Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

An efficient data layout scheme for better I/O balancing in RAID-6 storage systems*

Ping XIE[†1,2], Jian-zhong HUANG[†‡1], Er-wei DAI[1], Qiang CAO[†1], Chang-sheng XIE[†1]

([1]*Wuhan National Laboratory for OptoElectronics, Huazhong University of Science & Technology, Wuhan 430074, China*)

([2]*Computer College, Qinghai Normal University, Xining 810008, China*)

[†]E-mail: xieping@qhnu.edu.cn; hjzh@hust.edu.cn; caoqiang@hust.edu.cn; cs_xie@hust.edu.cn

Received Oct. 23, 2014; Revision accepted Nov. 20, 2014; Crosschecked Apr. 21, 2015

Abstract: Among redundant arrays of independent disks (RAID)-6 codes, maximum distance separable (MDS) based RAID-6 codes are popular because they have the optimal storage efficiency. Although vertical MDS codes exhibit better load balancing compared to horizontal MDS codes in partial stripes, an I/O unbalancing problem still exists in some vertical codes. To address this issue, we propose a novel efficient data layout, uniform P-code (UPC), to support highly balanced I/Os among P-coded disk arrays (i.e., PC). In UPC, the nonuniformly distributed information symbols in each parity chain of P-code are moved along their columns to other rows, thus enabling the parity chain to keep original parity relationships and tolerate double disk failures. The UPC scheme not only achieves optimal storage efficiency, computational complexity, and update complexity, but also supports better I/O balancing in the context of large-scale storage systems. We also conduct a performance study on reconstruction algorithms using an analytical model. Besides extensive theoretical analysis, comparative performance experiments are conducted by replaying real-world workloads under various configurations. Experimental results illustrate that our UPC scheme significantly outperforms the PC scheme in terms of average user response time. In particular, in the case of a 12-disk array, the UPC scheme can improve the access performance of the RAID-6 storage system by 29.9% compared to the PC scheme.

Key words: RAID-6, Data availability, High performance, I/O balancing
doi:10.1631/FITEE.1400362 **Document code:** A **CLC number:** TP311

1 Introduction

Redundant arrays of independent disks (RAID) techniques (Patterson *et al.*, 1988) are widely used in storage systems to achieve large capacity, high performance, and high reliability with acceptable spatial and monetary cost. Evidence shows that the possibility of disk failure occurrence grows along with the increasing scale of storage systems (Schroeder and Gibson, 2007). Researchers pay increasing attention to RAID-6 codes, because RAID-6 coding schemes can concurrently tolerate double disk failures.

Various erasure codes are applied to implement RAID-6 systems, in which a family of codes with a maximum distance separable (MDS) property is popular. The MDS property attains the Singleton bound (Blaum and Roth, 1999) and is of optimal storage efficiency. A RAID-6 storage system with an MDS property consists of $n = k+2$ disk drives, which means that if any two of the n disks fail, their content may be recomputed from the k surviving disks. In other words, the storage system can tolerate any $n-k = 2$ concurrent disk failures. MDS codes suffer from a common disadvantage that k symbols must be read to regenerate any one lost symbol. This

‡ Corresponding author

* Project supported by the National Basic Research Program (973) of China (No. 2011CB302303) and the National High-Tech R&D Program (863) of China (No. 2013AA013203)

ORCID: Ping XIE, http://orcid.org/0000-0001-9122-8534

limitation degrades the reconstruction performance with increasing RAID sizes. To improve the reconstruction performance, many recovery techniques have been designed for MDS-based RAID-6 storage systems (Wang *et al.*, 2010; Xiang *et al.*, 2011; Zhu *et al.*, 2012a).

According to the structure and distribution of parities, RAID-6 can be categorized mainly into horizontal codes (Reed and Solomon, 1960; Blaum *et al.*, 1995; Blaum and Roth, 1999; Corbett *et al.*, 2004) and vertical codes (Xu and Bruck, 1999; Jin *et al.*, 2009). However, most MDS-based RAID-6 storage systems suffer from unbalanced I/Os in a stripe. This performance problem becomes more pronounced for write-intensive applications. Typical horizontal codes have dedicated parity strips in a stripe, which means that corresponding parities need to be updated for any write operation, thereby causing heavier workload on parity strips, each of which corresponds to a unique disk. Rotating information strips and parity strips among all disks were introduced for load balancing (Plank *et al.*, 2009). Rotating stripes is carried out by switching the disk identities for each stripe. However, this needs additional overhead to maintain the mapping. As to the load balancing issue, vertical codes disperse parity symbols across all disks instead of existing dedicated parities as do horizontal codes. Although vertical codes achieve better load balancing relative to horizontal codes in a stripe, they suffer from the I/O unbalancing problem. For example, the P-code feature makes I/Os unbalanced because of the nonuniform arrangement of information symbols in a stripe. There exist some dynamic load balancing approaches designed for disk arrays (Jantz, 1999; Bachmat *et al.*, 2004), but they consume resources to monitor the status of storage devices and thus bring extra overhead to disk arrays.

Unbalanced I/Os adversely affect both data availability and storage reliability. To address this we propose a novel efficient data layout, based on P-code (PC), to support better load balancing for disk arrays, termed 'uniform P-code (UPC)'. We move the nonuniformly distributed information symbols in each parity chain of P-code along their columns onto other rows, while maintaining the original parity relationships and fault-tolerance capability.

The contributions of this paper are summarized as follows:

1. We propose a novel efficient data layout scheme (UPC), which not only exhibits the features of optimal storage efficiency, computational complexity, and update complexity, but also allows I/O load to be spread among disk arrays in a uniform manner.

2. We conduct extensive trace-driven experiments for both UPC and PC schemes under various configurations. The results illustrate that the UPC scheme offers better I/O balancing and higher data availability compared to the PC scheme.

2 Aims and background

2.1 Definitions

Symbol: A symbol is the fundamental unit for building codes. There are two types of symbols, information symbols and parity symbols.

