□ 기술해설 □

고성능 고신뢰도 입출력 시스템을 위한 디스크 배열 기술†

포항공과대학교 박찬익* · 강득윤**

목 차

1. 서 론
2. 디스크 배열
2.1 디스크 고장 허용성
2.2 운영 모드
2.3 데이타 및 패리티 배치
2.4 여분 공간(Sparing) 구성
3 RAID의 확장 구조 : RAID 5M
3.1 신뢰도 분석
3.2 성능 분석
4. 결 론

1. 서 론

저가의 마이크로프로세서 기술의 발전으로 말미암아 고성능 병렬컴퓨터를 매우 용이하게 구축할 수 있게 되었으며, 또한 병렬 컴퓨터의 사용이 다양한 분야에 걸쳐 확산되고 있다. 병렬처리 기술의 발전과 아울러 마이크로프로세서 성능의 급속한 발전은 컴퓨터 시스템 성능에 있어 입출력 부분이 차지하는 비중을 더욱 높이고 있다.

컴퓨터 응용 분야에서 multimedia, graphics, scientific visualization 등의 요소들이 점차 일반화되어 가고 있는 추세이다. 따라서 컴퓨터 시스템의 높은 계산 처리 성능과 아울러 대용량 고성능 입출력 시스템의 지원이 반드시 필요하다. 그러나 현재의 분산 병렬 컴퓨터 시스템은 이러한 I/O-intensive한 응용을 적절히 지원하지 못하고 있는 실정이다. 예를 들어, multimedia 응용의 경우 DVI로 압축시킨 NTSC-quality 비디오 재생을 위해 약 1.2 Mbps의 성능이, CD-quality 오디오를 위해 약 1.4 Mbps 정도의 성능이, 그리고 미래의 HDTV-quality 비디오의 경우 40~100 Mbps 정도의 성능이, 그리고 압축시키지 않은 칼라 오디오의 경우 약 900 Mbps의 성능이 요구된다. 또한 패턴 매칭의 예를 보더라도 일반적으로, 입력 데이타를 읽어 들이는 데 128개의 프로세서로 구성된 병렬 시스템이 대략 3.66 Mbyte/sec 정도의 입출력 성능을 요구한다[7]. 그러나 일반적으로 병렬 컴퓨터에서의 화일시스템은 단지 수백 Kbyte/sec 정도의 sustained 성능을 나타내므로[9,19,26], 따라서 앞서 언급한 응용 문제들을 효과적으로 지원하는 데 문제가 있다.

데이타가 디스크로부터 응용 프로그램의 목적지까지 이동하는 경로 상에는 여러 가지 잠재적인 입출력 병목 현상의 요인들이 있다. 디스크 장치 자체가 첫번째 위치하며, 두번째로 디스크 장치와 디스크 제어기를 연결하고 있는 링크가 있으며, 세번째로 디스크 제어기이고, 그리고 디스크 제어기로부터 디스크 버퍼로 이용되는 주기억장치로 데이타가 이동되어야 하므로 주기억장치에서 네번째의 요인을 가지며, 화일 시스템이 디스크 버퍼에 도달해 있는 데이타를 응용 프로그램이 지정한 목적지까지 전달하는 일을 해야 하므로 화일 시스템에서 다섯번째의 요인을 가진다. 이처럼 다양한 입출력 병목 요인을 해결하기 위해 오래전부터 운영 체제 화일 시스템 수준에서부터 디스크 하드웨어 장치 수준까지

†이 연구는 한국과학재단 1993년도 핵심 전문 연구와 국방과학연구소 장기 기초 연구의 지원을 받았음
*종신회원
**준회원

여러가지 방법들이 제안되었다. 운영 체제 수준에서 multiprogramming상에서의 계산 과정과 입출력 과정과의 overlapping 방법, single task 상에서의 read-ahead와 write-behind 방법[8] 등을 통한 주 메모리 hit ratio를 증가시키는 형태가 있으며, 좀 더 직접적으로는 화일 입출력 서비스 시간을 구성하는 운영체제 오버헤드(overhead). 디스크 contention, 디스크 서비스 시간(seek, rotation, transfer) 등의 각 요소들을 감소시키기 위한 chaining 방법[3], 디스크 블럭 할당 방법[18,23], zero latency read 방법[33], 다중 copy 방법[2], request reordering 방법[10, 34], I/O 프로세서 장치[16], 다중 seek head disk[7], disk caching[27,35] 등이 있다. 그러나, 이런 방법들은 근본적으로 한 개 디스크 장치의 성능에 제한적일 수 밖에 없다. 따라서 디스크 장치 자체의 성능 제한을 해결하기 위해 입출력 요구들 상에 나타나는 병렬성을 여러 개의 디스크들을 운영하여 이용하는 병렬 디스크 시스템에 대한 연구가 활발하다[4,17,21,29,33,39].

