J Inf Process Syst, Vol.15, No.1, pp.55~66, February 2019
https://doi.org/10.3745/JIPS.01.0035

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

HRSF: Single Disk Failure Recovery for Liberation Code Based Storage Systems

Jun Li* and Mengshu Hou*

Abstract

Storage system often applies erasure codes to protect against disk failure and ensure system reliability and availability. Liberation code that is a type of coding scheme has been widely used in many storage systems because its encoding and modifying operations are efficient. However, it cannot effectively achieve fast recovery from single disk failure in storage systems, and has great influence on recovery performance as well as response time of client requests. To solve this problem, in this paper, we present HRSF, a Hybrid Recovery method for solving Single disk Failure. We present the optimal algorithm to accelerate failure recovery process. Theoretical analysis proves that our scheme consumes approximately 25% less amount of data read than the conventional method. In the evaluation, we perform extensive experiments by setting different number of disks and chunk sizes. The results show that HRSF outperforms conventional method in terms of the amount of data read and failure recovery time.

Keywords

Erasure Codes, Disk Failure, Recovery Scheme, Reliability, Storage System

1. Introduction

In recent years, with the development of cloud and mobile computing technology, the storage system size along with the number of storage nodes has been increasing rapidly. All kinds of unpredictable failures may render storage nodes unavailable, which could seriously affect system reliability and availability. To address this issue, a variety of fault tolerance techniques have been proposed which can be mainly classified as replication and erasure codes. Replication is simple, but its storage overhead is large. In contrast, erasure codes have an advantage in optimal storage cost [1]. With increasing of system size, the amount of stored data increases rapidly. Replication technology is difficult to meet the requirements of mass storage systems in terms of storage utilization and fault tolerance. Therefore, erasure codes have been attracting more and more attention in academia and industry.

The researches of erasure codes are mainly focus on the aspects of encoding, decoding and updating complexity. There are some typical codes, such as RS code [2], RDP [3], EVENODD [4], STAR [5], and Short Code [6]. These codes can improve system reliability and availability. However, researchers find that if one disk fails, the probability of another disk failure will greatly increase. Once the number of disk

Manuscript received May 11, 2017; first revision July 12, 2017; accepted July 20, 2017.
Corresponding Author: Jun Li (suiyuanlj2006@gmail.com)
* School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China (suiyuanlj2006@gmail.com, mshou@uestc.edu.cn)

failures exceeds the capacity of fault tolerance in a system, data stored on the failed disks will not be restored [7]. Thus, for a storage system, repairing single disk failure timely is crucial.

There are many research findings in boosting single disk failure recovery process. RDOR [8] is an RDP-based scheme which can quickly recover failed disk. Wang et al. [9] introduced a similar scheme based on EVENODD. Based on X-code, Shen et al. [10] proposed SIOR. The authors of [11] proposed EDP based on P-code. Nakkiran et al. [12] presented a fast encoding method based on PM codes. Itani et al. [13] designed a single disk recovery method for fractional repetition (FR) code. When local reconstruction codes (LRC) were used in primary array storage systems, Sung and Park [14] observed that there was a major bottleneck in reconstruction. Then, they presented distributed reconstruction codes (DRC) that can rebuild disks rapidly. However, these methods are used for failure recovery of storage systems which use a specified code. Kahn et al. [15,16] introduced a scheme which is suitable for all codes. They found out a potential optimal way of recovering lost data. However, they observed that this scheme was NP-hard. To search for a fast recovery method, a replacement recovery algorithm EG was proposed. By using hilling-climbing method, EG find some optimal parity sets to minimize data transmission overhead [17]. These methods require a certain period of time for disk failure recovery. When there are a large number of nodes in the storage system, the recovery process is complicated and inefficient.

In order to reduce the amount of data read from disks, other methods have been proposed. For erasure-coded data recovery operations, Rashmi et al. [18,19] studied their impacts on datacenter networks, and then proposed a piggybacking framework to reduce disk space and network bandwidth overhead in a recovery process. In order to solve the single node failure recovery problem in a cluster file system setting, Shen et al. [20] proposed CAR, which is a cross-rack-aware failure recovery method. Sathiamoorthy et al. [21] proposed locally repairable codes to reduce the recovery overhead of RS code. However, these codes may increase the local parity blocks in a recovery process.

