Hindawi
Mobile Information Systems
Volume 2017, Article ID 1360413, 12 pages
https://doi.org/10.1155/2017/1360413

*Research Article*

# RAID-6Plus: A Comprised Methodology for Extending RAID-6 Codes

**Ming-Zhu Deng, Nong Xiao, Song-Ping Yu, Fang Liu, Lingyu Zhu, and Zhi-Guang Chen**

*State Key Laboratory of High Performance Computing, College of Computer, National University of Defense Technology, Changsha 410073, China*

Correspondence should be addressed to Ming-Zhu Deng; dk_nudt@126.com

Received 23 September 2016; Revised 26 December 2016; Accepted 10 January 2017; Published 23 February 2017

Academic Editor: Laurence T. Yang



Existing RAID-6 code extensions assume that failures are independent and instantaneous, overlooking the underlying mechanism of multifailure occurrences. Also, the effect of reconstruction window is ignored. Additionally, these coding extensions have not been adapted to occurrence patterns of failure in real-world applications. As a result, the third parity drive is set to handle the triple-failure scenario; however, the lower level failure situations have been left unattended. Therefore, a new methodology of extending RAID-6 codes named RAID-6Plus with better compromise has been studied in this paper. RAID-6Plus (Deng et al., 2015) employs short combinations which can greatly reuse overlapped elements during reconstruction to remake the third parity drive. A sample extension code called RDP+ is given based on RDP. Moreover, we extended the study to present another extension example called $X$-code+ which has better update penalty and load balance. The analysis shows that RAID-6Plus is a balanced tradeoff of reliability, performance, and practicality. For instance, RDP+ could achieve speedups as high as 33.4% in comparison to the RTP with conventional rebuild, 11.9% in comparison to RTP with the optimal rebuild, 47.7% in comparison to STAR with conventional rebuild, and 26.2% for a single failure rebuild.

## 1. Introduction

In modern data centers, RAID-6 credited for performance and reliability are among the most popular configurations to be deployed. However, more devices, larger disks, unimproved reliability, increased bit errors, and less-reliable hardware all expose modern storage systems to higher vulnerability, demanding RAID-6 evolvement with higher and more flexible reliability care [1]. Thus, the extension of the existing RAID-6 coding scheme is worth attention.

In fact, there are various RAID-6 coding algorithms available to be extended for higher failure tolerance. For example, RS codes [2] can be applied with various parameters while EVENODD [3] and RDP [4] are XOR-based for faster computation [5]. Regarding higher reliability, many attempts have been made by extending the existing RAID-6 codes. Blaum et al. generalized EVENODD for arbitrary failure scenarios [6]. Huang and Xu proposed the STAR code [7] to protect a storage array from triple failures. In fact, the STAR code is another extension of EVENODD. Goel and Corbett extend RDP to RTP [8] in 2012 to provide a faster coding algorithm for triple failures. Similarly, the Triple-STAR code [9] is extended from Rotary-code [10] by adding a more diagonal parity column to tolerate triple failures.

Codes along this direction can provide satisfactory solutions to triple failures, but the following problems still remain unsolved especially the lack of flexibility for enabling higher reliability:

(i) Existing codes assume that failures are independent, instantaneous, and occurrences of failures conform to the exponential distribution [11, 12]. This ideal assumption does not apply to the fault pattern of modern storage systems [13]. Furthermore, these codes are not designed to support multifailure degradations; such degradations aim to convert a higher level multifailure into separate low-level multifailures or single failures with a shorter reconstruction window.

(ii) Existing codes largely ignore the pattern of failure occurrences in practice. For example, 99.75% of recoveries are due to single disk failures [14], while

triple whole-disk failures are rare. However, the third parity drive in RTP is set to handle the triple-failure scenario only with single failure rebuild unattended. The third parity drive is then almost wasted.

(iii) Existing codes focus on whole-device level failures while mixed-fault modes are more common in practice [15]. For example, when a fault consisting of two erasures and a sector error occurs, all of the three parity drives must be used. This directly overkills the effects of the third parity drive.

(iv) As throughput is dwarfed by capacity [16], the reconstruction windows have grown exponentially from minutes to hours or even days in practice. This leads to a severe decrease of system performance and poor user experience.

Therefore a new methodology for RAID-6 code extension is in pressing need to support multifailure degradations and a smaller reconstruction window to deliver data reliability in a more flexible manner. In this paper, we propose a compromised code extending methodology with a shorter reconstruction window, named RAID-6Plus [14] to provide higher reliability at the expense of three parity drives.

Existing coding extensions provide absolute reliability for triple failures via full combinations. In contrast, RAID-6Plus employs short combinations which can effectively reuse overlapped elements during reconstruction to remake the third parity drive. This design shortens the reconstruction window of single failures by minimizing the total number of data reads. The possibility of multifailure overlapping in the reconstruction window is therefore significantly diminished. Such features provide RAID-6Plus with (1) a better system performance compared to the RTP and STAR codes and (2) an enhanced reliability compared to the RAID-6. An example extension code called RDP+ is given based on RDP (repetition). Moreover, we expand the study to present another extension example called $X$-code+ for the sake of vertical codes.

The analysis shows that RAID-6Plus is a balanced compromise among reliability, performance, and practicality. For example, RDP+ achieved least update penalty and far outperforms RTP and STAR both under their optimal reconstruction on encoding and decoding.

The main contributions of this study are as follows:

(i) We developed a new RAID-6 code extending methodology with shorter reconstruction window and lower risk of multifailure with no additional cost incurred compared to RTP and STAR, which provide a balanced tradeoff of flexible reliability and better system performance. This code can be applied to most XOR-based coding schemes and is orthogonal with some previous work on reconstruction speedup [17–20]. They can also be integrated together to further shorten reconstruction window.

(ii) An example extension code called RDP+ is presented based on RDP in terms of encoding and single failure reconstruction improvement.

(iii) Another extension example is given as $X$-code+ to apply RAID-6Plus with regard to vertical coding schemes. $X$-code+ shows good performance on single failure rebuild and load balance.

(iv) A new metric called $Q$-metric is proposed to validate and evaluate the presented extending methodology. $Q$-metric denotes induced benefit per cost. The higher the $Q$ value, the more competitive and useful the method. Furthermore, the $Q$-metric is intended to measure the performance improvement per cost, and it is an indicator of the correlation between gains and overheads.

In comparison to the previous work [14, 21], this paper not only provides more details of RDP+ and $X$-code+, but also reveals the generalized methodology for extending other codes. Also further identification and examination of the problem in the perspective of update penalty and load balance are presented. Additionally, a comprehensive evaluation including both two sample codes together is given. Further, a new metric called $Q$-metric is proposed to validate and evaluate the proposed extending methodology.

In the remainder of this paper, Section 2 introduces the background and motivation for RAID-6Plus. Section 3 explains the design of RAID-6Plus in detail and presents RDP+ and $X$-code+. Then, the performance is evaluated in Section 4 and related work is compared in Section 5. Finally, all of the findings of this paper are summarized and concluded in Section 6.

## 2. Backgrounds and Motivation

In this section, we describe the different failure modes and recovery methods in RAID systems and introduce the concept of multifailure degradation in reconstruction window. This motivates the need to design RAID-6Plus, which exploits multifailure degradation mechanism.

