논문 01-26-2A-11 한국통신학회논문지 '01-2 Vol.26 No.2A

RAID 시스템에서 3개의 디스크 오류에 대한 복구 방법

정회원 박 종 원*, 한 영 열**

A Method for Tolerating Triple Disk Failures in RAID Systems

Chong-Won Park*, Young-Yearl Han** *Regular Members*

요 약

디스크의 다중 오류 복구를 위하여 여러가지 이론적인 코딩 방법들이 제안되었으나 구현이 복잡하여 실용화하기가 매우 어렵다. 본 논문에서는 RAID 시스템에서 3개의 디스크가 동시에 오류를 일으켰을 때 복구할 수 있는 패리티 구성 방법을 제안한다. 이 방법은 먼저 행렬식 표현을 통해 문제를 정형화하였으며, 단지 3개의 패리티 디스크만을 사용하여 동시에 3개의 디스크에서 발생한 오류를 복구할 수 있다. 또한 이 방법은 단순한 산술 연산을 사용하기 때문에 기존의 RAID 제어기에도 적용할 수 있다.

ABSTRACT

Although theoretical methods were proposed for tolerating multiple disk failures, their complexities are too high to be practically applicable. In this paper, we proposed a practical parity scheme for tolerating simultaneous triple disk failures in RAID systems. We first formalized the problems with matrix operations. Our scheme is practical in the sense that it employs three redundant disks for tolerating triple disk failures. Furthermore, it requires simple arithmetic computations only, and it can be implemented on current RAID controllers.

I. 서 론

컴퓨터는 성능이나 응용 분야 면에서 급속히 발전하고 있다. 사용자 관점에서 보면 최악의 경우 2, 3년 밖에 되지 않은 컴퓨터가 거의 쓸모가 없어지는 경우도 있다. 그러나 입출력 시스템과 저장 시스템은 프로세서의 성능 증가에 비해 발전 속도가 느리다. 프로세서와 입출력 시스템의 성능의 차이가 점차 증대됨에 따라 컴퓨터 시스템의 성능은 입출력의 성능에 의해 제약을 받게 되었다[1][2]. 따라서 컴퓨터 시스템의 전체적인 성능을 개선하기 위해서는 연산 처리 능력과 입출력의 성능 및 저장 시스템 용량 사이의 균형을 유지하는 것이 매우 중요한 문제로 대두되고 있다.

데이터 클러스터링(clustering)과 디스크 스트라이핑(stripping)을 사용하는 디스크 어레이 시스템과 같이 입출력 시스템 성능을 개선하는 구체적인 방안이 최근 컴퓨터 시스템 연구에서 중요한 연구 과제가 되었다[2][3][4]. 디스크 어레이 시스템에서는 많은 수의 디스크를 사용하여 데이터를 병렬로 처리함으로써 입출력 성능을 개선하고 있다. 최근에 각광을 받고 있는 멀티미디어 시스템에서는 오디오, 비디오, 팩스와 같은 새로운 형태의 데이터들을 저장하기 위해 많은 수의 디스크들이 사용되고 있다. 한 시스템에서 많은 수의 디스크를 사용하는 디스크 어레이 시스템은 1개의 디스크를 사용하는 경우보다 고장에 더욱 취약하게 된다[3]. 따라서 디스크 어레이 시스템에서는 데이터를 보호하기 위해 디스크의 오류에 대한 복구가 중요한 문제로 대두된다.

Patterson 등[3]이 제안한 RAID(Redundant Arrays of Inexpensive Disks)에서는 데이터와 패리티의 위

* 한국전자통신연구원 책임연구원
** 한양대학교 전자통신공학과 교수
논문번호 : 00358-0915, 접수일자 : 2000년 9월 15일

치 및 구성에 따라 5개의 다른 레벨(RAID level 1~5)로 정의하였다. 기본적으로 RAID에서는 패리티를 사용하여 단지 1개의 디스크 오류 만을 감내할 수 있으며, 2개 이상의 디스크 오류에 대해서는 데이터를 복구하지 못하게 된다.

디스크의 다중 오류 복구를 위하여 Hamming 코드나 Reed-Solomon 코드와 같은 오류 정정 코드[5]가 사용되었지만 그것을 구현하는 것은 매우 복잡하다. 따라서 배타적 논리합(exclusive OR) 혹은 간단한 산술 계산과 같은 단순한 연산으로 수행되는 코드가 요구된다.

또한 길쌈 형태의 코드(convolution code)[6]들도 디스크 오류 복구에 사용할 수 있으나, 이 경우 코드의 허용 범위를 넘으면 오류가 무한히 전이되는 문제가 있다. 이것은 길쌈 코드의 인코딩 및 디코딩이 재귀적인(recursive) 방법을 사용하기 때문이다. 따라서 인코딩 및 디코딩이 재귀적이지 않으면서 간단한 동작을 하는 블록 형태의 코드가 필요하다.

다중 오류 복구를 위해 Blaum, Keren 등[7][8][9]이 이론적인 코딩 방법을 제안하였으며, 후에 Blaum[10]은 일반적인 코딩 방법을 제안하였다. 하지만 이러한 코딩 방법들은 인코딩과 디코딩이 매우 복잡하다는 문제가 있다[11]. 비록 다중 오류에 대해 실용적으로 구현 가능한 방법이 Blaum, Park 등[12][13]에 의해 제안되었지만, 그것은 2개의 디스크 오류 경우에만 가능하다.