Liberation code is proposed by Plank [22], the encoding and modifying performance of it is better than RDP, EVENODD and other types of RAID-6 codes. However, there is no effective recovery solution to the problem of single data disk failure. In this paper, in order to address this problem, we present a scheme called HRSF, which can boost the recovery process. Theoretical analysis and experimental results show that HRSF outperforms conventional Liberation code in terms of the amount of data read and failure recovery time. The contributions of this work are listed as follows.

1. For single disk failure, we present a recovery method called HRSF, which can reduce data read and speed up a failure recovery process.
2. In a single disk failure recovery process, we obtain a lower bound of data read. Through theoretical analysis, we can know that HRSF scheme consumes approximately 25% less amount of data read than conventional method.
3. In order to evaluate our proposed recovery scheme, we perform extensive experiments. The experimental results can verify the theoretical analysis. It has less amount of data read and recovery time than conventional method.

The rest of paper is organized as follows. In the next section, we introduce Liberation code, and then describe the construction of encoding matrix and conventional recovery method of single disk failure. In Section 3, we present a hybrid recovery method called HRSF to reduce the amount of data read during recovery. We describe the experiments in detail and discuss the results in Section 4. Section 5 concludes this paper and introduces future work.

2. Background of Liberation Code

In this section, we introduce Liberation code, and then describe the conventional method of single disk failure recovery.

2.1 Liberation Code

Fig. 1 describes a typical RAID-6 system. In a storage system, we assume that there are k data disks named D_0, D_1…D_{k-1}. In addition, there are two parity disks P and Q. It can tolerate any two disks failure. For each disk, it is divided into some strips, and each strip consisting of w elements. In the disks, all different strips at the same offset position can form a stripe. In a stripe, using Liberation code, the two parity elements are generated from the data elements. For different stripes, the encoding operations are independent. Without loss of generality, in this paper, we discuss operations in a stripe.

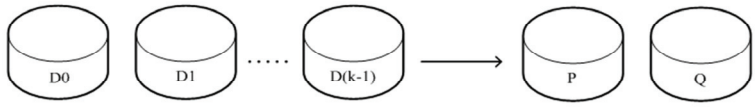

Fig. 1. The structure of a RAID-6 system.

Similar to other erasure codes, for Liberation code, its encoding and decoding operations depend on encoding and decoding matrices. The value of w is restricted which depends on k. In addition, it specifies that w must be a prime which is $\geq k$ and> 2. Bit matrix for encoding rules when $w = 5$, $k = 5$ is shown in Fig. 2. We know that P is a parity disk of the corresponding data disks, so each matrix is equal to a $w \times w$ identity matrix. For an identity matrix, the corresponding position of one is marked in the figure. The matrix X_i which generates the parity data of Q disk is described as follows [22].

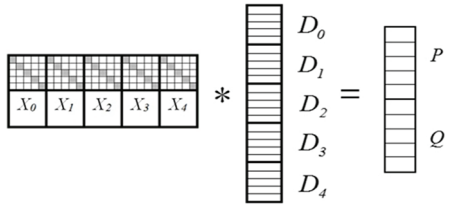

Fig. 2. The bit matrix for encoding rules of RAID-6 when k=5, w=5.

Before introducing the matrix X_i, we make the following definitions.

(1) Define $I^w_{\rightarrow j}$ to be a $w \times w$ identity matrix. For the matrix, its columns have been looped to the right by j columns. In particular, $I^w = I^w_{\rightarrow 0}$.

(2) Define $O^w_{i,j}$ to be a $w \times w$ matrix. In this matrix, every element is equal to zero, expect for the elements in (i mod w) row and (j mod w) column. In these positions, elements are equal to one.

For Liberation code, the matrix X_i is described as follows.

(1) $X_0 = I^w$.

(2) For $0 < i < k$, $X_i = I^w_{\rightarrow i} + O^w_{y,y+i-1}$. In the formula, $y = \frac{i(w-1)}{2}$. It is equivalent to $y = \frac{w-i}{2}$ when i is odd, and $y = w - \frac{i}{2}$ when i is even

According to the above definitions of Liberation code [22], we can obtain that for any given value of k and w, the total number of ones is $kw + k - 1$ for X_i matrices. Adding it to the kw ones of disk P's encoding matrices, so in the two types of matrices, there are $2kw + k - 1$ ones. The factor which affects the efficiency of encoding operation is the number of XOR-summing operations, and it is decided by the number of ones in matrices. As we all know, in matrices, if a coding bit's row has c ones, it needs $c - 1$ XOR-summing operations to encode that bit from the data bits. For Liberation code, the total number of parity elements is $2w$. Therefore, the average of XOR-summing operations to calculate each parity element p_i or q_i is $\frac{2kw+k-1-2w}{2w}$. It is equal to $k - 1 + \frac{k-1}{2w}$, and the optimal value is $k - 1$.