*2.1. Failure Modes.* Many researches on the massive disks have all shown that (1) mainstream disk drives have device failures or whole-disk failure and sector failures [22]. All these failures directly cause data unavailability. (2) Sector errors are not rare and increase with time [1]. (3) Despite infant mortality, multiple disks tend to fail at a similar age, indicating not only does single failure happen at the device level, but also multiple devices may fail almost simultaneously, calling for higher reliability care [1]. (4) In terms of the correlation between whole-disk failure and sector errors, [1] asserts whole-disk failure can be viewed as the consequence of accumulated sector error and uses the number of reallocated sectors to characterize the probability of whole-disk failure. (5) Further, the longer a functioning device endures, the higher the probability of device failure could be. In other words, other failures could happen in the ongoing process of failure recovery, aggravating system reliability and making it much more vulnerable [23]. In short, the single failure is of vital importance to reduce the window of system vulnerability than RAID-6.

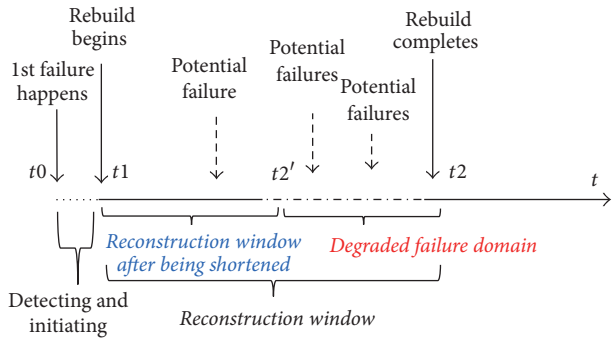


Figure 1: Successive failure happening.

However, real-world statistics show that, among all device failures, single failure accounts for the majority (99.75%), and double failures are not eligible (roughly 8%) and should be taken care of [24]. Meanwhile, often the cases are single failure coupled with other sector failures rather than multi-whole-disk failures [22]. Though there has not been any investigation published on massive SSD, SSD's failure modes must be similar while they differ in some of its vulnerability features, like its inborn limited endurance issues and wearing over time [25].

*2.2. Multifailure Degradation.* In fact, multiple failures happening at almost exactly the same time is hardly witnessed in real-world systems, which are quite different cases from ideal models for simplification of research. Strictly speaking, multiple failures happen in a successive manner, as shown in Figure 1.

When the first whole-disk failure takes place, the second failure or more will happen mainly in the reconstruction window for the first failure, thus making it a double-failure or multifailure.

It is clear that longer reconstruction window creates more space for multifailures to happen. If we purposely shorten the reconstruction window, the risk of multifailure could be degraded to a lower level, as shown in Figure 1. With proper shortening, a higher level of multifailure is degraded into a lower level of multifailure, alleviating threats to system reliability boundary and data loss [26].

Unfortunately, current codes with higher fault tolerance seldom make use of multifailure degradation mechanism and mainly concentrate on furthering reliability boundary (absolute reliability). They focus on providing inflexible reliability, unable to deliver flexible reliability regarding failure happening mode and probability.

For example, as an extension of the EVENODD code and a modification of the generalized triple-erasure-correcting EVENODD code, STAR code has the same recovery package when single failure happens, showing its inability to contract the reconstruction window [27].

*2.3. Motivation.* Above all, existing code extensions with higher fault tolerance, like RTP and STAR codes are originally designed for triple whole-device failures. Unfortunately, statistics have shown the probability of single failure overwhelmingly accounts for most while triple whole-device failures are relatively rare, thus making the current RAID-7 system with triple parity drives wasteful. Additionally, mixed-fault modes exhibited in modern storage systems overkill the solution of those codes with higher reliability boundaries. In short, the current RAID-7 system with triple parity coding is unpractical and needs to deliver more flexible reliability [14]. Further, the reconstruction window is exponentially increasing with device capacity, thus worsening user experience and leaving larger space of system vulnerability and data loss. More notably, current coding schemes for triple-failure are unable to shorten reconstruction window.

Therefore, all these factors above motivates extending $X$-code another way to shorten reconstruction window to provide flexible reliability and make it more practical.

## 3. Methodology Design

First in this section, the explicit definition and general ideal of the proposed RAID-6Plus is presented. Then RAID-6Plus based on RDP code is instantialized and its construction is illustrated. Further, with regard to vertical codes, some modification has been made and RAID-6Plus has been applied over $X$-code to get $X$-code+, which has better load balance.

*3.1. Definition and General Ideal.* In order to provide solid and flexible reliability against multifailure, RAID-6Plus is defined as a methodology to extend any conventional RAID-6 coding algorithm to delivery extra fault tolerance over double-failure tolerance in a new and practical way. Hereby, the meaning of "plus" is twofold by standing for that higher and more flexible reliability as well as extra cost of an added parity drive compared with RAID-6 configuration after extension. Explicitly, RAID-6Plus keeps the original encoding paradigm of a RAID-6 algorithm to maintain double-failure tolerance and adds one extra redundant drive to aid for accelerating any single failure rebuild scenario by reusing data elements. In order to reuse as many data elements as possible, the optimal reconstruction for single failure rebuild needs to be studied to find out the overlapping elements to be used in the third parity drive encoding. Note that how to find and reuse overlapped data elements would differ among different RAID-6 coding algorithms. In that way, a reasonable compromise is achieved between reliability level and system performance by not only accommodating nonnegligible double-failure but also shortening the reconstruction window to degrade failure and reduce user wait.

In detail, of all the three redundancy drives, the first two are devoted to maintaining a lower bound of reliability, which we denote as "base reliability," given RAID-6 coding scheme is widely deployed in diverse storage system for double-failure concern.

The remaining redundant $X$ drive is dedicated to reducing reconstruction window for single failure of any data drive, which is in charge of user access. Therefore, the key lies in the redundancy coding for $X$ drive. Conventionally, there are three ways for $X$ drive coding: (1) mirroring of a single data disk; (2) short combination; and (3) full combination, which

Table 1: Feature comparison of three coding ways for $X$ drive.

| Coding methods | Element example | Involved element | Merit | Shortcoming |
|---|---|---|---|---|
| Mirroring of single disk | $a1$ | 1 | Simple and can maximize reduction of reconstruction window for specific drive | Only covering replicated drive, not able to cover other drives, causing imbalance and fluctuation |
| Short combination | $a1 + b1$ | 2 | In between | |
| Full combination | $a1 + b1 + c1 + d1 + e1$ | 5 | Maximizing fault-tolerance for the whole system | Unable to reduce reconstruction window |

is the norm of existing extension methodology. The features of the three coding ways are so obvious that we summarize them in Table 1. Mirroring has a length of only one, suggesting any $X$ element is a replica of some element in other drives. Full combination has the same length of any element in $P$ or $Q$ drive, implying any element in $X$ is the combination of many elements in data drives while short combination is in between.

In order to reduce data reads, short combination with some two elements involved has been employed to encode for $X$ parity. The thought behind it is to find overlapping elements as many as possible on the basis of optimal reconstruction for single data disk in the base code and combine any two of them for the concern of data disk coverage.

In short, those three ways of coding for $X$ parity elements mainly differ in number of elements involved.

*3.2. RDP+ over RDP.* Since its birth, RDP code has been one of the most popular and efficient RAID-6 codes in academia and industry due to its performance. RTP code extended right from RDP has been proposed years ago. Therefore, another RDP-based code extension is offered by applying the proposed methodology and constructing RDP+.

*(i) Optimal Reconstruction for Single Erasure in RDP.* In order to explicitly illustrate the construction of RDP+, the optimal reconstruction for single erasure in RDP is provided to get a clear understanding of coding in the third redundant drive.