오늘날 사용자 데이터를 위한 고 신뢰성 저장 시스템에 대한 요구가 점차 증대되고 있어서, 디스크 어레이 시스템은 2개 이상의 디스크 오류에 대한 보다 실용적인 복구 방법이 필요하다.

본 논문에서는 동시에 디스크 3개에서 발생한 오류에 대해 복구할 수 있는 실용적인 방법을 제안하였다. 먼저 행렬식 표현을 통해 문제를 정형화하였으며, 이를 이용하여 단지 3개의 패리티 디스크만을 사용하여, 1개 및 2개의 디스크 오류를 포함한 3개의 디스크 오류를 복구할 수 있는 방법을 제안하였다. 제안된 방법은 배타적 논리합과 단순한 연산만을 사용하기 때문에 구현이 용이하며, 인코딩 및 디코딩 방법이 재귀적이지 않으므로 오류가 무한히 전이되지 않는다.

본 논문의 구성은 2장에서 인코딩 방법을, 3장에서 디코딩 방법을, 4장에서는 기존 RAID 시스템에의 적용에 대해 기술하였으며, 5장에서 결론을 기술하였다.

Ⅱ. 인코딩 방법

먼저 문제에 대한 표현을 간단히 하기 위하여 다음과 같이 가정한다. 첫째, m+3개의 디스크 중에 처음 m개의 디스크에 정보가 저장되며 마지막 3개의 디스크에 여분의 정보가 저장된다. 또한 m은 소수로 한다[8][10]. 만일 m이 소수가 아닌 임의의 수로 구성하고자 하면 임의 수보다 큰 소수를 사용하고, m보다 큰 인덱스를 갖는 디스크들에는 정보가 없는 것으로 간주한다. 즉, 이것은 실제 디스크를 사용하는 것이 아니라 단순히 계산을 위한 가상의 디스크이며, 이들 가상 디스크의 모든 정보는 0으로 간주한다.

다음에 예시되는 예제들에서는 표기의 단순화를 위해 각각의 심볼을 비트로 표기한다. 그러나 어떤 응용에서는 한 심볼이 512 바이트의 디스크 한 섹터보다 클 수도 있으므로 심볼이 꼭 바이너리(binary)라고 가정할 필요는 없다.

위와 같은 가정 하에서 3개의 디스크 오류 복구를 위해 다음과 같이 표시한다. 먼저, m이 소수인 $(m-1)\times(m+3)$ 어레이를 정의한다. 여기서 심볼 $a_{i,j}$, $0\le i\le m-2$, $0\le j\le m+2$는 j 번째 디스크의 i 번째 심볼이다. 어떤 응용에서는 어레이의 열이 디스크나 혹은 섹터로 가정된다. 마지막 3개의 디스크 m, $m+1$, $m+2$는 여분의 정보를 가지는 디스크, 즉 패리티 디스크이다.

여기서 $\langle n\rangle_m = j$ 는 $j \equiv n \pmod m$, $0\le j\le m-1$, 이다. 즉 $\langle 7\rangle_5 = 2$이며, $\langle -2\rangle_5 = 3$을 의미한다. 그리고 $A=\bigoplus_{t=l}^{m} a_{n,t}$ 는 $A = a_{n,l} \oplus a_{n,l+1} \oplus a_{n,l+2} \oplus \cdots \oplus a_{n,m}$ 이고, $\oplus$는 모듈로 (modulo) 더하기를 의미한다.

또한 마지막 행을 모두 0이라고 가정하면, $a_{m-1,j} = 0$, $0\le j\le m-1$이므로 이 경우 어레이는 $m\times(m+2)$로 간주할 수 있다.

따라서 l, $0\le l\le m-2$의 경우 여분의 심볼은 다음과 같이 표시할 수 있다.

$$a_{l,m} = \bigoplus_{t=0}^{m-1} a_{l,t} \tag{1}$$

$$a_{l,m+1} = \bigoplus_{t=0}^{m-1} a_{\langle l-t\rangle_m,t} \tag{2}$$

$$a_{l,m+2} = \bigoplus_{t=0}^{m-1} a_{\langle -2(l-t)\rangle_m,t} \tag{3}$$

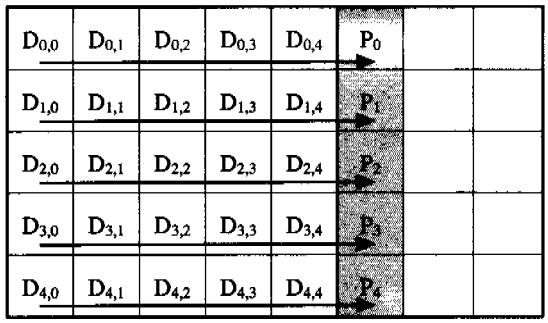


그림 1. 수평 패리티 계산 방법

식 (1), (2), (3)은 인코딩 방법을 기술한 것이다. 따라서 3가지 형태의 여분 정보가 있다. 즉 1개의 수평 여분 정보와 2개의 수직 여분 정보이다. 이것은 패리티와 동일한 개념이므로 각각 수평 패리티, 수직 패리티라고 부르기로 하자. 디스크 m은 식 (1)에 따라 단순히 디스크 0, 1, ……, m-1을 배타적 논리합으로 구할 수 있다. 디스크 (m+1)은 식 (2), 디스크 (m+2)는 식 (3)에 따라 수행하면 된다.

