디스크 배열 시스템에서의 이중 디스크 오류 허용을 위한 이중 패리티 배치 기법

(Dual Parity Placement Schemes for Tolerating Two Disk-Failures in Disk Array System)

이 남 규 † 한 탁 돈 ††
(Nam-Kyu Lee)(Tack-Don Han)

요 약 최근 처리해야 하는 정보의 양이 급속히 배가됨에 따라 디스크 배열(disk array) 시스템에서 사용하는 디스크의 수가 증가되는 경향을 보이고 있다. 그러나 많은 수의 디스크를 이용하여 디스크 배열을 구성하게 되면 디스크 오류 발생 확률을 높이는 결과로 이어지게 된다. 이 논문에서는 많은 수의 디스크를 사용하는 환경에서도 높은 신뢰성을 제공하기 위하여 DH(Diagonal-Horizontal) 기법이라 불리는 두 가지 형태의 이중 패리티 배치 기법들을 제안한다. 제안한 기법들은 사선 패리티와 수평 패리티를 인코딩하여 이용함으로써 디스크 배열에서 이중 디스크 오류의 복구가 가능하다. DH 기법들의 특징은 알고리즘이 단순하고, N을 임의의 소수라고 할 때 N 또는 N+1개의 디스크를 기반으로 쉽게 구현할 수 있다. 기본적으로 제안하는 기법들은 검사정보 저장을 위해 최적의 디스크 공간을 사용하고, 인코딩이나 디코딩 과정에서 단지 XOR 연산만을 필요로 하기 때문에 하드웨어의 수정 없이 기존의 디스크 배열 시스템에 쉽게 응용할 수 있다. 더욱이 치명적 오류를 방지할 수 있는 두 오류 디스크에 대한 복구 시간은 디코딩에서의 XOR 연산수를 최소화함으로써 신속하게 처리할 수 있다. 결과적으로 제안하는 기법들은 디스크 배열 시스템에서의 치명적 오류 확률을 낮출 수 있는 효과적인 방법이다.

Abstract Recently, as the amount of information processed by computers has been proliferated significantly, the number of disks in a disk array tends to be increased. However, constructing a disk array with such a large number of disks may result in a high probability of an occurrence of disk failures. In order to achieve high reliability under such an environment, two new methods using dual parity placement, called DH(Diagonal-Horizontal) schemes, are presented in this paper. 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. Because the encoding and decoding algorithms of both DH schemes are quite simple and efficient, disk arrays encoded with them can be easily implemented as either N or N+1 disks when N is a prime number. Both DH schemes use almost optimal disk space for storing the redundant information, and they can be easily implemented to an existing disk array system without updating hardware since they require only exclusive-or(XOR) operations. Moreover, both DH schemes can recover rapidly from any double disk failures since they use smaller number of XOR operations for decoding. As a result, DH schemes are efficient methods because the probability of an occurrence of catastrophic failures in a disk array system can also be lowered.

· 이 논문은 1998년도 연세대학교 학술연구비의 지원에 의하여 이루어진 것임.
† 학생회원 : 연세대학교 컴퓨터과학과
nklee@kurene.yonsei.ac.kr
†† 종신회원 : 연세대학교 컴퓨터과학과 교수
hantack@kurene.yonsei.ac.kr
논문접수 : 1999년 11월 29일
심사완료 : 2000년 6월 10일

1. 서 론

디스크들의 그룹으로 구성된 디스크 배열 시스템은 저장된 데이타를 병렬로 접근함으로써 저장장치 시스템의 성능을 높일 수 있을 뿐만 아니라 패리티와 같은 검사정보를 이용하여 디스크 오류 복구가 가능하므로 신

뢰성을 향상시킬 수 있다[1,2,3,4]. 따라서 최근 사용 범위가 확대되고 있고, 디스크 배열 시스템의 성능과 신뢰성을 보다 향상시키기 위하여 효과적인 검사정보의 배치 방법, 병렬 처리, 선인출과 캐싱 등과 같은 다양한 연구들이 진행되고 있다[5,6,7,8,9,10]. 그 중 데이타 손실과 관련되는 신뢰성 문제는 시스템 전체에 막대한 영향을 주기 때문에 성능 향상보다는 그 중요성이 크다고 하겠다[11]. 특히 정보의 바다로 불리는 인터넷과 같은 신규 응용 영역에서는 대량의 문자 정보뿐만 아니라 비디오, 오디오 데이타 등과 같은 대용량의 멀티미디어 데이타 처리에 대한 비중이 높아지고 있으므로 보다 많은 저장 공간이 필요하다[12]. 그러므로 대용량 저장장치가 필연적으로 요구되며 이를 위해 대용량 디스크가 사용되어야 함은 물론 현재 대용량 저장장치로 보편화된 디스크 배열 시스템을 구성하는 디스크 수가 더욱 증가되고 있다[13]. 저장장치에서 디스크 수가 증가된다는 의미는 곧 저장장치의 오류 발생 가능성을 높이는 결과로 이어지게 된다[14].

디스크 시스템의 오류 발생 평균 시간은 MTTF (Mean Time To Failure)로 표현한다. 만약 디스크 배열이 N개의 디스크로 구성되고 각각의 디스크에서 오류 발생 가능한 평균시간을 $MTTF_1$ 라고 할 때 디스크 배열의 오류 발생 평균 시간은 $MTTF_N = MTTF_1/N$ 이므로 디스크 배열을 구성하는 디스크의 수 N이 증가함에 따라 더욱 빈번하게 오류가 발생될 수 있다[4, 14, 15, 16]. 그리고 일단 디스크 그룹 중 하나의 디스크에서 오류가 발생되면 또 다른 디스크에서 오류가 발생될 가능성이 있으므로 오류에 대한 복구가 신속하게 이루어져야 한다. 오류 복구에 소요되는 시간은 MTTR (Mean Time To Repair)로 표현된다. 이 MTTR은 디스크 배열에 복구를 위해 디코딩되는 데이타의 복잡도와 복구해야 되는 데이타의 크기와 밀접한 연관 관계를 갖는다. 데이타의 크기 일정하다고 할 때 디코딩 복잡도에 따라 MTTR을 줄일 수 있고 이에 따라 좋은 성능을 나타낼 수 있다. 따라서 디스크 배열을 구성하는 디스크의 수가 증가하더라도 고신뢰성을 보장할 수 있는 방법은 기존의 일반적인 디스크 배열이 단지 하나의 디스크 오류에 대해서만 복구할 수 있는 중복성을 갖고 있는데 반해 두 개의 디스크에서 오류 발생되더라도 복구 가능할 수 있도록 디스크 배열이 구성되어야 한다[14,15]. 그와 더불어 오류가 발생된 디스크들을 복구하는 시간을 최적화 함으로써 시스템의 신뢰성을 증가시킬 수 있다[4].

디스크 배열에서 중복 디스크 오류를 복구하는 방법과 관련된 대표적 연구로는 확장 해밍 코드를 이용하는 방법[17], 이차원 패리티 기법[17], 리드-솔로몬 코드를 이용하는 방법[18], EVENODD 기법[15], 중복 행렬을 이용한 방법[14], PRIME과 PELPR 기법[19] 등이 있다. 확장된 해밍코드와 이차원 패리티 기법은 전체 디스크 배열에서 검사 디스크가 차지하는 비중이 높고, 검사정보가 특정 검사 디스크에 고정되어 있기 때문에 검사정보 수정 시 병목 현상이 나타날 가능성이 높다. 리드-솔로몬 코드를 이용한 방법은 무한 오류 전파 가능성이 있고 콘볼루션 타입의 코드(convolution type of code)를 사용하기 때문에 데이타에 오버헤드 중복이 있다[17]. 또한, EVENODD 기법은 패리티 연산에 있어서 특정 사선 방향에 위치한 데이타 블록들은 이용하여 사선 S(diagonal S)[17] 값을 구하고, 이를 다시 사선 패리티 연산에 이용하므로 사선 S 값을 구하는데 포함된 데이타 블록이 수정될 경우 모든 사선 패리티가 수정되어야 하므로 성능을 저하시키게 된다. 중복 행렬(Redundancy Matrix)을 이용한 기법은 패리티 디스크에 대한 병목 현상에 대한 문제점을 해결하기 위해 패리티 블록을 분산하는 방법의 일환으로 제안되었고 두 개 이상의 디스크에서 오류 발생 시에도 복구 가능함을 제시하였다. 그러나 알고리즘이 복잡하고 이에 이용되는 중복 행렬을 유지하기 위한 오버헤드가 크다. Alvarez [19]는 PRIME과 PELPR 기법 제안을 통해 디스크 배열에서의 다수 오류 허용 방법을 제안했으나 오류 복구에 필요한 검사정보의 양이 많기 때문에 상대적으로 데이타 저장을 위한 공간이 적다.