In RDP, there are two kinds of parity drives, where $P$ drive means slope 0 and $Q$ for slope 1. Whenever any single data disk fails, the conventional way to reconstruct a failed disk is merely using $P$ drive while the optimal way is using equal number of parities from $P$ and $Q$ drives, maximizing the overlapping data, as shown in Figure 2 [17].

*(ii) RDP+ Construction.* As proposed in RAID-6Plus, the original $P$ and $Q$ drives of RDP code are kept to maintain base reliability of double-failure tolerance. Much more attention is paid to the coding of the third redundant drive $X$. Getting insights from the optimal reconstruction of single failure in RDP, we understand the hybrid use of equal numbers of $P$ parities and $Q$ parities can maximize the overlapping data for single failure rebuild. Thus an attempt has been made to employ short combination with some two elements involved

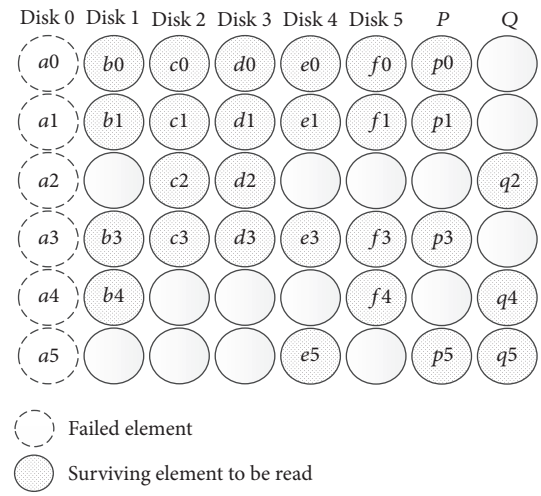


Figure 2: The optimal reconstruction sequence of single failure in RDP.

in optimal single rebuild to encode for $X$ parity, as shown in Figure 3.

In the view of $X$ drive in RDP+, nearly all the data drives are covered, because any new parity in the third parity drive $X$ is the XOR of some two data elements. Those data pairs or tuples to construct $X$ parity are specially chosen to satisfy fast reconstruction. For example, $a0$ and $a1$ of disk #0 are, respectively, included in $x0$ and $x1$, which will be used in the reconstruction of disk #0. In other words, disk coverage in some way guarantees the even and balanced distribution of speedup effect on multidisks. Those short combinations in $X$ drive constructed by data pairs are specially chosen to satisfy fast reconstruction. Similar algorithms to those in [18–20] can be easily constructed to find proper short combinations for $X$ drive.

*(iii) Single Failure Rebuild in RDP+.* Regarding single failure reconstruction, for example, if *disk#0* fails, reconstruction with the participation of related short combinations in $X$ will occur.

As shown in Figure 4, we use $x0$ and $f0$ to recover $a0$, $x1$, and $e1$ for $a1$. With the help of $p3$ and $p4$, respectively, $a3$ and $a4$ are reconstructed in the direction of slope 0 while $a2$ and $a5$ are, respectively, recovered from slope −1

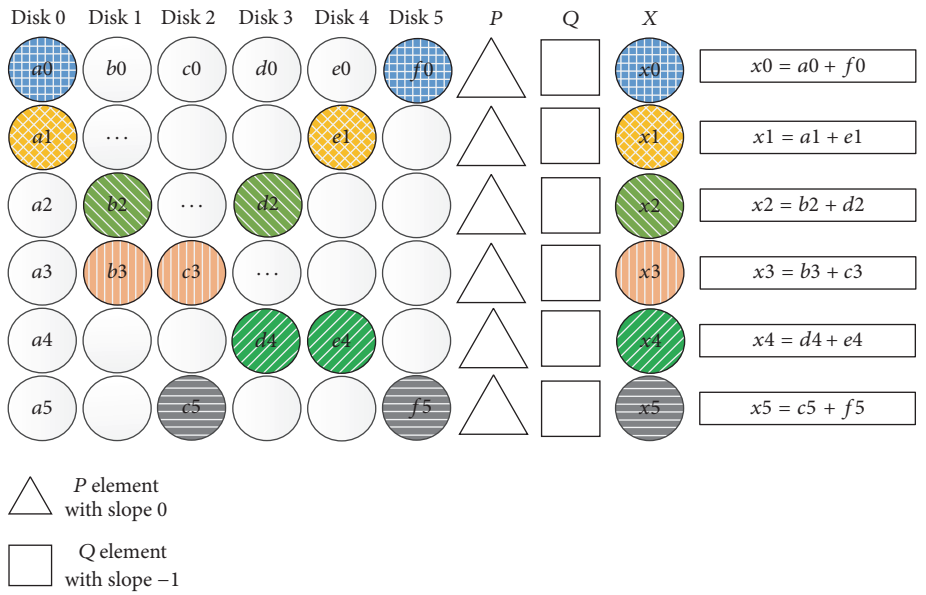


Figure 3: Encoding for $X$ drive in RDP+.

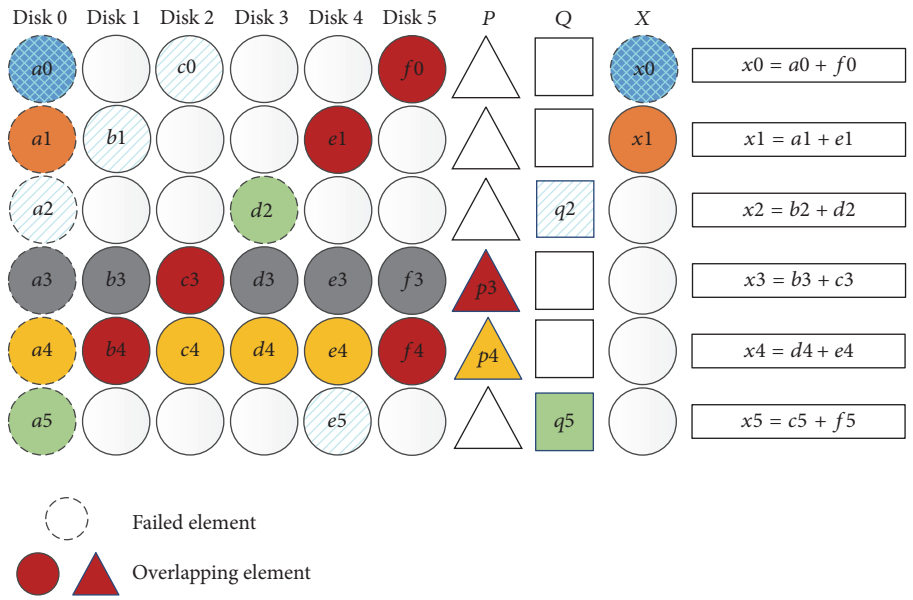


Figure 4: Rebuilding single failure with $X$ elements.

with $q2$ and $q5$. In total, there are 22 elements needed to be read, comparing with 27 element reads of RDP optimal recovery.

In the views of double failures, RDP+ maintains system reliability in two ways. First and foremost, RDP is included in RDP+; therefore there will be no data loss whenever any double failures happen.

Additionally, with shorter reconstruction window, some double-failure situations previous in traditional RAID-6 system could be converted to independent single failures, thus eliminating vulnerability undergone by the system. Also, the same way of using short combinations in $X$ can be applied to save data reads for double-failure scenarios.

In fact, triple failures happen at an extremely tiny probability; therefore the third parity drive in traditional codes only exists for extreme cases. According to Reliability Equation in [26], RDP+ could convert a portion of triple-failure situations in traditional coding schemes to lower possibility, leaving unconvertible triple failures at a negligible level.