이들 식들을 자세히 살펴보면 수직 패리티는 단순히 행 심볼 디스크의 패리티이다. 또한 식 (2)에 의한 첫번째 수직 패리티는 0번째 열은 변화를 주지 않고, 1번째 열을 한번 순환 자리 이동(cyclic shift)하고, 2번째 열을 2번 순환 자리 이동하며, 같은 방법으로 (m-1)번 열을 (m-1)번 순환 자리 이동한 후 행 디스크를 모듈로 가산하는 것이다.

식 (3)에 의한 두번째 수직 패리티는 0번째 열은 변화를 주지 않고, 1번째 열을 2번 순환 자리 이동하고, 2번째 열을 4번 순환 자리 이동하며, 같은 방법으로 (m-1)번 열을 2×(m-1)번 순환 자리 이동한 후 행 디스크를 모듈로 가산하는 것이다. 이렇게 하여 만든 (m-1)×(m+3) 어레이는 3개 열의 정보에 오류가 발생하여도 복구할 수 있다.

다음은 m = 5일 때, 위의 설명에 따른 식(1), (2), (3)에 의한 패리티 계산 방법을 예로 도시하였다.

(예 1) 그림 1은 식 (1)에 따른 수평 패리티를, 그림 2는 식 (2)에 따른 첫번째 수직 패리티를, 그림 3은 식(3)에 따른 두번째 수직 패리티를 계산하는 것이다.

위에서 보는 바 같이 인코딩 방법이 단순하여 하드웨어로 간단히 구성할 수 있으며, 또한 소프트웨어적으로도 현재의 RAID 제어기에 쉽게 적용시킬 수 있다.

다음은 m = 5일 때, 인코딩 예를 제시하였다.

(예 2) m = 5 이고, 정보 $a_{i,j}$가 $0 \le i \le 3$, $0 \le j \le 7$ 일 때, 행 5, 6, 7에 패리티 정보가 있다. 여기에서는 0번에서 7번까지 8개의 디스크가 있고 각 디스크는 4개의 섹터가 있다고 가정한다. 데이터는 0, 1, 2, 3, 4번 디스크 섹터에 있으며, 패리티 정보

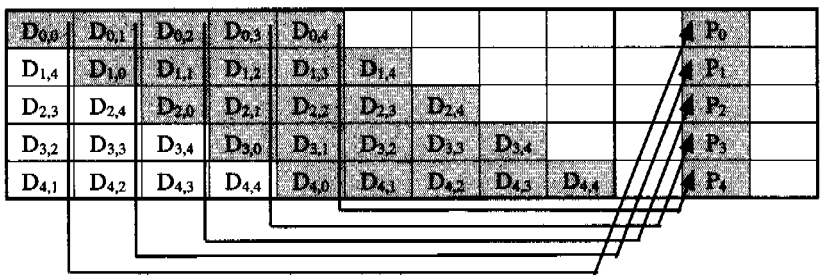


그림 2. 첫번째 수직 패리티 계산 방법

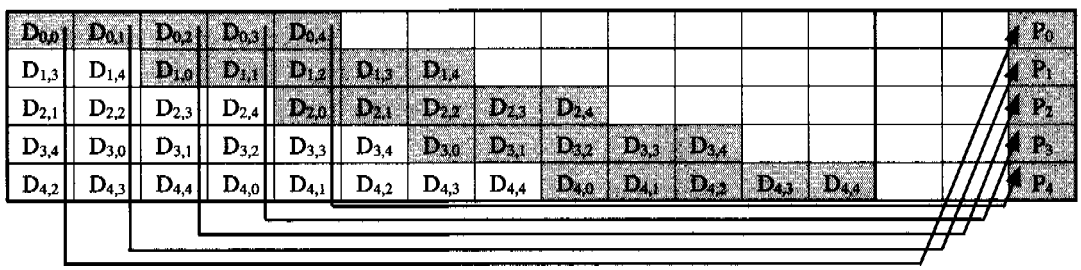


그림 3. 두번째 수직 패리티 계산 방법

는 5, 6, 7번 디스크 섹터에 있다. 그러면 식 (1), (2), (3)에 따라 다음과 같이 패리티를 계산할 수 있다.

$$a_{l,5} = a_{l,0} \oplus a_{l,1} \oplus a_{l,2} \oplus a_{l,3} \oplus a_{l,4}\ ,\ 0 \le l \le 3$$

$$a_{0,6} = a_{0,0} \oplus a_{3,2} \oplus a_{2,3} \oplus a_{1,4}$$

$$a_{1,6} = a_{1,0} \oplus a_{0,1} \oplus a_{3,3} \oplus a_{2,4}$$

$$a_{2,6} = a_{2,0} \oplus a_{1,1} \oplus a_{0,2} \oplus a_{3,4}$$

$$a_{3,6} = a_{3,0} \oplus a_{2,1} \oplus a_{1,2} \oplus a_{0,3}$$

$$a_{0,7} = a_{0,0} \oplus a_{2,1} \oplus a_{1,3} \oplus a_{3,4}$$

$$a_{1,7} = a_{3,0} \oplus a_{0,1} \oplus a_{2,2} \oplus a_{1,4}$$

$$a_{2,7} = a_{1,0} \oplus a_{3,1} \oplus a_{0,2} \oplus a_{2,3}$$

$$a_{3,7} = a_{1,1} \oplus a_{3,2} \oplus a_{0,3} \oplus a_{2,4}$$

이를테면, 그림 4와 같이 인코딩하고자 하는 5개의 열이 있다고 가정한다.

