ELSEVIER

Journal of Systems Architecture 46 (2000) 1383–1402

JOURNAL OF SYSTEMS ARCHITECTURE

www.elsevier.com/locate/sysarc

Efficient parity placement schemes for tolerating up to two disk failures in disk arrays

Nam-Kyu Lee [a], Sung-Bong Yang [a,*], Kyoung-Woo Lee [b]

[a] *Department of Computer Science, Yonsei University, 134 Shinchon-dong, Secodaemum-ku, Seoul 120-749, South Korea*
[b] *LG Electronics Inc., Seoul, South Korea*

Received 13 May 1998; received in revised form 10 June 2000; accepted 1 September 2000

Abstract

In order to achieve high reliability in disk array systems, two new schemes using dual parity placement, called DH1 (diagonal–horizontal) and DH2 schemes, are presented. Both DH schemes can tolerate up to two disk failures by using two types of parity information placed in the diagonal and the horizontal directions, respectively, in a matrix of disk partitions. DH1 scheme can reduce the occurrences of the bottleneck problem significantly because the parity blocks are evenly distributed throughout the disk array. DH2 scheme uses one more disk than DH1 scheme in order to store the horizontal parities, while the diagonal parities are placed in the same way as in DH1 scheme with a minor change. Even though both DH schemes use almost optimal disk space for storing the redundant information, the encoding algorithms for them are quite simple and efficient. Moreover, both DH schemes can recover rapidly from any two disk failures.

Keywords: Disk arrays; DH schemes; Parity placement methods; Fault tolerence; Two disk failures

1. Introduction

In order to construct an optimal storage system, it should provide both high performance and high reliability. A disk array with a group of disks is able to improve I/O performance by accessing several disks in parallel [1–3]. To achieve the more performance improvement in disk arrays, many researches have been done such as the optimal stripe unit size [4], data striping and parity placement [5,6], and data prefetching and catching [7]. However, the reliability issue should be more seriously taken into account than I/O performance, because even a single disk failure may cause a catastrophic result to the overall disk array system [8]. In order to increase the reliability of the system, some redundant information, called *parities*, on the data in a disk array should be *encoded*. And the parities should be placed in the disk array along with the data for recoveries from disk failures.

Nowadays, a disk array system usually has a large group of disks, each of which has an enlarged capacity owing to the current technology. However, as the number of disks in a system increases, so does the

* Corresponding author. Tel.: +82-02-3612716; fax: +82-02-3652579.
E-mail address: yang@mythos.yonsei.ac.kr (S.-B. Yang).

1383-7621/00/$ - see front matter
PII: S1383-7621(00)00031-X

probability of the occurrence of a disk failure. Therefore, disk failures in a disk array may occur more frequently in a disk array system with a large number of disks [1,9–11], that is, once a disk fails in the system, the next failure on another disk is highly likely to occur in the very near future. Thus, the failed disk should be rapidly reconstructed (or *decoded*) to prevent catastrophic failures to the system. For a fixed amount of information to recover, the lower the complexity of a decoding algorithm gets, the shorter the reconstruction time a disk array takes.

Many researches on designing a reliable disk array system with high I/O performance have been done [9,11–15]. MacWilliams and Sloane gave two methods exploiting the extended hamming code and the Reed-Solomon code. The extended hamming code has a capability of double erasure correcting as well as triple-erasure-correcting, however this code shows a higher update penalty [12]. This code can attain low check disk overhead by using only two check disks. However, highly-integrated encoding and decoding chip sets should be used due to the complexity of Reed-Solomon code [12]. In addition, the method suffers from the high probability of the infinite error propagation and has an overhead redundancy at the end of the data because of using the convolutional type of code [9]. Gibson [12,13] presented a method using the 2D parity scheme to decrease update penalty. Even though the 2D parity scheme changes only two check disks for updating a data block, this scheme has higher check disk overheads [12]. Moreover, the three methods aforementioned have a common drawback that the severe bottleneck problem can occur when updating a check disk, since they use designated check disks to store the redundant information.

In order to cope with frequent disk failures, a few schemes tolerating up to two disk failures at the same time have been recently suggested in [9,11,15]. Park has given a method which alleviates the bottleneck problem by distributing the parity blocks over the disk array space [11]. However, the overhead for the management of the redundancy matrix that holds the redundant information cannot be ignored since the redundant matrix should be searched for the positions of the associated blocks whenever updating a data block or reconstructing failed disks is done. Blaum et al. [9] have developed a scheme called EVENODD which has better performance than the methods described so far. EVENODD scheme uses optimal redundant storage while tolerating up to two disk failures. In EVENODD scheme, however, when the data blocks contained in *diagonal S* are to be updated, every diagonal parity block should also be updated. Moreover, Alvarez [15] proposed a method that can tolerate multiple disk failures in a disk array. However, the method has a defect that a large amount of redundant information should be included in the disk array.

In this paper, we present a couple of new efficient parity placement methods called DH (diagonal–horizontal) 1 and DH2 schemes. Both are dual parity schemes based on the diagonal and horizontal parities for tolerating up to two disk failures in a disk array. DH1 scheme places the horizontal parities in the antidiagonal direction and the diagonal parities in the horizontal direction. DH1 scheme does not use any dedicated parity disks. Therefore, the bottleneck problem occurring when updating the parities can be avoided in DH1 scheme. DH2 scheme uses one more disk than DH1 scheme to store the horizontal parities in the vertical direction, that is the horizontal parities are stored into a disk. This allows for DH2 scheme to reduce the overheads in updating a certain parity block. Therefore, when small writes are frequently required, DH2 scheme performs better than DH1 scheme.

DH1 and DH2 schemes can be implemented by pure software technology. In other words, they use only exclusive-or (XOR) operations that are already used in the existing disk array systems for computing parities, and can be implemented without modifying the hardware. Moreover, they are simpler and more efficient than other schemes that tolerate two disk failures. If two controllers are available in the disk array system, the reconstruction time for any pair of failed disks can be reduced significantly, since the decoding algorithms for both DH schemes allow two different directions of the recoveries at the same time. Hence, the probability of an occurrence of catastrophic failures in the system can be lowered.

Gibson [12] showed that 2D scheme performs better than the extended hamming code even though it has a higher check disk overhead. Blaum et al. [9] also showed that EVENODD scheme is more efficient than both 2D and Reed-Solomon code schemes. The redundancy matrix scheme in [11] requires more parity

space than EVENODD scheme, and the management overheads for the redundancy matrix cannot be negligible. Therefore, we have compared our DH schemes only with EVENODD scheme for performance analysis. The analysis has shown that the encoding schemes for both DH schemes are more efficient than EVENODD scheme. It has also shown that reconstruction using either DH1 or DH2 scheme has taken shorter time for any two failed disks than EVENODD scheme. The ratio of the number of the parity blocks to that of the overall blocks (data and parity blocks) in a given disk array for each scheme is also compared. It has been shown that both DH schemes use near optimal disk space for storing the redundant information.

The rest of this paper is organized as follows. In Section 2, the encoding method of DH1 scheme for the parity placement is described. The decoding algorithm for recovering from disk failures is also explained. Section 3 illustrates both the encoding and decoding methods of DH2 scheme. In Section 4, the performances of DH1 and DH2 schemes are evaluated by comparing with EVENODD scheme.

2. DH1 scheme

In order to tolerate up to two disk failures, DH1 scheme places the horizontal parities in the diagonal direction and the diagonal parities in the horizontal direction. When a single disk failure occurs, DH1 scheme reconstructs it easily by using either the horizontal or the diagonal parities. In the case of two disk failures, the faulty blocks can be sequentially reconstructed by using both the diagonal and the horizontal parities alternately.