본 논문에서는 디스크 배열에서 이중 오류가 발생되어도 복구 가능한 두 가지 형태의 패리티 배치 기법을 제안한다. 제안하는 두 가지 방법 중 하나인 DH_N 기법은 별도의 패리티 디스크 없이 수평 패리티와 사선 패리티가 정보 데이타와 함께 순환되면서 저장된다. 또한 DH_{N+1} 기법은 기본적으로 DH_N 기법과 유사하게 수평 패리티와 사선 패리티를 이용하지만 수평 패리티 저장을 위해 디스크 하나를 추가하여 확장함으로써 DH_N 기법 보다 나은 성능을 보인다. 제안된 두 DH 기법은 하드웨어의 수정 없이 기존의 디스크 배열에서 사용하는 간단한 XOR 연산을 이용하고 단순히 소프트웨어적으로 구현할 수 있다. 그리고 오류 복구를 위한 검사정보 저장에 있어서 최적에 가까운 공간을 사용하면서도 동시에 두 디스크에서 오류 발생 시에 기존에 제안된 기법보다 알고리즘이 간단하고, 이중 오류 시 복구 시간을 최소화할 수 있는 효과적인 데이타 및 패리티 블록 배치 기법이다.

서론에 이어 2장에서는 DH_N 기법의 데이타 및 패리티 블록의 저장 방법과 이중 디스크 오류에 따른 복구 방법에 대해 설명한다. 3장에서는 DH_{N+1}기법의 데이타 및 패리티 블록의 저장 방법과 오류 복구 방법에 대해 서술한다. 그리고 4장에서는 제안한 기법에 대한 효과를 기존의 대표적인 방법과 비교 분석하고, 마지막 장에서 결론을 맺는다

2. DHN 기법

이 장에서는 DH_N 기법의 패리티 배치 방법과 디스크 오류 발생 시 복구 방법에 대해 설명한다. DH_N 기법에서의 패리티 블록들은 디스크 배열을 구성하는 모든 디스크에 분산되어 저장되므로 동시에 여러 블록들의 쓰기 시 패리티 수정에 따른 병목을 피할 수 있다. 기본적으로 디스크 배열 중 하나의 디스크에서 오류가 발생된 경우 수평 오류 정정 그룹(Horizontal Error Correction Group)을 이용하거나 사선 오류 정정 그룹(Diagonal Error Correction Group)을 이용하여 쉽게 복구할 수 있다. 그리고 두 개의 디스크에서 동시에 오류 발생 시에는 사선 오류 정정 그룹과 수평 오류 정정 그룹을 반복적으로 적용하여 오류가 발생된 두 디스크의 모든 블록들을 차례로 복구할 수 있다.

2.1 데이타와 패리티 배치

DH_N 기법은 소수 N을 기반으로 디스크 배열이 구성된다. 디스크 배열을 구성하는 디스크들은 N 개로 이루어지고, 각 디스크에는 (N-1) 개의 논리적인 블록들이 연속적으로 저장된 형태를 갖는다. 결국 디스크 배열의 기본적인 구조는 (N-1) x N 행렬로서 정의된다. 만약 행렬에서 각각의 블록을 A[i,j]로 나타낸다고 하면 i와 j의 범위는 각각 0 i N-2, 0 j N-1로 나타내고, 이 블록은 디스크 j에서의 논리적인 i번째 블록을 의미한다.

디스크 배열을 나타내는 (N-1) x N 행렬에서 각 행들을 수평 오류 정정 그룹으로 정의된다. 수평 오류 정정 그룹에는 데이타 블록들과 한 개의 패리티 블록으로 구성된다. 각 그룹의 패리티는 첫 번째 디스크의 (N-1)번째 블록으로부터 우상 방향으로 (N-1)번째 디스크의 첫 번째 블록까지 사선 방향으로 위치한 블록들에 저장된다. 그리고 대각선으로 배치된 블록들은 사선 오류 정정 그룹으로 정의된다. 사선 오류 정정 그룹에서 사선 패리티는 각 디스크의 (N-1)번째 행에 저장되며, 각 사선 오류 정정 그룹은 사선 패리티를 포함하여 패리티가 저장된 블록으로부터 우측 사선 방향에 위치한 블록들의 순환 집합으로 구성된다.

그림 1은 N=7일 때 DH_N 기법에서 데이타와 수평 패리티 및 사선 패리티의 위치를 나타낸다. 디스크 배열에서 각각의 데이타 블록이 속한 수평 오류 정정 그룹과 사선 오류 정정 그룹을 표현하기 위하여 (i,j) 형태로 표현한다고 할 때 앞의 숫자 i는 수평 오류 정정 그룹을 나타내고, 뒤의 숫자 j는 동일한 사선 오류 정정 그룹에 속함을 의미한다. 수평 패리티 H_i에는 i 그룹에 속한 데이타 블록들의 XOR 연산 결과가 저장되고, 사선 패리티 D_j에는 j 그룹의 패리티가 위치한다. 그림 1에서 D_0는 수평 패리티 블록들의 XOR 연산 값이 저장되고, H_5는 다른 모든 사선 패리티 블록들의 XOR 연산 결과이다. 따라서 이들 두 값은 결국 서로 중복되지 않는 모든 데이타 블록들의 XOR 연산 결과를 뜻하므로 서로 항상 동일함을 의미한다.

(0, 2)	(0, 3)	(0, 4)	(0, 5)	(0, 6)	H_0	(0, 1)
(1, 3)	(1, 4)	(1, 5)	(1, 6)	H_1	(1, 1)	(1, 2)
(2, 4)	(2, 5)	(2, 6)	H_2	(2, 1)	(2, 2)	(2, 3)
(3, 5)	(3, 6)	H_3	(3, 1)	(3,2)	(3, 3)	(3, 4)
(4, 6)	H_4	(4, 1)	(4, 2)	(4, 3)	(4, 4)	(4, 5)
H_5/D_0	D_1	D_2	D_3	D_4	D_5	D_6

그림 1 DH_N 기법에서 패리티 및 데이타 배치(N=7).

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

그림 2 DH_N 기법에서 패리티 및 데이타 배치(N=7).

우선 앞으로 전개되는 내용의 행렬 A[i,j]에서 수식 표현을 단순하게 하고, 인덱스 i,j 값의 범위를 제한하기 위하여 나머지를 구하는 연산 기호(%)를 사용하는 다음과 같은 표현은 동일한 의미를 갖는 것으로 정의한다.

$$A[i\%N,\ j\%N] == A[i,j]^{\%N}$$

소수 N에 대하여 그림 1에서 수평 패리티 H_i와 사선 패리티 D_j를 XOR 연산을 기반으로 하여 구하는 수식을 나타내면 다음과 같다.

[예 1] N=7인 경우의 데이타 및 패리티 블록의 저장 방법

다음의 그림 2는 디스크 배열이 7개의 디스크, 즉 N=7이라 가정하고 각 데이타 블록에 임의의 이진수 데이타가 저장되어 있는 상태를 나타낸 것이다.

앞의 그림2의 빈 패리티 블록들에 식(1)과 식(2)를 이

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***

그림 3 DH_N 기법의 데이타 및 패리티 배치(N=7).

용하여 수평 패리티와 사선 패리티를 구하면 그림 3과 같다.

2.2 오류 복구

제안하는 DH_N 기법에서는 동시에 두 개의 디스크 오류까지 복구가 가능하며, 오류 복구는 디스크 배열에 인코딩되어 있는 수평 패리티와 사선 패리티가 포함된 오류 정정그룹을 이용하여 이루어진다. 한 개의 디스크에서 오류가 발생되었을 경우 수평 오류 정정 그룹 또는 사선 오류 정정 그룹을 적용하여 복구할 수 있다. 그리고 두 개의 디스크에서 동시에 오류가 발생된 경우 사선 오류 정정 그룹과 수평 오류 정정 그룹을 교대로 적용하여 두 디스크 오류를 복구 할 수 있다. 먼저 디스크 배열의 행 방향을 기준으로 보면 두 블록에서 오류가 발생되었으므로 수평 오류 정정 그룹을 이용한 복구는 불가능하다. 따라서 사선 오류 정정 그룹을 이용하게 되는데 이 그룹들 중에는 한 블록만이 오류가 발생된 사선 오류 정정 그룹이 존재한다. 따라서 이 그룹을 선택하여 오류 복구의 시발점으로 삼는다. 일단 선택된 사선 오류 정정 그룹을 이용하여 한 블록을 복구하게 되면 복구한 블록이 포함된 수평 오류 정정 그룹에는 다른 하나의 블록만이 오류 상태에 놓이게 되므로 이 블록이 속하는 수평 오류 정정 그룹을 이용하여 복구할 수 있다. 이렇게 사선 오류 정정 그룹과 수평 오류 정정 그룹을 반복적으로 이용하여 복구를 진행하면 결국 오류가 발생한 두 디스크의 모든 블록들을 복구할 수 있다. 다음은 임의의 디스크 i와 디스크 j에서 오류 발생되었다는 가정 하에 사선 오류 정정 그룹과 수평 오류 정정 그룹을 반복적으로 이용하여 오류 복구 과정을 수행하는 알고리즘이다.