Stripe: A stripe is an independent set of information and parity symbols in an erasure code, which can tolerate the failure of both its information and parity symbols.

Strip: A strip is a set of all the continuous symbols on a disk drive in a stripe.

Parity chain: A parity chain contains the parity symbol and all the symbols used to build it.

An erasure-coded storage system is usually partitioned into fixed-size stripes, each of which is a two-dimensional array with a fixed number of rows and columns. Each column corresponds to a unique disk. Each stripe stores a fixed number of symbols. A symbol refers to a fixed-size block (or chunk) in a practical storage system.

2.2 Load balancing problem in partial stripes within MDS codes

Some MDS coding approaches have been proposed for RAID-6 storage systems to improve the efficiency, performance, and reliability. However, they suffer from the problem of unbalanced I/O in partial stripes. In this section we discuss the unbalanced I/O problem and present the aim of our work.

2.2.1 Unbalanced I/Os in horizontal codes

RAID-5 keeps a highly balanced load based on the parity declustering approach (Holland and Gibson, 1992). Unfortunately, due to the dedicated distribution of parity strips, load balancing is a serious

issue for horizontal codes in a stripe. For example, as shown in Fig. 1, RDP code for a $(p+1)$-disk array is defined in a matrix with $p-1$ rows and $p+1$ columns. The first $p-1$ columns (i.e., information strips) store all information symbols, while the last two columns (i.e., parity strips) hold all parity symbols. Assume $C_{i,j}$ represents a symbol in the ith row and jth column, and there are uniform single writes (USW) to all information symbols in a period. For a read of a single information symbol, just one I/O operation occurs. However, for a write of a single information symbol, there are at least six I/O operations (i.e., one read and one write on the information symbol, two reads and two writes on the corresponding parity symbols), and the total numbers of I/O operations on information strips, row parity strip, and diagonal parity strip are $2(p-1)$, $2(p-1)^2$, and $2(p-2)^2+2(p-1)^2$, respectively. Fig. 1c shows that it is an extremely unbalanced I/O. Meanwhile, Fig. 1d shows that the unbalance impact exhibits an exponential growth trend with the increasing disk array size (i.e., $2p+1/(p-1)-4$). It is deduced that the number of I/O operations in the parity strips is about six to ten times higher than that in the information strips. Therefore, it would sharply decrease the reliability and performance of a storage system. For the load balancing issue of horizontal codes, a rotating stripes (Plank *et al.*, 2009) technique is introduced in a practical storage system.

2.2.2 Unbalanced I/Os in vertical codes

Compared to horizontal codes, vertical codes achieve better load balancing by dispersing parity symbols across all disks/columns (see, X-code and P-code in Figs. 2 and 3 respectively) in a stripe. The difference existing on information symbols distribution in parity chains (i.e., the set of symbols with the same shape) between X-code and P-code is as follows: for X-code, the information symbols in each parity chain are uniformly dispersed across all information rows; however, for P-code, the information symbols in each parity chain are nonuniformly distributed among information rows, thus leading to unbalanced I/Os. For example, Fig. 3 depicts the load balancing problem in P-code—there is a nonuniform information symbol layout in the parity chains; e.g., the parity symbol $C_{1,1}$ is the

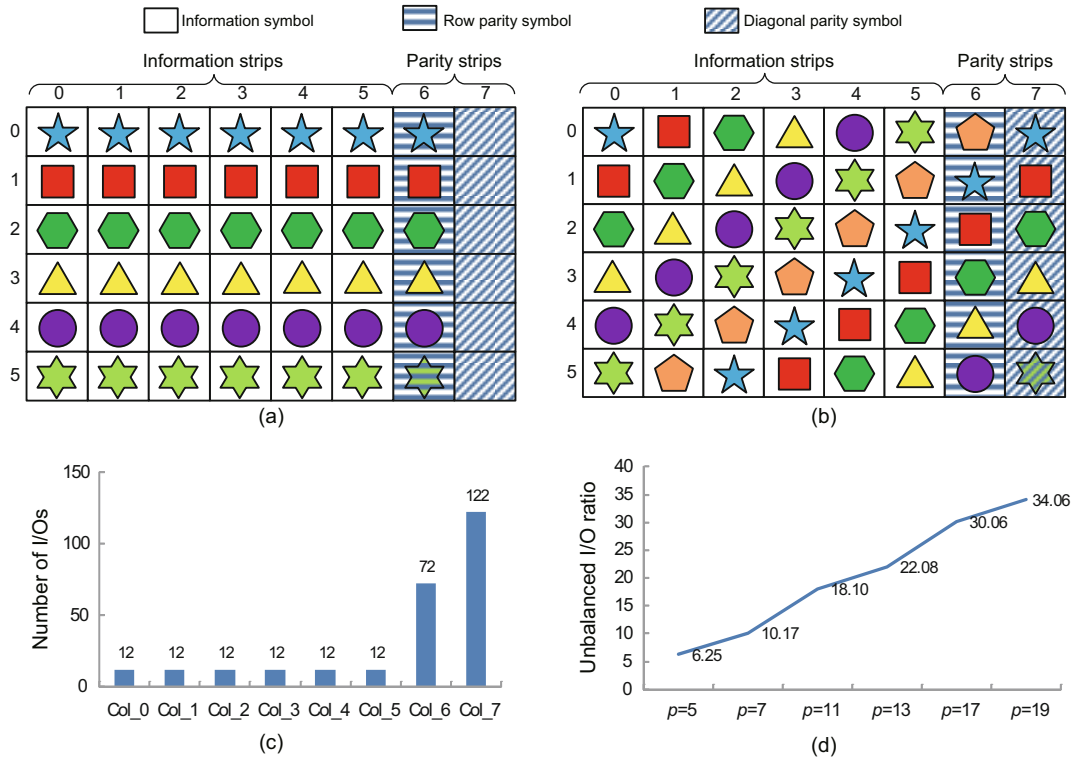

Fig. 1 Load balancing problem in a stripe of row-diagonal parity (RDP): (a) row parity layout in RDP ($p=7$); (b) diagonal parity layout in RDP ($p=7$); (c) I/O distribution in the USW; (d) trend of unbalance in RDP

