An Extension of RDP Code with Parallel Decoding Procedure

[†]Jun Feng, [†]Yu Chen[*], [†]Douglas Summerville, [‡]Zhou Su

[†]State University of New York – Binghamton, Binghamton, NY 13902, USA

[‡]Waseda University, Ohkubo 3-4-1, Shinjyuku, Tokyo 169-8555, Japan

Abstract **– XOR based RAID 6 systems outperform other RAID systems. Among the XOR (exclusive OR) based RAID-6 schemes, RDP has better performance than others by a narrow margin. However, the RDP code scheme cannot take full advantage of parallel hardware implementation of XOR codes. In this paper, we propose an extension of the double–erasure-correcting RDP code called EDP, which consists of a parallel decoding scheme. Thus, EDP can improve the decoding velocity of RDP by about 40% without any change to the current RDP configuration for storage.**

Keywords: XOR, Erasure Code, Fault Tolerate, Storage System.

1. Introduction

In modern storage systems, RAID (Redundant Array of Independent Disks) techniques are known to be the preferable ones that achieve higher performance and more reliability. RAID systems protect the data against disk failures by constructing redundant information and storing them on an array of hard disks. RAID-6, which can tolerate two failure-disks, has become more popular.

Some RAID codes have been successfully designed to recover double storage node failures, including Reed-Solomon codes, EVEN-ODD [1], Row Diagonal Parity (RDP) [2] and Liberation codes [5]. X-code [7] an elegant two-erasure code, is not a RAID code. It does not meet the RAID-6 specification of having two independent parity devices, *P* and *Q* [5]. A recent examination on the performance of the codes for RAID-6 had concluded that special purpose RAID-6 codes vastly outperform their general purpose counterparts such as Reed-Solomon code, and RDP performs the best of these by a narrow margin [6].

However, the RDP code cannot take full advantage of parallel implementation of the multi-processer. For example, RDP code can start two iterative procedures at the beginning, but one of them has to stop quickly when it hits the block in the "missing" chain.

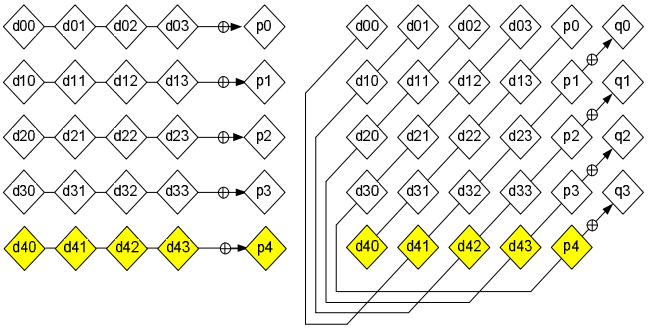
This paper proposes EDP scheme (an Extension of RDP). It is an improved decoding method based on RDP. When implemented in parallel, the decoding velocity of EDP scheme can be improved by about 40% in theory without changing any disk configuration for RDP code. The only additional cost is to recover a missing parity in one extra step. However, after the step the EDP scheme can be effectively implemented in parallel on multi-processer or reconfigurable hardware devices.

The rest of this paper is arranged as follows. Section 2 provides background on RDP code. It focuses on the decoding algorithm that leads to the EDP scheme. Section 3 gives EDP decoding scheme. Section 4 presents experimental results. Section 5 concludes the paper.

2. RDP Code

RDP uses the two slopes 0 and 1 to construct the parity devices *P* and *Q*. Figure 1(a) shows how parity blocks on the *P* device are created by data blocks, and Figure 1(b) shows the generation of parity blocks on the *Q* device. In Figure 1(b), the yellow colored blocks are imaginary blocks.