According to the encoding rules of EVENODD, we know that the number of XOR-summing operations is $k - \frac{1}{2}$ for each parity element. For RDP, when $k + 1$ and $k + 2$ are prime numbers, the number of XOR-summing operations achieves optimal value $k - 1$. However, we can know from the experiments of Plank, in some cases, Liberation code outperforms RDP [22]. In terms of encoding performance, Liberation code is better than other erasure codes, such as EVENODD, RDP. For a data element $d_{i,j}$, the factor which affects the efficiency of modifying operation is the number of ones of column $wi + j$ in the encoding matrices. The number of columns in encoding matrices is kw, so the average of ones for each column is $\frac{2kw+k-1}{kw} = 2 + \frac{k-1}{kw}$, which is approximately equal to 2. From the above analysis, we conclude that encoding and modifying operations performance of Liberation code is optimal.

2.2 Single Disk Failure Recovery

When $k = 5, w = 5$, the encoding rules of Liberation code are shown in Fig. 3. Firstly, we define data elements and the corresponding parity elements. We can know that in the same parity set, parity elements are generated from the data elements by XOR-summing operations. These operations are described in the following equations. As we all know, in Fig. 3, parity disks are P and Q. According to the encoding rules of Liberation code, we can get the following results. For example, $p_0 = d_{0,0} \oplus d_{0,1} \oplus d_{0,2} \oplus d_{0,3} \oplus d_{0,4}$, data elements $d_{0,0}$,$d_{0,1}$,$d_{0,2}$,$d_{0,3}$,$d_{0,4}$ and parity element p_0 are in the same parity set P_0. $P_0 = \{d_{0,0},d_{0,1},d_{0,2},d_{0,3},d_{0,4},p_0\}$. Data elements $d_{0,0}, d_{1,1}, d_{2,2}, d_{3,3}, d_{4,4}$ and parity element q_0 are in the same parity set Q_0. $Q_0 = \{d_{0,0}, d_{1,1}, d_{2,2}, d_{3,3}, d_{4,4}, q_0\}$. According to the encoding rules, we can get the rest of parity sets in the same way.

$$
\begin{aligned}
p_1 &= d_{1,0} \oplus d_{1,1} \oplus d_{1,2} \oplus d_{1,3} \oplus d_{1,4} \\
p_2 &= d_{2,0} \oplus d_{2,1} \oplus d_{2,2} \oplus d_{2,3} \oplus d_{2,4} \\
p_3 &= d_{3,0} \oplus d_{3,1} \oplus d_{3,2} \oplus d_{3,3} \oplus d_{3,4} \\
p_4 &= d_{4,0} \oplus d_{4,1} \oplus d_{4,2} \oplus d_{4,3} \oplus d_{4,4} \\
q_0 &= d_{0,0} \oplus d_{1,1} \oplus d_{2,2} \oplus d_{3,3} \oplus d_{4,4} \\
q_1 &= d_{1,0} \oplus d_{2,1} \oplus d_{3,2} \oplus d_{3,3} \oplus d_{4,3} \oplus d_{0,4} \\
q_2 &= d_{2,0} \oplus d_{2,1} \oplus d_{3,1} \oplus d_{4,2} \oplus d_{0,3} \oplus d_{1,4} \\
q_3 &= d_{3,0} \oplus d_{4,1} \oplus d_{0,2} \oplus d_{1,3} \oplus d_{1,4} \oplus d_{2,4} \\
q_4 &= d_{4,0} \oplus d_{0,1} \oplus d_{0,2} \oplus d_{1,2} \oplus d_{2,3} \oplus d_{3,4}
\end{aligned}
$$

Liberation code uses conventional method to recover from a single disk failure. The recovery process is described as follows. It uses the surviving elements in a P_i parity set to recover the lost data element through XOR-summing operations. The remaining data elements in the failed disk are recovered by the

same method. In the following, we will describe conventional method of single disk failure recovery in detail.

