + 1$ disks is $\frac{m*(m-1)}{m*(m+1)+v} = \frac{(n-1)*(n-2)}{(n-1)*n+v}$. The results are shown in Figure 18 and we notice that virtual disks have minor effect on the storage efficiency (less than 3.8%).

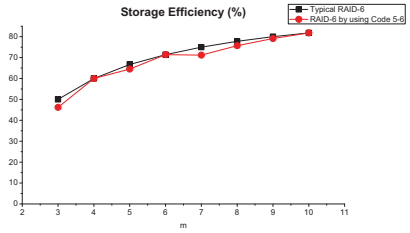


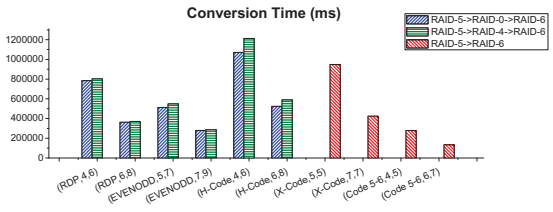
Fig. 18. Storage efficiency comparison between typical RAID-6 and RAID-6 using Code 5-6.

C. Simulation Results

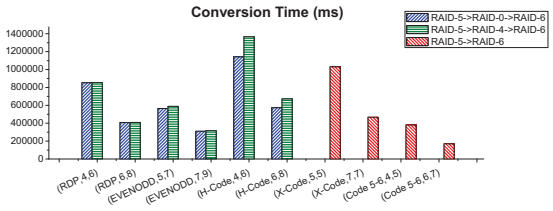
In this section, we give the simulation results on conversion time by using various conversion approaches. First we generate different synthetic traces for the migration I/Os by using various coding schemes, which is based on the results of mathematical analysis in Figures 14 and 15. The block size is set to 4KB or 8KB (typically the stripe size is 128KB or 256KB) and the total number of data blocks (B) is set to 0.6 million. Then we use Disksim [7] as the simulator to evaluate the conversion time, which is the overall time to handle all I/O requests in these traces. The results are shown in Figure 19 and Table V. It is clear that, with the same value of p, Code 5-6 saves the conversion time by up to 89.0%. When p becomes larger (from 5 to 7), Code 5-6 achieves higher speedup.

D. Analysis

From above results, compared to other codes, Code 5-6 has great advantages on conversion from a RAID-5 to a RAID-6 (a type of RAID level migration). There are several reasons



(a) block size is 4KB.



(b) block size is 8KB.

Fig. 19. Comparison on conversion time under different conversion approaches by using various RAID-6 codes with load balancing support.

TABLE VI
RELIABILITY OF CONVERSIONS (FROM A RAID-5 TO A RAID-6)

Conversions Type	Conversions RAID-6 codes	Reliability	Note
RAID-5→ RAID-0→ RAID-6	EVENODD RDP H-Code	Low	No fault tolerance in RAID-0
RAID-5→ RAID-4→ RAID-6	EVENODD RDP H-Code	Medium	Errors may occur when old parity blocks are migrated (RAID-5→RAID-4)
RAID-5→ RAID-6	X-Code P-Code HDP	High	Old parity blocks in RAID-5 should be retained until conversion is done
RAID-5→ RAID-6	Code 5-6	High	No risk on parity loss

to achieve these gains. First, the layout of Code 5-6 is based on RAID-5 by adding a parity column, which minimizes the parity migration/generation cost, computation cost and total I/Os, and thus it accelerates the conversion process. Second, Code 5-6 distributes the conversion I/Os and online application I/Os among different disks, which guarantees well online bidirectional conversions between RAID-5 and RAID-6 with high reliability compared to other transformations (shown in Table VI). Third, Code 5-6 provides high write performance after conversion due to its property on single write performance.

VI. CONCLUSIONS

In this paper, we propose a novel MDS code in RAID-6 called Code 5-6, to improve the efficiency of RAID level migration from a RAID-5 to a RAID-6. Based on our comprehensive analysis, compared to other approaches, Code 5-6 not only offers properties provided by typical MDS codes such as optimal storage efficiency, optimal encoding/decoding computational complexity, optimal single write performance, but also achieves high efficiency to convert a RAID-5 to a RAID-6 as evidenced by low computation cost, less I/Os and faster conversion process.