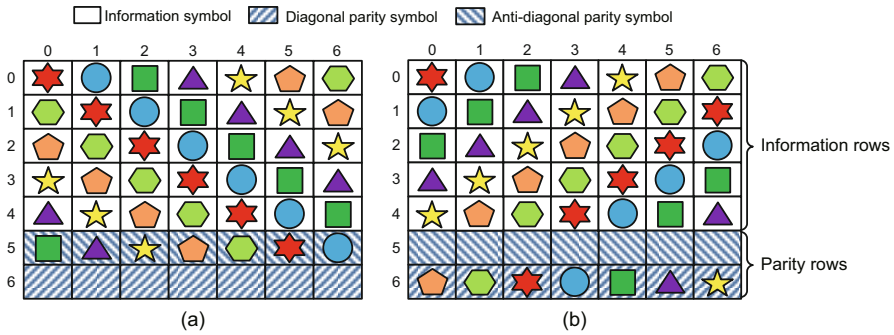

Fig. 2 Diagonal (a) and anti-diagonal (b) parity layouts in X-code with $p = 7$

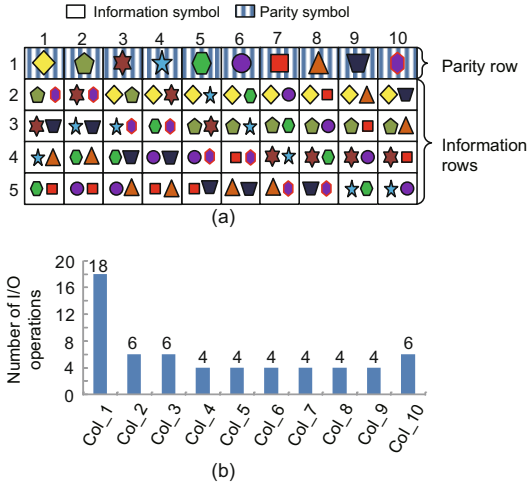

Fig. 3 Load balancing problem in a stripe of P-code with $p = 11$: (a) parity layout of P-code with a 10-disk array; (b) I/O distribution among different columns when 10 writes to information symbols in the second row occur

XOR-sum of eight continuous information symbols $\{C_{2,3}, C_{2,4}, \ldots, C_{2,10}\}$ residing in the second row (Fig. 3a). Furthermore, there exists an unbalanced I/O issue when 10 writes to different columns occur (e.g., as shown in Fig. 3b, 10 continuous information symbols $\{C_{2,1}, C_{2,2}, \ldots, C_{2,10}\}$ are written, and each column writes just one information symbol, and thus it is a uniform write mode for all disks). We can see that column 1 suffers from very high workload (i.e., 18 I/Os) while other columns' workload is very low (i.e., only 4 I/Os in columns 4–9). This unbalanced I/O problem becomes more pronounced for write-intensive operations.

2.3 Recovery approaches of RAID-coded storage systems

In RAID-6 storage systems, the recovery performance of single-disk failure has drawn much attention over the years and a number of approaches have been proposed to address the issue from various perspectives in research community. Two recovery approaches emerge as follows.

2.3.1 Conventional recovery approach

In the conventional recovery approach (Blaum *et al.*, 1995; Corbett *et al.*, 2004), all the symbols of the failed disk are reconstructed only by the set of a kind of parity chain, in which common symbols among parity chains do not exist. Specifically, if a disk fails, all symbols of the failed disk are regenerated only by the set of its row parity chains, or only by the set of its diagonal parity chains. We illustrate the idea of the conventional recovery approach using Fig. 1. For example, if disk 1 fails, one can read the set $\{C_{0,0}, C_{0,2}, C_{0,3}, C_{0,4}, C_{0,5}, C_{0,6}\}$ to regenerate $C_{0,1}$ of disk 1. Thus, the total number of read symbols for recovery disk 1 is 36.

2.3.2 Hybrid recovery approach

In contrast, all symbols of a failed disk in the hybrid recovery approach are reconstructed by the set of different kinds of parity chains (i.e., row parity chains and diagonal parity chains), in which there exist common symbols among parity chains. Each of the common blocks is read once but utilized twice, which means that some parts of data blocks, instead of all, are read from surviving disks in the conventional recovery scheme, such as

$\{C_{0,0}, C_{0,2}, C_{0,3}, C_{0,4}, C_{0,5}, C_{0,6}\}$ to recover $C_{0,1}$
$\{C_{1,0}, C_{1,2}, C_{1,3}, C_{1,4}, C_{1,5}, C_{1,6}\}$ to recover $C_{1,1}$
$\{C_{2,0}, C_{2,2}, C_{2,3}, C_{2,4}, C_{2,5}, C_{2,6}\}$ to recover $C_{2,1}$
$\{C_{4,0}, C_{2,2}, C_{1,3}, C_{0,4}, C_{5,6}, C_{4,7}\}$ to recover $C_{3,1}$

$\{C_{5,0},\ C_{3,2},\ C_{2,3},\ C_{1,4},\ C_{0,5},\ C_{5,7}\}$ to recover $C_{4,1}$

$\{C_{5,0},\ C_{5,2},\ C_{5,3},\ C_{5,4},\ C_{5,5},\ C_{5,6}\}$ to recover $C_{5,1}$

Since the above 36 symbols contain seven common symbols $C_{2,2}$, $C_{1,3}$, $C_{0,4}$, $C_{2,3}$, $C_{1,4}$, $C_{0,5}$, $C_{5,6}$, the total number of read symbols for recovering disk 1 is reduced to 29. Thus, the core idea of hybrid recovery is to find the set of maximum-common symbols to minimize the number of read symbols for recovery. Note that the hybrid recovery approach provides an optimal recovery scheme of single-disk failure for RDP (Xiang *et al.*, 2011) and EVENODD (Wang *et al.*, 2010).