In Fig. 3, we assume that D_0 fails, and the corresponding data $d_{1,0}$,$d_{2,0}$,$d_{3,0}$,$d_{4,0}$, $d_{0,0}$ are lost. For conventional recovery method, we need $d_{0,1}$, $d_{0,2}$, $d_{0,3}$, $d_{0,4}$, p_0 and through XOR-summing $\{d_{0,1},d_{0,2},d_{0,3},d_{0,4},p_0\}$ to restore $d_{0,0}$. Using this method, the data $d_{1,0}$ is recovered by XOR-summing the elements $\{d_{1,2}$, $d_{1,3}$, $d_{1,4}$, $d_{1,1}, p_1\}$. The data $d_{2,0}$ is recovered by XOR-summing the elements $\{d_{2,2},d_{2,3},d_{2,4},d_{2,1},p_2\}$. The data $d_{3,0}$ is recovered by XOR-summing the elements $\{d_{3,2},d_{3,3},d_{3,4},d_{3,1},p_3\}$. The data $d_{4,0}$ is recovered by XOR-summing the elements $\{d_{4,2},d_{4,3},d_{4,4},d_{4,1}, p_4\}$. In this case, for data elements recovery, the total number of data and parity elements read is 25. For conventional recovery method, the number of elements read is w^2 in a recovery process.

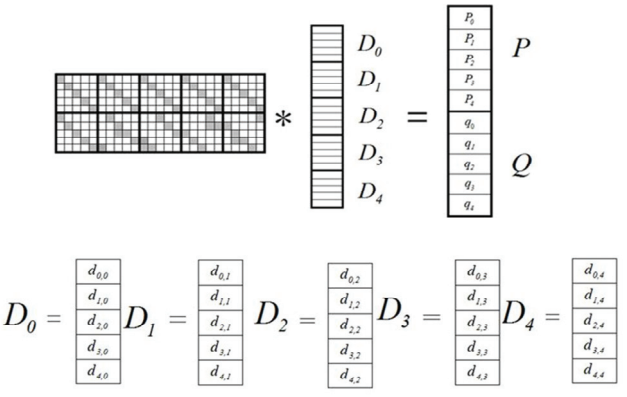

Fig. 3. Encoding rules of Liberation code when *k*=5 and *w*=5.

The conventional recovery method only uses parity set P_i and recovers the lost data by XOR-summing the surviving elements in P_i parity sets. According to the above analysis of encoding rules of Liberation code, a data element is in different types of parity sets, P_i or Q_j. The conventional recovery method ignores this fact, so it has to read more data. We observe that there are overlapping elements when some lost data elements are recovered by using P_i parity sets and the rest of w data elements are recovered by using Q_j parity sets. Therefore, in a disk failure recovery process, the common elements only need to be read once from the disks. The total number of elements read can be reduced. In the next section, we will describe our failure recovery scheme HRSF.

3. Hybrid Recovery Method of Single Disk Failure

From the introduction of encoding rules and parity sets of Liberation code in Section 2, we can find that. (1) Between each pair of parity sets P_i and P_j ($i \neq$j), there is no overlapping element. (2) There are overlapping elements between each pair of parity sets P_i and Q_j. The first parity set Q_0 and any parity set P_i have one overlapping element. The rest of parity sets Q_j $(1 \leq j \leq w-1)$ and the only one special parity set P_i have two overlapping elements. Except for the special parity set P_i, the remaining $(w-1)$ P_i and the parity set Q_j $(1 \leq j \leq w-1)$ have one overlapping element. (3) In order of j$(0 \leq j \leq w-1)$, for the pair of parity sets Q_j and Q_{j+1}, there is only one overlapping element between each pair of Q_0 and

Q_1, Q_1 and Q_2, Q_2 and Q_3, $\cdots$, Q_{w-2} and Q_{w-1}.

From the above analysis, we find that if we use two types of parity sets P_i and Q_j in a disk failure recovery process, the number of elements read can be reduced. It is because that the overlapping elements only need to be read once from the disks. Our target is to choose w parity sets to recover the lost data elements and hence the number of elements read during the recovery can be minimized. In this way, we can reduce the recovery overhead and speed up the recovery process.

Assume that t lost elements are recovered using P_i parity sets. The remaining $(w-t)$ elements are recovered from Q_j parity sets. From the following formulas, we can obtain the number of overlapping elements.