VII. Acknowledgements

We thank anonymous reviewers for their insightful comments. This work is partially sponsored by the National 973 Program of China (No.2015CB352403), the National Natural Science Foundation of China (No.61261160502, No.61272099, No.61332001, No.61303012, No. 61202025 and No. 61428204), the Program for Changjiang Scholars and Innovative Research Team in University (IRT1158, PCSIRT), the Scientific Innovation Act of STCSM (No.13511504200), the Shanghai Natural Science Foundation (No.13ZR1421900), and the Scientific Research Fund for the Returned Overseas Chinese Scholars. The work performed at VCU is partially sponsored by the U.S. NSF Grants CNS-1218960 and CNS-1320349. 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 the funding agencies.

References

[1] Adaptec Inc. Adaptec Storage Manager: User's Guide. http://download.adaptec.com/pdfs/user_guides/adaptec_storage_manager_v4_users_guide_00030-01-a.pdf.

[2] L. Bairavasundaram et al. An analysis of latent sector errors in disk drives. In *Proc. of the SIGMETRICS'07*.

[3] 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.

[4] M. Blaum et al. Partial-mds codes and their application to raid type of architectures. *IEEE Trans. on Inf. Theory*, 59(7):4510–4519, Jul. 2013.

[5] M. Blaum and R. Roth. On lowest density MDS codes. *IEEE Trans. on Information Theory*, 45(1):46–59, Jan. 1999.

[6] J. Blomer et al. An XOR-based Erasure-Resilient coding scheme. Technical Report TR-95-048, Int'l Comp. Sci. Institute, Aug. 1995.

[7] J. Bucy, J. Schindler, et al. The disksim simulation environment version 4.0 reference manual. Technical Report CMU-PDL-08-101.

[8] Y. Cassuto and J. Bruck. Cyclic lowest density MDS array codes. *IEEE Trans. on Information Theory*, 55(4):1721–1729, Apr. 2009.

[9] P. Chen et al. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, June 1994.

[10] Dell Compellent. Fluid data storage: Driving flexibility in the data center. A Dell Technical White Paper, February 2011.

[11] P. Corbett et al. Row-Diagonal Parity for double disk failure correction. In *Proc. of the FAST'04*.

[12] EMC Corporation. EMC CLARiiON RAID 6 technology: A detailed review. Technical Report H-2891, EMC Corporation, July 2007.

[13] R. DeKoning. Methods and structure for raid level migration within a logical unit. US Patent No. 6275898 B1, Aug. 2001.

[14] Dell Inc. Manuals: About PERC 6 and CERC 6/i Controllers. http://support.dell.com/support/edocs/storage/RAID/PERC6/en/UG/HTML/chapterd.htm#wp1070638.

[15] J. Elerath and M. Pecht. Enhanced reliability modeling of RAID storage systems. In *Proc. of the DSN'07*.

[16] G. Feng, R. Deng, et al. New efficient MDS array codes for RAID part I: Reed-Solomon-like codes for tolerating three disk failures. *IEEE Trans. on Computers*, 54(9):1071–1080, Sep. 2005.

[17] G. Feng, R. Deng, et al. New efficient MDS array codes for RAID part II: Rabin-like codes for tolerating multiple (≥ 4) disk failures. *IEEE Trans. on Computers*, 54(12):1473–1483, Dec. 2005.

[18] C. Franklin et al. System and method for accomplishing data storage migration between raid levels. US Patent No. 6510491 B1, Jan. 2003.

[19] R. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, 1962.

[20] K. Greenan et al. Flat XOR-based erasure codes in storage systems: Constructions, efficient recovery, and tradeoffs. In *Proc. of the MSST'10*.

[21] J. Hafner. HoVer erasure codes for disk arrays. In *Proc. of the DSN'06*.

[22] J. Hafner. WEAVER codes: Highly fault tolerant erasure codes for storage systems. In *Proc. of the FAST'05*.

[23] J. Hafner et al. Undetected disk errors in RAID arrays. *IBM Journal of Research and Development*, 52(4/5):413–425, July/September 2008.

[24] HP Inc. Configuring Arrays on HP Smart Array Controllers Reference Guide. http://h10032.www1.hp.com/ctg/Manual/c02289065.pdf.

[25] C. Huang et al. Erasure coding in windows azure storage. In *Proc. of the USENIX ATC'12*.

[26] C. Huang et al. Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems. In *Proc. of NCA'07*.

[27] C. Huang and L. Xu. STAR: An efficient coding scheme for correcting triple storage node failures. In *Proc. of the FAST'05*.