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

그림 4. 인코딩을 위한 4×5 어레이

인코딩 절차에 따라 계산을 하면 마지막 3개 열을 그림 5와 같이 나타낼 수 있다.

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

그림 5. 그림 1의 인코딩 결과

위에서 제안한 인코딩 방법은 다음과 같은 전체 계산량과 소요 메모리 공간에 관한 정의를 만족시킨다.

정의 1: 정보 디스크의 수가 m일 때, 제안된 방법의 인코딩에서 3개의 디스크 오류를 복구하기 위한 패리티의 계산량은 $O(m^2)$ 이다.

증명: 식 (1), (2) , (3)에서의 계산량은 $O(m)$ 이고 각 식은 m번 수행된다. 따라서 인코딩을 위한 전체 계산량은 $O(m^2)$ 이다.

따름정의 1: 정보 디스크의 수가 m일 때, 제안된 방법의 인코딩에서 3개의 디스크 오류를 복구하기 위한 메모리 공간은 $O(m^2)$ 이다.

증명: 가산을 위한 메모리 공간은 행렬식 $(m\text{-}1)\times(m+3)$의 크기에 의해 결정된다. 따라서 인코딩을 위한 메모리 공간은 $O(m^2)$ 경계 안에 있다.

Ⅲ. 디코딩 방법

본 논문에서 제안한 디코딩 알고리즘의 목표는 동시에 발생한 3개의 디스크 오류에 대한 복구 방법이다. 이 알고리즘은 응용에 따라 하드웨어 또는 소프트웨어로 모두 구현될 수 있다. 즉, 1개의 디스크 오류와 2개 및 3개의 디스크에 동시에 오류가 발생하여도 이를 복구할 수 있다. 그림 6과 같이 디코딩을 위하여 $(m\text{-}1)\times(m+3)$ 어레이가 있다고 가정한다. 다음과 같이 3개의 오류를 복구할 수 있는 알고리즘을 제안한다.

여기서 $i<j<k$ 의 경우로 3개의 행 i, j, k 에서 오류가 발생했다고 가정하자. 그러면 식 (1), (2), (3)에 의해 $(m\text{-}1)\times 3$개의 방정식을 만들 수 있다. 이것은 $(m\text{-}1)\times 3$개의 미지수를 가지는 식 (4), (5), (6)으로 구성된 $(m\text{-}1)\times 3$개의 방정식을 만들 수 있으므로, 이로부터 그 해를 구할 수 있다[14].

식 (4), (5), (6)은 식 (7)과 같이 정리된다.

그러면 식 (7)의 $[\Phi]$를 단위 행렬 $[I]$로 만들면 식 (8)의 $[\Omega]$가 미지수에 대한 해가 된다.

$a_{0,0}$	$a_{0,1}$	....	$a_{0,i}$	....	$a_{0,j}$	....	$a_{0,k}$	....	$a_{0,m-1}$	$a_{0,m}$	$a_{0,m+1}$	$a_{0,m+2}$
$a_{1,0}$	$a_{1,1}$	....	$a_{1,i}$	....	$a_{1,j}$	....	$a_{1,k}$	....	$a_{1,m-1}$	$a_{1,m}$	$a_{1,m+1}$	$a_{1,m+2}$
$a_{2,0}$	$a_{2,1}$	....	$a_{2,i}$	....	$a_{2,j}$	....	$a_{2,k}$	....	$a_{2,m-1}$	$a_{2,m}$	$a_{2,m+1}$	$a_{2,m+2}$
....	....	....	....	....	....	....	....	....	....	....	....	....
$a_{m-2,0}$	$a_{m-2,1}$	....	$a_{m-2,i}$	....	$a_{m-2,j}$	....	$a_{m-2,k}$	....	$a_{m-2,m-1}$	$a_{m-2,m}$	$a_{m-2,m+1}$	$a_{m-2,m+2}$

그림 6. 디코딩을 위한 (m-1)×(m+3) 어레이

$$
\begin{array}{llllllllll}
a_{0,0} & \oplus a_{0,1} & \oplus a_{0,2} & \oplus\Lambda\oplus a_{0,i} & \oplus\Lambda\oplus a_{0,j} & \oplus\Lambda\oplus a_{0,k} & \oplus\Lambda\oplus a_{0,m-1} & = a_{0,m} \\
a_{1,0} & \oplus a_{1,1} & \oplus a_{1,2} & \oplus\Lambda\oplus a_{1,i} & \oplus\Lambda\oplus a_{1,j} & \oplus\Lambda\oplus a_{1,k} & \oplus\Lambda\oplus a_{1,m-1} & = a_{1,m} \\
a_{2,0} & \oplus a_{2,1} & \oplus a_{2,2} & \oplus\Lambda\oplus a_{2,i} & \oplus\Lambda\oplus a_{2,j} & \oplus\Lambda\oplus a_{2,k} & \oplus\Lambda\oplus a_{2,m-1} & = a_{2,m} \\
\mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M}\ \mathrm{M} & \mathrm{M} \\
a_{m-2,0} & \oplus a_{m-2,1} & \oplus a_{m-2,2} & \oplus\Lambda\oplus a_{m-2,i} & \oplus\Lambda\oplus a_{m-2,j} & \oplus\Lambda\oplus a_{m-2,k} & \oplus\Lambda\oplus a_{m-2,m-1} & = a_{m-2,m}
\end{array}
\tag{4}
$$