(i) Through the above analysis of parity sets, we know that when there is an overlapping element between t P_i and $-t$ Q_j, the number of overlapping elements is calculated as follows.

$$\begin{aligned}(w-t)t+\big((w-t)-1\big) &= -t^2+t(w-1)+(w-1) \\ &= -\left(t-\frac{w-1}{2}\right)^2+\left(\frac{w-1}{2}\right)^2+(w-1)\end{aligned} \tag{1}$$

When $t=\frac{w-1}{2}$, the number of overlapping elements is maximized.

(ii) When there are two overlapping elements between t P_i and $-t$ Q_j, we should add the number of overlapping elements between the pair of parity sets Q_j and the special parity set P_i in the above formula. The number of overlapping elements is described as follows.

$$(w-t)t+\big((w-t)-1\big)+(w-t-1)=-t^2+(w-2)t+(2w-2) \tag{2}$$

Because w must be a prime and >2, for this case, the number of overlapping elements is maximized when $t=\frac{w-1}{2}$.

From the above analysis, we can get the following results. When $t=\frac{w-1}{2}$, the number of overlapping elements is maximized. From the analysis of parity sets, we can know that in order of $0\le j\le w-1$, there is a common element between each pair of continuous parity sets Q_j. For the lost data, when the first $\frac{w-1}{2}$ elements are recovered from P_i and the remaining $w-\frac{w-1}{2}=\frac{w+1}{2}$ elements are recovered form Q_j, the number of overlapping elements is maximized.

In this paper, we discuss the situation when $k=w$. For our hybrid recovery scheme of single disk failure, the total number of elements read in a recovery process is obtained from formula (3). The first item in formula (3) is the number of elements read for recovering the first $\frac{w-1}{2}$ lost elements. The number of elements read for the remaining $\frac{w+1}{2}$ lost elements recovery is described in the last two items.

$$\begin{aligned}w\tfrac{(w-1)}{2}+\left(w+1-\tfrac{w-1}{2}\right)+(\tfrac{w+1}{2}-1)(w+1-\tfrac{w-1}{2}-1) \\ =\frac{3}{4}w^2+\frac{5}{4}\end{aligned} \tag{3}$$

As we all know, for conventional method, the lost elements recovery operations need P_i parity sets. The number of elements read is w^2 in the recovery process.

The number of elements read of our proposed method, which adopts the P_i and Q_j hybrid recovery is

reduced by $\frac{1}{4}+\frac{5}{4w^2}$ compared to the conventional recovery method.

For example, in Fig. 3, $k=5$ and $w=5$. When disk D_0 fails, P_0 and P_1 are used to recover $d_{0,0}$ and $d_{1,0}$ separately. The elements $d_{2,0}$, $d_{3,0}$, $d_{4,0}$ are recovered from Q_2, Q_3, Q_4 separately. For the failure recovery, the number of elements read from disks is 20. However, for the conventional method, it is 25. The number of elements that need to be read can be reduced by 20%. Likewise, when $k=7$, $w=7$, the number of elements read from disks of hybrid recovery scheme is 38. For the conventional method, it is 49. The number of elements that need to be read can be reduced by 22.4%. In a single disk recovery process, our method can effectively reduce the number of elements read, and further speed up the recovery process.

For any k, w, $w=k$, when D_m $(0 \le m \le k)$ fails, our hybrid recovery scheme can be described in Algorithm 1.

Algorithm 1. Hybrid recovery scheme of single data disk failure for Liberation code

Assume that the data disk D_m fails.

(1) Based on the value of w, k, calculate the number of lost elements on D_m.

(2) For the first $\frac{w-1}{2}$ lost data elements, use P_i parity sets to recover, and recover the element by XOR-summing all surviving elements in P_i.

(3) For the last $\frac{w+1}{2}$ lost data elements, use Q_j parity sets to recover, and recover the element by XOR-summing all surviving elements in Q_j. If an element that need to be read from the disk has be stored in memory, there is no need to read the overlapping element.

(4) If the lost data elements in D_m have been recovered, they are written back to an available disk.

(5) When the w lost elements are successfully restored, the recovery process can move on to recover the next stripe.

4. Performance Evaluation

In order to evaluate the performance of our proposed recovery scheme HRSF, in this section, we conduct extensive experiments to compare HRSF with conventional method. In a disk recovery process, for these two schemes, we compare the total amount of data read from disks and recovery time by setting different number of disks and chunk sizes. Then we choose five workloads, financial1, financial2, websearch1, websearch2, websearch3 from several international enterprise data centers [23,24], and evaluate the recovery time under these workloads.