알고리즘 1: DH_N 기법에서의 두 디스크 오류 복구를 위한 알고리즘 (0 i < j (N-1))

디스크 i와 디스크 j에서 오류 발생되었다고 가정하자. 만약 두 디스크 i와 j의 관계를 i < j라고 하면 두 디스크의 차이 값은 k = j - i로 나타낼 수 있다. 그리고 사선 패리티 D_i 속해있는 사선 오류 정정 그룹을 $DECG_i$ 라고 하고, 수평 패리티 H_i 가 속해있는 수평 오류 정정 그룹을 $HECG_i$ 라고 정의하자. 오류 정정을 위해 선택되는 시작점은 디스크 j의 n번째 블록(n = N-1-k)이 된다. 이 블록은 $DECG_{i+1}$ 에 속하게 되는데 이 그룹에는 단 한 개의 오류 블록만이 존재한다. 따라서 이 그룹에 속하는 디스크 j의 오류 블록을 복구할 수 있다. 다음에는 디스크 i의 오류 블록을 이 블록이 속하는 수평 오류 정정 그룹을 통해 복구할 수 있다. 다시 디스크 i의 복구 블록이 속하는 사선 패리티 오류 정정 그룹을 통해 디스크 j의 오류 블록을 복구할 수 있고, 수평 오류 정정 그룹을 이용해 디스크 i의 오류 블록을 복구 할 수 있다. 이 블록은 다시 사선 패리티 그룹의 한 블록이므로 그 그룹의 오류 블록을 복구할 수 있으며 같은 방법으로 모든 블록에 적용된다. 이러한 방법으로 소수 N 개로 이루어진 디스크 배열에서 임의의 두 디스크에서 오류 발생 시 오류블록을 모두 복구할 수 있다.

[예 2] N=7일 때, 두 개의 디스크에서 오류 발생 시 복구 방법 다음의 그림 4는 이전에 예시한 그림 3에서 두 번째와 네 번째 디스크가 오류인 상황을 보여주고 오류 블록을 '?'로 표시하였다.

알고리즘 1에서 N=7이고, 두 오류 디스크는 i=1, j=3 이므로 k = j-i = 2 이다. 이 값들은 step 3과 step 4의 수식 n에 대입하면 n=(4-2m)%7이 되고 다음과 같은 수식으로 표현할 수 있다.

$$A[(4-2m),3]^{\%7} = \bigoplus_{l=0}^{5} A[5-l,(2-2m+l)]^{\%7}, \quad \text{where } (2-2m+l)\%7 \neq 3 \tag{3}$$

$$A[(4-2m),1]^{\%7} = \bigoplus_{l=0}^{6} A[(4-2m),l]^{\%7}, \quad \text{where } l \neq 1 \tag{4}$$

m	연산	값
0	A[4,3]=A[5,2]⊕A[3,4]⊕A[2,5]⊕A[1,6]⊕A[0,1]	1
	A[4,1]=A[4,0]⊕A[4,2]⊕A[4,3]⊕A[4,4]⊕A[4,5] A[4,6]	1
1	A[4,3]=A[5,2]⊕A[3,4]⊕A[2,5]⊕A[1,6]⊕A[0,1]	1
	A[4,1]=A[4,0]⊕A[4,2]⊕A[4,3]⊕A[4,4]⊕A[4,5]⊕A[4,6]	1
2	A[0,3]=A[5,5]⊕A[4,6]⊕A[3,0]⊕A[2,1]⊕A[1,2]	1
	A[0,1]=A[0,0]⊕A[0,2]⊕A[0,3]⊕A[0,4]⊕A[0,5]⊕A[0,6]	0
3	A[5,3]=A[4,4]⊕A[3,5]⊕A[2,6]⊕A[1,0]⊕A[0,1]	1
	A[5,1]=A[5,0]⊕A[5,2]⊕A[5,3]⊕A[5,4]⊕A[5,5]⊕A[5,6]	1
4	A[3,3]=A[5,1]⊕A[4,2]⊕A[2,4]⊕A[1,5]⊕A[0,6]	1
	A[3,1]=A[3,0]⊕A[3,2]⊕A[3,3]⊕A[3,4]⊕A[3,5]⊕A[3,6]	0
5	A[1,3]=A[5,6]⊕A[4,0]⊕A[3,1]⊕A[2,2]⊕A[0,4]	0
	A[1,1]=A[1,0]⊕A[1,2]⊕A[1,3]⊕A[1,4]⊕A[1,5]⊕A[1,6]	1

위의 식(3)과 식 (4)를 이용하여 변수 m을 증가시키면서 오류가 발생한 두 디스크 i와 j의 오류 블록 의 데이타를 다음과 같이 복구할 수 있다.

위 과정을 거쳐 복구된 값을 그림 4의 손상된 블록 '?' 부분에 대치하면 그림 3에서와 같은 복구된 결과를

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

그림 4 두 개의 디스크에서 오류가 발생한 경우(N=7).

얻을 수 있다.

2.3 오류 복구의 증명

DH_N 기법에서의 오류 복구 방법에 대한 올바름을 증명하기 위하여 다음과 같은 패리티 블록과 오류 정정 그룹에 관한 세 가지 정의를 사용한다.

[정의 1]

임의의 N-1 개의 데이타 블록에 1개의 XOR 패리티 블록을 추가하여 N개로 이루어진 오류 정정 그룹을 구성할 경우 만약 1 개의 데이타 블록이 손실되어도 나머지 (N-1) 개의 데이타를 XOR하여 손실된 블록을 복구할 수 있다.

[정의 2]

N 개의 디스크로 이루어진 디스크 배열 (N-1) x N 행렬에서 임의의 수평 오류 정정 그룹 i($HIECG_i$) 는 동일한 i 행에 놓인 N개의 블록들의 집합으로 구성된다.

$$HECG_i = (A[i,0], A[i,1], \ldots A[i,N-2], A[i,N-1])$$

$HECG_i$ 에 속한 수평 패리티 블록 H_i는 식 (1)에서 나타난 바와 같이 (N-1)개의 데이타 블록들을 XOR하여 구해진다.

$$H_i = A[i,N-2-i]$$
$$= A[i,0] \; A[i,1] \; \ldots \; A[i,N-3-i] \; A[i,N-1-i] \; \ldots \; A[i,N-2] \; A[i,N-1]$$

[정의 3]

N개의 디스크로 이루어진 디스크 배열 (N-1) x N 행렬에서 임의의 사선 오류 정정 그룹 j, ($DECG_j$)는 대각선상으로 인접한 (N-1) 개 블록들의 집합으로 구성된다.

$$DECG_j = (A[N-2,j]^{\%N}, A[N-3,(i+1)]^{\%N}, \ldots A[1,(i+N-3)]^{\%N}, A[0,(i+N-2)]^{\%N})$$

앞의 식(2)는 사선 패리티를 구하는 식으로 이 식에서 j의 범위는 0부터 (N-3)이므로 사선 오류 정정 그룹은 (N-2) 개의 데이타 블록과 이들 데이타 블록들을 XOR하여 구해진 사선 패리티 블록을 포함하여 총 (N-1) 개 블록으로 구성된다.

$DECG_j$에 속한 사선 패리티 블록 D_i는 식(2)에서 나타난 바와 같이 (N-2)개의 데이타 블록을 XOR하여 구해진다.

$$D_i = A[N-2,i]$$
$$= A[N-3,(i+1)]^{\%N} A[N-4,(i+2)]^{\%N} \ldots A[1,(i+N-3)]^{\%N} A[0,(i+N-2)]^{\%N}$$

데이타 블록들은 우측 사선 방향으로 분산되며 D_i를 기준으로 행은 1씩 감소하고 열은 1씩 증가하면서 이루어진 (N-2)개로 구성된다. 따라서 임의의 $DECG_i$ 사선 오류 정정 그룹의 블록들은 순서 상으로 D_i 사선 패리티 블록의 바로 이전인 디스크 (j-1)을 제외한 다른 모든 디스크에 분산되어 구성된다.

DH_N 기법의 오류 정정 방법을 증명하기 위하여 다음과 같은 보조정리들을 정리하고 증명하고자 한다.

[보조정리 1] DH_N 기법으로 인코딩된 디스크 배열에서 디스크 i 와 j에서 오류가 발생되었을 경우 $DECG_{i+1}$ 오류 정정 그룹은 오직 한 개의 오류 블록만을 포함하고 있고, 이 블록은 k=j-1(j > i) 일 때 디스크 j에 속한 $A[N-1-k,j]^{\%N}$ 블록이다.

(증명)

사선 오류 정정 그룹은 사선 패리티 블록이 위치한 바로 이전 디스크를 제외한 다른 모든 디스크에 분산되어 저장되므로 정의 3에 의해 $DECG_{i-1}$ 오류 정정 그룹에 속한 블록들은 오류가 발생된 디스크 i를 제외한 다른 모든 디스크에 균일하게 분산되어 있다. 따라서 $DECG_{i+1}$ 사선 오류 정정 그룹에서는 디스크 j의 한 블록만이 손상된 상태이므로 정의 1에 의해 이 블록을 복구할 수 있다. 디스크 j의 복구될 첫 번째 블록을 $DECG_{i+1}$ 오류 정정 그룹에서 살펴보면 k번째 블록에 해당한다. 이는 $A[N-2-(k-1),i+1+(k-1)]^{\%N}$ 이고, k=j-i 일 경우 이 블록은 $A[N-1-k,j]^{\%N}$ 이 된다.

$$A[N-2-(k-1),i+1+(k-1)]^{\%N} = A[N-1-k,i+k]^{\%N}$$
$$= A[N-1-k,((j-k)+k)]^{\%N}$$
$$= A[N-1-k,[(j-k)+k]]^{\%N}$$
$$= A[N-1-k,j]^{\%N}$$

[보조정리 2] 수평 오류 정정 그룹을 이용하여 디스크 i의 $A[N-1-k,i]^{\%N}$ 블록은 복구된다.