$$
\begin{array}{llllllllll}
a_{0,0} & \oplus a_{m-1,1} & \oplus a_{m-2,2} & \oplus\Lambda\oplus a_{\langle -i\rangle_m,i} & \oplus\Lambda\oplus a_{\langle -j\rangle_m,j} & \oplus\Lambda\oplus a_{m-k,k} & \oplus\Lambda\oplus a_{1,m-1} & = a_{0,m+1} \\
a_{1,0} & \oplus a_{0,1} & \oplus a_{m-1,2} & \oplus\Lambda\oplus a_{\langle 1-i\rangle_m,i} & \oplus\Lambda\oplus a_{\langle 1-j\rangle_m,j} & \oplus\Lambda\oplus a_{\langle 1-k\rangle_m,k} & \oplus\Lambda\oplus a_{2,m-1} & = a_{1,m+1} \\
a_{2,0} & \oplus a_{1,1} & \oplus a_{0,2} & \oplus\Lambda\oplus a_{\langle 2-i\rangle_m,i} & \oplus\Lambda\oplus a_{\langle 2-j\rangle_m,j} & \oplus\Lambda\oplus a_{\langle 2-k\rangle_m,k} & \oplus\Lambda\oplus a_{3,m-1} & = a_{2,m+1} \\
\mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M}\ \mathrm{M} & \mathrm{M} \\
a_{m-2,0} & \oplus a_{m-2,1} & \oplus a_{m-2,2} & \oplus\Lambda\oplus a_{\langle m-2-i\rangle_m,i} & \oplus\Lambda\oplus a_{\langle m-2-j\rangle_m,j} & \oplus\Lambda\oplus a_{\langle m-2-k\rangle_m,k} & \oplus\Lambda\oplus a_{0,m-1} & = a_{m-2,m+1}
\end{array}
\tag{5}
$$

$$
\begin{array}{llllllllll}
a_{0,0} & \oplus a_{2,1} & \oplus a_{4,2} & \oplus\Lambda\oplus a_{\langle 2i\rangle_m,i} & \oplus\Lambda\oplus a_{\langle 2j\rangle_m,j} & \oplus\Lambda\oplus a_{\langle 2k\rangle_m,k} & \oplus\Lambda\oplus a_{\langle 2(m-1)\rangle_m,m-1} & = a_{0,m+2} \\
a_{\langle -2\rangle_m,0} & \oplus a_{0,1} & \oplus a_{2,2} & \oplus\Lambda\oplus a_{\langle 2(i-1)\rangle_m,i} & \oplus\Lambda\oplus a_{\langle 2(j-1)\rangle_m,j} & \oplus\Lambda\oplus a_{\langle 2(k-1)\rangle_m,k} & \oplus\Lambda\oplus a_{\langle 2(m-2)\rangle_m,m-1} & = a_{1,m+2} \\
a_{\langle -4\rangle_m,0} & \oplus a_{\langle -2\rangle_m,1} & \oplus a_{0,2} & \oplus\Lambda\oplus a_{\langle 2(i-2)\rangle_m,i} & \oplus\Lambda\oplus a_{\langle 2(j-2)\rangle_m,j} & \oplus\Lambda\oplus a_{\langle 2(k-2)\rangle_m,k} & \oplus\Lambda\oplus a_{\langle 2(m-3)\rangle_m,m-1} & = a_{2,m+2} \\
\mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M}\ \mathrm{M} & \mathrm{M} \\
a_{4,0} & \oplus a_{6,1} & \oplus a_{8,2} & \oplus\Lambda\oplus a_{\langle 2(i+2)\rangle_m,i} & \oplus\Lambda\oplus a_{\langle 2(j+2)\rangle_m,j} & \oplus\Lambda\oplus a_{\langle 2(k+2)\rangle_m,k} & \oplus\Lambda\oplus a_{2,m-1} & = a_{m-2,m+2}
\end{array}
\tag{6}
$$

$$
\begin{bmatrix} \\ \\ \Phi \\ \\ \\ \end{bmatrix}
\begin{bmatrix} a_{0,i} \\ a_{1,i} \\ \mathrm{M} \\ a_{n-2,i} \\ a_{0,j} \\ a_{1,j} \\ \mathrm{M} \\ a_{n-2,j} \\ a_{0,k} \\ a_{1,k} \\ \mathrm{M} \\ a_{n-2,k} \end{bmatrix}
=
\begin{bmatrix} \\ \\ \Psi \\ \\ \\ \end{bmatrix}
\tag{7}
$$

$$
\begin{bmatrix} \\ \\ \mathrm{I} \\ \\ \\ \end{bmatrix}
\begin{bmatrix} a_{0,i} \\ a_{1,i} \\ \mathrm{M} \\ a_{n-2,i} \\ a_{0,j} \\ a_{1,j} \\ \mathrm{M} \\ a_{n-2,j} \\ a_{0,k} \\ a_{1,k} \\ \mathrm{M} \\ a_{n-2,k} \end{bmatrix}
=
\begin{bmatrix} \\ \\ \Omega \\ \\ \\ \end{bmatrix}
\tag{8}
$$

다음의 예는 m = 5일 때, 디코딩 과정을 기술하였다.