2.1. Encoding method

For DH1 scheme, we assume that there are N disks in a disk array, where N is a prime number as the same as in [9,15]. N must be prime in order for the decoding algorithm to reconstruct two failed disks successfully. In DH1 scheme a disk array can be considered as an $(N-1)\times N$ matrix, A, where each disk contains $(N-1)$ logical blocks. Let $A[i,j]$ refer to the block in the ith row and in the jth column of the matrix A, where $0\leqslant i\leqslant N-2$ and $0\leqslant j\leqslant N-1$, that is, $A[i,j]$ denotes the ith logical block of $disk_j$. In order to simplify the description for the encoding and decoding algorithms of both DH schemes, the following notations and definitions are used. Let $A[i,j]_{\%N}$ be $A[i \bmod N, j \bmod N]$, and $(k_1,k_2,\ldots)_{\%N}$ denote $(k_1 \bmod N,\ k_2 \bmod N,\ldots)$. We also define *an error correcting group* as a group of blocks such that if any block in the group has a fault, the block can be reconstructed by XORing the rest of the blocks in the group.

Each row in the matrix A can then be defined as a horizontal error correcting group (HECG). The ith row of A, $0\leqslant i\leqslant N-2$, can be denoted as HECG_i. The horizontal parity H_i of HECG_i can be obtained with Eq. (1) and is placed into $A[i,N-2-i]$ as shown in Fig. 1, where $N=7$. In this figure row i holds $(N-1)$ i's and an H_i in order to indicate each element (block) of the row belongs to HECG_i.

0	0	0	0	0	H_0	0
1	1	1	1	H_1	1	1
2	2	2	H_2	2	2	2
3	3	H_3	3	3	3	3
4	H_4	4	4	4	4	4

Fig. 1. Placement of the horizontal parities and HECGs for DH1 scheme.

$$H_i = \bigoplus_{j=0}^{N-1} A[i,j], \quad \text{where } 0 \leqslant i \leqslant (N-3) \text{ and } j \neq N-2-i. \tag{1}$$

That is, H_i is computed by XORing the ith data block of each disk except the ith block of $disk_{N-2-i}$. Observe that there are $N-1$ data blocks and a horizontal parity block in each HECG.

Another type of an error correcting group called a diagonal error correcting group (DECG) is also required for tolerating two disk failures. Let DECG_j, $0 \leqslant j \leqslant N-1$, be the blocks that are located in the antidiagonal direction as shown in Fig. 2. The diagonal parity D_j of DECG_j can be calculated with Eq. (2) and is placed into $A[N-2,j]$. In this figure DECG_j consists of the elements holding j or D_j.

$$D_j = \bigoplus_{i=0}^{N-3} A[N-3-i, j+1+i]_{\%N}, \quad \text{where } 0 \leqslant j \leqslant (N-1). \tag{2}$$

That is, D_j is computed by XORing the data blocks in DECG_j. Observe that the last block of disk j holds D_j, and the number of the blocks in each DECG is $N-1$ while each HECG has N blocks.

Fig. 3 shows the locations of the horizontal and the diagonal parities along with HECGs and DECGs in DH1 scheme by combining Figs. 1 and 2, where $N=7$. In Fig. 3 the last block of $disk_0$ holds D_0 which is the diagonal parity for H_i's, where $0 \leqslant i \leqslant (N-3)$. And D_0 happens to be the horizontal parity for D_j's, where $1 \leqslant j \leqslant (N-1)$. Hence D_0 also plays the role of H_{N-2}. The following lemma proves that $D_0 = H_{N-2}$.

Lemma 1. *The diagonal parity D_0 and the horizontal parity H_{N-2} are always the same.*

Proof. By Eq. (1), H_{N-2} is obtained as follows:

$$H_{N-2} = \bigoplus_{j=1}^{N-1} A[N-2,j]$$

The above equation can be unfolded and then each term of the right-hand side of the above equation can be replaced by the terms using Eq. (2).

2	3	4	5	6	0	1
3	4	5	6	0	1	2
4	5	6	0	1	2	3
5	6	0	1	2	3	4
6	0	1	2	3	4	5
D_0	D_1	D_2	D_3	D_4	D_5	D_6

Fig. 2. Placement of the diagonal parities and DECGs for DH1 scheme.

02	03	04	05	06	H_0	01
13	14	15	16	H_1	11	12
24	25	26	H_2	21	22	23
35	36	H_3	31	32	33	34
46	H_4	41	42	43	44	45
D_0	D_1	D_2	D_3	D_4	D_5	D_6

Fig. 3. The diagonal and horizontal parity placements in DH1 scheme.

$$
\begin{aligned}
H_{N-2} &= A[N-2,1] \oplus A[N-2,2] \oplus A[N-2,3] \\
&\quad \oplus \cdots \oplus A[N-2,N-1] \\
&= D_1 \oplus D_2 \oplus D_3 \oplus \cdots \oplus D_{N-1} \\
&= (A[N-3,2] \oplus A[N-4,3] \\
&\quad \oplus \cdots \oplus A[0,N-1]) \quad /* \text{ for } D_1 \ */ \\
&\quad \oplus (A[N-3,3] \oplus A[N-4,4] \\
&\quad \oplus \cdots \oplus A[0,0]) \quad /* \text{ for } D_2 \ */ \\
&\quad \oplus (A[N-3,4] \oplus A[N-4,5] \\
&\quad \oplus \cdots \oplus A[0,1]) \quad /* \text{ for } D_3 \ */ \\
&\quad \cdots \\
&\quad \oplus (A[N-3,0] \oplus A[N-4,1] \\
&\quad \oplus \cdots \oplus A[0,N-3]) \quad /* \text{ for } D_{N-1} \ */.
\end{aligned}
$$

When the above terms are rearranged in row major order, H_{N-2} can be rewritten as in the following:

$$
\begin{aligned}
H_{N-2} &= (A[N-3,0] \oplus A[N-3,2] \\
&\quad \oplus A[N-3,3] \oplus \cdots \oplus A[N-3,N-1]) \\
&\quad \oplus (A[N-4,0] \oplus A[N-4,1] \\
&\quad \oplus A[N-4,3] \oplus \cdots \oplus A[N-4,N-1]) \\
&\quad \cdots \\
&\quad \oplus (A[0,0] \oplus A[0,1] \oplus A[0,2] \\
&\quad \oplus \cdots \oplus A[0,N-3] \oplus A[0,N-1]) \\
&= A[N-3,1] \oplus A[N-4,2] \\
&\quad \oplus \cdots \oplus A[0,N-2] \\
&= H_{N-3} \oplus H_{N-4} \oplus \cdots \oplus H_0 \\
&= \bigoplus_{i=0}^{N-3} H_i = D_0. \qquad \square
\end{aligned}
$$

In the following example the encoding of DH1 scheme is illustrated when the number of disks is 7.

Example 1. In the figure below, each data block contains an arbitrary binary digit for the simplicity. Empty blocks (elements) in the figure are the places where the horizontal and diagonal parities are to be placed.

1	0	1	1	0		0
0	1	1	0		0	1
1	1	1		0	0	0
0	0		1	0	1	0
0		0	1	0	1	1

Using Eqs. (1) and (2), the horizontal and the diagonal parities are calculated and deposited to the predefined locations as follows.