In order to avoid confusion and inconsistency, we equal "*n*" to a prime number, the number of data disks is *n*-1 by default. If it is less than *n*-1, we can suppose there are imaginary data disks to form the *n*-1 data disks. Each disk has *n*-1 blocks. So the storage can be considered at as *n*-1 rows × *n*+1 columns array. One column stands for a disk. We use "*i*" to specify the rows and "*j*" to the columns. If two disks failed, we mark them as "*a*" and "*b*", where $a < b$. We call the set of blocks in one line to generate parity *x* the "chain *x*". For example, the set of $\{q_0, p_1, d_{23}, d_{32}, d_{00}\}$ is chain q_0. There are five parity chains for the *Q* device generation, but there are only four parity blocks in the *Q* device. Since one of the parity blocks cannot be stored in the *Q* device, the corresponding chain



(a) P device construction. **(b) Q device construction.**

Figure 1. RDP encoding procedure (n=5).

* Manuscript submitted on July 8, 2011 to *the 9th IEEE Consumer Communications and Networking Conference* (CCNC 2012), Las Vegas, NV, USA, Jan. 14 – 17, 2012. Corresponding author: Yu Chen, Dept. of Electrical & Computer Eng., SUNY–Binghamton, Binghamton, NY 13902. E-mail: ychen@binghamton.edu, Tel.: (607) 777-6133, Fax: (607) 777-4464.

is called the "missing chain". There is no restriction on which diagonal should be selected to store parity blocks. By default, diagonal $n-1$ is the missing diagonal.

The idea of the decoding process will be illustrated by an example. Assume that data devices 1 and 3 are erased. In this example, two starting chains, q_0 and q_2, can be found. The two missing blocks d_{23} and d_{11} are reconstructed immediately. Then, the row parity can be used to recover two more missing blocks d_{21} and d_{13} in the two rows.

Then, because the block d_{13} lies on the missing chain, it cannot start the next recovery process. It has to stop. The remaining failed blocks have to be recovered by another iterative process one by. Figure 2 illustrates that the sequence for the recovery procedure.

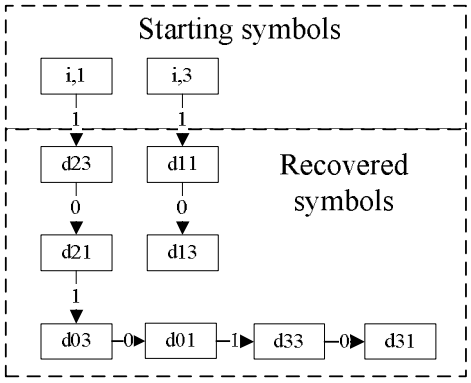


Figure 2. Illustration of the RDP decoding procedure.

Figure 2 also shows that when one of them hits the missing parity chain and stops another chain can traverse the remaining erased data blocks. Since a recovery time depends on the long chain, if each of the two iterative processes recovers half of the erased blocks in parallel, the recovery process can achieve the highest decoding efficiency. Based on this observation, we proposed the EDP method that achieves the highest speed by making each chain recover half of the lost blocks in parallel.

3. EDP Decoding Scheme

EDP decoding scheme can be described as "accelerator plus the RDP decoding algorithm". The advantage this scheme offers is that the current RDP based systems can be upgraded to the new EDP based systems smoothly without any change, which is preferable for users.

In this section, an overview of various missing data scenarios is first introduced, and then the theoretical foundations of the EDP code are discussed. In subsection 3.3, the detailed EDP decoding algorithm is presented.

Since it is trivial to recover a single failed column, this section describes how to recover two failed data devices which is most likely to occur. It is infeasible to recover the failed blocks separately. The scheme consists of two steps.

Step #1: calculates the accelerator:

$$q_{n-1} = \bigoplus_{i=0}^{n-2} q_i \qquad (1)$$

where q_{n-1} is the parity of the missing chain.

Step #2: Run the RDP decoding procedure.

Below is a short proof of Equation (1).

Proof:

Let's define $q_i = Q_i \oplus p_{i+1}$, where Q_i is the parity of the data blocks in the chain. Then, $q_0 = Q_0 \oplus p_1, \ldots q_{n-1} = Q_{n-1} \oplus p_n$.