(예 3) (예 2)에서와 같이 m = 5인 경우, 그림 7과 같이 행 1, 3, 4에서 오류가 발생했다고 가정한다. 여기서 ×는 오류가 발생한 것을 표시하였다.

1	×	1	×	×	1	1	1
0	×	1	×	×	0	1	0
1	×	0	×	×	0	0	0
0	×	0	×	×	1	1	0

그림 7. 3개의 디스크 오류의 예

식 (1), (2), (3)으로 부터 다음의 12개의 식 (9)를 만들 수 있다.

$$
\begin{array}{l}
1 = 1 \oplus a_{0,1} \oplus 1 \oplus a_{0,3} \oplus a_{0,4} \\
0 = 0 \oplus a_{1,1} \oplus 1 \oplus a_{1,3} \oplus a_{1,4} \\
0 = 1 \oplus a_{2,1} \oplus 0 \oplus a_{2,3} \oplus a_{2,4} \\
1 = 0 \oplus a_{3,1} \oplus 0 \oplus a_{3,3} \oplus a_{3,4}
\end{array}
$$

$$
\begin{array}{ll}
1 = 1 \oplus\ 0\ \oplus a_{2,3} \oplus a_{1,4} & 1 = \ 1\ \oplus a_{2,1} \oplus a_{1,3} \oplus a_{3,4} \\
1 = 0 \oplus a_{0,1} \oplus a_{3,3} \oplus a_{2,4} & 0 = \ 0\ \oplus a_{0,1} \oplus\ 0\ \oplus a_{1,4} \\
0 = 1 \oplus a_{1,1} \oplus\ 1\ \oplus a_{3,4} & 0 = \ 0\ \oplus a_{3,1} \oplus\ 1\ \oplus a_{2,3} \\
1 = 0 \oplus a_{2,1} \oplus\ 1\ \oplus a_{0,3} & 0 = a_{1,1} \oplus\ 0\ \oplus a_{0,3} \oplus a_{2,4}
\end{array}
\tag{9}
$$

그러면 12개의 미지수와 12개의 식을 가진다. 따라서 식 (9)로 부터 식 (10)을 유도해 낼 수 있다.

$$\begin{bmatrix}1&1&1&0&0&0&0&0&0&0&0&0\\0&0&0&1&1&1&0&0&0&0&0&0\\0&0&0&0&0&0&1&1&1&0&0&0\\0&0&0&0&0&0&0&0&0&1&1&1\\0&0&0&0&1&0&1&0&0&0&0&0\\1&0&0&0&0&0&0&0&1&0&1&0\\0&0&0&1&0&0&0&0&0&0&0&1\\0&1&0&0&0&0&0&1&0&0&0&0\\0&0&0&1&0&0&0&0&1&0&0&0\\1&0&0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&1&0&0&1&0&0\\0&1&0&1&0&0&0&0&0&0&0&0\end{bmatrix}\begin{bmatrix}a_{0,1}\\a_{0,3}\\a_{0,4}\\a_{1,1}\\a_{1,3}\\a_{1,4}\\a_{2,1}\\a_{2,3}\\a_{2,4}\\a_{3,1}\\a_{3,3}\\a_{3,4}\end{bmatrix}=\begin{bmatrix}1\\1\\1\\1\\0\\1\\0\\0\\0\\0\\1\\0\end{bmatrix} \quad (10)$$

$$\begin{bmatrix}1&0&0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&0&0&1\end{bmatrix}\begin{bmatrix}a_{0,1}\\a_{0,3}\\a_{0,4}\\a_{1,1}\\a_{1,3}\\a_{1,4}\\a_{2,1}\\a_{2,3}\\a_{2,4}\\a_{3,1}\\a_{3,3}\\a_{3,4}\end{bmatrix}=\begin{bmatrix}0\\1\\0\\1\\0\\0\\1\\0\\0\\1\\1\\1\end{bmatrix} \quad (11)$$

식 (10)은 식 (11)로 정리할 수 있다. 그러면 식 (11)로 부터 미지수에 대한 해를 구할 수 있다. 따라서 그림 8과 같이 어레이를 재구성할 수 있다.

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

그림 8. 3개의 디스크 오류에 대한 복구

또한 1개 혹은 2개의 디스크 오류에 대한 경우도 식 (4)에서 (8)로부터 재구성할 수 있다. 이 경우, 미지수는 $(m-2)$개 혹은 $2\times(m-2)$개가 된다.

부가적으로, 다음과 같이 오류가 발생한 디스크의 위치가 특별한 경우에는 행렬식을 사용하지 않고 복구할 수 있으므로 디코딩 과정의 계산량을 줄일 수 있다.

3개의 디스크 오류의 경우 :

- $m \le i, j, k \le m+2$: 모든 패리티 디스크에서 오류가 발생한 경우로, 디스크 m은 식 (1)을 사용하여, 디스크 $m+1$은 식 (2)를 사용하여, 디스크 $m+2$는 식 (3)을 사용하여 복구할 수 있다. 다시 말하면 복구가 인코딩 과정과 같다.
- $i < m$, $m+1 \le j, k \le m+2$: 1개의 데이터 디스크와 2개의 수직 패리티 디스크의 오류의 경우이다. 데이터 디스크는 식 (1)을 사용하여, 패리티 디스크는 식 (2)와 식 (3)을 사용하여 복구할 수 있다.