1	0	1	1	0	**1**	0
0	1	1	0	**1**	0	1
1	1	1	**1**	0	0	0
0	0	**0**	1	0	1	0
0	**1**	0	1	0	1	1
0	**1**	**1**	**1**	**0**	**0**	**1**

2.2. Decoding method

If a single disk has a failure in a disk array encoded with the encoding algorithm of DH1 scheme, the disk can be recovered easily using either the horizontal or the diagonal parities. When two disk failures in the disk array have occurred at the same time, DH1 scheme performs error correction using the diagonal and the horizontal parities alternately. First, the recovery procedure is initiated with a certain DECG. Among DECGs, there exist two groups each of which has only one faulty block. This fact will be proved in the following subsection. Thus, one of the two DECGs can be selected arbitrarily for initiating the recovery procedure. The faulty block can then be reconstructed with the rest of the blocks in the DECG. After the reconstruction of the faulty block, the HECG to which the recovered block belongs now has only one faulty block. The blocks with unfaulty information in the HECG can be used to reconstruct the faulty block. The block just recovered with the HECG allows the recovery of the next faulty block using another DECG. Using DECGs and HECGs repeatedly in this way, all the rest of the faulty blocks of two failed disks can be entirely recovered. The following algorithm shows the error recovery procedure of DH1 scheme when two disks are faulty at the same time.

Decoding algorithm 1. Assume that two disks, $disk_{j_1}$ and $disk_{j_2}$, are faulty at the same time, where $0 \leqslant j_1 < j_2 \leqslant (N-1)$:

Step 1: $k = j_2 - j_1$;
for $(m = 0;\ m \leqslant (N-2); m++)$ {

Step 2: /* r finds the row index such that $A[r, j_2]$ can be recovered in step 3. */
$r = (N - 1 - k(m+1))_{\%N}$;

Step 3: /* reconstruct $A[r, j_2]$ using the DECG to which $A[r, j_2]$ belongs */
$A[r, j_2] = \bigoplus_{l=0}^{N-2} A[N-2-l, j_1 + 1 - mk + l]_{\%N}$, where $(j_1 + 1 - mk + l)_{\%N} \neq j_2$;

Step 4: /* reconstruct $A[r, j_1]$ using the HECG to which $A[r, j_2]$ belongs */ $A[r, j_1] = \bigoplus_{l=0}^{N-1} A[r, l]$, where $l \neq j_1$
}

In the following example the above algorithm is applied to a disk array with 7 disks. Each block in the disk array holds an arbitrary bit for the simplicity.

Example 2. Assume that $disk_1$ and $disk_3$ have failed at the same time, when a disk array has 7 disks. Then the following table can be viewed as a snap shot before decoding.

1	?	1	?	0	1	0
0	?	1	?	1	0	1
1	?	1	?	0	0	0
0	?	0	?	0	1	0
0	?	0	?	0	1	1
0	?	1	?	0	0	1

In the above figure, ?s indicate that $disk_1$ and $disk_3$ are faulty. In step 1 of Algorithm 1 we get $k = j_2 - j_1 = 2$. Then in the **for**-loop, the following are executed varying m from 0 to $N - 2$, that is, steps 3 and 4 can be given as the following equations after r has been replaced:

$$A[4-2m,3]_{\%7} = \bigoplus_{l=0}^{5} A[5-l, 2-2m+l]_{\%7},$$

$$A[4-2m,1]_{\%7} = \bigoplus_{l=0}^{6} A[4-2m, l]_{\%7}.$$

The following table shows the sequence of the recovery of each faulty blocks in $disk_1$ and $disk_3$ as the **for**-loop is iterated.

m	Calculations	Results
0	$A[4,3] = A[5,2] \oplus A[3,4] \oplus A[2,5] \oplus A[1,6] \oplus A[0,0]$	1
	$A[4,1] = A[4,0] \oplus A[4,2] \oplus A[4,3] \oplus A[4,4] \oplus A[4,5] \oplus A[4,6]$	1
1	$A[2,3] = A[5,0] \oplus A[4,1] \oplus A[3,2] \oplus A[1,4] \oplus A[0,5]$	1
	$A[2,1] = A[2,0] \oplus A[2,2] \oplus A[2,3] \oplus A[2,4] \oplus A[2,5] \oplus A[2,6]$	1
2	$A[0,3] = A[5,5] \oplus A[4,6] \oplus A[3,0] \oplus A[2,1] \oplus A[1,2]$	1
	$A[0,1] = A[0,0] \oplus A[0,2] \oplus A[0,3] \oplus A[0,4] \oplus A[0,5] \oplus A[0,6]$	0
3	$A[5,3] = A[4,4] \oplus A[3,5] \oplus A[2,6] \oplus A[1,0] \oplus A[0,1]$	1
	$A[5,1] = A[5,0] \oplus A[5,2] \oplus A[5,3] \oplus A[5,4] \oplus A[5,5] \oplus A[5,6]$	1
4	$A[3,3] = A[5,1] \oplus A[4,2] \oplus A[2,4] \oplus A[1,5] \oplus A[0,6]$	1
	$A[3,1] = A[3,0] \oplus A[3,2] \oplus A[3,3] \oplus A[3,4] \oplus A[3,5] \oplus A[3,6]$	0
5	$A[1,3] = A[5,6] \oplus A[4,0] \oplus A[3,1] \oplus A[2,2] \oplus A[0,4]$	0
	$A[1,1] = A[1,0] \oplus A[1,2] \oplus A[1,3] \oplus A[1,4] \oplus A[1,5] \oplus A[1,6]$	1

Observe that the following figure is the same as the original disk array in the previous example.

1	**0**	1	**1**	0	1	0
0	**1**	1	**0**	1	0	1
1	**1**	1	**1**	0	0	0
0	**0**	0	**1**	0	1	0
0	**1**	0	**1**	0	1	1
0	**1**	1	**1**	0	0	1

2.3. Correctness of the decoding method

Let a disk array have N disks, where N is prime. Also, let two failed disks be $disk_{j_1}$ and $disk_{j_2}$, and $k = j_2 - j_1$ where $j_1 < j_2$. If the encoding method of DH1 scheme has been applied to the disk array, then the following lemmas and a theorem prove that the decoding algorithm for DH1 scheme always recovers successfully from two disk failures.

Lemma 2. *Among DECGs of the disk array described above, there exist two DECGs such that each of theses DECGs has only one faulty block.*

Proof. When two disks have failed, the total number of the faulty blocks is $2(N-1)$, since there are $(N-1)$ blocks in each disk. Observe that there are N DECGs in the disk array and each DECG consists of $(N-1)$ blocks. It can be easily seen that each DECG has at most two faulty blocks since each block in a DECG is placed into a different disk. Therefore, there are at most $2N$ faulty blocks. But we know that there are exactly $2(N-1)$ faulty blocks. Hence there should exist some DECGs that have less than two faulty blocks. However, it is not possible that a DECG has no faulty blocks and each of the rest of DECGs has two faulty blocks, because each block in a DECG is placed into a different disk, and the blocks in each DECG span $(N-1)$ disks. This implies that each DECG has at least one faulty block. Therefore, there should exist two DECGs each of which has only one faulty block. □

The following lemma shows that $DECG_{j_1+1}$ is one of the two DECGs which has only one faulty block.

Lemma 3. *$DECG_{j_1+1}$ has only one faulty block which is in $A[N-1-k, j_2]_{\%N}$.*

Proof. If we look into $disk_{j_1+1}$ which is the next disk to $disk_{j_1}$, the last block of $disk_{j_1+1}$ is supposed to contain D_{j_1+1}. Recall that D_{j_1+1} was obtained during the encoding by XOR operations over the rest of the blocks in DECG_{j_1+1} (refer to Eq. (2)). Since each DECG has $N-1$ blocks and each block in a DECG is located at different disk in the antidiagonal direction as shown in Fig. 2, none of the blocks in DECG_{j_1+1} is in $disk_{j_1}$ and only one faulty block b in DECG_{j_1+1} is in disk j_2.