4.1 Experiment Settings

We use a popular simulator to perform experiments, DiskSim [25], which was developed by Carnegie Mellon University. The simulated disks are 15000-RPM, 146 GB. Recovery process is in an offline mode, and there is no request from the front-end access. The data is stored on disks according to the corresponding encoding rules. We suppose that each element consists of a data chunk, and the lost data is recovered stripe by stripe. Similar to conventional recovery method, HRSF reads all the required data chunks of a stripe, and then puts them in memory. If all lost data elements are recovered, they are

immediately written to an available disk. In order to evaluate recovery performance, we use the data recovery time per MB of data as a metric. For each data disk, we assume that the failure probability is equal, and implement the program for 6 times. We conduct a failure recovery process on all data disks, and then use the average as final experimental result.

4.2 The Number of Elements Read During Recovery

When a disk fails, we use conventional recovery method and our hybrid recovery scheme to recover lost data respectively, and then compare the number of elements read in a recovery process. As we can see in Fig. 4, when k is increased, the hybrid recovery scheme HRSF reduces the amount of data read from disks by approximately 25% compared to conventional method. The results are consistent with our expectations. The amount of data read can be effectively reduced by using HRSF.

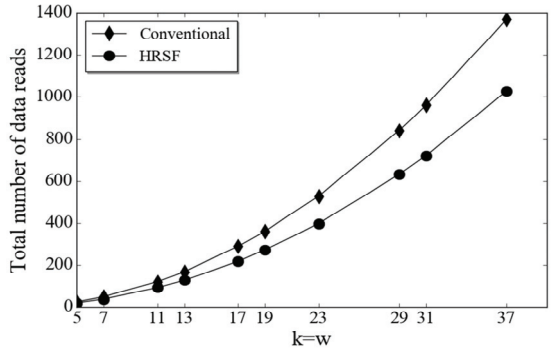

Fig. 4. The number of elements read from disks.

4.3 Performance Evaluation under Different Chunk Sizes

In distributed storage systems, the size of a chunk is usually larger. Therefore, we evaluate the recovery performance under different chunk sizes, which is from 256 kB to 8196 kB. The number of disks is fixed, which is $k = w = 5$, $k = w = 11$, $k = w = 17$, respectively. For conventional recovery method and our hybrid recovery scheme, we use the recovery time overhead per MB to evaluate recovery performance.

Fig. 5 shows the recovery time of the conventional and hybrid recovery scheme. When the chunk size is increased from 256 kB to 8196 kB, HRSF outperforms the conventional method consistently. It has less time overhead during a recovery process. As the size of chunk increases, we find that the recovery time is reduced. We can predict that for large chunk size, the failure recovery time will tend to be stable.

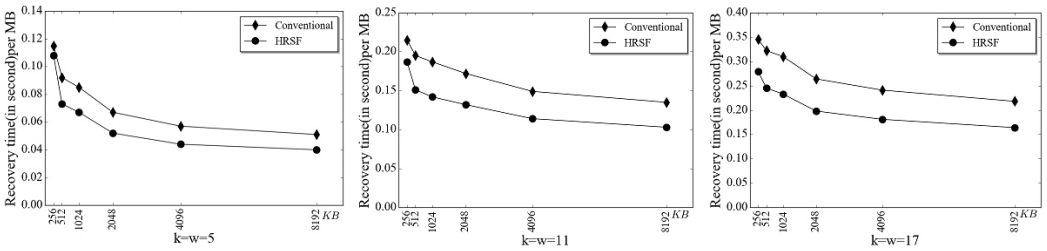

Fig. 5. The impact of chunk size on recovery performance.

4.4 Performance Evaluation under Different Number of Disks

For different number of disks, we analyze the impact of these situations on recovery performance in this subsection. When the number of disks is 5, 7, 11, 13, 17, we perform experiments. The chunk size is fixed, which is 256 kB, 512 kB, and 1024 kB, respectively.