Suppose all imaginary data blocks are zero. Then their parity result p_{n-1} is zero. $\bigoplus_{i=0}^{n-1} p_i$ is equal to $\bigoplus_{i=0}^{n-1} Q_i$ because both of them are XOR result of all data blocks. Then:

$$\bigoplus_{i=0}^{n-2} q_i = (\bigoplus_{i=0}^{n-2} Q_i) \oplus (\bigoplus_{i=1}^{n-1} p_i)$$

$$= (\bigoplus_{i=0}^{n-1} Q_i) \oplus (\bigoplus_{i=0}^{n-1} p_i) \oplus Q_{n-1} \oplus p_0 = Q_{n-1} \oplus p_0 = q_{n-1}$$

For example, in Figure 1, we have

$q_0 \oplus q_1 \oplus q_2 \oplus q_3$

$= Q_0 \oplus p_1 \oplus Q_1 \oplus p_2 \oplus Q_2 \oplus p_3 \oplus Q_3 \oplus p_4$
$= (Q_0 \oplus Q_1 \oplus Q_2 \oplus Q_3) \oplus (p_1 \oplus p_2 \oplus p_3 \oplus p_4)$
$= (Q_0 \oplus Q_1 \oplus Q_2 \oplus Q_3 \oplus Q_4)$
$\oplus (p_1 \oplus p_2 \oplus p_3 \oplus p_4 \oplus p_0) \oplus Q_4 \oplus p_0$
$= Q_4 \oplus p_0$
$= q_4$.

Theorem. *Assume there are two columns a and b in the array, where $0 \le a < b \le n-1$. The two iterative chains start from (n-1, a) and (n-1, b) independently, and then take slope 1 and slope 0 alternatively. Using the same speed, each of the chains covers half of the lost blocks when they encounter. The union of the two sequence chains traverses all the failed blocks and each of them traverse half of the failed blocks.*

Based on the theorem, the two iterative chains will only encounter in the middle, either of them traverses half of the failed blocks. Therefore, EDP can improve the recovery speed in an optimal way. Another advantage of the EDP algorithm is that each chain can be processed independently without sharing any information. This is preferable for parallel programming with multi-core or hardware implementation.

Due to the limited space, the detailed proof of the theorem is not included. Interested readers are referred to our technical report [8].

4. Experiment

There are many factors that influence the performance, including memory utilization, cache size, CPU speed, disk type, OS type, data read/write type, word size, thread

priority, and specific coding/decoding algorithm implementation. The bottleneck can be any one or any combination of them. It is extremely difficult, if not impossible, to compare all the combinations.

4.1 Setup

The experimental results could be more persuasive if the data were read from and written into different disks. However, in the current RAID systems, the bottleneck is the disk IO speed which has existed for more than 20 years and beyond our paper's scope. Moreover, there are many uncontrolled factors when transferring data between machines and disks [6]. Our focus is the performance of the decoding algorithm. Therefore, our experiments have been conducted in a single machines' memory, just as was done in [6]. Otherwise, it would be too difficult to make the result convincible. Actually, to the best of our knowledge, there are no such practical experimental results reported in the literature of any codes including EVENODD [1], RDP [2], X-code [7], STAR code [4], Weaver code [3] etc.

Table 1. Machine Configuration.

Brand	HP6120t	DELL760	HP6340f
Intel CPU	E5300	E8400	Q8400
Frequency	2.6GHz	3GHz	2.6GHz
Cores	2	2	4
Memory	3GB	3GB	8GB
L1 Cache	2×32KB	2×32KB	4×32KB
L2 Cache	2MB	6MB	4MB
OS	WindowsXP 32-bit	WindowsXP 32-bit	Windows7 64-bit

Since our focus is the performance of the decoding procedure, the influences of other factors should be negligible or be reduced to a negligible level. As we only consider the operation in the memory, for the disk type, the disk utilization factor is negligible. Regarding the CPU speed, because the timestamp counter in each core cannot be guaranteed to be strictly synchronous in a multi-core platform, the RTDSC (read time stamp counter) function is not considered accurate, instead, functions QueryPerformanceCounter() and QueryPerformance Frequency() are selected to get accurate relative timing to cancel the speed influence. Three machines with different configurations were used to conduct comparison experiments to reduce the influence of the cache size. Details of the three machines are listed in Table 1.