(증명)

보조정리 1에서 사선 오류 정정 그룹에서 선택된 오류 블록, $A[N-1-k,j]$는 정의 1에 의하여 복구할 수 있다. 그러므로 이 오류 블록이 속하는 수평 오류 정정 그룹에서 오류 블록의 수가 두 개에서 한 개로 감소하였다. 따라서 같은 수평 오류 정정 그룹에 포함된 디스크 i의 오류 블록 $A[N-1-k,i]$을 정의 1에 의해 복구할 수 있다.

[보조정리 3] $A[N-1-k,i]^{\%N}$ 은 $DECG_{(i+1-k)\%N}$ 사선

오류 정정 그룹에 포함된다. 따라서 이 그룹에 속한 또 하나의 오류 블록을 복구할 수 있다.

(증명)

우선 정의 3에 의해 $DECG_i$ 오류 정정 그룹은 다음과 같이 나타낼 수 있다.

$(A[N-2,i]^{\%N}, A[N-3,i+1]^{\%N}, A[N-4,i+2]^{\%N}, \ldots, A[1,i+N-3]^{\%N}, A[0,i+N-2]^{\%N})$

이 그룹에서 k번째 블록은 $A[N-2-(k-1),i+(k-1)]^{\%N}$ 이므로 $DECG_{(i+1-k)\%N}$의 k번째 블록은 $A[N-2-(k-1),i+1-k+(k-1)]^{\%N}$ 이고, 이것은 $A[N-1-k,i]^{\%N}$ 임을 알 수 있다. 따라서 보조정리 2에서 복구된 블록은 $DECG_{(i+1-k)\%N}$ 오류 정정 그룹의 k번째 블록이고, 2k번째 블록은 $A[N-2-(2k-1),i+1-k+(2k-1)]^{\%N}$ 이며 k=j-i이므로 디스크 j의 한 블록인 $A[N-2-(k-1),j]^{\%N}$ 이다. 그러므로 보조정리 2에서 $DECG_{i+1}$ 오류 정정 그룹을 이용하여 복구한 뒤 다음 순서에 복구에 이용되는 오류 정정 그룹은 $DECG_{(i+1-k)\%N}$임을 알 수 있다. 그리고$DECG_{(i+1-k)\%N}$ 의 한 오류 블록이 복구되었으므로 나머지 오류 블록도 복구될 수 있다.

[보조정리 4] 디스크 i와 디스크 j의 복구를 위해 적용될 사선 오류 정정 그룹의 순서는 알고리즘 1에서와 같이 다음과 같음을 알 수 있고, 이 그룹들은 서로 중복되지 않는다.

$DECG_{(i+1)\%N}$, $DECG_{(i+1-k)\%N}$, $DECG_{(i+1-2k)\%N}$, ... , $DECG_{(i+1-(N-2)k)\%N}$, $DECG_{(i+1-(N-1)k)\%N}$

(증명)

앞의 사선 오류 정정 그룹의 아래 첨자만을 표현하면 다음과 같다.

$(i+1)\%N, (i+1-k)\%N, (i+1-2k)\%N, \ldots,$
$(i+1-(N-2)k)\%N, (i+1-(N-1)k)\%N$

위의 순서에서 (i+1)은 임의의 고정된 숫자이므로 그것을 제외한 나머지 부분을 차례로 나타내면 다음과 같이 된다.

$0\ \%\ N, k\ \%\ N, 2k\ \%\ N, \ldots, (N-2)k\ \%\ N,$
$(N-1)k\ \%\ N$

위와 같은 N개의 숫자들은 k와 N이 상대적인 소수이면, 즉 상대적으로 나누어 떨어지지 않는다면 중복되지 않는 숫자로 구성된다는 사실이 증명되었다[20]. 따라서 N은 소수이고, k는 1 k N-1 이므로 N과 k는 상대적인 소수이며, 위의 숫자들은 N개로 구성되어 있으므로 이들을 낮은 순서로 다시 배열한다면 다음과 같은 N개의 숫자로 이루어짐을 알 수 있다.

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

결국 임의의 수 (i+1)은 1 (i+1) N-1 사이의 특정 숫자이므로 (i+1)%N, (i+1-k)%N, (i+1-2k)%N, ... , (i+1-(N-2)k)%N, (i+1-(N-1)k)%N 은 중복되지 않는 N개의 숫자들로 구성된다.

앞의 정의들과 보조정리들을 이용하여 다음과 같은 정리 1의 임의의 두 디스크 오류를 복구할 수 있다.

[정리 1]

디스크 배열 시스템이 DH_N 기법으로 인코딩되어 있다면 임의의 두 디스크 i, j (0 i < j (N-1))에서 오류가 발생되었을 경우 모두 복구할 수 있고, 이는 알고리즘 1과 같다.

(증명)

먼저 디스크의 수를 N이라 할 때 오류가 발생된 두 디스크를 각각 i와 j라고 가정하자. 그리고 두 디스크의 관계를 j > i라 하고 k=j-i로 가정하자.

모든 수평 오류 정정 그룹에는 각각 두 개씩의 오류가 발생되었으므로 일단 수평 오류 정정 그룹을 이용할 수 없다. 그러므로 사선 오류 정정 그룹을 이용하여 복구를 시작할 수 있다.

보조정리1과 보조정리 2에서 $DECG_{i+1}$ 오류 정정 그룹을 이용해 디스크 j에 속한 한 오류 블록을 복구하고, 그 블록이 포함된 수평 오류 정정 그룹을 이용해 디스크 i의 한 오류 블록을 복구할 수 있다. 그리고 보조정리 3에서는 다음으로 적용될 사선 오류 정정 그룹이 $DECG_{(i+1-k)\%N}$ 이고, 이 사건 오류 정정 그룹에 속한 오류블록은 복구된다는 사실을 보여 주었다. 이런 방식으로 사건 오류 정정 그룹과 수평 오류 정정 그룹을 교대로 적용할 수 있고, 알고리즘 1에서와 같이 디스크 i와 디스크 j의 복구를 위해 적용될 사선 오류 정정 그룹의 순서는 다음과 같이 됨을 알 수 있다.

$DECG_{(i+1)\%N}$, $DECG_{(i+1-k)\%N}$, $DECG_{(i+1-2k)\%N}$, ... ,
$DECG_{(i+1-(N-2)k)\%N}$, $DECG_{(i+1-(N-1)k)\%N}$

보조정리 4에 의하여 위의 사선 오류 정정 그룹들은 서로 다른 그룹으로 $DECG_0$, $DECG_1$, ..., $DECG_{N-1}$을 이룸을 알 수 있는데 이를 통하여 N 개의 사선 오류 정정 그룹을 중복 없이 방문할 수 있음을 알 수 있다. 그러므로 디스크 배열 시스템에서 임의의 두 개의 디스크들의 오류에 대하여 디스크들의 복구 과정에서 사선 오류 정정 그룹들은 정확하게 모든 그룹들을 중복 없이 방문함을 알 수 있다. 그리고 각 디스크는 (N-1) 개의 블록으로 구성되므로 마지막으로 적용될 사선 패리티 그룹을 적용하기 전에 두 디스크의 모든 블록들을 복구할 수 있다.

2.4 오류 복구 방법의 병렬성

오류 복구 방법의 병렬성이라 함은 복구 과정을 순차

적으로 해야만 하는 것이 아니라 그 과정을 동시에 진행할 수 있음을 말한다. 다음의 정리 2는 오류 복구를 병렬로 수행할 수 있음을 보여 준다.

[정리 2] DH_N 기법으로 인코딩된 (N-1) x N 디스크 배열 시스템에서 두 개의 디스크들이 오류일 경우 단지 한 개의 블록만이 오류인 사선 오류 정정 그룹이 두 개 존재한다.

(증명)

디스크 배열에서 각각의 열들은 디스크를 의미하고, 각 디스크는 (N-1)개의 블록으로 이루어져 있으므로 두 개의 디스크가 오류일 경우 총 오류 블록의 수는 2(N-1)이다. 한편 디스크 배열에는 N개의 사선 오류 정정 그룹이 있고, 각 그룹에 속한 블록들은 각각 다른 디스크에 배치되어 있다. 따라서 사선 오류 정정 그룹에는 최대 2N개의 오류 블록이 있을 수 있다. 그러나 디스크 배열의 오류 블록의 수는 정확히 2(N-1) 이고, 임의의 사선 오류 정정 그룹에는 적어도 한 개의 오류 블록이 포함되어 있으므로 N 개의 사선 오류 정정 그룹들 중에는 한 개의 오류 블록만을 포함하고 있는 사선 오류 정정 그룹이 두 개 존재한다.