Fig. 6 shows the recovery time of these two schemes and the improvement of hybrid recovery scheme over conventional recovery method. The results show that the recovery time of our scheme is reduced by 5.80%–19.1% when chunk size is 256 kB. When chunk size is 516 kB, the recovery time of our scheme is reduced by 20.9%–23.9%. When chunk size is 1024 kB, the recovery time of our scheme is reduced by 21.5%–24.9%. We know that the number of elements read from disks is reduced by approximately 25% in subsection 4.2. However, in this subsection, the percentage reduction in amount of data read does not match to the percentage reduction in recovery time. This is due to the disk access pattern of our scheme, it needs more time for additional disk seeks.

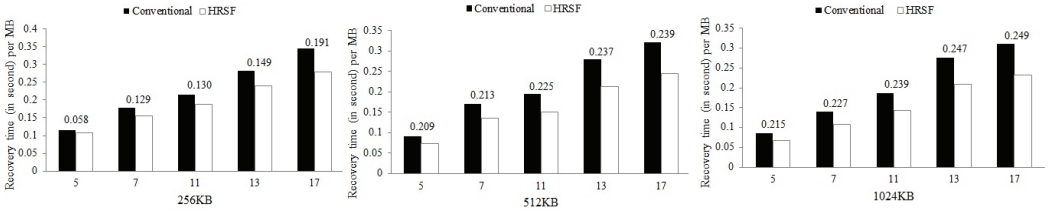

Fig. 6. The impact of disks number on recovery performance.

4.5 Performance Evaluation under Different Workloads

We perform experiments and analyze the impact of different workloads on recovery performance in this subsection. We fix the number of disks, $k = w = 11$. The fixed chunk size is 512KB.

Fig. 7 shows that compared with conventional method, the recovery time of HRSF is reduced by 22.3% in offline mode. For all workloads, the recovery time of our scheme is reduced by 22.7%–24.9%. As we can see from experimental results, using conventional method to recover lost data, the failure recovery time of online modes is longer than offline mode. This is because for online modes, applications and recovery I/O may use disk bandwidth at the same time. For the conventional recovery process, there is a large amount of data reading, which leads to a slower recovery speed. Like the conventional method, our scheme still needs to read data from disks during recovery. However, the amount of data read during the failure recovery process is minimized. Therefore, recovery speed is faster than conventional method.

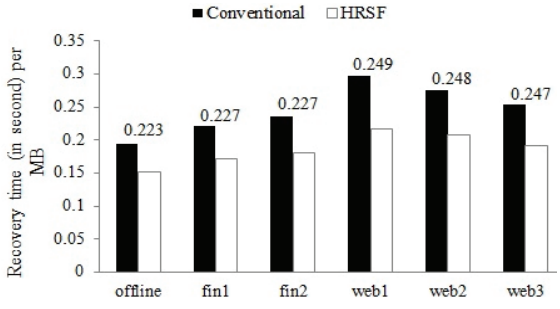

Fig. 7. Recovery performance under different workloads.

5. Conclusion and Further Work

For Liberation code based storage systems, in this paper, we present a single disk failure recovery scheme HRSF. HRSF can choose optimal parity sets. Using this scheme, we can minimize the number of elements read from surviving disks to reduce the failure recovery time. We obtain the lower bound of elements read and then describe our recovery scheme. Theoretical analysis and experimental results show that HRSF outperforms conventional recovery method in terms of the amount of data read and failure recovery time. In this paper, we present the recovery method when $k=w$. The discussion of other situations will be our future work. In addition, we will evaluate the performance of our scheme in a practical environment. In the case of two disks failure, decoding algorithm of Liberation code is complex. The recovery performance of it is still inefficient. Therefore, in our future work, we will try to address these problems.

Acknowledgement

This work was funded by the NSF China Project (No. 61472067) and the Science and Technology Project of Sichuan Province (No. 2013GZ0006).

References

[1] M. Deng, Z. Chen, Y. Du, N. Xiao, and F. Liu, "Erasure codes in big data era," in *Proceedings of International Conference on Control, Automation and Information Sciences (ICCAIS)*, Gwangju, Korea, 2014, pp. 218-223.

[2] X. Pei, Y. Wang, X. Ma, and F. Xu, "A decentralized redundancy generation scheme for codes with locality in distributed storage systems," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 8, article no. e3987, 2017.

[3] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-diagonal parity for double disk failure correction," in *Proceeding of the 3rd USENIX Conference on File and Storage Technologies*, San Francisco, CA, 2004, pp. 1-14.

[4] 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.

[5] 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.