The program was executed in user space and no other user programs were executing. For all of the experiments, each test case ran on the three machines. On each machine, each test case was also repeated 10 times and the average results were calculated. Every "repeat" includes "run, exit, and restart the test program again". In the experiments, the memory data transfer and memory XOR operation were used as a baseline. Then the influence of cache and thread priority were studied. The performance of the EDP decoding algorithm has been compared with RDP based on all the experimental results.

The memcpy() speed and an XOR speed of each machine have been recorded. For HP6120t, they are 1.24 GB/s and 1.43 GB/s. For DELL760, they are 1.97GB/s, and 1.67GB/s. For HP6340f, they are 2.22GB/s and 1.54GB/s, respectively. The memory copy speed is measured by copying 768 MB data from one buffer to another. XOR speed is measured by the following steps:

- The initial value is 0xaaaaaaaa;
- XOR with the value 0xffffffff in serial directly with 2 million times;
- Loop 100 times.

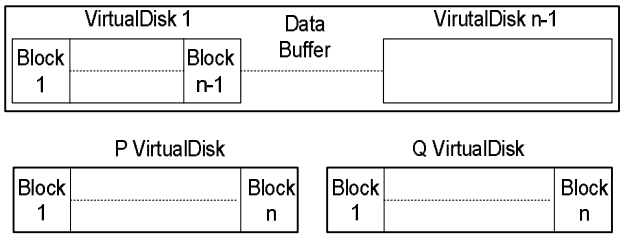


Figure 3. Top view of memory allocation.

For the following experiment, three buffers were allocated in memory to simulate data *P* and *Q* disks separately. Data buffer was partitioned into *n*-1 virtual disks, and each disk had *n*-1 blocks. Each parity buffer (*P* or *Q*) was partitioned into *n* blocks. The high-level view of the setup is shown in Figure 3.

4.2 Cache

Cache miss impacts the performance significantly. If the data is not in cache, it has to be fetched from memory or disk. It leads to extra time cost. Therefore, each data

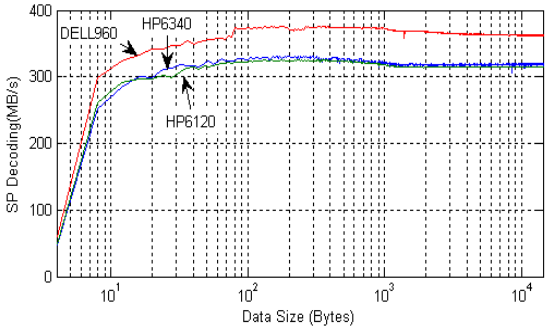


Figure 4. Decoding Speed on different Cache Size (SP).

block size was expanded from 4B to 4MB with step size of 16 bytes to measure the influence of cache size. For convenience, the maximum data disk number was fixed to 13 because of the memory allocation limitation.

Figure 4 shows the EDP performance of the three machines as the data block size changes. SP (Single-core Processor) indicates the case when the decoding process was running on only one processor core. MP (Multi-core Processor) means that the decoding process was running on multi-cores in parallel, only two threads can be started at the same time. MP also implies that the decoding process is running on two processor cores.

As the data block size increases, the decoding speed remains within a reasonable range. For small data block size, the total data buffer is from ten to hundreds bytes. This leads to inaccurate results because of the cold start of the program. Due to the combination of inaccurate result and some initial timing cost, the final result is not precise for small data block size. In addition, the curve is not monotonically increasing and strictly stable. The reasons include cache misses and collisions between cache entries, which are random and unpredictable.

Figure 5 shows the difference of the SP decoding speed on other machines compared to that on the HP6120t. The upper curve shows that the DELL 760 performs the best since it has the biggest cache size, which implies the least probability of a cache miss. HP6340 achieved performance similar to HP6120t.