*3.3. X-Code+ over X-Code.* Unlike aforementioned RAID-6 codes, which are all horizontally aligned, $X$-code [6] stands out as a vertical code and has the unique property in update complexity, which is denoted by the penalized writes to parity caused by a write request to a single data element. Additionally, nonvolatile memory (NVM) is gaining

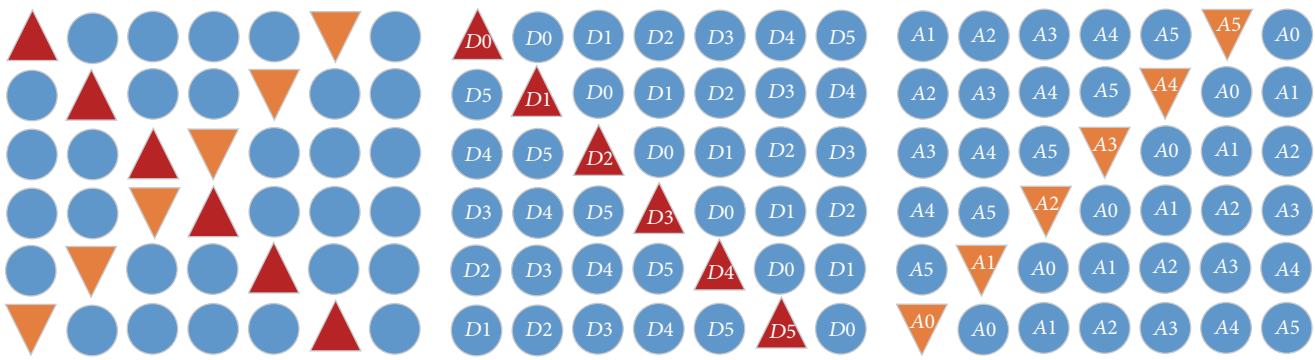


FIGURE 5: (a) presents the layout of modified $X$-code. (b) and (c), respectively, give the detailed construction of $D$ parity and $A$ parity.

popularity and much sensitive to write operation [12]. With its born optimality on update penalty, $X$-code is very suitable to be used in nonvolatile memory systems for its optimal update complexity to mitigate write operations. However, no extension based on $X$-code exists, leaving us possibility of extending $X$-code with the proposed RAID-6Plus methodology.

Nevertheless, since its different data layout, what has been done to RDP cannot simply be applied to RDP to get the extended code. A particular modification of data layout on $X$-code is needed.

*(i) Layout Modification of Original X-Code*. Originally, all parity elements in $X$-codes are aligned downward horizontally. When being extended, the layout of $X$-code needs to be modified to maintain storage efficiency. Therefore, all parity elements in *the shape of* $X$ among the data elements are aligned and one more column of data elements is added for balance, as shown in Figure 5.

In this way, original $X$-code of size $p \times p$ has been modified to size $(p-1)p$ and with all the following properties preserved: (1) parity construction, (2) update complexity, and (3) MDS property. The things changed are array size and element layout.

The objective for doing so is to achieve the following:

(1) *Base reliability*: the modified $X$-code is aimed at maintaining a lower bound of reliability, denoted as "base reliability," given that RAID-6 is widely deployed in various storage system for double-failure concern.

(2) Optimal update complexity before extension so as to mitigate media wear-out penalized with write operations.

*(ii) X-Code+ Construction*. The modified $X$-code is extended by adding one more drive of parity as existing extensions do, but in a new and more practical way.

Note that different from RDP+, load balance is taken into consideration when constructing $X$-code+. It is determined by the layout of $X$-code, where data elements and parity elements are stored together in each drive. Therefore $X$-code+ is intended to achieve a reasonable compromise among (1) reliability, (2) performance, and (3) load balance, where,

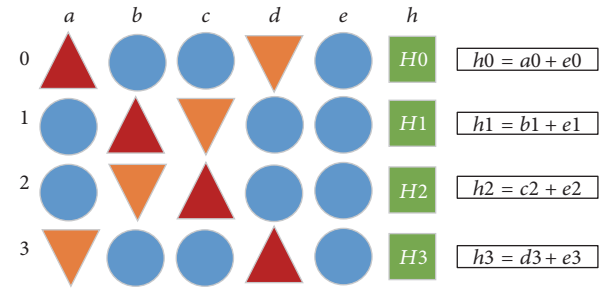


FIGURE 6: The coding for parity elements in $H$ drive.

on the one hand, we maintain a basic reliability guarantee to accommodate nonnegligible double-failure, while, on the other hand, reconstruction window is shortened to degrade failure and offload bottleneck access and reduce user wait.

In the proposed $X$-code+, the added parity drive is denoted as $H$ *drive* for clarity, whose purpose is to reduce reconstruction window and balance load for single failure and whose parity is constructed *horizontally*. Though $H$ *drive* is a pure parity drive, free of user access; it plays a key role in various failure modes.

In terms of constructing parity elements in $H$ drive, two-element tuples are selected horizontally. By getting insight from the optimal single failure recovery of modified $X$-code, it is aimed at (1) minimizing total data reads by reusing overlapping data and (2) balancing load by offloading bottleneck workload to the $H$ drive as much as possible.

With the concrete example in Figure 6, the proposed construction of parity elements in $H$ drive is illustrated.

As can be seen from Figure 6, parity elements in $H$ drive are aligned vertically and each one is the XOR sum of a two-element tuple; for example, $h0$ is the XOR sum of $a0$ and $e0$. In order to cover all existing drives when any of them fails, some elements are included in the components of $H$ drives. For example, $a0$ in h0 is included, which will be used in face of column $a$. Likewise, other drives are covered by introducing some elements. To pay attention, $E$ drive is the only one that stores only data to bear much user access; therefore we have all its elements ($e0$, $e1$, $e2$, and $e3$) as components for $H$ parity.

Table 2: Recover $a0$ with parity of different length in Figure 8.

| Parity | Parity type | Parity length | Recovery sequence for $a0$ | Data needed |
|---|---|---|---|---|
| $H0 = a0 + e0$ | Horizontal | 2 | $h0$, $e0$ | 2 |
| $A0 = b0 + d2 + e3$ | Diagonal | 3 | $b0$, $d2$, $e3$ | 3 |

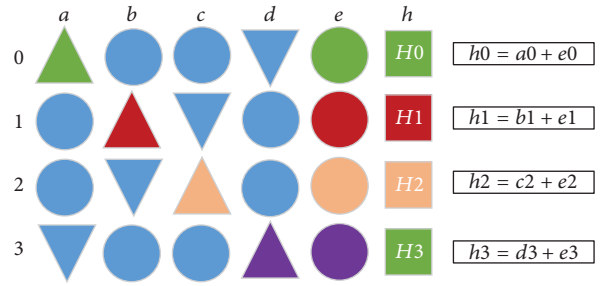


Figure 7: The general construction of $H$ parity as the XOR sum of $D$ parity with its corresponding element in $E$ drive.

In other words, storage drive coverage in this way guarantees as much as we can the even and balanced distribution of speedup effect on multiple drives.

In essence, $H$ parity elements are combinations with the length shorter than original $D$ (diagonal) parity or $A$ (antidiagonal) parity so that its components could be recovered with fewer data. For example, any $D$ or $A$ parity consisted of 3 elements while $H$ parity is comprised of 2 elements. Furthermore, the gap will be bigger and pronounced with the array size growing up.