We claim that the block, b, is in $A[N-1-k, j_2]_{\%N}$. Since the blocks in DECG_{j_1+1} are placed into from $A[N-2, j_1+1]$ in the antidiagonal direction, if $j_2 = j_1 + 1$, the block b is in $A[N-2, j_2]$. In general, the block b is $A[N-2-(k-1), j_1+k]_{\%N}$, since $j_2 = j_1 + k$, and since the blocks in DECG_{j_1+1} are placed in the antidiagonal direction. Therefore, the block b is $A[N-1-k, j_1+k]_{\%N} = A[N-1-k, j_2]_{\%N}$. □

Note that the other DECG that has only one faulty block can be found to be $\mathrm{DECG}_{(j_2+1)_{\%N}}$ similarly. In the decoding algorithm of DH1 scheme, the row index r is obtained to reconstruct the faulty block $A[r, j_2]$ in the beginning of the recovery. After the block has recovered, the faulty block of $disk_{j_1}$ in the same row can be fixed. Then the next row index is calculated to reconstruct the faulty block of $disk_{j_2}$ in another row, and so on. If we enumerate these row numbers that the decoding algorithm traverses during the recovery, the enumeration sequence σ can be given as follows:

$$\sigma = (N-1-k, N-1-2k, N-1-3k, \ldots, N-1-(N-1)k)_{\%N}, \quad \text{where } k = j_2 - j_1.$$

In the following lemma we prove that the row numbers are unique and are within the range between 0 and $N-2$. For the proof of the lemma, we use the following fact which was proven in [16]. The following N numbers in σ' are distinct each other, when N and k are relatively prime.

$$\begin{aligned}\sigma' &= (0 \bmod N, k \bmod N, 2k \bmod N, \ldots, \\ &\quad (N-2)k \bmod N, (N-1)k \bmod N) \\ &= (0, k, 2k, 3k, \ldots, (N-2)k, (N-1)k)_{\%N}.\end{aligned}$$

Therefore, the numbers in σ' can be rewritten as follows, since each number has been obtained by $\bmod N$ and is distinct each other.

$$0, 1, 2, \ldots, N-2, N-1.$$

Lemma 4. *Each number in σ is unique and the numbers in σ can be reordered as follows*:

$$(0, 1, 2, \ldots, (N-2), (N-1)).$$

Proof. The sequence σ' can be converted into the following sequence while the numbers in this new sequence are distinct each other.

$$\begin{aligned}\sigma'' &= (N-0, N-k, N-2k, \ldots, N-(N-2)k, \\ &\quad N-(N-1)k)_{\%N}.\end{aligned}$$

The numbers in the above sequence are merely shifted of their positions within the sequence σ'. We now shift each number in σ'' as follows once more:

$$\begin{aligned}\sigma''' &= (N-1-0, N-1-k, N-1-2k, \ldots, \\ &\quad N-1-(N-2)k, N-1-(N-1)k)_{\%N}.\end{aligned}$$

The numbers in the sequence σ''' are still distinct each other, because each number has been merely subtracted by 1 and $(\bmod N)$ has been applied to each number. Hence the numbers in σ''' can be reordered from 0 to $N-1$. The sequence σ is, therefore, equivalent to σ''' except for the first term of σ''', i.e., $(N-1-0)_{\%N}$. But the value of this term is $N-1$. Therefore, the numbers in σ can be rewritten as claimed. □

We now prove the following theorem for the correctness of the decoding algorithm in DH1 scheme.

Theorem 1. *Let a disk array have N disks, where N is prime. If the disk array is constructed with the encoding method of DH1 scheme, then Algorithm* 1 *can recover from any two disk failures successfully.*

Proof. When two disks are faulty at the same time, each row of the disk array has exactly two faulty blocks. Lemma 3 has shown that DECG_{j_1+1} has only one faulty block, $A[N-1-k, j_2]_{\%N}$. Therefore, the faulty block can be reconstructed by XORing the remaining blocks in DECG_{j_1+1}. Then the faulty block $A[N-1-k, j_1]_{\%N}$ in disk j_1 in the same row can be reconstructed by using the unfaulty blocks in HECG_{N-1-k}. It can be seen that the reconstructed block $A[N-1-k, j_1]_{\%N}$ belongs to $\mathrm{DECG}_{(j_1+1-k)_{\%N}}$. This DECG contains only one faulty block in $disk_{j_2}$, since each DECG has at most two faulty blocks and one of the two blocks has been just recovered. So this block in $disk_{j_2}$ can be reconstructed with $\mathrm{DECG}_{(j_1+1-k)_{\%N}}$. Then the faulty block of $disk_{j_1}$ in the row at which the block of $disk_{j_2}$ just recovered is located can now be recovered using the rest of the blocks in the corresponding HECG. When we continue this recovery process as in the decoding algorithm using both DECGs and HECGs alternately, the sequence of the row numbers visited by Algorithm 1 becomes σ as illustrated before this theorem. In Lemma 4 we showed that σ can be rewritten as follows:

1	**(6)**	1	**(5)**	0	1	0
0	**(12)**	1	**(11)**	1	0	1
1	**(4)**	1	**(3)**	0	0	0
0	**(10)**	0	**(9)**	0	1	0
0	**(2)**	0	**(1)**	0	1	1
0	**(8)**	1	**(7)**	0	0	1

Fig. 4. The recovery sequence of the decoding algorithm for DH1 scheme, where $disk_1$ and $disk_3$ are faulty and $N = 7$.

$$0, 1, 2, \ldots, N-2.$$

This illustrates that Algorithm 1 visits each row exactly once and traverses all the rows of the disk array. Whenever the algorithm visits a row, it uses the corresponding DECG to reconstruct the faulty block in disk j_2, then recovers the faulty block of disk j_1 in the same row. Therefore, all the blocks in the two failed disks can be recovered entirely. □

2.4. Two sequences in the recovery procedure

A recovery procedure is generally proceeded in a predetermined sequence. Therefore, if some block of a disk is not accessible during the recovery due to some bad sectors of the disk, the recovery procedure cannot be proceeded further on. In order to enhance the reliability, an alternative sequence in a recovery procedure may be provided. DH1 scheme is able to provide an alternative sequence which is the reverse of the sequence σ. Fig. 4 shows the sequence of the reconstructed blocks in the recovery procedure of the decoding algorithm for DH1 scheme.

The recovery procedure can be proceeded either from (1) to (12) (the *forward* direction) or from (12) down to (1) (the *backward* direction). For example, if a certain faulty block cannot be recovered while the recover procedure follows the forward direction of the sequence, the recovery using the backward direction can be processed so that the number of faulty blocks that could not be recovered can be reduced. If two controllers are available in the disk array system, the recovery procedure can proceed using both directions of the sequence at the same time. As a result, the reconstruction time for any pair of failed disks can be reduced significantly. Also, the probability of an occurrence of catastrophic failures in the system can be lowered.

3. DH2 scheme

DH2 scheme also uses both DECGs and HECGs as in DH1 scheme. In DH2 scheme, however, the horizontal parities are placed in the vertical direction using an extra parity disk, i.e., $(N+1)$ disks are used, where N is prime. We assume that each disk has $(N-1)$ logical blocks as in DH1 scheme. By placing the horizontal parities vertically, the computations for the parity block, D_0 (or H_{N-2}), in DH1 scheme can be eliminated.

In order to reduce the occurrences of the bottleneck problem that may occur when updating some horizontal parities, the horizontal parities can be stored rotationally as shown in Fig. 5, since a disk array can be viewed, without loss of generality, as a multiple of $(N-1) \times (N+1)$ matrices.