4.3 Thread Priority

Windows is a time-slotted OS. Thus, for any cores, the threads or processes are scheduled and executed in different time slots. Higher priority threads will have a higher probability of being executed. In order to find the influence, an SP decoding process was executed with different priority level. The normal priority level is set to 15 and high priority level is set to 31 by SetPriorityClass() and SetThreadPriority() functions. The data block size was set to 128KB, 256KB and 512KB since they can be divided by 512Byte or 4KB, which is the disk sector size. The prime number n is changed from 5 to 29.

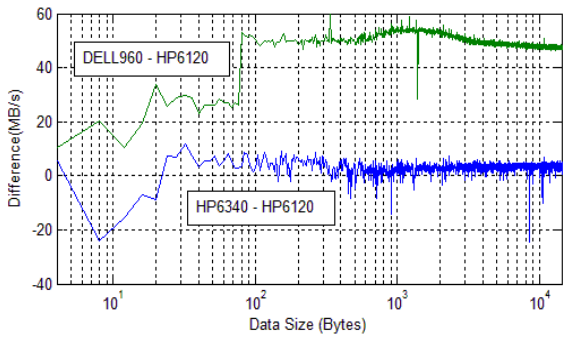


Figure 5. Coding Speed Difference in Different Machine.

Figure 6 shows the influence of thread. Three lines for each machine stand for results with 128KB, 256KB and 512KB data block sizes respectively. It is obvious that the priority factor does not have much impact on performance in our environment in the same machine. Based on this, in the other test cases, we set the priority to the high level.

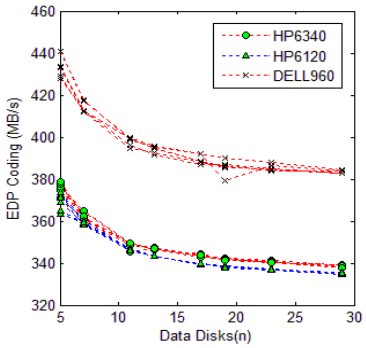


Figure 6. Thread Priority (EDP).

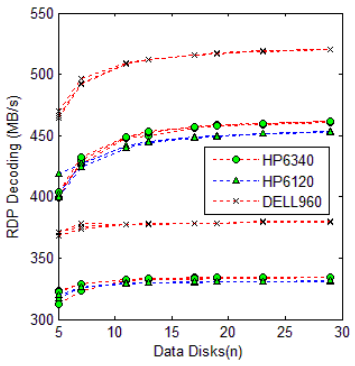
4.4 Coding and Decoding Performance

After the thread priority and cache analysis, the decoding performance was compared. Since there were no open codes for the RDP, we wrote the code according to reference [3]. Although the results are different from those in other papers [6] the differences are mainly caused by the different configurations and optimization methods. They can be optimized using the same approach and similar results can be obtained.

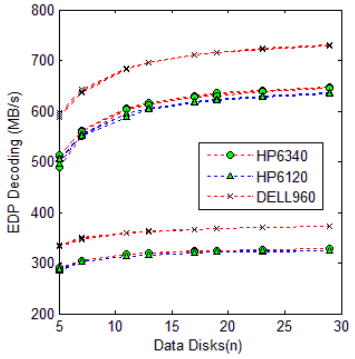
The result is shown on Figure 7. Similarly, the data block size was set to 128, 256, and 512KB. Therefore, there are three lines which stand for results with 128KB, 256KB and 512KB data block sizes respectively on each machine. The prime number was in the range from 5 to 29. Figure 7(a) and 7(b) show the SP and MP decoding speed of RDP and EDP. MP speed is shown above the legend. SP speed is shown below the legend. Different blocks stand for the results on different machines. The figure confirms that the speed in the multi-core can be faster than that in the single core.

Figure 7(c) shows the ratio of MP to SP. The ratio of EDP is close to 2 and that of RDP is close to 1.4. This means EDP can achieve better performance in MP than RDP does. In Figure 7(c), no matter what the machine is, the ratio is almost the same as the ratio is the relative parameter and the machine factor is cancelled.