2개의 디스크 오류의 경우 :

- $m+1 \le i, j \le m+2$: 2개의 수직 패리티 디스크 오류의 경우로, 디스크 $m+1$은 식 (2)를 사용하여, 디스크 m+2는 식 (3)을 사용하여 복구할 수 있다. 다시 말하면 인코딩 과정과 같다.
- $i=m$, $m+1 \le j \le m+2$: 수평 패리티 디스크와 하나의 수직 패리티 디스크 오류의 경우로, 수평 패리티 디스크는 식 (1)을 사용하여, 수직 패리티 디스크는 식 (2) 또는 (3)을 사용하여 복구할 수 있다.
- $i<m$, $m+1 \le j \le m+2$: 1개의 데이터 디스크와 1개의 수직 패리티 디스크오류의 경우로, 데이터 디스크는 식(1)을 사용하여, 수직 패리티 디스크는 식 (2) 또는 (3)을 사용하여 복구할 수 있다.

하나의 디스크 오류의 경우:

- $m \le i \le m+2$: 하나의 패리티 디스크 오류의 경우로, 디스크가 m인 경우 식 (1)을 사용하여, 디스크가 $m+1$인 경우 식 (2)를 사용하여, 디스크가 $m+2$인 경우 식 (3)을 사용하여 복구할 수 있다. 다시 말하면 인코딩 과정과 같다.
- $i<m$: 하나의 데이터 디스크 오류의 경우로, 식 (1)을 사용하여 복구할 수 있다.

위와 같은 디코딩 방법은 다음과 같이 전체 계산량과 소요 메모리 공간에 관한 정의를 만족시킨다.

정의 2: 정보 디스크의 수가 m일 때, 제안된 방법의 디코딩에서 3개의 디스크 오류를 복구하기 위한 패리티의 계산량은 $O(m^3)$이다.

증명: 식 (8)에서의 계산량은 $O(m^3)$이다. 따라서

디코딩을 위한 전체 계산량은 $O(m^3)$이다.

따름정의 2: 정보 디스크의 수가 m일 때, 제안된 방법의 디코딩에서3개의 디스크 오류를 복구하기 위한 메모리 공간은 $O(m^2)$ 이다.

증명: 가산을 위한 메모리 공간은 행렬식 $(m-1)\times(m+3)$의 크기에 의해 결정된다. 따라서 디코딩을 위한 메모리 공간은 $O(m^2)$ 경계 안에 있다.

Ⅳ. 기존 RAID 시스템에의 적용

기존의 RAID 시스템의 구성은 그림 9와 같이 디스크의 집합(pool)과 디스크를 제어하는 RAID 제어기로 구성되어 있다. RAID 제어기에는 RAID의 구동 소프트웨어와 패리티 제어를 위한 패리티 제어기가 포함되어 있다. RAID 제어기는 소프트웨어 혹은 하드웨어로 구현할 수 있다.

본 논문에서 제안한 알고리즘을 기존의 RAID 시스템에 구현하기 위해서는 그림 10과 같이 패리티

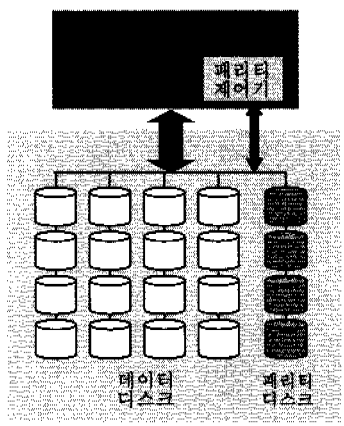


그림 9. 기존 RAID의 구성

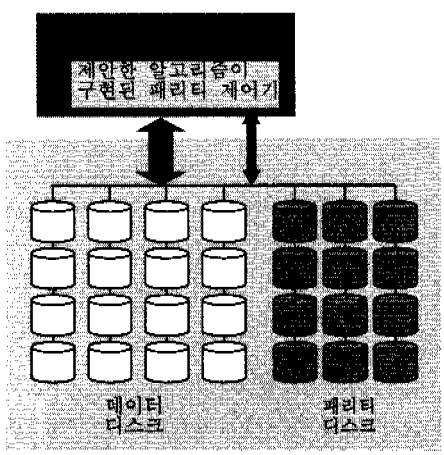


그림 10. 기존 RAID에 제안한 알고리즘을 적용한 구성

디스크의 추가와 패리티 제어기에서의 패리티 제어 방법을 제안한 알고리즘으로 변경하면 된다.

패리티 제어기는 일반적으로 하드웨어적인 XOR와 소프트웨어 연산 기능이 포함되어 있으므로, 본 논문에서 제안한 알고리즘을 소프트웨어 연산 기능에 구현하면 하드웨어의 변경 없이 기존의 RAID 제어기에 적용할 수 있다.

Ⅴ. 결 론

본 논문에서는 RAID 시스템에서 동시에 3개의 디스크 오류를 복구할 수 있는 방법에 대해 제안하고 이를 상세히 기술하였다. 먼저 행렬식 표현을 통해 문제를 정형화하였고 이를 이용하여 3개의 디스크 오류를 단지 3개의 패리티 디스크를 사용하여 복구할 수 있도록 하였다. 이는 단순한 산술적 계산만을 사용함으로써 기존의 RAID 제어기에도 구현될 수 있다. 또한 데이터들의 크기는 응용되는 분야에 따라 임의의 크기로 사용할 수 있다.