In the above figure each disk consists of a sequence of $(N-1)$ logical blocks. Hence the horizontal parities can be distributed into different disks with minor changes in the encoding and decoding methods for DH2 scheme. However, for the sake of simplicity, we assume that a disk array is viewed only a $(N-1) \times (N+1)$ matrix from now on.

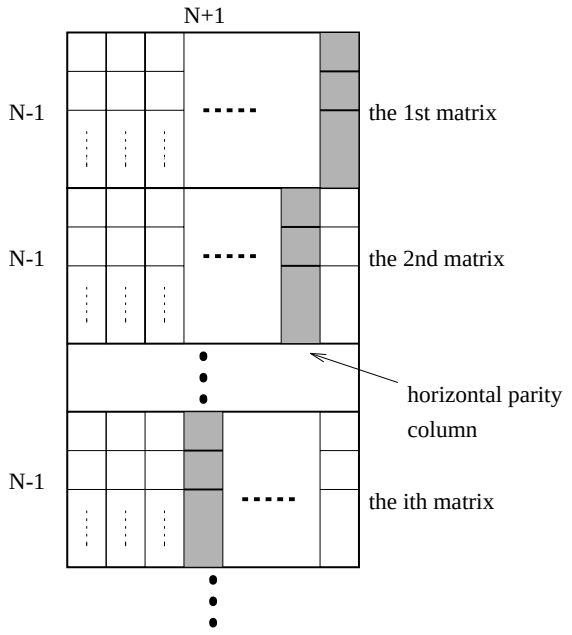


Fig. 5. Rotational placement of the horizontal parities in DH2 scheme.

3.1. Encoding method

In DH2 scheme a disk array can be viewed as an $(N-1) \times (N+1)$ matrix, since there are $(N+1)$ disks and each disk is assumed to have $(N-1)$ logical blocks. We call the matrix B. In DH2 scheme the last disk, $disk_N$, holds all the horizontal parities.

In the matrix B, the blocks in each row are grouped into a HECG as in DH1 scheme. Let HECG_i denote the ith HECG, $0 \leqslant i \leqslant N-2$. The parity H_i in HECG_i can be obtained with Eq. (3) and is placed into $B[i, N]$ as shown in Fig. 6, where $N = 7$. In this figure each block in the ith row holds i or H_i to indicate that the blocks in the row belong to the same HECG_i.

$$H_i = \bigoplus_{j=0}^{N-1} B[i, j], \quad \text{where } 0 \leqslant i \leqslant (N-3). \tag{3}$$

That is, H_i can be obtained by XORing the ith block of each disk except the ith block of the last disk. Observe that the number of the data blocks in each HECG is N.

0	0	0	0	0	0	0	H_0
1	1	1	1	1	1	1	H_1
2	2	2	2	2	2	2	H_2
3	3	3	3	3	3	3	H_3
4	4	4	4	4	4	4	H_4

Fig. 6. Placement of the horizontal parities and their corresponding HECGs in DH2 scheme.

Let a DECG be a group of blocks located in the antidiagonal direction as in DH1 scheme. The diagonal parity D_j in DECG_j is obtained with Eq. (4) and is placed into $B[N-2,j]$, the last block of $disk_j$.

$$D_j = \bigoplus_{i=0}^{N-3} B[N-3-i, j+1+i]_{\%N}, \quad \text{where } 0 \leqslant j \leqslant (N-1). \tag{4}$$

That is, D_j is computed by XORing the data blocks in DECG_j. Fig. 7 shows the locations of the diagonal parities. In this figure, the element holding j or D_j indicates that it belongs to DECG_j. Note that there are $(N-1)$ blocks in each DECG including its parity block.

Fig. 8 illustrates the placement of the horizontal and diagonal parities along with the HECGs and DECGs by combining both Figs. 6 and 7.

The block indicated by 'E' in Fig. 8 represents an empty block and remains empty after the encoding. However, this block may be used for temporary storage such as a swap space during runtime. UNIX, for example, allocates some disk space for temporary storage which does not require any redundancy for reliability since it holds valid data only during runtime. The block E may also be used to substitute for a physically defective block such that the defects may occur when the disk drive is newly manufactured. The following example illustrates the encoding of DH2 scheme when N is 7.

Example 3. The following matrix depicts a disk array such that each data block is assumed to have an arbitrary bit for the simplicity.

1	0	1	1	0	1	0	
0	1	1	0	0	0	1	
1	1	1	0	0	0	0	
0	0	1	1	0	1	0	
0	1	0	1	0	1	1	

2	*3*	*4*	*5*	*6*	*0*	*1*	
3	*4*	*5*	*6*	*0*	*1*	*2*	
4	*5*	*6*	*0*	*1*	*2*	*3*	
5	*6*	*0*	*1*	*2*	*3*	*4*	
6	*0*	*1*	*2*	*3*	*4*	*5*	
D_0	D_1	D_2	D_3	D_4	D_5	D_6	

Fig. 7. Placement of the diagonal parities and their corresponding DECGs in DH2 scheme.

02	03	04	05	06	00	01	H_0
13	14	15	16	10	11	12	H_1
24	25	26	20	21	22	23	H_2
35	36	30	31	32	33	34	H_3
46	40	41	42	43	44	45	H_4
D_0	D_1	D_2	D_3	D_4	D_5	D_6	E

Fig. 8. The diagonal and horizontal parity placements in DH2 scheme.

For the encoding of the disk array using DH2 scheme, the horizontal and diagonal parities are calculated with Eqs. (3) and (4), respectively. The following matrix shows the disk array after placing the parities appropriately.

1	0	1	1	0	1	0	**0**
0	1	1	0	0	0	1	**1**
1	1	1	0	0	0	0	**1**
0	0	1	1	0	1	0	**1**
0	1	0	1	0	1	1	**0**
1	**1**	**1**	**1**	**0**	**0**	**1**	E

3.2. Decoding method

A single disk failure can be recovered easily by using either DECGs or HECGs. However, in case when any pair of disks are faulty at the same time, DH2 scheme uses two different recovery procedures. One procedure takes care of the case when the last disk holding the horizontal parities has no failure. The other procedure is used when the last disk is one of the two disks with failures. Algorithms 2-1 and 2-2 illustrate these two procedures, respectively. For the algorithms, we assume that $disk_j$ and $disk_{j2}$ are faulty, where $0 \leqslant j_1 \leqslant j_2 \leqslant N$.

Decoding algorithm 2-1. In case when the last disk has no failure, i.e. $0 \leqslant j_1 < j_2 \leqslant (N-1)$:

Step 1: $k = j_2 - j_1$;
for $(m = 0; m \leqslant N-2; m++)$ {
Step 2: /* r finds the row at which the next faulty block of $disk_{j_2}$ is located. */
$r = (N-1-k(m+1))_{\%N}$;
Step 3: /* The block at $B[r, j_2]$ of $disk_{j_2}$ is recovered using $\mathrm{DECG}_{(j_1+1-mk)_{\%N}}$. */
$B[r, j_2] = \bigoplus_{l=0}^{N-2} B[N-2-l, j_1+1-mk+l]_{\%N}$,
where $(j_1+1-mk+l)_{\%N} \neq j_2$;
Step 4: **if** $r \neq (N-2)$
/* The faulty block at $B[r, j_1]$ of $disk_{j_1}$ is recovered using HECG_r */
$B[r, j_1] = \bigoplus_{l=0}^{N} B[r, l]$, where $l \neq j_1$;
else
/* The block at $B[N-2, j_1]$ is recovered using H_i's and HECG_{N-2} */
$B[r, j_1] = (\bigoplus_{h=0}^{N-3} B[h, N]) \oplus (\oplus_{l=0}^{N} B[N-2, l])$, where $l \neq j_1$;
}