사선 오류 정정 그룹과 수평 오류 정정 그룹을 이용하는 기존의 EVENODD 기법인 경우 사선 패리티와 수평 패리티를 통한 복구가 순차적이어야만 한다. 그러나 본 논문에서 제안하는 DH_N 기법은 정리 2에서와 같이 오류 복구 진행을 시작할 수 있는 블록이 두 개이므로 양방향에서 복구를 병렬적으로 수행할 수 있고, 디스크 배열 시스템 복구에 소요되는 시간을 크게 줄일 수 있다. 따라서 두 디스크의 오류 복구 중에 또 다른 하나의 오류가 발생되어 디스크 배열 전체가 복구 불가능한 상태로 빠지는 치명적인 오류의 발생 가능성을 크게 낮출 수 있다. 다음의 그림 5는 앞 절에서 살펴본 예로 두 번째 디스크와 네 번째 디스크에서 오류가 발생한 후 복구 과정의 순서에 따라 각 블록에 번호를 붙인 것이다

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

그림 5 두 개의 디스크에서 오류 발생 시 복구 과정 순서(N=7).

이 과정은 i=1이고 j=3인 경우로 디스크 i+1, 즉 세 번째 디스크의 사선 패리티 블록을 이용하여 디스크 j의 해당 블록을 복구하는 것이 (1)블록에 해당되고 이 블록이 속하는 수평 패리티 그룹을 통해 디스크 i의 블록을 복구하는 것이 (2)블록이다. 같은 과정으로 (3)과 (4), (5)와 (6) 등등을 계속 복구해 나간다. 여기서 홀수 번째 블록들은 사선 오류 정정 그룹을 이용해 복구되고 짝수 번째 블록들은 수평 오류 정정 그룹을 통해 복구되어지는 것을 알 수 있다. 그런데 이 과정은 역으로도 진행될 수 있다. 즉, 디스크 (j+1), 본 예에서는 다섯 번째 디스크로부터 시작되는 사선 오류 정정 그룹을 이용하여 디스크 i의 오류 블록을 복구할 수 있음을 알 수 있는데 이 블록이 바로 (12)블록이다. 복구된 (12)블록과 (11)블록은 동일한 수평 오류 정정 그룹에 속하므로 (11) 블록은 복구될 수 있고, 마찬가지로 이번에는 (10) 블록을 그 블록이 속한 사선 오류 정정 그룹을 통해 복구할 수 있다. 다시 말하면 (12),(11),(10),...,(1)의 순서로 복구가 될 수 있음을 의미한다. 그러므로 이러한 과정은 디스크 i+1 로부터 시작되는 사선 오류 정정 그룹에 속한 블록부터 복구되는 순서의 역 방향이고, 중간 과정에 있어서 중첩되는 부분은 없다. 따라서 추가적인 하드웨어 장치를 이용하여 복구 과정을 양방향으로 병렬로 진행시킬 경우 복구 시간을 크게 줄일 수 있다.

3. DHN+1 기법

본 장에서는 다른 하나의 DH 기법인 DH_{N+1} 기법에 대해 설명한다. 기본적으로 DH_{N+1} 기법은 DH_N 기법과 유사하게 수평 패리티와 사선 패리티를 이용하지만 한 개의 추가적인 패리티 디스크를 확장하여 수평 패리티를 위치시킴으로써 성능 향상을 가져올 수 있다. 저장되는 수평패리티는 한 디스크에 고정되는 것이 아니라 배열 단위로 순환되므로 패리티 수정으로 인한 병목은 큰 문제가 되지 않는다.

3.1 데이타와 패리티 배치

DH_{N+1} 기법에서는 소수 N 개의 디스크에 하나의 디스크를 추가한 (N+1) 개의 디스크로 구성된다. 이 디스크들로 디스크 배열을 구성할 경우 DH_N 기법에서와 같이 각 디스크에는 (N-1) 개의 논리적인 블록들이 연속적으로 저장된 형태를 갖는다. 기본적인 디스크 배열 구조는 (N-1) x (N+1) 행렬로 정의된다. 여기서 각 행들을 수평 오류 정정 그룹으로 정의되고 패리티 값은 마지막 디스크, 즉 (N+1)번째 디스크에 저장된다. 사선 패리티는 DH_N 기법에서와 마찬가지로 각 디스크의 (N-1)번째 블록에 저장하도록 한다.

데이타 블록들의 위치를 (i,j)로 표현할 때 i는 수평 패리티 그룹을, j는 사선 패리티 그룹을 의미한다. H_i는

(0, 2)	(0, 3)	(0, 4)	(0, 5)	(0, 6)	(0, 0)	(0, 1)	H_0
(1, 3)	(1, 4)	(1, 5)	(1, 6)	(1, 0)	(1, 1)	(1, 2)	H_1
(2, 4)	(2, 5)	(2, 6)	(2, 0)	(2, 1)	(2, 2)	(2, 3)	H_2
(3, 5)	(3, 6)	(3, 0)	(3, 1)	(3, 2)	(3, 3)	(3, 4)	H_3
(4, 6)	(4, 0)	(4, 1)	(4, 2)	(4, 3)	(4, 4)	(4, 5)	H_4
D_0	D_1	D_2	D_3	D_4	D_5	D_6	*E*

그림 6 DH_{N-1} 기법에서 패리티 및 데이타 배치(N=7).

i 수평 오류 정정 그룹에 속한 패리티 블록을 의미하고, D_i는 j 사선 오류 정정 그룹에 속한 패리티 블록을 나타낸다. 다음의 그림 은 N=7인 경우에 DH_{N-1} 기법의 데이타 및 패리티 배치를 나타낸 예이다.

위의 그림 6에서 블록 *E*는 인코딩 이후에도 비어 있는 상태로 남아 있는 공간이다. 웹브라우저의 캐쉬에 저장되는 임시 파일이나 스왑 파일(swap file)과 같은 일시적인 파일들은 오류와 무관하므로 이 블록은 검사정보를 필요로 하지 않는 데이타를 위한 공간으로 사용될 수 있다.

다음의 식들은 디스크 배열을 행렬 B로 가정하고, XOR연산을 이용하여 각 블록들의 패리티 값을 구하는 방법으로 식 (3)은 수평 패리티, 식 (4)는 사선 패리티를 구하는 식이다.

[예 3] N=7일 때 DH_{N-1}기법에서의 패리티 블록 저장 방법

다음의 그림 7은 N=7일 때, 즉 8 개의 디스크로 디스크 배열을 구성할 경우에 각 데이타 블록에 임의의 이진수 데이타가 저장되어 있는 예제이다.

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	

그림 7 DH_{N-1} 기법의 데이타 배치(N=7).

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*

그림 8 DH_{N-1}기법의 데이타 및 패리티 배치(N=7).

식(3)과 식(4)를 이용하여 각각 수평 패리티와 사선 패리티를 구하고 그림 7에 저장하면 그림 8과 같이 인코딩이 완료된 형태를 얻을 수 있다.

3.2 오류 복구

하나의 디스크에서 오류 발생 시에는 사선 오류 정정 그룹이나 수평 오류 정정 그룹을 이용하여 간단히 복구할 수 있다. 반면에 두 개의 디스크에서 오류 발생 시에는 손상된 두 디스크 중 한 디스크에 수평 패리티 블록이 포함되어있는지에 따라 다르게 처리된다. 두 디스크 i와 j에서 오류가 발생되었다고 가정하자. 만약 디스크 j에 수평 패리티 블록이 포함되어 있지 않으면, 즉 j (N-1), DH_N 기법에서와 마찬가지 방법으로 복구할 수 있다. 우선 두 디스크의 차이 값인 k를 구한다. 이때 디스크 j의 N-1-k번째 블록을 이 블록이 포함된 사선 오류 정정 그룹의 손상되지 않은 나머지 데이타 블록을 이용하여 복구한다. 다음에는 디스크 i의 N-1-k 블록을 여기에 속하는 수평 오류 정정 그룹을 통해 복구한다. 이와 같이 사선 오류 정정 그룹과 수평 오류 정정 그룹을 교대로 반복해서 이용하여 오류가 발생한 두 디스크의 모든 블록을 차례로 복구한다. 그리고 오류가 발생한 두 디스크 중 한 디스크에 수평 패리티 블록이 포함되어 있으면, 즉 j=N 일 경우 사선 오류 정정 그룹들에는 디스크 i의 한 블록만이 손상된 상태에 있다. 따라서 사선 오류 정정 그룹을 이용하여 디스크 i의 오류 블록을 복구하고 디스크 j의 오류 블록을 수평 오류 정정 그룹을 이용하여 복구한다. 이렇게 사선 오류 정정 그룹과 수평 오류 정정 그룹을 반복해서 이용하여 모든 블록들을 복구할 수 있다. 또한 사선 오류 정정 그룹을 연속적으로 이용하여 디스크 i의 블록들을 모두 복구한 후 수평 오류 정정 그룹들을 적용하여 디스크 j의 블록들을 복구할 수도 있다.

다음의 알고리즘 2-1은 디스크 j가 j (N-1)인 경우를 그리고 알고리즘 2-2는 디스크 j가 j = N인 경우를 가정하고 오류 복구 과정을 나타낸 것이다.