3 Uniform P-code

To overcome the insufficiency in I/O balancing of P-code, we propose an efficient data layout scheme better to support I/O balancing in disk arrays, termed 'uniform P-code'. The uniform P-code takes the uniform information symbols layout into account among the P-code. Compared to P-code, uniform P-code achieves better I/O balancing in large-scale RAID-6 storage systems.

3.1 An architecture of P-code

P-code storage systems can be realized by multiple inexpensive disks. Disks of the same size constitute a disk array storage system. Among these disks, parity symbols have uniform distribution among all columns, and information symbols and parity symbols are mixed in each column.

Erasure codes are defined by parity symbol construction mechanisms to protect data from disk failure. Fig. 4a depicts a specific coding scheme in P-code. P-code is one of the most popular double fault tolerant erasure codes and it achieves both optimal computational complexity and update complexity. P-code for a $(p-1)$-disk array is defined in a matrix with $(p-1)/2$ rows and $p-1$ columns (or what we call a stripe), where p is a prime number greater than 3. The first row in the array holds all parity symbols and the remaining rows hold all information symbols. For each disk/column, the first symbol is assigned to be a parity symbol and the remaining $(p-3)/2$ symbols are assigned to be information symbols. To facilitate analysis, we label the $p-1$ columns sequentially as $c_1, c_2, \cdots, c_{p-1}$, and label the $(p-1)/2$ rows sequentially as $r_1, r_2, \cdots, r_{(p-1)/2}$.

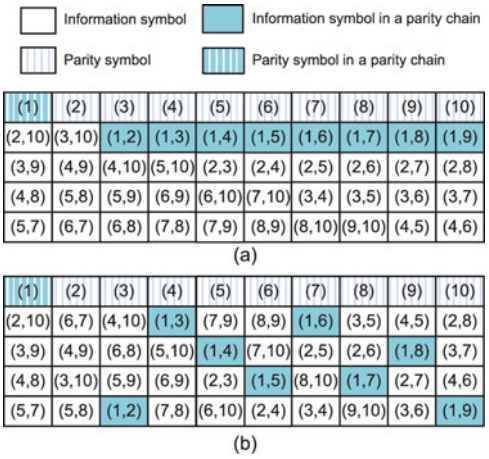

Fig. 4 Data layout comparison between the P-code (a) and uniform P-code (b) with a 10-disk array

Each parity symbol is labeled with an integer equal to the index of the disk in which it resides; that is, the parity symbol on the first disk is labeled with '(1)' and on the last disk/column with '$(p-1)$'. P represents the set of all parity symbols (e.g., $P = \{(1), (2), \cdots, (10)\}$), defined as

$$P = \{(j) | 1 \le j \le p-1\}. \tag{1}$$

Each information symbol is labeled with an unordered two-integer-tuple (x, y), for (x, y) in the set D defined in Eq. (2). It is easy to see that there are $(p-1)\cdot(p-3)/2$ information symbols in set D. We define $p-1$ subsets of D denoted by $d_1, d_2, \cdots, d_{p-1}$, where d_j is defined as in Eq. (2) and represents the set of information symbols in the jth column, e.g., $d_1 = \{(2,10), (3,9), (4,8), (5,7)\}$.

$$\begin{aligned} &D = \{(x,y) | 1 \le x,\ y \le p-1,\ x \ne y,\ x+y \ne p\},\\ &d_j = \{(x,y) | (x,y) \in D,\ j \equiv\ (x+y) \bmod p\}. \end{aligned} \tag{2}$$

Each column set consists of one parity symbol and $(p-3)/2$ information symbols. C is the superset of all column sets c_j, which is defined as in Eq. (3), e.g., $C = \{c_1, c_2, \cdots, c_{10}\}$. c_j represents the set of parity symbol and information symbols in the jth column; e.g., c_1 contains a parity symbol $\{(1)\}$ and four information symbols {(2,10), (3,9), (4,8), (5,7)}.

$$C = \{c_j | 1 \le j \le p-1\},\ c_j = \{(j), d_j\}. \tag{3}$$

From Fig. 4a, each parity chain is the set of symbols labeled as the same integer, e.g., the parity symbol $C_{1,1}$ and information symbols $C_{2,3}$, $C_{2,4}$, $C_{2,5}$, $C_{2,6}$, $C_{2,7}$, $C_{2,8}$, $C_{2,9}$, and $C_{2,10}$ with the integer 1 constituting a parity chain. In other words,

the parity symbol $C_{1,1}$ is the XOR-sum of information symbols $\{C_{2,3}, C_{2,4}, C_{2,5}, C_{2,6}, C_{2,7}, C_{2,8}, C_{2,9}, C_{2,10}\}$ with the integer 1.

3.2 Construction of uniform P-code

P-code achieves the optimal computational complexity and update complexity. So, one method to realize our proposed code is to change the nonuniform arrangement of information symbols among P-code into a uniform layout, while keeping the original calculation relationships in each parity chain unchanged, to ensure that computational complexity and update complexity do not rise.

In I/O balancing vertical X-code, information symbols in each parity chain are uniformly allocated to all information rows. However, in P-code, the information symbols in each parity chain are nonuniformly allocated to information rows. So, we need to move the nonuniformly distributed information symbols in each parity chain of P-code onto other rows to realize our idea. To maintain the original calculation relationships and failure tolerance ability, we just move the information symbols along their columns onto other rows, to ensure that all information symbols in each parity chain are uniformly dispersed over all information rows instead of a few information rows. After these moves, the new structure is as shown in Fig. 4b.

The specific constructing process of uniform P-code is illustrated in Algorithm 1. Fig. 4 depicts the data layout comparison between the P-code and uniform P-code with a 10-disk array. From the data layout of P-code described in Fig. 4a, it is observed that all information symbols $C_{2,3}$, $C_{2,4}$, $C_{2,5}$, $C_{2,6}$, $C_{2,7}$, $C_{2,8}$, $C_{2,9}$, $C_{2,10}$ produced the parity symbol $C_{1,1}$ and resided in row r_2, which means nonuniform information symbols layout in the vertical direction of the disk array. However, the data layout of uniform P-code described in Fig. 4b exhibits a uniform arrangement; e.g., all information symbols producing the parity symbol $C_{1,1}$ are uniformly allocated to all information rows. In other words, each information row among r_2, r_3, r_4, and r_5 holds two information symbols from any parity chain. Obviously, the information symbol layout of uniform P-code is a uniform distribution in the vertical direction of the disk array.