Algorithm 2-1 reconstructs the faulty blocks of $disk_{j_1}$ and $disk_{j_2}$ as in the decoding algorithm of DH1 scheme, except the last block of $disk_{j_1}$. Step 1 finds the difference k between the indices of the two failed disks. In step 2 the row number r is computed for finding the faulty block of $disk_{j_2}$. In step 3, the rth block of $disk_{j_2}$ can be reconstructed by XORing the rest of the unfaulty blocks of $\mathrm{DECG}_{(j_1+1-mk)_{\%N}}$. Next, the rth block in $disk_{j_1}$ can be recovered with the blocks in HECG_r, if r is not $(N-2)$. But when r is $(N-2)$, the last block of $disk_{j_1}$ should be recovered by XORing all the horizontal parities as well as the unfaulty blocks of the last row, i.e., HECG_{N-2}, because the horizontal parity for the last row is not stored in the array. As in the decoding algorithm for DH1 scheme, every faulty block of the two failed disks can be sequentially recovered using DECGs and HECGs alternately.

Decoding algorithm 2-2. In case when one of the two failed disks is the last disk, i.e., $0 \leqslant j_1 \leqslant (N-1)$ and $j_2 = N$:

for $(r = 0; r \leqslant N-3; r++)$ {
Step 1: /* Find the DECG_c to recover $B[r, j_1]$ $*$ $/c = (j_1 + 2 + r)_{\%N}$;
Step 2: /* Reconstruct $B[r, j_1]$ using the blocks in DECG_c */ $B[r, j_1] = \bigoplus_{l=0}^{N-2} B[N-2-l, c+l]_{\%N}$, where $(c+l)_{\%N} \neq j_1$;
Step 3: /* Reconstruct $B[r, j_2]$ using the blocks in HECG_r */ $B[r, j_2] = \bigoplus_{l=0}^{N-1} B[r, l]$;
}

Algorithm 2-2 describes the decoding method when the last disk is one of the two failed disks, i.e., $j_2 = N$. This decoding algorithm is simpler than Algorithm 2-1, since in this case every DECG has only one faulty block which is in $disk_{j_1}$. Algorithm 2-2 shows the recovery process that uses DECGs and HECGs alternately as in Algorithm 2-1 or the decoding algorithm of DH1 scheme. However, another recovery can be proceeded by recovering each faulty block of $disk_{j_1}$ by XORing of the unfaulty blocks of the corresponding DECG first. And then each faulty block of $disk_{j_2}$ can be reconstructed by the blocks in the corresponding HECG. The following example shows a recovery of a disk array of size 7 for Algorithm 2-1.

Example 4. The following matrix shows the disk array when two disks, $disk_1$ and $disk_3$, are faulty,

1	?	1	?	0	1	0	0
0	?	1	?	0	0	1	1
1	?	1	?	0	0	0	1
0	?	1	?	0	1	0	1
0	?	0	?	0	1	1	0
1	?	1	?	0	0	1	E

In step 1 of Algorithm 2-1, we obtain $k = j_2 - j_1 = 2$. In step 2 $r = (N-1-k(m+1))_{\%N} = (4-2m)_{\%7}$. Then in steps 3 and 4 the following equations can be obtained, where $0 \leqslant m \leqslant 5$:

$$B[4-2m, 3]_{\%7} = \bigoplus_{l=0}^{5} B[5-l, 2-2m+l]_{\%7},$$

$$B[4-2m, 1]_{\%7} = \begin{cases} \left(\bigoplus_{h=0}^{4} B[h, N]\right) \oplus \left(\bigoplus_{l=0}^{7} B[4-2m, l]\right), & l \neq 1, \quad \text{if } m = 3, \\ \bigoplus_{l=0}^{7} B[4-2m, l]_{\%7}, & \text{otherwise.} \end{cases}$$

Using the above equations and varying m, the two failed disks can be sequentially recovered as shown in the following figure.

1	**0**	1	**1**	0	1	0	0
0	**1**	1	**0**	0	0	1	1
1	**1**	1	**0**	0	0	0	1
0	**0**	1	**1**	0	1	0	1
0	**1**	0	**1**	0	1	1	0
1	**1**	1	**1**	0	0	1	E

We omit a decoding example for Algorithm 2-2 since it is immediate. The decoding algorithms for DH2 scheme can also be verified. But we omit the proof for the correctness of Algorithms 2-1 and 2-2 because it is very similar to the one shown for DH1 scheme.

4. Performance analysis

In [9], the performance of EVENODD scheme was compared with 2D scheme and Reed-Solomon code scheme in recovering from two disk failures. EVENODD scheme outperformed these schemes. Hence, in this section we have compared our DH schemes only with EVENODD scheme. For the analysis, we assume that the disk arrays for EVENODD, DH1, and DH2 schemes have the dimensions as in Table 1. Note that N is a prime number, since each scheme works only if N is prime.

The efficiency of each scheme can be measured by the numbers of XOR operations required for encoding and decoding for a given disk array, and by the number of XOR operations needed for a small write. The ratio of the number of the parity blocks to that of the data blocks for each scheme is also compared in this section.

4.1. Encoding efficiencies

The encoding efficiency of each scheme is compared using the number of XOR operations. In order to perform fair comparisons, the ratio R_s defined below is used, where s is a scheme, since each scheme uses an array of different size

$$R_s = \frac{\text{the number of XOR operations for encoding a disk array}}{\text{the number of data blocks in the array}}$$

That is, R_s indicates the number of XOR operations to encode a single data block using scheme s. We now calculate R_{DH1}, R_{DH2}, and R_{EVENODD} for their encoding efficiencies.

For DH1 scheme, each horizontal parity can be calculated with $(N-2)$ XOR operations using Eq. (1). Since there are $(N-2)$ horizontal parities in a disk array, the number of XOR operations for all the horizontal parities is $(N-2)^2$. Similarly, all the diagonal parities can be computed with $(N-3)N$ XOR operations as shown in Eq. (2). Therefore, the total number of XOR operations to encode a disk array of size $(N-1)\times N$ becomes $(N-2)^2+(N-3)N=(2N^2-7N+4)$. Hence, R_{DH1} can be found as follows:

Table 1
The dimensions of disk arrays for EVENODD, DH1, and DH2 schemes

	EVENODD	DH1	DH2
Total array size	$(N-1)\times(N+2)$	$(N-1)\times N$	$(N-1)\times(N+1)$
No. of data blocks	$(N-1)\times N$	$(N-2)\times(N-1)$	$(N-2)\times N$

$$R_{\mathrm{DH1}} = \frac{(2N^2 - 7N + 4)}{(N-2)(N-1)}.$$

In DH2 scheme, all the horizontal parities can be obtained with $(N-1)(N-2)$ XOR operations by Eq. (3), and all the diagonal parities require $(N-3)N$ XOR operations by Eq. (4). The total number of XOR operations is, therefore, $(N-1)(N-2) + (N-3)N = (2N^2 - 6N + 2)$. We now get R_{DH2} as in the following:

$$R_{\mathrm{DH2}} = \frac{(2N^2 - 6N + 2)}{(N-2)N}.$$

In the encoding of EVENODD scheme, the diagonal S, the horizontal parities, and the diagonal parities should be calculated. The equations for computing these parities are given below [9]:

$$S = \bigoplus_{l=1}^{N-1} A[N-1-l, l], \tag{5}$$

$$A[i, N] = \bigoplus_{l=0}^{N-1} A[i, l], \quad \text{where } 0 \leqslant i \leqslant N-2, \tag{6}$$

$$A[i, N+1] = S \oplus \left(\bigoplus_{l=0}^{N-1} A[i-l, l]_{\%N} \right), \quad \text{where } 0 \leqslant i \leqslant N-2. \tag{7}$$