알고리즘 2-1: DH_{N-1} 기법을 이용한 이중 디스크 오류에 따른 복구 알고리즘 (0 i < j (N-1))

step 1: $k = j - i$ **and** $m = 0$

step 2: $x = (N-1-k-m*k) \% N$

$y = (i+1-m*k)$

step 3: $B[x,j] = \bigoplus_{l=0}^{N-1} B[(N-2-l, y+l)]^{T_4 N}$, **where** $(y+l) \% N \neq j$

step 4: **if** $(x == (N-2))$

then $B[x,i] = \bigoplus_{l=0}^{N-1} B[x,l] \oplus \bigoplus_{h=0}^{N-3} B[h,N]$

else $B[x,i] = \bigoplus_{l=0}^{N} B[x,l]$ **where** $l \neq i$

step 5: **if** $(m < (N-1))$

then $m = m + 1$ **and go to step 2**

else stop

알고리즘 2-2: DH_{N-1} 기법을 이용한 이중 디스크 오

류에 따른 복구 알고리즘 (0 i (N-1), j = N)

step 1: $m = 0$

step 2: $y = (i+2+m)$

step 3: $B[m,i] = \bigoplus_{l=0}^{N-2} B[N-2-l, y+l \ \% \ N]$, **where** $((y+l) \ \% \ N) \neq i$

step 4: $B[m,j] = \bigoplus_{l=0}^{N-1} B[x,l]$

step 5: if $(m < (N-2))$

then $m = m + 1$ **and go to step 2**

else stop

[예 4] N=7일 때, 두 개의 디스크에서 오류 복구

다음의 그림 9는 그림 8에서 두 번째와 네 번째 디스크가 오류 발생한 상황을 보여준다. 즉, i=1이고 j=3이며 따라서 k=2이다.

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	

그림 9 두 개의 디스크에서 오류가 발생한 경우(N=7).

알고리즘 2-1의 step 3과 step 4의 수식을 이용하여 다음과 같은 식(7)과 식(8)을 구할 수 있다.

$$B[4-2m,3]^{(a,7)} = \bigoplus_{l=0}^{5} B[5-l, 2-2m+l]^{(a,7)} \quad (7)$$

$$B[4-2m,1]^{(a,7)} = \bigoplus_{l=0}^{2} B[4-2m,l]^{(a,7)} \quad \text{if } m=3 \quad (8)$$

$$= \bigoplus_{l=0}^{3} B[4-2m,l]^{(a,7)} \oplus \bigoplus_{h=0}^{4} B[h,N] \quad \text{otherwise}$$

위의 식(7)과 식 (8)을 이용하여 변수 m을 증가시키면서 오류가 발생한 두 디스크 i와 j의 오류 블록 의 데이타를 다음과 같이 복구할 수 있다.

m	연산	값
0	B[4,3]=B[5,2]⊕B[3,4]⊕B[2,5]⊕B[1,6]⊕B[0,0] B[4,1]=B[4,0]⊕B[4,2]⊕B[4,3]⊕B[4,4]⊕B[4,5]⊕B[4,6] ⊕B[4,7]	1 1
1	B[2,3]=B[5,0]⊕B[4,1]⊕B[3,2]⊕B[1,4]⊕B[0,5] B[2,1]=B[2,0]⊕B[2,2]⊕B[2,3]⊕B[2,4]⊕B[2,5]⊕B[2,6] ⊕B[2,7]	1 1
2	B[0,3]=B[5,5]⊕B[4,6]⊕B[3,0]⊕B[2,1]⊕B[1,2] B[0,1]=B[0,0]⊕B[0,2]⊕B[0,3]⊕B[0,4]⊕B[0,5]⊕B[0,6] ⊕B[0,7]	1 0
3	B[5,3]=B[4,4]⊕B[3,5]⊕B[2,6]⊕B[1,0]⊕B[0,1] B[5,1]=B[5,0]⊕B[5,2]⊕B[5,3]⊕B[5,4]⊕B[5,5]⊕B[5,6] ⊕B[0,7]⊕B[1,7]⊕B[2,7]⊕B[3,7]⊕B[4,7]	1 1
4	B[3,3]=B[5,1]⊕B[4,2]⊕B[2,4]⊕B[1,5]⊕B[0,6] B[3,1]=B[3,0]⊕B[3,2]⊕B[3,3]⊕B[3,4]⊕B[3,5]⊕B[3,6] ⊕B[3,7]	1 0
5	B[1,3]=B[5,6]⊕B[4,0]⊕B[3,1]⊕B[2,2]⊕B[0,4] B[1,1]=B[1,0]⊕B[1,2]⊕B[1,3]⊕B[1,4]⊕B[1,5]⊕B[1,6] ⊕B[1,7]	0 1

4. 성능 분석

논문 [15]에서는 디스크 배열의 중복 오류를 복구와 관련된 대표적 연구 중 2-D 기법, RS 기법과 몇 가지 평가 지표를 통한 비교로 EVENODD 기법의 장점을 보였다. 이 절의 성능 분석에서는 검사정보 저장을 위해 최적의 동간을 사용하면서도 좋은 성능을 보이는 EVENODD 기법과 본 논문에서 제안한 두 가지 DH_N 기법과 DH_{N-1} 기법을 몇 가지 평가 지표에 의해 비교함으로써 제안한 방법들이 효과적임을 보이고자 한다.

표 1 기본적인 배열 크기

기법	DH_N	DH_{N-1}	EVENODD
전체 배열 크기	(N-1)xN	(N-1)x(N+1)	(N-1)x(N+2)
데이타 배열 크기	(N-2)x(N-1)	(N-2)xN	(N-1)xN

4.1 데이타 인코딩

DH_N기법 및 DH_{N-1} 기법을 EVENODD 기법과 인코딩에 필요한 XOR 연산을 비교할 수 있다. 각각의 기법은 기본적으로 구성되는 전체 배열의 크기와 패리티를 제외한 데이타 배열의 크기가 서로 다르기 때문에 배열 단위로 인코딩하는데 필요한 XOR 연산수를 구해서 직접 비교하는 것은 공정하지 못하다. 그러므로 데이타 배열 전체를 인코딩하는데 필요한 XOR 연산수를 계산하고, 데이타 배열의 크기로 나누면 각 데이타 블록에 필요한 평균 XOR 연산수를 구할 수 있으므로 이를 비교한다. DH_N 기법인 경우 수평 패리티 연산 식(1)에서 각 행에 대한 수평 패리티의 XOR 연산의 수는 (N-2)이고, 배열에서 수평 패리티가 (N-2)개 있으므로 $(N-2)^2$이다. 사선 패리티는 식(2)를 이용해서 구하면 (N-3)N이 된다. 결국 (N-1) x N 배열의 데이타 인코딩에 필요한 XOR 연산을 블록 단위로 계산하면 수평 패리티와 사선 패리티 연산을 합하여 $(N-2)^2 + (N-3)N = (2N^2-7N+4)$이 된다. 단순화하기 위해서 한 블록에 한 바이트로 구성되었다고 가정한다. 전체 디스크 배열에서 데이타 배열의 크기는 표 1 에서와 같이 (N-1) x N 이므로 각 데이타 블록의 인코딩에 필요한 XOR 연산의 평균은 다음과 같다.

$$\frac{(2N^2-7N+4)}{(N-2)(N-1)}$$

DH_{N-1} 기법에서는 수평 패리티를 구하는데 필요한 블록의 수는 식(5)에서 구하면 (N-1)(N-2)이고, 사선 패리티를 구하는데 이용되는 블록의 수는 식(6)을 이용하여 구하면 (N-3)N이다. 두 부분의 합을 구하고 역시

하나의 블록을 1비트로 가정한다면 $(2N^2 - 6N + 2)$가 된다. 마찬가지로 표 1에서 유효 데이타 배열 블록의 크기는 (N-2)N이므로 다음과 같이 인코딩시 각 데이타 블록에 대한 XOR 연산의 평균을 나타낼 수 있다.

$$\frac{(2N^2 - 6N + 2)}{(N-2)N}$$

비교 대상인 EVENODD 기법에서는 사선 S, 수평 패리티, 그리고 사선 패리티를 인코딩하는 방법은 다음과 같은 식으로 나타낼 수 있다[15].