In these movements, we do not break the basic calculation relationships producing each parity symbol in P-code, keeping the information symbols in the last $(p-3)/2$ rows with an updating complexity of two, while the uniform P-code achieves better I/O balancing compared to P-code.

Algorithm 1 Construction of uniform P-code

Input: Information symbols (x, y) in the PC, $m = (p-1)/2$, and $n = p-1$
Output: Information symbols (x', y') in the UPC
Start the construction of UPC:
for $j = 1$ **to** n **do**
  $c[j] = j$;
end for
for $r[1] = 2$ **to** m **do**
  for $r[2] = 2$ **to** m **do**
    for $r[3] = 2$ **to** m **do**
      ⋮
      for $r[n] = 2$ **to** m **do**
        (1) $N[i] = 0$ $(1 \le i \le n)$; flag $= 0$;
        (2) $x = a[r[j]][c[j]]$, $y = b[r[j]][c[j]]$;
            $(x, y) \to N[i] + +$;
        (3) **if** $N[i] = 2$
              flag $+ +$;
            end if
        (4) **if** flag $= n$
              $(x', y') \leftarrow (x, y)$; $(x, y) \leftarrow (0, 0)$;
            end if
      end for
      ⋮
    end for
  end for
end for

4 Performance evaluation

4.1 Experimental setup

We conduct our performance evaluation upon a high-performance storage server, where all disks are organized in a form of RAID-6. The server's hardware configuration is with two Intel Xeon E7540@2.00 GHz (six cores) CPUs, 16 GB DDR3 main memory. All disks are Seagate ST9300605SS SAS-II, and they are connected by a MegaRAID SAS 1078 controller with 512 MB dedicated cache. The operating system is Ubuntu 10.04 LTS X86-64 with the Linux Kernel 2.6.32.

4.2 Evaluation methodology

To measure the average user response time of the underlying storage system under our UPC scheme

and the PC scheme, we adopt a kind of open-loop measuring approach by implementing and executing an application-level trace re-player.

In a RAID-6 storage system, basic I/O operations are read, write, and update. For a read request, it is sent to the designated information symbol according to the size of I/O request in the trace file, which does not cause any additional overhead; in a write request, however, the written data is stored to the destination location, and at least two corresponding parity symbols are updated (i.e., updating I/O).

Generally speaking, there are two models for trace replay: open-loop and closed-loop. The former has the potential to measure the user response time since the I/O arrival rate is independent of the underlying system, whereas the latter is aimed to test the service capability of the underlying system, where the I/O arrival rate is dominated by the processing speed of the underlying system. In this paper, we use an open-loop model to evaluate the performance of the storage system using the UPC scheme, where all traces are re-played according to their time stamps recorded in the trace file.

Here, we choose SPC-Financial1 as the user workload trace of write ratio 76.8% (http://www.storageperformance.org/home). The write-intensity Financial1 trace is collected from OLTP applications running at a large financial institution, which is widely used in the performance evaluation of RAID storage systems (Wu *et al.*, 2009). In our experiments, we conduct extensive comparative experiments under different RAID sizes, workload intensities, and stripe unit sizes.

The Financial1 trace has a limited footprint (e.g., the max-offset is 1.35 GB), meaning that the user I/O requests are congregated on a small part of the RAID space (e.g., the capacity of Seagate ST9300605SS is 300 GB). Just as mentioned in Wu *et al.* (2009), it is appropriate to scale up the address coverage of the I/O requests by multiplying the address of each request with an appropriate factor without changing the size of each request. Therefore, we also adopt the trace scaling method to replay the Financial1 trace in our tests.

4.3 Trace-driven evaluations

The performance of RAID is likely influenced by several important factors, including the RAID size, workload intensities (i.e., IOPS), and stripe unit size. We conduct a wide range of experiments to evaluate user response time of the RAID-6 storage system using both PC and UPC schemes, as well as using both UPC and X-code schemes, under different configurations (i.e., RAID sizes, IOPSs, and stripe unit sizes), with replaying real-world workload traces.

4.3.1 RAID sizes

To examine the advantage of the UPC scheme over the PC scheme, we implement our trace-driven experiments on the two data layout schemes of different p values (e.g., $p = 7$, 11, and 13), where p must be a prime number defined by the PC coding scheme, with a stripe unit size of 64 KB and IOPS of 140. Fig. 5 shows the experimental results for PC and UPC schemes.

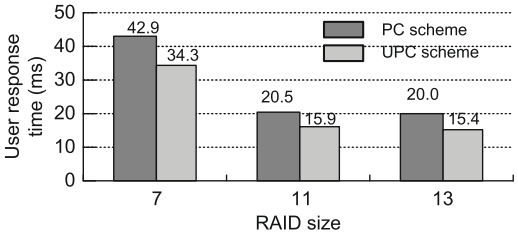

Fig. 5 Average user response time under different RAID sizes

From Fig. 5, we observe that the UPC scheme consistently outperforms the PC scheme in terms of average user response time, which means our UPC scheme achieves better I/O balancing than the PC scheme. The reason is that in the UPC case, the information symbols of a parity chain are uniform across all information rows in a stripe, while in the PC case, the information symbols of a parity chain are spread across a few information rows in a stripe, resulting in a nonuniform data distribution. Take 'vast concurrent I/Os accessing an information row in a stripe' as an example. For the disk array using the PC scheme, vast updating I/Os occur on a few disks rather than all disks, and thus some disks suffer from high load while the remaining disks have low load. Such unbalanced I/Os in the RAID system would lengthen the I/O access latency; for the RAID system using the UPC scheme, the updating I/O requests are uniform among all disks. The experimental results validate our analysis—the access performance of the RAID-6 storage system using the UPC scheme outperforms that using the PC scheme.