In terms of which elements should be selected to be components, a simple and straightforward way is introduced by adding one element from $D$ parity and its corresponding element horizontally in $E$ drive, as shown in Figure 7.

Because $D$ parity and $A$ parity are symmetrical, therefore it is acceptable to use either $D$ parity or $A$ parity.

*(iii) Single Failure Rebuilt in X-Code+.* $X$-code+ is intended to strike a balance between reliability and performance and load balance. This property is better illustrated in the face of single failure reconstruction. It has been reported that 99.75% of recoveries are for the single disk failure [24]. Accordingly, the performance of recovery for single disk failure is of high importance to reduce window of vulnerability.

Regarding single failure reconstruction, for example, if column a fails, reconstruction with the participation of related parity in $H$ drive will occur. As shown in Figure 8(a), before extension, $a0$ and $a1$ are recovered along the diagonal direction while $a2$ and $a3$ are fixed with the help of their corresponding antidiagonal elements.

In contrast as shown in Figure 8(b), $h0$ and $f0$ are used to recover $a0$. With the help of parity $D2$ and $D3$, respectively, $a2$ and $a3$ are reconstructed in the direction of slope 1 while parity $A0$ is, respectively, recovered from slope −1 with element $b3$, $d1$, and $e0$. Therefore, with the help of parity $h0$, $f0$ is recovered horizontally instead of diagonally and meanwhile, the overlapped $e0$ is preserved.

In order to better measure reconstruction performance, we present a comparison on total data reads and bottleneck drive.

*(iv) Total Data Reads.* In the erasure coding field, reducing total amount of data reads in the recovery is the main consideration behind code design. There are examples in non-MDS (Maximum Distance Separable) [5] codes such as Pyramid code [28]. Further, many excellent works of optimization are also measured by total data reads, like Khan et al. [29]. In this way, it could be seen that $X$-code+ is able to reduce total data reads for single failure recovery from 10 to 8.

The reason behind total data reads reduction is because of the following:

(1) *H parity* with shorter length is used to replace original *D parity* or *A parity* with longer length; thus less elements are needed, as shown in Table 2.

(2) Overlapped elements are preserved and reused with the use of $H$ parity. This is our consideration in the construction of parity elements in $H$ drive. For example, in Figure 3, $e0$ is one of the overlapped elements and in order to preserve and reuse it after extension, we have $e0$ in $h0$ in Figure 8.

In this way, $X$-code+ is conducive to saving total reads. This effect could be greater when array size grows bigger and bigger.

*(v) Load Balance.* There are two flaws to use total data reads as an indicator for recovery performance: (1) total data read is a static metric with particular codes. Their recovery performance is determined as long as the codes are designed. (2) Given the distribution and parallel IO across multidrives, minimizing the total amount of data accessed for the recovery does not necessarily translate into minimal recovery time [23, 30].

Therefore, with multiple drives serving the read requests for a recovery with parallel I/O, it is the service time of the disk that reads the largest amount of data that determine the recovery time.

In terms of this, it can be seen in Figure 8(a) that the bottleneck before extension is *column d* with 4 elements read while the bottleneck after extension is reduced to 2 elements read in *columns c, d*, or *e*, as shown in Figure 8(b). Therefore the proposed $X$-code+ is able to offload recovery IO with help of newly added *H parity*. This would translate directly into recovery performance.

In summary, $X$-code+ outperforms modified $X$-code on both total data reads and load balance.

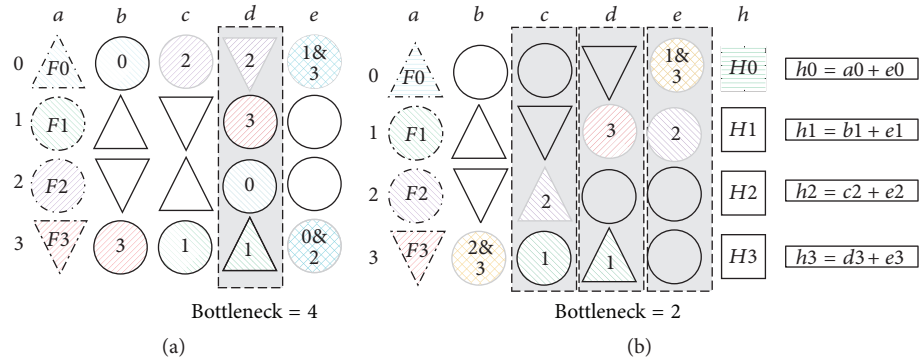


Figure 8: (a) Recovery of column $a$ before extension; (b) recovery of column $a$ after extension.

## 4. Evaluation

In this section, RDP+ and $X$-code+ along with its counterpart codes are compared at the same cost in the aspects of parity computation complexity, update penalty, reconstruction cost, and storage ratio. The basic size for STAR and modified $X$-code is $(p-1)p$ and $(p-1)(p-1)$ for RTP. Further, a $Q$ metric is proposed to denote the induced performance improved per cost to validate the proposed RAID-6Plus methodology.

*4.1. Encoding Complexity.* The encoding process is complicated and influenced by many aspects. However in theory, parity computation complexity is reasonably used to denote the encoding complexity. Hereby, given XOR as a basic operation, XORs per element is used to quantitatively represent parity computation complexity among different codes.

The total XORs are the sum of XORs for different parity types. For example, the number of XORs for writing a stripe in RDP+ consisted of XORs for row parity, diagonal parity, and $X$ parity, while those of $X$-code+ are of XORs for diagonal parity, antidiagonal parity, and horizontal parity, given $H$ parity could be calculated with existing $D$ parity cached and data. Thus, we have

$$\begin{aligned} A_{\text{RDP+}} &= \frac{\text{XOR}_{\text{row}} + \text{XOR}_{\text{dia}} + \text{XOR}_{x}}{\text{Blocknum}} \\ &= \frac{(p-1)(p-2) + (p-1)(p-2) + p - 1}{(p-1)(p-1)} \\ &= \frac{2p-3}{p-1} = 2 - \frac{1}{p-1}, \\ A_{X\text{-code+}} &= \frac{\text{XOR}_{\text{Dia}} + \text{XOR}_{\text{anti-dia}} + \text{XOR}_{\text{Hori}}}{\text{Elementnum}} \\ &= \frac{(p-1)(p-3) + (p-1)(p-3) + (p-1)}{(p-1)(p-2)} \\ &= 3 - \frac{1}{p-2}. \end{aligned} \tag{1}$$

Similarly, we can get that of RTP is $A_{\text{RTP}} = 3 - 3/(p-1)$ and $A_{\text{STAR}} = 3 - 1/(p-1)$ for STAR. The results are shown in Figure 9.

Obviously, RDP+ uses least XORs per data block for parity computation and is fastest. The difference is more obvious when disk array size is smaller, which is the typical case of most storage systems. During encoding process, the complexity of $X$-code+ is between RTP and STAR code. The differences in between will become less and less with array size growing bigger. In the best case, $X$-code+ is 3.1% faster than STAR.