[28] I. Iliadis et al. Disk scrubbing versus intra-disk redundancy for high-reliability RAID storage systems. In *Proc. of the SIGMETRICS'08*.

[29] C. Jin et al. P-Code: A new RAID-6 code with optimal properties. In *Proc. of the ICS'09*.

[30] A. Krioukov et al. Parity lost and parity regained. In *Proc. of the FAST'08*.

[31] M. Li et al. Grid codes: Strip-based erasure code with high fault tolerance for storage systems. *ACM Trans. on Stor.*, 4(4):15, Jan. 2009.

[32] M. Luby et al. Practical loss-resilient codes. In *Proc. of the STOC'97*.

[33] C. Lueth. RAID DP: Network appliance implementation of RAID double parity for data protection. Technical Report TR-3298.

[34] N. Brown. Converting RAID5 to RAID6 and other shape changing in md/raid. http://neil.brown.name/blog/20090817000931, August 2009.

[35] N. Brown. Road map for md/raid driver - sort of. http://neil.brown.name/blog/20090129234603, January 2009.

[36] A. Oprea and A. Juels. A Clean-Slate look at disk scrubbing. In *Proc. of the FAST'10*.

[37] J. Paris and D. Long. Using device diversity to protect data against batch-correlated disk failures. In *Proc. of the StorageSS'06*.

[38] D. Patterson et al. A case for Redundant Arrays of Inexpensive Disks (RAID). In *Proc. of the SIGMOD'88*.

[39] E. Pinheiro, W. Weber, and L. Barroso. Failure trends in a large disk drive population. In *Proc. of the FAST'07*.

[40] J. Plank. A new minimum density RAID-6 code with a word size of eight. In *Proc. of the NCA'08*.

[41] J. Plank. The RAID-6 liberation codes. In *Proc. of the FAST'08*.

[42] J. Plank et al. A practical analysis of low-density parity-check erasure codes for wide-area storage applications. In *Proc. of the DSN'04*.

[43] J. Plank et al. SD Codes: Erasure codes designed for how storage systems really fail. In *Proc. of the FAST'13*.

[44] I. Reed and G. Solomon. Polynomial codes over certain finite fields. *J. of the Society for Indus. and Applied Math.*, pages 300–304, 1960.

[45] E. Rozier et al. Evaluating the impact of undetected disk errors in RAID systems. In *Proc. of the DSN'09*.

[46] M. Sathiamoorthy et al. Xoring elephants: Novel erasure codes for big data. In *Proc. of the VLDB'13*.

[47] B. Schroeder et al. Understanding latent sector errors and how to protect against them. In *Proc. of the FAST'10*.

[48] B. Schroeder and G. Gibson. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In *Proc. of the FAST'07*.

[49] Seagate LLC. Barracuda ES Serial ATA Product Manual. http://www.seagate.com/staticfiles/support/disc/manuals/enterprise/Barracuda%20ES/SATA/100424667b.pdf.

[50] Z. Shen and J. Shu. Hv code: An all-around mds code to improve efficiency and reliability of raid-6 systems. In *Proc. of the DSN'14*.

[51] H. Smith. *Data Center Storage: Cost-Effective Strategies, Implementation, and Management*. CRC Press, 2011.

[52] S. Wan et al. Code-m: A non-mds erasure code scheme to support fast recovery from up to two-disk failures in storage systems. In *Proc. of the DSN'10*.

[53] Western Digital Inc. Thermal Reliability: Cool-Running WD Hard Drives Demonstrate Exceptional Reliability in High Duty Cycle Environments. http://www.wdc.com/wdproducts/library/other/2579-001134.pdf.

[54] J. Wilkes et al. The HP AutoRAID hierarchical storage system. *ACM Trans. on Comp. Sys.*, 14(1):108–136, Feb. 1996.

[55] 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*.

[56] 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*.

[57] J. Wylie and R. Swaminathan. Determining fault tolerance of XOR-based erasure codes efficiently. In *Proc. of the DSN'07*.

[58] L. Xiang et al. Optimal recovery of single disk failure in RDP code storage systems. In *Proc. of the SIGMETRICS'10*.

[59] L. Xu and J. Bruck. X-Code: MDS array codes with optimal encoding. *IEEE Trans. on Information Theory*, 45(1):272–276, Jan. 1999.

[60] L. Xu et al. Low-density MDS codes and factors of complete graphs. *IEEE Trans. on Information Theory*, 45(6):1817–1826, Sep. 1999.