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

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

$$A[i,N+1] = S \oplus \bigoplus_{l=0}^{N-1} A[(N-1-l)\%N, l], \quad where\ 0 \le i \le N-2$$

EVENODD 기법의 경우 데이타 부분이 (N-1) x N 배열로 구성된다. 수평 패리티 연산에 이용되는 각 행에는 N개의 데이타 블록이 있으므로 (N-1) XOR 연산이 필요하다. 그리고 배열에는 (N-1)개의 행이 있으므로 필요한 XOR 연산수는 $(N-1)^2$이다. 사선 패리티 연산을 하기 위해 먼저 사선 S 값을 구하여 사용하는데 이에 필요한 XOR 연산수는 (N-2)가 된다. 사선 패리티 연산에는 사선 블록들과 사선 S 값이 이용되므로 각 사선 패리티를 구하기 위해 N개의 사선 블록과 사선 S 값의 XOR 연산이 필요하므로 N이고, 이러한 사선 패리티가 (N-1)개 있으므로 N(N-1)이 된다. 그러므로 이들 세 가지를 합산하여 필요한 XOR 연산수를 구하면 $(2N^2 - 2N - 1)$이 된다. 결국 이 결과를 데이타 블록의 배열 수로 나누면 한 데이타 블록 평균 필요한 XOR 연산수를 구할 수 있다.

$$\frac{(2N^2 - 2N - 1)}{(N-1)N}$$

앞에서 계산된 식들을 기반으로 하나의 블록 크기를 일반적으로 사용하는 8Kbyte로 하고, XOR 연산 시 사용하는 데이타 레지스터를 64bit(8byte)로 가정한다면 8byte 씩 1024번의 연산이 필요하게 된다. 디스크 배열을 구성하는 디스크 수를 증가시키면서 데이타 블록을 인코딩하는데 필요한 XOR 연산의 평균을 나타내면 그림 10과 같다. 결과에서 보여지듯이 DH_{N+1} 기법이 데이타 인코딩에 필요한 XOR 연산이 가장 적게 필요한데 이는 EVENODD 기법에서는 사선 S 값에 대한 오버헤드가 반영되었음을 의미하고, DH_N 기법에서는 D_0 또는 H_0 패리티의 중복 사용에 따른 오버헤드가 포함되었음을 알 수 있다.

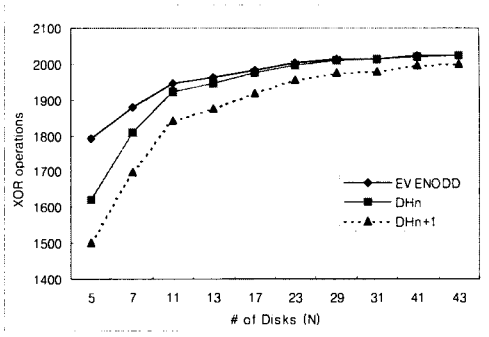

그림 10 인코딩에 필요한 블록 당 평균 XOR 연산수

4.2 이중 디스크 오류 복구

디스크 배열에서 두 디스크 오류 시 복구 가능한 방법들이 제시되었지만 디스크 오류 가능성은 여전히 포함되어 있다. 따라서 두 디스크 복구 중에 또 다른 오류가 발생되어 디스크 배열 전체의 데이타를 모두 잃어버리게 되는 최악의 상태를 줄이는 방법은 디스크 오류에 대한 복구 시간을 최소화하는데 있다. 만약 복구에 사용 가능한 제한된 메모리 크기를 이용한다면 메모리가 허용하는 한도내의 복구에 연관되는 연속적인 디스크 블록을 읽은 후 그 블록들을 XOR 연산하여 다시 저장하면 된다. 이때 비교하는 방법들의 입출력 속도가 동일하다면 XOR 연산수를 줄임으로써 복구시간을 최소화할 수 있다.

두 디스크에서 동시에 오류가 발생된 경우 복구에 필요한 배열 평균 XOR 연산의 수를 각각 구해 보면 다음과 같다. DH_N 기법의 경우에는 모든 디스크에 수평 패리티와 사선 패리티가 분산되어 있으므로 임의의 두 디스크 오류에 대해 모든 경우가 동일하게 취급된다. DH_{N+1} 기법의 경우에는 두 디스크 중에서 수평 패리티 디스크가 포함되어 있는지의 여부에 따라 다른 결과가 나온다. 각각 일반 정보가 저장된 디스크를 I, 수평 패리티가 저장된 디스크를 H라 할 때 표2에서 결과를 나타내었다. EVENODD 기법에서는 네 가지 다른 경우의 오류 발생 가능성이 있다. 수평 패리티가 저장되는 디스크를 H, 사선 패리티가 저장되는 디스크를 D, 일반적인 정보가 저장되는 디스크를 I라고 할 때, 각각의 경우에 대해 다른 결과를 보인다. 비교하는 세 가지 기법에 대해 두 오류 디스크 복구 시 필요한 XOR 연산수, 오류 발생 경우의 수, 그리고 오류 발생 확률을 나타내면 다음의 표2와 같다. 표2를 이용하면 각 기법에서 임의의 두 디스크에서 오류가 발생되었을 경우에 이들을 복구하기 위해 필요한 XOR 연산수를 알 수 있다. 하나의

블록의 크기는 인코딩에서와 마찬가지로 일반적으로 사용하는8Kbytes로 가정하고, 전체 배열의 크기가 각각 다르므로 공정한 비교를 위해 전체 디스크 배열 크기로 나누면 그림 11의 결과를 산출할 수 있다.

표 2 두 디스크 오류에 대한 세 가지 기법의 비교

기법	두 오류 디스크	필요한 XOR 연산	경우의 수	오류 발생 확률
DH_N	(I,I)	$2N^2-6N+2$	$N(N-1)/2$	1.0
DH_{N+1}	(I,H) (I,I)	$2N^2$ 5N+2 $2N^2$ 5N+1	N $N(N-1)/2$	2/(N+1) (N-1)/(N+1)
EVENODD	(H,D) (I,H) (I,D) (I,I)	$2N^2-2N$ 1 N^2 $2N^2-2N$ 1 $2N^2+N$ 7	1 N N $N(N-1)/2$	2/((N+2)(N+1)) 2N/((N+2)(N+1)) 2N/((N+2)(N+1)) N(N-1)/((N+2)(N+1))

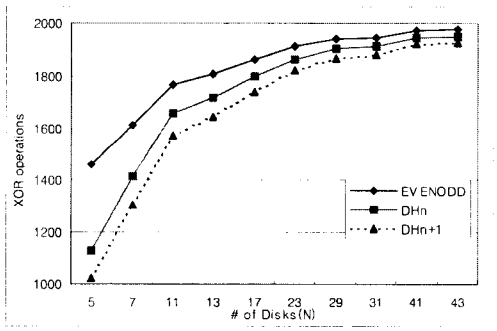

그림 11 이중 디스크 오류 복구를 위한 블록 당 평균 XOR 연산수

앞에서와 같이 DH_{N+1} 기법은 다른 방법에 비해 XOR 연산수를 줄임으로써 가장 빠른 복구를 할 수 있다. 또한 DH_N 기법도 디스크 배열에서 두 디스크 오류 복구 시 기존의 방법보다는 좋은 성능을 보일 것으로 예측된다.

4.3 패리티 블록의 비율

디스크 배열에서 오류 복구를 위해 사용하는 패리티 데이타의 양이 적을 수록 더 많은 정보를 저장할 수 있다. 따라서 패리티와 같은 검사정보가 차지하는 공간을 최적화 하여 실제 데이타 저장을 위한 디스크 배열의 가용성을 높여야 한다. 이론적으로 임의의 디스크 개수로 디스크 배열을 구성할 경우 두 디스크 오류를 복구하기 위해서는 적어도 서로 다른 두개의 디스크에 복구에 필요한 정보를 저장해 놓아야 하고, 그리고 최소한의 복구에 필요한 정보는 두개의 디스크 분량이 된다. 이는 코딩 이론에서 싱글톤 한계(Singleton bound)로 알려져 있다[15].

논문 [19]에서 제안한 방법 중 PRIME 기법에서는 이중 디스크 오류 허용을 위하여 전체 디스크 배열의 절반을 검사정보로 사용하고 있고, PELPR 기법에서는 단지 하나의 오류 허용만 가능하게 인코딩 할 경우에도 N=7일 때 DH_{N+1} 기법에서와 같은 공간을 검사정보 저장을 위해 사용하고 있다. 반면에 두개의 디스크 오류 허용을 위해 사용되는 기법 중 EVENODD 기법은 검사정보를 위해 단지 최적인 두개의 디스크만을 사용하고 있는데 본 논문에서 제안한 두 가지 DH 기법들 역시 최적에 가까운 공간을 검사정보 저장을 위해 사용하고 있다. 예를 들어 디스크 배열에 사용하는 디스크 수가 각각 7개, 43개일 경우 EVENODD 기법과 DH_N 기법은 디스크 배열에서의 차지하는 검사정보의 비율은 각각 0.286, 0.046으로 동일하다. 그리고 같은 디스크 수로 디스크 배열 구성을 할 수 없기 때문에 직접 비교는 안되지만 DH_{N+1} 기법의 경우에는 6개, 8개의 디스크를 사용할 때 각각 0.333, 0.25이고, 42개, 44개 일 경우 각각 0.047, 0.045로 EVENODD 기법의 경우와 거의 비슷한 공간을 검사정보 저장에 사용한다. 그림 12는 제안하는 DH 기법들이 검사정보의 저장을 위해 PRIME 기법에 비해서는 현저하게 작은 공간을 사용하며, EVENODD 기법과 마찬가지로 싱글톤 한계 이론에 따라 최적의 공간을 사용하는 결과를 보여 준다.