[6] Y. Fu, J. Shu, X. Luo, Z. Shen, and Q. Hu, "Short code: an efficient RAID-6 MDS code for optimizing degraded reads and partial stripe writes," *IEEE Transactions on Computers*, vol. 66, no. 1, pp. 127-137, 2017.

[7] 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.

[8] L. Xiang, Y. Xu, J. Lui, and Q. Chang, "Optimal recovery of single disk failure in RDP code storage," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 1, pp. 119-130, 2010.

[9] Z. Wang, A. G. Dimakis, and J. Bruck, "Rebuilding for array codes in distributed storage systems," in *Proceedings of 2010 IEEE Globecom Workshops*, Miami, FL, 2010, pp. 1905-1909.

[10] Z. Shen, J. Shu, P.P. Lee, and Y. Fu, "Seek-efficient I/O optimization in single failure recovery for XOR-coded storage systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 877-890, 2017.

[11] Z. Shen, P.P. Lee, J. Shu, and W. Guo, "Encoding-aware data placement for efficient degraded reads in XOR-coded storage systems," in *Proceedings of IEEE 35th Symposium on Reliable Distributed Systems*, Budapest, Hungary, 2016, pp. 239-248.

[12] P. Nakkiran, K. V. Rashmi, and K. Ramchandran, "Optimal systematic distributed storage codes with fast encoding," in *Proceedings of IEEE International Symposium on Information Theory*, Barcelona, Spain, 2016, pp. 430-434.

[13] M. Itani, S. Sharafeddine, and I. Elkabbani, "Practical single node failure recovery using fractional repetition codes in data centers," in *Proceedings of IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, Crans-Montana, Switzerland, 2016, pp. 762-768.

[14] B. Sung and C. Park, "Fast reconstruction for degraded reads and recovery process in primary array storage systems," *IEICE Transactions on Information and Systems*, vol. 100, no. 2, pp. 294-303, 2017.

[15] O. Khan, R. C. Burns, J. S. 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)*, San Jose, CA, 2012.

[16] O. Khan, R. C. Burns, J. S. Plank, and C. Huang, "In search of I/O optimal recovery from disk failures," in *Proceedings of the 3rd USENIXCconference on Hot Topics in Storage and File Systems*, Portland, OR, 2011.

[17] Y. Zhu, J. Lin, P. P. Lee, and Y. Xu, "Boosting degraded reads in heterogeneous erasure-coded storage systems," *IEEE Transaction on Computers*, vol. 64, no. 8, pp. 2145-2157, 2015.

[18] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: a study on the Facebook warehouse cluster," in *Proceedings of the 5th USENIX Conference on Hot Topics in Storage and File System*, Berkeley, CA, 2013, pp. 8-13.

[19] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A "hitchhiker's" guide to fast and efficient data reconstruction in erasure-coded data centers," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 331-342, 2015.

[20] Z. Shen, J. Shu, and P. P. Lee, "Reconsidering single failure recovery in clustered file systems," in *Proceedings of 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, Toulouse, France, 2016, pp. 323-334.

[21] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "XORing elephants: novel erasure codes for big data," *Proceedings of the VLDB Endowment*, vol. 6, no. 5, pp. 325-336, 2013.

[22] J. S. Plank, "The RAID-6 Liber8Tion Code," *The International Journal of High Performance Computing Applications*, vol. 23, no. 3, pp. 242-251, 2009.

[23] D. Narayanan, A. Donnelly, and A. Rowstron, "Write off-loading: practical power management for enterprise storage," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST)*, San Jose, CA, 2008, pp. 253-267.

[24] S. Wu, H. Jiang, D. Feng, L. Tian, and B. Mao, "Workout: I/O workload outsourcing for boosting RAID reconstruction performance," in *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST)*, San Francisco, CA, 2009, pp. 239-252.

[25] The DiskSim simulation environment (v4.0) [Online]. Available: http://www.pdl.cmu.edu/DiskSim/index.shtml.

Jun Li https://orcid.org/0000-0002-7383-5864

She is currently working toward the PhD candidate of computer science in University of Electronic Science and Technology of China. Her current research interests include distributed storage system, data recovery and cloud storage.

Mengshu Hou https://orcid.org/0000-0002-5283-7318

He received his Ph.D. degree in computer science and engineering from the University of Electronic Science and Technology of China in 2005. He is a full professor and PhD supervisor in University of Electronic Science and Technology of China. His research interests include wireless sensor networking, distributed storage and mobile computing.