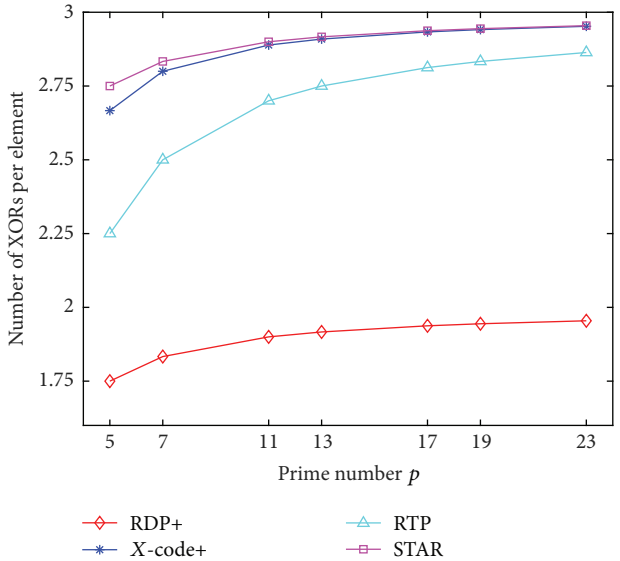


Figure 9: Parity computation complexity.

*4.2. Update Penalty.* Redundancy incurs update penalty for consistency between data and its related parity. Hereby, the update penalty is measured by number of introduced writes except writing on data itself. For example, in original $X$-code, any update on a single data block will result in two more writes to, respectively, its corresponding diagonal parity and antidiagonal parity; thus its update penalty is 2. Through similar calculation [31], the update penalty of RTP turns out to be $U_{\text{RTP}} = 3 - 2/(p-1) + 2/(p-1)^2$ [8] while STAR is $U_{\text{STAR}} = 5 - 4/p$ in [32].

For RDP+, update penalty of all data elements requires

$$\begin{aligned} &[(p-1)(p-1) - 2(p-1) - (p-2)] \times 2 + 2 \\ &\quad \times (p-1) \times 3 + (p-2) \times 1 = 2p^2 - 3p + 2. \end{aligned} \tag{2}$$

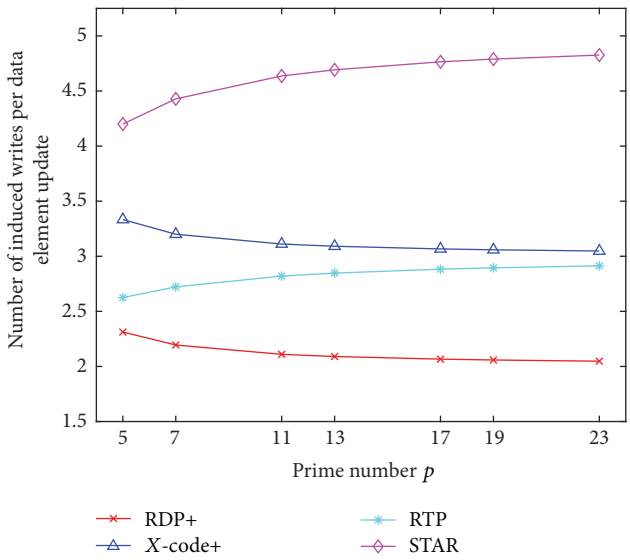


Figure 10: Update penalty.

Thus, the update penalty per block is

$$U_{\text{RDP+}} = \frac{\left(2p^2 - 3p + 2\right)}{(p-1)^2} = 2 + \frac{1}{p-1} + \frac{1}{(p-1)^2} \tag{3}$$

which is smaller than RTP and STAR. Likewise, update penalty of all data elements for $X$-code+ is

$$\begin{aligned} U_{X\text{-code+}} &= \frac{3\left[(p-1)^2 - 2(p-1)\right] + 4(p-1)}{(p-1)\times(p+1) - 3(p-1)} \\ &= 3 + \frac{1}{p-2}. \end{aligned} \tag{4}$$

Comparison results are plotted in Figure 10. Significantly, RDP+ bears least update penalty, and it converges toward optimal update penalty of double-failure-tolerant codes with bigger array size while $X$-code+ converges toward optimal update penalty of triple-failure-tolerant codes. For example, when $p$ equals 11, RDP+ has 2.11 update penalties compared with 2.82 of RTP and 4.636 of STAR. The difference will be more obvious when disk array size grows bigger. The reason is the proposed RDP+ methodology that has least parity nesting in the third parity drive, while RTP and STAR have more parity nesting and use full combinations with larger length, therefore incurring more update penalty.

*4.3. Reconstruction Cost for Single Failure.* Specifically, the reconstruction performance for single failure is evaluated on both total data read and load balance.

In terms of single failure, because RTP and RDP share exactly the same reconstruction sequences, they have the same reconstruction cost, which are, respectively $(p-1)(p-1)$, with conventional recovery and $(3/4)(p-1)^2$ with optimal recovery. Similarly, STAR rebuilds a failed drive with the same cost of EVENODD, which is $(p-1)p$ conventionally and $(3p+1)(p-1)/4$ by optimal recovery.

When a drive fails in RDP+, two elements are recovered by two $X$ parities and half of the rest elements are, respectively, rebuilt with $P$ and $Q$ parity as shown in Figure 5. Therefore, we compute data elements read for rebuild in RDP+ as

$$\begin{aligned} &2\times 2 + \frac{(p-1)-2}{2}\times(p-1) + \frac{(p-1)-2}{2}\times(p-1) \\ &\quad - \frac{(p-1)-2}{2}\times\frac{(p-1)-2}{2} - 2 \\ &= 2 + \frac{(p-3)(3p-1)}{4}. \end{aligned} \tag{5}$$

Initially, the total data read of modified $X$-code is

$$\begin{aligned} T_{M\text{-}X\text{-code}} &= (p-1)(p-2) - \frac{p-1}{2}\times\frac{p-3}{2} \\ &= \frac{p-1}{4}(3p-5). \end{aligned} \tag{6}$$

With the help of *H parity* participating in reconstruction, total data reads could be reduced with shorter parity length and reuse of overlapping elements. Therefore, through calculation, the total data read of $X$-code+ is achieved as

$$\begin{aligned} T_{X\text{-code+}} &= 2 + (p-2)(p-2) - 1 - \frac{p-1}{2}\times\frac{p-3}{2} \\ &= \frac{3(p-1)(p-3)}{4} + 2. \end{aligned} \tag{7}$$

Hereby results normalized by the total data read of modified $X$-code are shown in Figure 11.

Clearly, $X$-code+ reads the least data and RDP+ is the second least. For example, when $p$ is 7, RDP+, respectively, have a normalized speedup of 33.4%, 11.9%, 47.7%, and 26.2%, respectively, over RTP under the conventional rebuild (labeled as RTP_C), RTP under the optimal rebuild (labeled as RTP_O), STAR under that conventional rebuild (labeled at STAR_C), and STAR under optimal rebuild (labeled at STAR_O), respectively.

The secret of the proposed methodology for single erasure rebuild is that it uses short combinations in the third redundant drive to minimize elements needed and meanwhile reuse overlapping elements as much as possible by optimal reconstruction sequence of RDP.

*(i) Load Balance for X-Code+.* Upon normal user access, RTP and STAR will experience the hot-drive effect because parity elements are placed in drives independent of data, causing serious update bottleneck. In contrast, $X$-code+ employs a hybrid placement policy by mixing diagonal and antidiagonal parity with data and meanwhile storing horizontal parity independently, thus mitigating the hot-drive effect and balance access.