In order to obtain diagonal S, $(N-2)$ XOR operations are required because there are $(N-1)$ diagonal blocks. In calculating the diagonal parities using Eq. (7), we need $N(N-1)$ XOR operations since finding each diagonal parity needs N XOR operations and the number of all the diagonal parities is $(N-1)$. For the horizontal parities, we need $(N-1)^2$ operations because there are $(N-1)$ rows in a disk array and each row needs $(N-1)$ XOR operations. The total number of XOR operations for encoding in EVENODD scheme is, therefore, $(2N^2 - 2N - 1)$. R_{EVENODD} can then be given as follows:

$$R_{\mathrm{EVENODD}} = \frac{(2N^2 - 2N - 1)}{(N-1)N}.$$

In Fig. 9, R_s for each scheme s is compared while N increases. DH2 scheme requires the smaller number of XOR operations than others. This result is mainly because EVENODD scheme needs to calculate the diagonal S for the diagonal parities and DH1 scheme should calculate the parity of the parities, i.e., D_0 (or H_{N-2}). DH1 scheme also shows better performance than EVENODD scheme when N is less than 17. This figure shows only little differences among the schemes, because we have used only 1 bit as the size of a data block. When a data block gets a realistic size, the differences should be much greater.

4.2. Decoding efficiencies

Even though the reliability of a disk array can be increased by using an encoding scheme for tolerating two disk failures, a catastrophic failure such as an another disk failure during a recovery from two disk failures may occur. Thus if we can reduce the reconstruction time, the probability of occurring of failures of this type could be lowered. In a decoding process, the unfaulty blocks associated with the block to be reconstructed are read into memory and are XORed. Then the reconstructed block is stored into its place in the disk array. Therefore, if I/O time of a block for each scheme is assumed to be the same, we could say

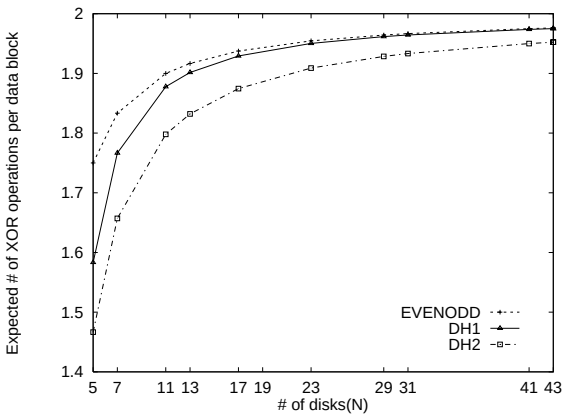


Fig. 9. The number of XOR operations to encode a single data block.

that the smaller the number of XOR operations for decoding a scheme has, the shorter the reconstruction time of a block for two disk failures it takes.

In order to compare the decoding efficiencies of DH1, DH2, and EVENODD schemes, we classify the disks in a disk array into threes types; H, D, and I. Let H denote a dedicated disk storing all the horizontal parities, D be a dedicated disk holding all the diagonal parities, and I be a disk that is neither H nor D. I may hold data and some parities together, but not all parities. In DH1 scheme, any pair of disks with failures are of the types (I,I) since both the horizontal and the diagonal parities are distributed over the disk array, i.e., there is no dedicated parity disk. In DH2 scheme, since there exists a dedicated disk to hold all the horizontal parities, two failed disks have type (I,I) or type (I,H). However, in EVENODD scheme there are two dedicated parity disks for the horizontal and diagonal parities. Therefore, (H,D), (I,H), (I,D), and (I,I) are all possible types for any pair of disks with failures. Table 2 shows the expected numbers of XOR operations for recovering two failed disks, the numbers of occurrences of different types of two failed disks, and the probabilities of the occurrences in different types of two disk failures with the assumption that any pair of disks in the disk array are equally likely to fail.

As in the comparisons for encoding the expected number of XOR operations each scheme is divided by the number of data blocks used in the scheme for a fair comparison. These results have been drawn in Fig. 10 while N varies.

As shown in Fig. 10, the decoding in DH2 scheme can be done faster than other schemes. Also, DH1 scheme is better than EVENODD scheme, since EVENODD scheme suffers from the overheads in computation involving diagonal S. This figure again shows only little differences among the schemes, because

Table 2
Comparison of three schemes for decoding for two disk failures

Schemes	Types of two failed disks	Expected no. of XOR operations	No. of types	Probability of occurring each pair of types
DH1	(I, I)	$2N^2-6N+2$	$N(N-1)/2$	1.0
DH2	(I, H)	$2N^2-5N+2$	N	$2/(N+1)$
	(I, I)	$2N^2-5N+1$	$N(N-1)/2$	$(N-1)/(N+1)$
EVENODD	(H, D)	$2N^2-2N-1$	1	$2/(N+2)(N+1)$
	(I, H)	N^2	N	$2N/(N+2)(N+1)$
	(I, D)	$2N^2-2N-1$	N	$2N/(N+2)(N+1)$
	(I, I)	$2N^2+N-7$	$N(N-1)/2$	$N(N-1)/(N+2)(N+1)$

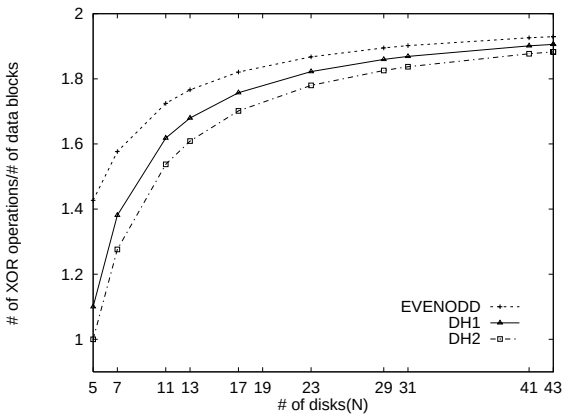


Fig. 10. The expected numbers of XOR operations required for decoding.

we have used only 1 bit as the size of a block. When a block size gets realistic, the differences should be much greater.

4.3. Efficiencies in small writes

Whenever a data block is updated, the associated parity blocks should also be updated. In DH1 scheme the number of block accesses for such updates is 8, since there are three parity blocks related with the block to be updated. These are the corresponding horizontal and diagonal parity blocks along with the special block D_0 (or H_{N-2}). Each of these blocks should be also read and written. Therefore, DH1 scheme needs four reads and four writes for updating a block, regardless of the number of disks used for a disk array. DH2 scheme requires only three reads and three writes, i.e., 6 block accesses, since it does not keep the special parity block, i.e., D_0. Recall that DH2 scheme spreads the diagonal parities throughout the disk array as in DH1 scheme, but uses a dedicated disk for all the horizontal parities.

However, in EVENODD scheme, if a data block is not included in the diagonal S, the number of block accesses only is 6. Otherwise, the number of block accesses becomes $2(N+1)$, where N is the number of disks. When N ranges from 5 to 43, the average number of block accesses for EVENODD scheme is 7.2–7.9. So the expected number of block accesses for updating a block in EVENODD scheme is smaller than that of DH1 scheme in this range. However, when two or more data blocks should be updated simultaneously, EVENODD scheme should suffer from updating the parity blocks sequentially because all the diagonal and the horizontal parities are placed into the two dedicated disks. Moreover, when a data block contained in diagonal S should be updated, the waiting time takes much longer, because every diagonal parity should also be updated.