The improvement factors are 25.1%, 28.9%, and 29.9% when p is 7, 11, and 13, respectively.

4.3.2 Workload intensities (IOPS)

To examine the two data layout schemes' sensitivity to workload density, we change the IOPS of replayed trace by multiplying the timestamp of the Financial1 trace with a factor. Fig. 6 shows the average user response time using the two data layout schemes under different IOPSs (116, 124, 133, 140, and 155), when p equals 7 and the stripe unit size is 64 KB.

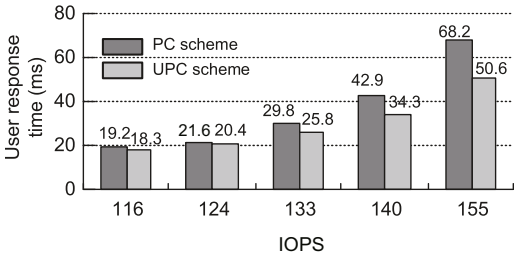

Fig. 6 Average user response time under different IOPSs ($p = 7$)

From Fig. 6, it is observed that the average user response time of both UPC and PC schemes grows with the increasing workload density, and the two schemes have similar sensitivity to the workload density. There are two kinds of overhead during the data access in the context of the RAID-6 storage system: I/O overhead caused by disk read/write and CPU overhead incurred by parity computation. CPU overhead is the smaller. With the growth of workload density, the I/O request queue will be lengthened, and thus the access latency increases. If the level of unbalanced I/O among all disks becomes more serious, the latency will increase accordingly. The two data layout schemes represent two kinds of I/O balancing scenarios. The results show that the UPC scheme has shorter average response time regardless of different IOPSs (e.g., UPC has an improvement factor of 25.1% over PC in the case of IOPS=140), which means that the UPC scheme achieves better I/O balancing than the PC scheme.

4.3.3 Stripe unit sizes

To assess the impact of the stripe unit size, we carry out experiments on a 6-disk (i.e., p=7) RAID-6 storage system under different stripe unit sizes (32, 64, 128, and 256 KB) using the UPC scheme and the PC scheme, with IOPS=140. As shown in Fig. 7, our UPC scheme consistently outperforms the PC scheme in terms of average user response time for all the stripe unit sizes. Furthermore, we observe that the RAID-6 storage system has better access performance when the stripe unit size is larger for each scheme. The reason lies in the fact that a larger stripe unit helps to improve access sequentiality.

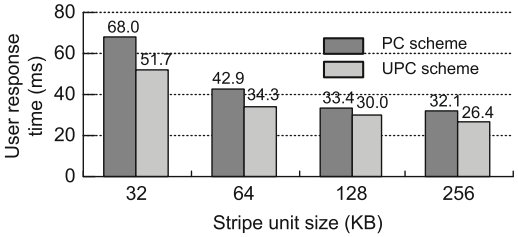

Fig. 7 Average user response time under different stripe unit sizes ($p = 7$)

4.3.4 Comparison of UPC and X-code

To evaluate the I/O performance of UPC and X-code, we implement a set of comparative experiments under different stripe unit sizes (32, 64, 128, and 256 KB) on storage systems using the two schemes, with IOPS=133 and $p = 7$. Fig. 8 illustrates the results of experiments.

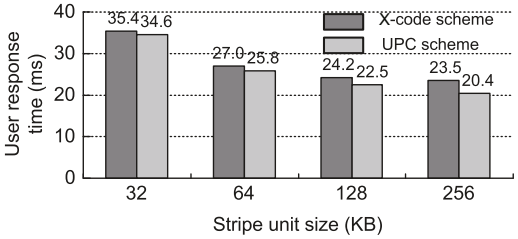

Fig. 8 The performance evaluation of UPC and X-code ($p = 7$)

From Fig. 8, we observe that regardless of different stripe unit sizes, the UPC and X-code schemes exhibit similar I/O performance. The reason lies in the fact that there exists a uniform distribution of information symbols for the UPC and X-code schemes.

4.4 Recovery performance of UPC

Compared to P-code, our UPC scheme moves the nonuniform information symbols only along their columns. That is, UPC has the same construction mechanism of erasure codes as the P-code scheme,

and thus the recovery process of UPC is equivalent to that of P-code.

We consider a UPC-based disk array system with $p>2$. According to the UPC data layout, the storage system has $p-1$ disks, denoted by D_1, D_2, ..., D_{p-1}. When a disk (e.g., D_f, $1 \le f \le p-1$) fails, it is required to reconstruct all lost symbols of the failed disk D_f. We assume that the reconstruction operation needs to read x_i symbols from disk D_i ($i \neq f$) in a stripe.

4.4.1 Recovery scheme

When a single-disk failure occurs in UPC-coded storage systems, $(p-1)/2$ symbols (i.e., a parity symbol and $(p-3)/2$ information symbols) are lost in the failed disk D_f. According to construction mechanisms of parity chains in the UPC code, a parity chain consists of a parity symbol and $p-3$ information symbols; each parity symbol exactly belongs to a parity chain; each information symbol is involved in the generation of two parity chains; and each lost symbol is regenerated by the remaining symbols in its parity chain. Thus, it is deduced that there are $L = 2^{(p-3)/2}$ feasible recovery schemes for the case of single-disk failure. Each recovery scheme corresponds to a reconstruction data distribution, which is featured with an individual number of data blocks retrieved from surviving disks to recover the failed disk.

4.4.2 Reconstruction data distribution

Suppose that the jth recovery scheme is determined. We can achieve its reconstruction data distribution $\{\{x_{i,j}\}_{1\le i\le p-1, i\neq f}\}_j$ associated with the jth recovery scheme.