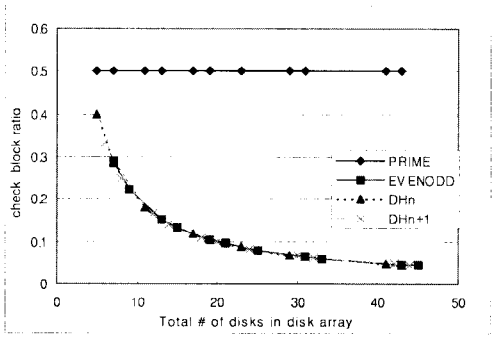

그림 12 디스크 배열 시스템에서의 검사 블록이 차지하는 비율

4.4 작은 블록 쓰기

작은 쓰기 문제는 데이타의 갱신에 따라 연관되는 패리티 블록이 갱신되어야 하므로 디스크 배열에서 성능 저하의 요인으로 지적되어 왔다. DH_N 기법의 경우 작은 크기 블록의 쓰기 연산 시 읽기와 쓰기를 합하여 디스크 배열에서 사용하는 디스크 개수에 관계없이 총 8

번의 블록 접근이 필요하다. 그리고 DH_{N+1} 기법의 경우에는 6번의 블록 접근만을 필요로 한다. 반면에 DH 기법들과 유사하게 사선 패리티와 수평 패리티를 사용하는 EVENODD 기법에서는 쓰기 연산 시 수정되는 블록이 사선 S 에 위치한 블록이 아닌 경우에는 6번의 블록 접근이 필요하고, 만약 사선 패리티 연산에 영향을 미치는 사선 S에 포함된 블록이 수정되는 경우 N이 디스크 수일 때 2(N+1)번의 블록 접근이 필요하다. 만약 5 ~ 43개의 디스크를 사용할 경우 평균적으로 7.2~7.9번의 블록 접근이 필요하다. 이러한 결과로 볼 때 외견상으로는 EVENODD 기법이 DH_N 기법보다는 좋은 성능을 가진다고 생각할 수도 있다. 그러나 동시에 두개 이상의 블록이 수정될 경우 EVENODD 기법은 사선 패리티와 수평 패리티가 특정 디스크에 배치되어 있으므로 패리티 블록들이 순차적으로 수정되어야 하므로 이에 따른 대기 시간이 생기게 된다. 심지어 사선 S에 속한 블록이 수정될 경우에는 모든 사선 패리티들이 수정되어야 하므로 대기 시간이 더 길어지게 되어 성능을 떨어뜨리게 된다.

5. 결 론

본 논문에서는 기존 디스크 배열 시스템의 신뢰성을 향상시키기 위해 이중 디스크 오류, 즉 동시에 임의의 두 디스크에서 오류가 발생되더라도 복구가 가능한 두 가지 효과적인 방법을 제시하였다. 디스크 배열에 수평 패리티와 사선 패리티를 인코딩하여 이용하는 EVENODD 기법이 N+2개의 디스크가 필요한데 반해 제안한 두 가지 DH 기법들은 각각 N 또는 N+1개의 디스크만을 필요로 한다. 두 가지 기법 중 하나인 DH_N 기법은 단일 디스크 오류만 복구 가능한 단순 패리티 기법 중 RAID 5에서와 같이 패리티 블록을 모든 디스크에 분산시킴으로써 기존 기법들에서 병목으로 인해 나타날 수 있는 성능 저하 요인을 미리 제거하였다. 또한, DH_{N+1} 기법은 DH_N 기법에서와 같이 사선 패리티와 수평 패리티를 이용하여 오류 복구를 할 수 있으나 사선 패리티 블록들은 데이타 디스크에 분산하고 수평 패리티 블록들을 패리티 디스크에 배치시키고 있다. 앞에서 보여준 바와 같이 제안한 방법들은 EVENODD 기법에 비해 인코딩에 필요한 평균 XOR 연산수나 오류복구를 위한 디코딩에 필요한 XOR연산수를 비교할 때 보다 효과적임을 알 수 있었다. 그리고 디스크 배열 시스템에서 두 디스크 오류 시 복구 과정을 양방향에서 병렬로 처리할 수 있으므로 복구 시간을 줄일 수 있는 장점도 가지고 있다. 제안한 이중 디스크 오류 복구를 위한 알고리즘들은 기존의 방법들에 비해 매우 단순하고, 하드웨어의 수정 없이 쉽게 상용화된 디스크 배열 시스템에 적용할 수 있다. 결과적으로 제안한 기법들은 효과적인 인코딩/디코딩 방법을 이용하므로 중복 디스크 오류와 관련하여 기존의 방법보다 좋은 성능을 기대할 수 있다.

참 고 문 헌

[1] D. A. Patterson, G. Gibson, and R. H. Katz, A Case for Redundant Arrays of Inexpensive Disks(RAID), Proc. ACM SIGMOD Conf., Chicago, pp.109-116, 1988.

[2] Randy H. Katz, Garth A. Gibson, and David A. Patterson, Disk System Architecture for High Performance Computing, Proceedings of the IEEE, Vol. 77, No. 12, Dec. 1989.

[3] Edward K. Lee, and Randy H. Katz, The Performance of Parity Placements in Disk Arrays, IEEE Trans. on Computers, Vol. 42, No. 6, pp. 651-664, Jun. 1993.

[4] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson, RAID: High-Performance, Reliable Secondary Storage, ACM Computing Surveys, Vol. 26, No. 2, pp. 145-185, Jun. 1994.

[5] Szu-Wen Kuo, Efficient Data Organization and Load Balancing on Parallel Disks, PhD Thesis, University of Illinois at Urbana-Champaign, 1999.

[6] Gregory R. Ganger, Bruce L. Worthington, Robert Y. Hou, and Yale N. Patt, Disk Arrays: High-Performance, High-Reliability Storage Subsystems, IEEE Computer, Mar. 1994.

[7] Vincenzo Catania, Antonio Puliafito, Salvatore Riccobene, and Lorenao Vita, Design and Performance Analysis of a Disk Array System, IEEE Trans. on Computer, Oct. 1995.

[8] Ramakrishna Karedla, J. Spencer Love, and Bradley G. Wherry, Caching Strategies to Improve Disk System Performance, IEEE Computer, Mar. 1994.

[9] Arif Merchant, and Philip S. Yu, Analytic Modeling and Comparisons of Striping Strategies for Replicated Disk Arrays, IEEE Trans. on Computers, Vol.44, No.3, Mar. 1995.

[10] E.J. Schwabe and I.M. Sutherland, Flexible Usage of Parity Storage Space in Disk Arrays, The 8th Annual ACM Symposium on Parallel Algorithms and Architectures, pp99-108, 1996.

[11] X. Wu, J. Li, and H. Kameda, Reliability Analysis of Disk Organization by Considering Uncorrectable Bit Errors, 16th Symposium Reliable Distributed

Systems pp. 2-9, 1997.
[12] Fouad A. Tlbag, Joseph Pang, Randall Baird, and Mark Gang, Streaming RAID-A Disk Array Management System for Video Files, Proceedings of the first ACM Conf. Multimedia '93, 1993.
[13] Nisha Talagala, Satoshi Asami, Thomas Anderson and David Patterson, Tertiary Disk: Large Scale Distributed Storage, UC Berkeley Technical report CSD-98-989, Jan. 28, 1998.
[14] Chan-Ik Park, Efficient Placement of Parity and Data to Tolerate Two Disk Failures in Disk Array Systems, IEEE Trans. on Parallel and Distributed Systems, Vol. 6, No. 11, pp.1177-1184, Nov. 1995.
[15] Mario Blaum, Jim Brady, Jehoshua Bruck, and Jai Menon, EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures, IEEE Trans. on Computers, Vol. 44, No. 2, pp. 192-202, Feb. 1995.
[16] Nisha Talagala and David Patterson, An Analysis of Error Behaviour in a Large Storage System, UC Berkeley Technical report CSD-99-1042, Feb. 26, 1998.
[17] Garth A. Gibson, Redundant Disk Arrays: Reliable, Parallel Secondary Storages, PhD dissertation, Univ. of California at Berkeley, Dec. 1990.
[18] F.J. MacWilliams and N.J. A. Solane, The theory of Error-Correcting Codes, Amsterdam, The Netherlands: North-Holland, 1977.
[19] Guillermo A.Alvarez, Walter A.Burkhard, Larry J.Stockmeyer, and Flaviu Cristian, Declustered disk array architectures with optimal and near-optimal parallelism, Proceedings of the 25th annual international symposium on computer architecture, 1998, pp109120.
[20] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik, Concrete Mathematics-A Foundation for Computer Science, Addison-Wesley Publishing Company, pp.129-130, May 1989.

이 남 규

1988년 연세대학교 전산과학과 졸업 (학사). 1988년 ~ 1993년 삼성종합기술원 연구원. 1905년 연세대학교 컴퓨터과학 (이학석사). 1995년 1997년 삼성전자 선임연구원. 1995년 ~ 현재 연세대학교 공과대학 컴퓨터과학과 박사과정. 관심분야는 고성능 디스크 구조, RAID, Mobile Computing, HCI, Wearable computer.

한 탁 돈

1978년 연세대학교 전자공학과 졸업 (학사). 1983년 Wayne State University 컴퓨터공학 (공학석사). 1987년 University of Massachusetts 컴퓨터공학 (공학박사). 1987년 ~ 1989년 Cleveland 주립대학 조교수. 1989년 ~ 현재 연세대학교 공과대학 컴퓨터과학과 교수. 관심분야는 Wearable computer, HCI, ASIC 설계, 고성능 컴퓨터구조.