기존의 방법들 중 길쌈 형태의 코드를 사용하면 디코딩 과정에서 오류 발생시 그 오류가 무한히 전이되지만, 본 논문에서 제안한 방법은 블록 형태의 코드이므로 오류 전이에 대한 문제가 없다.

본 논문에서는 단순히 RAID의 경우만을 생각하였지만 이 방법을 자기 테이프와 같은 멀티 트랙 응용 분야에서도 적용할 수 있다.

참 고 문 헌

[1] Randy H. Katz, Garth A. Gibson and David A. Patterson, *Disk system architectures for high performance computing,* Proceedings of the IEEE, vol.77, no.12, pp.1842-1858, December 1989.

[2] A. L. Narasimha Reddy and Prithviraj Banerjee, *An evaluation of multiple-disk I/O systems, IEEE Transactions on Computers,* vol.38, no.12, pp.1680-1690, December 1989.

[3] D. Patterson, G. Gibson, and R. Katz, *A case for redundant arrays of inexpensive disks (RAID),* Proceedings of the ACM SIGMOD International Conference on Management of Data, pp.109-116, June 1988.

[4] K. Salem and H. Garcia-Molina, *Disk striping,* Proceedings of the International Conference of

Data Engineering, pp.336-342, February 1986.

[5] Shu Lin and Daniel J. Costello Jr., *Error Control Coding: Fundamentals and Applications*, Prentice Hall, New Jersey, 1983.

[6] T. Fuja, C. Heegard, and M. Blaum, *Cross parity check convolutional codes*, IEEE *Transactions on Information Theory*, vol.35, no.6, pp.1264-1276, November 1989.

[7] M. Blaum, J. Bruck, and A. Vardy, *MDS array codes with independent parity symbols*, IEEE Transactions on Information Theory, vol.42, no.2, pp.529-542, March 1996.

[8] O. Keren and S. Litsyn, *A class of array codes correcting multiple column erasures*, IEEE Transactions on Information Theory, vol.43, no.6, pp.1843-1851, November 1997.

[9] M. Blaum, J. Bruck, and A. Vardy, *Binary codes with large symbols*, Proceedings of IEEE International Symposium on Information Theory, pp.508, June 1994.

[10] M. Blaum and R. Roth, *New array codes for multiple phased burst correction*, IEEE Transactions on Information Theory, vol.39, no.1, pp.66-77, January 1993.

[11] G. Alvarez, W. Burkhard, and F. Cristian, *Tolerating multiple failures in RAID architectures with optimal storage and uniform declustering*, Proceedings of the 24th Annual ACM/IEEE International Symposium on Computer Architecture, pp.62-72, June 1997.

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

[13] C. Park, *Efficient placement of parity and data to tolerate two disk failures in disk array systems*, IEEE Transactions on Parallel and Distributed Systems, Vol.6, No.11, pp.1177-1184, November 1995.

[14] Eric W. Weisstein, *Concise Encyclopedia of Mathematics(CD- ROM)*, CRC Press, 1998.

[15] D. Patterson, P. Chen, G. Gibson, and R. Katz, *Introduction to redundant arrays of inexpensive disks(RAID)*, Proceedings of COMPCON Spring '89, pp.112-117, February 1989.

[16] W. Nurkhard and J. Menon, *Disk array storage system reliability*, Proceedings of the Twenty-third International Symposium on Fault-Tolerant Computing, pp.432-441, June 1993.

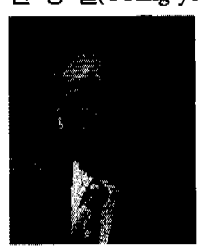
[17] W. Courtright II., G. Gibson, M. Holland, and J. Zelenka, *A structured approach to redundant disk array implementation*, Proceedings of IEEE International Computer Performance and Dependability Symposium, pp.11-20, September 1996.

박 종 원(Chong-Won Park) 정회원

1981년 2월 : 한양대학교 전자통신공학과(공학사)
1983년 2월 : 한양대학교 전자통신공학과(공학석사)
1984년~현재 : 한국전자통신연구원 책임연구원

<주관심 분야> RAID 시스템, 컴퓨터 네트워크, 시스템 아키텍춰, 데이터 통신

한 영 열(Young-yeral Han) 정회원



1960년 2월 : 서울대학교 전자공학과 (공학사)
1976년 5월 : Univ. of Missouri 통신공학 (공학석사)
1979년 5월 : Univ. of Missouri 통신공학 (공학박사)
1985년~1987년 : 전자통신에 관한 국제기구 국내연구단(운영위원)
1988년~1989년 : 미국 콜로라도 주립대(교환교수)
1991년~1994년 : 체신부 전파육성협의회(운영의원)
1981년~1995년 : 한국 통신 학회(이사, 상임이사, 부회장)
1995년~1996년 : 미국 오레곤 주립대학(교환교수)
1997년~현재 : 한국 통신 학회 (감사)
1993년~현재 : 특허청(객원 심사 위원)
1980년~현재 : 한양대학교 전기전자공학부 교수

<주관심 분야> 확률/통계통신, 디지털 통신이론, 이동통신, 통신시스템