4.4.3 Recovery performance evaluation

The performance of single-disk recovery is dictated by the total number of data blocks retrieved from surviving disks, i.e., the total number of block read requests. To improve recovery performance, it is a trend to minimize the total number of read blocks from surviving disks. We formulate the recovery performance via an optimization model as follows:

$$\min \sum_{j\in L} \{x_{i,j}\}_{1\le i\le p-1, i\neq f}. \tag{4}$$

5 Related work

Various erasure codes have been developed to protect data from disk failures in large-scale storage systems, for example, Reed-Solomon codes, LDPC codes, and parity array codes. Compared to replication, erasure-coded storage systems have better storage efficiency. If an erasure code attains the Singleton bound, we call it an MDS code, which attains the optimal storage efficiency. In theory, Reed-Solomon codes (Reed and Solomon, 1960) can offer any fault-tolerant storage system, which makes them popular in massive distributed storage systems, and they gain MDS property. Low-density parity check (LDPC) codes are constructed using bipartite graphs, and can be encoded and decoded by using simple XOR operations. LDPC codes are featured with irregular structure, which makes them unpopular in production storage systems.

Compared with the above two types of erasure codes, parity array codes have regular structure and depend on simple XOR operations in encoding and decoding, which makes them more popular in RAID storage systems. In recent years, parity array codes have been widely developed to satisfy the reliability requirements of storage systems. Many RAID-6 codes have been proposed and widely used in production storage systems, for example, RDP (Corbett *et al.*, 2004), P-code (Jin *et al.*, 2009), EVENODD (Blaum *et al.*, 1995), X-code (Xu and Bruck, 1999), and Blaum-Roth (Blaum and Roth, 1999). All the above codes are MDS codes. Non-MDS codes are featured with better reconstruction performance over MDS codes, for example, Pyramid code (Huang *et al.*, 2007), WEAVER (Hafner, 2005), V^2-code (Xie *et al.*, 2015), Flat XOR-code (Greenan *et al.*, 2010), and Code-M (Wan *et al.*, 2010). In this paper, we focus on parity array codes for RAID-6 storage systems, which can be divided into horizontal codes and vertical codes.

5.1 Horizontal codes

EVENODD and RDP are two typical horizontal codes in RAID-6 storage systems, consisting of information strip and parity strip in a stripe. Information strip and parity strip hold original data and parity data, respectively. So, the existence of horizontal code dedicated parity strip in a stripe, and any write operation to information strips will lead

to parity updating. For the write bottled problem, a parity rotating technique has been developed to solve unbalanced I/Os for storage systems. EVENODD codes are usually regarded as a matrix with $p-1$ rows and $p+2$ columns. Similarly, RDP codes are defined in a matrix with $p-1$ rows and $p+1$ columns.

5.2 Vertical codes

In vertical codes, information symbols and parity symbols are mixed on each disk/column, instead of separately stored on different disks. X-code and P-code are two typical vertical MDS codes for RAID-6 storage systems, and achieve optimal computational complexity and update complexity. Parity symbols of both X-code and P-code are evenly dispersed over all disks rather than in dedicated parity disks. Such a layout makes them achieve better load balancing compared to horizontal codes.

5.3 Load balancing in disk arrays

Load balancing is an important issue in parallel and distributed storage systems (Zomaya and Teh, 2001), and there are many approaches aiming to achieve the load balancing in disk arrays. For example, Holland and Gibson (1992) showed that parity declustering is an effective way to keep load balancing in RAID-5. Ganger *et al.* (1993) found that disk striping is a better method and could significantly improve load balance with reduced complexity for various applications. Scheuermann *et al.* (1998) proposed a data partitioning method to optimize disk striping and to achieve load balance using proper file allocation and dynamic access redistributions. Additionally, a patent from industry (Jantz, 1999) solves the load balancing problem in a disk array using a host-based I/O schedule strategy.

5.4 Recovery schemes of RAID codes

Recovery performance is a very important metric for RAID-coded storage systems. For existing RAID-coded storage systems, the single-disk failure recovery has drawn much attention over the years and a number of approaches have been proposed to address the issue from various perspectives. For example, Wang *et al.* (2010) and Xiang *et al.* (2011) proposed the optimal single-disk failure recovery scheme for EVENODD and RDP, respectively. For any erasure code, Khan *et al.* (2012) proposed an exhaustive approach for single-disk failure recovery, while Zhu *et al.* (2012a) proposed a replacement recovery algorithm to speed up the search process. For the heterogeneous distributed environment, Zhu *et al.* (2012b) proposed a cost-based recovery scheme for single-disk failure. In addition, various non-MDS codes have been proposed to improve reconstruction performance, such as WEAVER (Hafner, 2005), Pyramid (Huang *et al.*, 2007), and Code-M (Wan *et al.*, 2010).

6 Conclusions and future work

We addressed the I/O balancing issue in vertical RAID-6 codes in this paper. In particular, we proposed a novel I/O balancing scheme (i.e., UPC) for the P-code. In UPC, the distribution of nonuniform information symbols in each parity chain of P-code is changed by moving them along their columns onto other rows, while keeping the original parity relationships and fault-tolerance capability. The UPC scheme enables disk I/Os to be distributed in a balanced way in the context of large-scale RAID-6 storage systems. On the one hand, extensive theoretical analysis has been presented to validate the effectiveness of the UPC scheme. That is, the UPC scheme exhibits significant benefits in terms of I/O balancing. On the other hand, comparative performance evaluations have been conducted by replaying real-world workloads for both UPC and PC schemes under various configurations. The experimental results showed that the UPC scheme outperforms the PC scheme in terms of user average response time; e.g., the UPC improved the access performance of the PC by 25.1%, 28.9%, and 29.9% for p=7, 11, and 13 respectively, under IOPS of 140 and stripe unit size of 64 KB.

The design of the UPC scheme was based on the assumption that the disks in the RAID-6 storage system are homogeneous. However, it is normal that the disks in RAID-6 are not homogeneous. In the future, we plan to evaluate the UPC scheme within a real-world heterogeneous RAID-6 storage system.