When failure happens, $X$-code+ reconstructs failed drives with the help of horizontal parity to alleviate the recovery bottleneck with most IO on it by offloading some load to the *H drive*, as shown in Figure 8. Under some

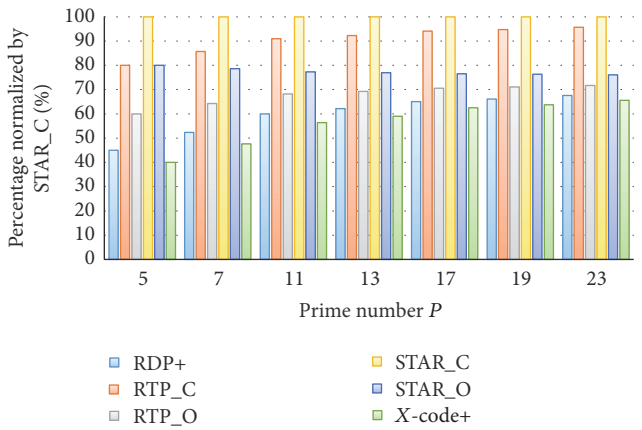


Figure 11: Data reads for rebuilding single failure.

specific reconstruction sequence, the bottleneck load could be reduced from $p-1$ to $p-3$.

In this way, in both normal and failure situations, $X$-code+ provides better load balance among its peer codes.

*4.4. Storage Overhead and Reliability.* RTP and STAR are both MDS codes [5] while RDP+ and $X$-code+ are non-MDS. Regarding storage overheads, all the three codes have three drives dedicated to parities. In terms of absolute failures, RTP and STAR are strictly triple-failure tolerant while both RDP+ and $X$-code+ can tolerate only double failures at device level. But they provide flexible and higher relative reliability by shorter single failure reconstruction. This effect is hard to be explicitly characterized because there are not any plausible reliability model for RAID systems.

*4.5. Q Metric for Reconstruction.* Better performance comes at some cost. Usually, the gains and the pains are independently evaluated, without any indicator to suggest their correlation. However, the $Q$ metric we propose intend to bridge this gap and is truly an indicator of improvement correlated with cost we spent. Hereby we use RDP+ as an example to illustrate.

Essentially, coding schemes like RTP, STAR, and the proposed RDP+ are extending existing RAID-6 array codes by adding extra parity drive. In order to evaluate the validity and utility of adding extra parity drive, a $Q$ metric is proposed to denote the induced benefit per cost. The higher the $Q$ value is, the more competitive and useful the method will be. Exactly in this paper,

$$Q = \frac{\text{Induced benefit}}{\text{Cost}} = \frac{\text{Speedup of single failure rebuild}}{\text{ratio of extra parity over data drives}}. \quad (8)$$

When adding a third parity drive, we have three ways to encode the third parity drive, as mentioned in Table 1. RTP is using full stripe while the proposed RAID-6Plus is short striped. We will compare RTP, RAID-6Plus, mirroring for single drive, and mirroring for all drives with $Q$ metric and take RDP_optimal as the baseline performance.

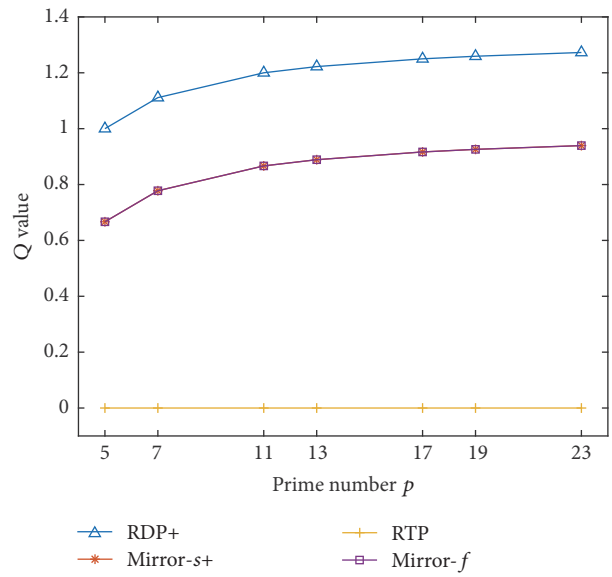


Figure 12: $Q$ comparison of four ways of adding third parity drive.

$$Q_{\text{RTP}} = \frac{\left[(3/4)(p-1)^2 - (3/4)(p-1)^2\right] \div (3/4)(p-1)^2}{1/(p-1)} = 0,$$

$$Q_{\text{RAIS-}P} = \frac{\left[(3/4)(p-1)^2 - (2+(p-3)(3p-1)/4)\right] \div (3/4)(p-1)^2}{1/(p-1)} = \frac{4(p-2)}{3(p-1)},$$

$$Q_{\text{Mirror\_}s} = \frac{\left[(3/4)(p-1)^2 - 3(p-1)(p-2)/4 - 1\right] \div (3/4)(p-1)^2}{1/(p-1)} = \frac{3p-7}{3p-3}, \quad (9)$$

$$Q_{\text{Mirror\_}f} = \frac{\left[(3/4)(p-1)^2 - (p-1)\right] \div (3/4)(p-1)^2}{1} = \frac{3p-7}{3p-3}.$$

The results are shown in Figure 12.

In this way, it can be clearly seen that the existing RTP has no contribution in accelerating single failure rebuild with a zero $Q$, which is the very motivation of RAID-6Plus. Among possible way to encode the third parity drive, RAID-6Plus prevails. The reason behind this is RAID-6Plus has nearly an even and balanced coverage for all data drives and short combinations in $X$ drive use as less data elements as possible.

In short, $Q$ metric is proving that, unlike RTP and STAR, RAID-6Plus is a valid way to extend RDP from the perspective of single failure reconstruction.

## 5. Related Work

Typically, in erasure codes, there are two basic arithmetic to be in use: XOR and Galois Field. XOR-based codes are faster than GF-based codes while GF-based code has wider flexibility in code construction [5]. RAID-6Plus, as an extension on the basis of XOR-based codes, is still XOR-based, such as RTP [8] and STAR [27]. RAID-6Plus is faster with less complexity incurred in parity computation.

MDS codes like RS code are traditional focus of storage reliability for they offer further reliability boundary, that is, absolute fault tolerance [5]. Both RTP and STAR are MDS codes, but they are unable to accelerate single failure reconstruction. Non-MDS codes are thus invented. A typical example is the LRC codes from Microsoft [33]. However, LRC is horizontal codes and GF-based while RAID-6Plus is array code and XOR-based.

Efforts have also been made to accelerating single failure. They focus either on finding optimal reconstruction sequences [17–20] or on load-balancing [23, 30]. However all these work are limited to the given codes, without modification or extension. To our knowledge, RAID-6Plus is first to extend given codes from the perspective of speedup single failure instead of providing reliability boundary. Further, RAID-6Plus [14] is orthogonal with previous work above and can be integrated together to further shorten reconstruction window.

## 6. Conclusions

The existing methodology of extending RAID-6 codes considers only the fault tolerance level by utilizing the third redundant drive to recover from triple-failure scenarios. As a result, no contribution is made to speed up rebuilding single failures, and consequently, the third parity drive is almost idle.

This paper proposes RAID-6Plus, allowing the reuse of overlapped elements during reconstruction to balance the reliability and performance of the resulting coding scheme. By applying the proposed methodology to RDP and $X$-code, respectively, we generate two new coding schemes named RDP+ and $X$-code+ to provide a better balance between performance and reliability. The performance evaluation indicated that RAID-6Plus exhibited (1) a better system performance with no extra cost compared to RTP and STAR and (2) an enhanced reliability compared to RAID-6. With the proposed $Q$-metric, a detailed justification of the proposed methodology is provided.