4.4. Parity block ratio

In a disk array, the blocks holding the parities cannot be used to hold the actual data. So, it is desirable to keep as small a number of parity blocks as possible in a given disk array. Theoretically, any scheme that can tolerate up to two simultaneous disk failures needs at least two redundant disks. This is known as *singleton bound* [9]. EVENODD scheme uses exactly two redundant disks for the parities. Both DH1 and DH2 schemes also use approximately two disks for them. Fig. 11 shows the ratio of the number of the

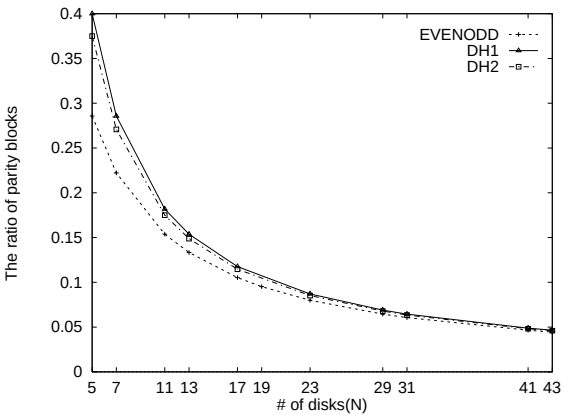


Fig. 11. The ratio of parity blocks to the overall blocks in a disk array.

parity blocks to that of the overall blocks for each scheme. It can be seen that DH1 and DH2 schemes use near optimal disk space for storing the redundant information.

5. Conclusion

In this paper, two efficient parity placement schemes, called DH1 and DH2 schemes, are presented for tolerating up to two disk failures. These schemes use both the diagonal and the horizontal parities for encoding a disk array. DH1 scheme places all the parity blocks evenly throughout the disk array, while DH2 scheme places the horizontal parity blocks into a dedicated disk and stores the diagonal parity blocks evenly as in DH1 scheme. DH1 and DH2 schemes can be implemented by pure software technology. In other words, they use only exclusive-or operations that are already used in the existing disk array systems for computing parities, and can be implemented without modifying the hardware.

The algorithms of both DH schemes can be implemented with lower complexities than that of EVENODD scheme. Moreover, DH1 scheme can reduce the occurrences of the bottleneck problems when updating parities, because the data and parity blocks are evenly distributed throughout the disk array. In DH1 and DH2 schemes, the recovery procedure can be overlapped by proceeding the procedure in two opposite directions and can also be processed in parallel with some additional hardware. Therefore, the reconstruction time for the disk array with two disk failures can be reduced significantly. As a result, the probability of an occurrence of catastrophic failures in a disk array system can also be lowered.

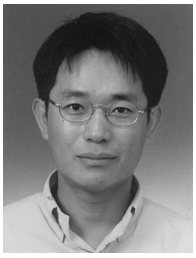
It has been shown that both DH schemes require smaller number of XOR operations for encoding a disk array, and need smaller number of XOR operations for recovering two disk failures than EVENODD scheme, respectively. DH2 scheme is more efficient for small writes than EVENODD scheme, too, while both DH schemes use near optimal disk space for the redundant information.

References

[1] D.A. Patterson, G.A. Gibson, R.H. Katz, A case for redundant arrays of inexpensive disks (RAID), in: Proceedings of the ACM SIGMOD Conference, 1988, pp. 109–116.

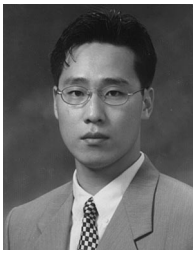
[2] R.H. Katz, G.A. Gibson, D.A. Patterson, Disk system architecture for high performance computing, IEEE Proc. 77 (12) (1989) 1842–1850.

[3] G.R. Ganger, B.L. Worthinton, R.Y. Hou, Y.N. Patt, Disk arrays: high-performance high-reliability storage subsystems, IEEE Comput. 27 (3) (1994) 30–36.
[4] P.M. Chen, D.A. Patterson, Maximizing performance in a striped disk array, Berkeley Technical Report, 1989.
[5] P. Shenoy, H.M. Vin, Efficient striping techniques for various bit rate continuous media file servers, University of Massachusetts, Amherst, UM-CS-1998-053, December 1998.
[6] E.K. Lee, R.H. Katz, The performance of parity placements in disk arrays, IEEE Trans. Comput. 42 (6) (1993) 651–664.
[7] R. Karedla, J.S. Love, B.G. Wherry, Caching strategies to improve disk system performance, IEEE Comput. 27 (3) (1994) 38–46.
[8] M. Schulze, G.A. Gibson, R.H. Katz, D.A. Patterson, How reliable is a RAID, IEEE COMPCON (Spring) (1989) 118–123.
[9] M. Blaum, J. Brady, J. Bruck, J. Menon, EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures, IEEE Trans. Comput. 44 (2) (1995) 192–202.
[10] P.M. Chen, E.K. Lee, G.A. Gibson, R.H. Katz, D.A. Patterson, RAID: high-performance, reliable secondary storage, ACM Comput. Surveys 26 (2) (1994) 145–185.
[11] C. Park, Efficient placement of parity and data to tolerate two disk failures in disk array systems, IEEE Trans. Parallel Distribut. Syst. 6 (11) (1995) 1177–1184.
[12] G.A. Gibson, Redundant disk arrays: reliable, parallel secondary storages, Ph.D. Thesis, Department of Computer Science, University of California, Berkeley, 1990.
[13] G.A. Gibson, L. Hellerstein, R.M. Karp, R.H. Katz, D.A. Patterson, Coding techniques for handling failures in large disk arrays, Computer Science Technical Report, CSD88-477, University of California, Berkeley, 1988.
[14] F.J. MacWilliams, N.J.A. Sloane, The Theory of Error-Correcting Codes, vol.16, Elsevier, Amsterdam, 1977.
[15] G.A. Alvarez, W.A. Burkhard, L.J. Stockmeyer, F. Cristian, Declustered disk array architectures with optimal and near-optimal parallelism, Proceedings of the 25th Annual International Symposium on Computer Architecture 6 (3) (1998) 109–120.
[16] R.L. Graham, D.E. Knuth, O. Patashnik, Concrete Mathematics: A Foundation for Computer Science, second ed., Addison-Wesley, Reading, MA, 1994, pp. 129–130.



Nam-Kyu Lee received his B.S. and M.S. degree in computer science from the Yonsei University, Seoul, Korea, in 1988 and 1995, respectively. He was employed at Samsung Advanced Institute of Technology from 1988 to 1995, and Samsung Electronics Co. Ltd. in Korea from 1995 to 1997. He is currently a Ph.D. student of computer science at the Yonsei University. His research interests are high performance I/O architecture, advanced computer architecture design, human computer interaction (HCI), and mobile computing.

Sung-Bong Yang received his B.S. degree form the Yonsei University, Seoul, Korea, in 1981, and his M.S. and Ph.D. degrees from the University of Oklahoma, in 1986 and 1992, respectively. He was an adjunct assistant professor at the University of Oklahoma for Fall semester of 1992. He is currently an associate professor of computer science at Yonsei University, Seoul, Korea. His research interests include parallel processing, 3D graphics, and GIS.



Kyoung-Woo Lee received his B.S. and M.S. degree in computer science from Yonsei University, Seoul, Korea, in 1995 and 1997, respectively. Since 1997, he has been working at Digital TV Research Laboratory, LG Electronics Inc. as a research engineer. His research interests include high performance parallel architecture, disk array, multimedia server design, VOD (Video On-Demand), SMIL (Synchronized Multimedia Integration Language), and mobile computing.