References

Bachmat, E., Ofek, Y., Zakai, A., *et al.*, 2004. Load Balancing on Disk Array Storage Device. US Patent 6711649.

Blaum, M., Roth, R.M., 1999. On lowest density MDS codes. *IEEE Trans. Inform. Theory*, **45**(1):46-59. [doi:10.1109/18.746771]

Blaum, M., Brady, J., Bruck, J., *et al.*, 1995. EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures. *IEEE Trans. Comput.*, **44**(2):192-202. [doi:10.1109/12.364531]

Corbett, P., English, B., Goel, A., *et al.*, 2004. Row-diagonal parity for double disk failure correction. Proc. 3rd USENIX Conf. on File and Storage Technologies, p.1-14.

Ganger, G.R., Worthington, B.L., Hou, R.Y., *et al.*, 1993. Disk subsystem load balancing: disk striping vs. conventional data placement. Proc. 26th Hawaii Int. Conf. on System Sciences, p.40-49. [doi:10.1109/HICSS.1993.270759]

Greenan, K.M., Li, X.Z., Wylie, J.J., 2010. Flat XOR-based erasure codes in storage systems: constructions, efficient recovery, and tradeoffs. Proc. IEEE 26th Symp. on Mass Storage Systems and Technologies, p.1-14. [doi:10.1109/MSST.2010.5496983]

Hafner, J.L., 2005. WEAVER codes: highly fault tolerant erasure codes for storage systems. Proc. 4th USENIX Conf. on File and Storage Technologies, p.16.

Holland, M., Gibson, G.A., 1992. Parity declustering for continuous operation in redundant disk arrays. Proc. 5th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, p.23-35. [doi:10.1145/143365.143383]

Huang, C., Chen, M., Li, J., 2007. Pyramid codes: flexible schemes to trade space for access efficiency in reliable data storage systems. Proc. 6th IEEE Int. Symp. on Network Computing and Applications, p.79-86. [doi:10.1109/NCA.2007.37]

Jantz, R.M., 1999. Method for Host-Based I/O Workload Balancing on Redundant Array Controllers. US Patent 5937428.

Jin, C., Jiang, H., Feng, D., *et al.*, 2009. P-code: a new RAID-6 code with optimal properties. Proc. 23rd Int. Conf. on Supercomputing, p.360-369. [doi:10.1145/1542275.1542326]

Khan, O., Burns, R.C., Plank, J.S., *et al.*, 2012. Rethinking erasure codes for cloud file systems: minimizing I/O for recovery and degraded reads. Proc. 11th USENIX Conf. on File and Storage Technologies, p.20.

Patterson, D.A., Gibson, G., Katz, R.H., 1988. A case for redundant arrays of inexpensive disks (RAID). Proc. ACM SIGMOD Int. Conf. on Management of Data, p.109-116. [doi:10.1145/50202.50214]

Plank, J.S., Luo, J., Schuman, C.D., *et al.*, 2009. A performance evaluation and examination of open-source erasure coding libraries for storage. Proc. 8th USENIX Conf. on File and Storage Technologies, p.253-265.

Reed, I.S., Solomon, G., 1960. Polynomial codes over certain finite fields. *J. Soc. Ind. Appl. Math.*, **8**(2):300-304. [doi:10.1137/0108018]

Scheuermann, P., Weikum, G., Zabback, P., 1998. Data partitioning and load balancing in parallel disk systems. *VLDB J.*, **7**(1):48-66. [doi:10.1007/s007780050053]

Schroeder, B., Gibson, G.A., 2007. Disk failures in the real world: what does an MTTF of 1,000,000 hours mean to you? Proc. 6th USENIX Conf. on File and Storage Technologies, p.1-16.

Wan, S., Cao, Q., Xie, C.S., *et al.*, 2010. Code-M: a non-MDS erasure code scheme to support fast recovery from up to two-disk failures in storage systems. Proc. IEEE/IFIP Int. Conf. on Dependable Systems and Networks, p.51-60. [doi:10.1109/DSN.2010.5545009]

Wang, Z.Y., Dimakis, A.G., Bruck, J., 2010. Rebuilding for array codes in distributed storage systems. Proc. IEEE GLOBECOM Workshops, p.1905-1909. [doi:10.1109/GLOCOMW.2010.5700274]

Wu, S., Jiang, H., Feng, D., *et al.*, 2009. WorkOut: I/O workload outsourcing for boosting RAID reconstruction performance. Proc. 8th USENIX Conf. on File and Storage Technologies, p.239-252.

Xiang, L.P., Xu, Y.L., Lui, J., *et al.*, 2011. A hybrid approach to failed disk recovery using RAID-6 codes: algorithms and performance evaluation. *ACM Trans. Stor.*, **7**(3), Article 11. [doi:10.1145/2027066.2027071]

Xie, P., Huang, J.Z., Cao, Q., *et al.*, 2015. A new non-MDS RAID-6 code to support fast reconstruction and balanced I/Os. *Comput. J.*, in press. [doi:10.1093/comjnl/bxv006]

Xu, L.H., Bruck, J., 1999. X-code: MDS array codes with optimal encoding. *IEEE Trans. Inform. Theory*, **45**(1):272-276. [doi:10.1109/18.746809]

Zhu, Y.F., Lee, P.P.C., Hu, Y.C., *et al.*, 2012a. On the speedup of single-disk failure recovery in XOR-coded storage systems: theory and practice. Proc. IEEE 28th Symp. on Mass Storage Systems and Technologies, p.1-12. [doi:10.1109/MSST.2012.6232371]

Zhu, Y.F., Lee, P.P.C., Xiang, L.P., *et al.*, 2012b. A cost-based heterogeneous recovery scheme for distributed storage systems with RAID-6 codes. Proc. 42nd Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks, p.1-12. [doi:10.1109/DSN.2012.6263934]

Zomaya, A.Y., Teh, Y.H., 2001. Observations on using genetic algorithms for dynamic load-balancing. *IEEE Trans. Parall. Distr. Syst.*, **12**(9):899-911. [doi:10.1109/71.954620]