In short, RAID-6Plus held the potential of practical uses in modern disk systems, flash-based systems, and even hybrid storage systems on any array codes. As for the future work, it would be interesting to investigate how RAID-6Plus should be applied in real storage systems and also how such algorithm could be optimized in terms of flash memory. For example, $X$-code+ would be helpful to mitigate the write amplification problem of flash memory with less update penalty in providing system reliability.

## Disclosure

This work is an extended version of the conference papers presented at 4th International Conference on Computer Science and Network Technology (ICCSNT) [21] and 9th Asia-Pacific Services Computing Conference, APSCC 2015.

## Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant nos. 61433019, U1435217, 61232003, 61502514, 61402503, and 61402501 and National High Technology Research and Development 863 Program of China under Grant 2015AA015305.

## References

[1] A. Ma, R. Traylor, F. Douglis et al., "RAIDShield: characterizing, monitoring, and proactively protecting against disk failures," *ACM Transactions on Storage (TOS)*, vol. 11, no. 4, article 17, 2015.

[2] J. S. Plank, "A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems," *Software, Practice & Experience (SPE)*, vol. 27, no. 9, pp. 995–1012, 1997.

[3] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures," *IEEE Transactions on Computers*, vol. 44, no. 2, pp. 192–202, 1995.

[4] C. Lueth, "RAID-DP: network appliance implementation of RAID double parity for data protection," Tech. Rep. 3298, Network Appliance, 2004.

[5] J. S. Plank, "T1: erasure codes for storage applications," in *Proceedings of the 4th USENIX Conference on File and Storage Technologies (FAST '05)*, San Francisco, Calif, USA, December 2005.

[6] M. Blaum, T. Cortes, and H. Jin, "The EVENODD code and its generalization," *High Performance Mass Storage and Parallel I/O*, pp. 187–208, 2001.

[7] C. Huang and L. Xu, "STAR: an efficient coding scheme for correcting triple storage node failures," *IEEE Transactions on Computers*, vol. 57, no. 7, pp. 889–901, 2008.

[8] A. Goel and P. Corbett, "RAID triple parity," *ACM SIGOPS Operating Systems Review*, vol. 46, no. 3, pp. 41–49, 2012.

[9] Y. Wang, G. Li, and X. Zhong, "Triple-star: a coding scheme with optimal encoding complexity for tolerating triple disk failures in raid," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 3 A, pp. 1731–1742, 2012.

[10] Y. Wang and G. Li, "Rotary-code: efficient MDS array codes for RAID-6 disk arrays," *WSEAS Transactions on Computers*, vol. 8, no. 12, pp. 1917–1926, 2009.

[11] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: high-performance, reliable secondary storage," *ACM Computing Surveys (CSUR)*, vol. 26, no. 2, pp. 145–185, 1994.

[12] A. Amer, D. D. E. Long, and S. J. Thomas Schwarz, "Reliability challenges for storing exabytes," in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC '14)*, pp. 907–913, IEEE, Honolulu, Hawaii, USA, February 2014.

[13] B. Schroeder and G. A. Gibson, "Disk failures in the real world: what does an MTTF of 1, 000, 000 hours mean to you?" *FAST*, vol. 7, pp. 1–16, 2007.

[14] M.-Z. Deng, Y. Ou, N. Xiao et al., "RAID-6Plus: a fast and reliable coding scheme aided by multi-failure degradation," in *Advances in Services Computing*, vol. 9464, pp. 210–221, Springer, Berlin, Germany, 2015.

[15] J. S. Plank and M. Blaum, "Sector-disk (SD) erasure codes for mixed failure modes in RAID systems," *ACM Transactions on Storage*, vol. 10, article 4, 2014.

[16] A. Leventhal, "Triple-parity RAID and beyond," *Queue*, vol. 7, no. 11, 2009.

[17] L. Xiang, Y. Xu, J. C. Lui, and Q. Chang, "Optimal recovery of single disk failure in RDP code storage systems," in *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '10)*, pp. 119–130, ACM, New York, NY, USA, 2010.

[18] L. Xiang, Y. Xu, J. C. S. Lui, Q. Chang, Y. Pan, and R. Li, "A hybrid approach to failed disk recovery using RAID-6 codes: algorithms and performance evaluation," *ACM Transactions on Storage*, vol. 7, article 11, 2011.

[19] Y. Zhu, P. P. C. Lee, L. Xiang, Y. Xu, and L. Gao, "A cost-based heterogeneous recovery scheme for distributed storage systems with RAID-6 codes," in *Proceedings of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '12)*, Boston, Mass, USA, June 2012.

[20] O. Khan, R. Burns, J. Plank, and W. Pierce, "Rethinking erasure codes for cloud file systems: minimizing I/O for recovery and degraded reads," in *Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST '12)*, San Jose, Calif, USA, February 2012.

[21] M. Deng, L. Zhu, N. Xiao, Z. Chen, and F. Liu, "X-code+: a compromised coding scheme with smaller rebuild window and load-balance," in *Proceedings of the 4th International Conference on Computer Science and Network Technology (ICCSNT '15)*, December 2015.

[22] J. S. Plank, M. Blaum, and J. L. Hafner, "SD codes: erasure codes designed for how storage systems really fail," in *Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST '13)*, San Jose, Calif, USA, February 2013.

[23] X. Luo and J. Shu, "Load-Balanced recovery schemes for single-disk failure in storage systems with any erasure code," in *Proceedings of the 42nd Annual International Conference on Parallel Processing (ICPP '13)*, pp. 552–561, Lyon, France, October 2013.

[24] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population," *FAST*, 2007.

[25] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti, "Introduction to flash memory," *Proceedings of the IEEE*, vol. 91, no. 4, pp. 489–502, 2003.

[26] J. G. Elerath and J. Schindler, "Beyond MTTDL: a closed-form RAID 6 reliability equation," *ACM Transactions on Storage*, vol. 10, no. 2, article 7, Article ID 2577386, 2014.

[27] C. Huang and L. Xu, "STAR: an efficient coding scheme for correcting triple storage node failures," *Institute of Electrical and Electronics Engineers. Transactions on Computers*, vol. 57, no. 7, pp. 889–901, 2008.

[28] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," *ACM Transactions on Storage*, vol. 9, no. 1, article 3, 2013.

[29] O. Khan, R. Burns, J. Plank, W. Pierce, and C. Huang, "Rethinking erasure codes for cloud file systems: minimizing I/O for recovery and degraded reads," in *Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST '12)*, San Jose, Calif, USA, 2012.

[30] Y. Fu, J. Shu, and X. Luo, "A stack-based single disk failure recovery scheme for erasure coded storage systems," in *Proceedings of the 33rd IEEE International Symposium on Reliable Distributed Systems (SRDS '14)*, pp. 136–145, IEEE, Nara, Japan, October 2014.

[31] R. Hu, G. Liu, and J. Jiang, "An efficient coding scheme for tolerating double disk failures," in *Proceedings of the 12th IEEE International Conference on High Performance Computing and Communications (HPCC '10)*, pp. 707–712, September 2010.

[32] H. Rongdong, L. Guangming, and J. Jingfei, "An efficient coding scheme for tolerating double disk failures," in *Proceedings of the 12th IEEE International Conference on High Performance Computing and Communications (HPCC '10)*, Melbourne, Australia, 2010.

[33] C. Huang, H. Simitci, Y. Xu et al., "Erasure coding in windows azure storage," in *Proceedings of the USENIX Annual Technical Conference (USENIX ATC '12)*, Boston, Mass, USA, June 2012.