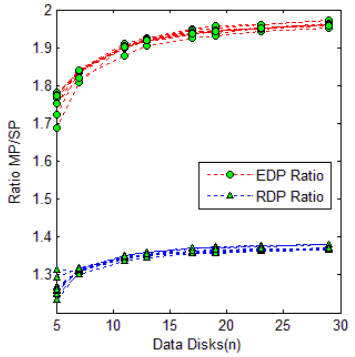
Figure 7(d) shows the speedup of EDP over RDP. The red lines stand for the results on different machines with 128, 256, and 512 KB block sizes. For comparison, the MatLab simulation result is plotted with the color blue. It shows that the experiment results match the simulation quite well. In our experiment, EDP can improve the decoding speed over RDP by about 40% when the data disk number is close to 30.



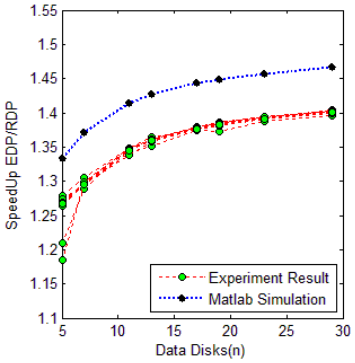
(a) Decoding procedure on MP/SP (RDP).



(b) Decoding procedure on MP/SP (EDP).



(c) Ratio of Speed on dual to that on single processor.



(d) Speedup of EDP over RDP.

Figure 7. Performance Comparison.

5. Conclusions

In this paper, we proposed the EDP scheme, which is an optimal parallel efficient double disk failure recovering algorithm. The EDP scheme is an extension of the RDP code, and requires only n-1 iterative recovery steps in its decoding operations. In comparison with RDP code, the EDP scheme can significantly reduce the data recovery time due to the two optimal parallel iterative chains. Detailed simulations and experiments show that the EDP scheme has the faster decoding process between the existing two iterative recovery chain codes.

EDP algorithm is more suitable for parallel implementation because each thread covers exactly the same amount of failed nodes with the same speed. Along with simplicity and flexibility, this property makes the EDP scheme an ideal candidate for parallel implementation.

Another advantage of the EDP scheme is that it can be used in the current RDP based systems smoothly without any configuration change. Hence, the EDP scheme is very suitable for achieving high availability in practical data storage systems.

References

[1] 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* (44:2), 1995, pp. 192-202.

[2] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong and S.Sankar, "Row-Diagonal Parity for Double Disk Failure Correction," *Proc.of USENIX FAST 2004*, Mar. 31 to Apr. 2, San Francisco,CA, USA.

[3] J. L. Hafner, "WEAVER Codes: Highly Fault Tolerant Erasure Codes for Storage Systems," *the Fourth USENIX Conference on File and Storage Technologies* (FAST '05)', pp. 211-224, 2005

[4] C. Huang and L.H. Xu, "STAR: An Efficient Coding Scheme for Correcting Triple Storage Node Failures," *IEEE Transactions on Computers*, pp. 889-901, 2008

[5] J. S .Plank, "The RAID-6 Liberation Codes," *6th USENIX Conference on File and Storage Technologies* (FAST '08), pp. 97–110, San Jose, CA, Feb., 2008.

[6] J. S .Plank, J. Luo, C. D. Schuman, L. Xu, and Z. Wilcox-O'Hearn, "A Performance Evaluation and Examination of Open-Source Erasure Coding Library for Storage," *7th USENIX Conference on File and Storage Technologies* (FAST '09), San Francisco, CA, February, 2009.

[7] L. Xu and J. Bruck, "X-Code: MDS Array Codes with Optimal Encoding," *IEEE Trans. on Information Theory* (45), 1999, pp. 272--276.

[8] J. Feng, Y. Chen, and D. Summerville, "EDP: An Extension of RDP Code with Optimal Decoding Procedure," *technical report*, Dept. of Electrical and Computer Engineering, Binghamton University, Dec. 2010.