그러한 병렬 입출력 시스템을 설계하는 데 있어 중요한 과제들은 실제 응용 문제에 대해 추상화되는 수준에 따라 다음과 같은 세 가지로 나눌 수 있다 : 입출력 서브시스템의 구조, 디스크 캐싱, 화일 저장 구조.

본 논문에서는 이러한 배경하에 최근 많은 관심이 고조되고 있는 병렬 입출력 시스템에 관하여, 특히 디스크배열(disk array)에 대하여 문제점 및 현재 기술 수준을 살펴보고, 기존의 구조를 확장하여 신뢰도를 향상시킨 RAID 5 M을 제안하고 분석한다.

2. 디스크 배열

디스크 배열은 하드디스크들을 병렬적으로 운영함으로써 입출력 성능을 개선시키며 아울러 중복(redundancy)을 이용하여 신뢰도를 향상시킨 데이타 저장 장치이다. 운영체제의 입장에서 보면 디스크 배열은 하나의 하드디스크로 간주된다. 그러나 최적의 성능을 얻기 위해서는 한 개의 하드디스크의 경우와는 달리 그 자체로의 기본 운영 소프트웨어 지원이 반드시 필요하다.

디스크 배열 방법은 Cray나 Connection Machine 등의 과학 계산용 수퍼 컴퓨터 시스템에서는 과학 계산이 요구하는 큰 입출력 요구를 만족시키기 위해 오래전부터 연구 개발되어 왔으며, 최근에는 DEC, HP, IBM, NEC, Sony 등의 워크스테이션 제작사들 뿐 아니라 Compaq, Dell 등 개인용 컴퓨터 시스템 회사들의 네트워크 화일 서버 등의 기종에 까지 적용될 만큼 컴퓨터 시스템에서 핵심적인 요소 기술로 자리잡아 가는 상태이다[14].

이러한 디스크 배열에 대한 연구는 향후 디스크 form factor가 감소하고, Mbyte 당 cost가 점차적으로 소형 디스크들에서 최적으로 나타나고 있으며, 또한 디스크 신뢰도, 무게, 소비 전력 등이 크게 향상되고 있는 현 추세에 힘입어 더욱 활발해질 것으로 예상된다[12]. 대표적으로 Univ. of California at Berkeley와 IBM에서 많은 연구들이 발표되었으며, 특히 UC Berkeley에서 1988년에 디스크 배열 구조를 다섯가지 level로 구분하여 RAID level 1-5을 제안한 바 있다[29].

2.1 디스크 고장 허용성

디스크 배열의 경우 많은 수의 디스크들을 이용하게 됨으로써 전체 신뢰도가 감소되는 문제점이 있다. 이를 위해 저장되는 데이타 수준에서 중복 정보(redundant information)를 부가적으로 저장해둠으로써 해결하는 데, 이러한 중복 정보를 어떻게 구성할 것인가에 따라 다음과 같은 여러 종류가 있다.

- 미러링(mirroring)
- 해밍 코드(Hamming code)
- 패리티 코드(Parity code)
- 그외 에러 수정 코드들(Error Correcting Codes)

먼저 미러링의 경우, 중복 데이타를 원래 데이타와 동일하게 만들어 내어 저장하는 형태인데 RAID level 1에서 사용하는 방법이다. 따라서 중복 데이타를 저장하기 위해서는 원래 데이타를 저장하고 있는 용량만큼의 디스크가 더 필요하다. 그외의 해밍 코드나 패리티 코드 등은 일반

적인 에러 수정 코드에 속하는 것으로 이를 이용하여 중복 데이타를 구성한다. 이 때 원래 데이타를 저장하는 디스크를 데이타 디스크라 하고, 중복 데이타를 저장하는 디스크를 확인(check) 디스크라 한다.

RAID level 2에서는 해밍 코드를 이용하여 중복 데이타를 만든다. 그러나, 해밍 코드와 같은 일반적인 에러 수정 코드를 사용하는 경우 에러 발생 위치를 검출하기 위해 많은 중복 데이타를 저장해야 한다. 디스크 배열에서의 경우에는 고장이 발생한 디스크 위치는 그와 연결된 디스크 제어기에서 검출할 수 있으므로 에러 위치 검출을 위해 필요한 부가적인 중복 데이타를 없앨 수 있다. 이에 따라 해밍코드를 이용한 RAID level 2와는 달리 RAID level 3, 4, 5에서는 에러 위치 검출은 디스크 제어기에서 담당한다고 가정하고 에러 수정만을 위해 패리티 코드를 이용한다. 예를 들어, A와 B 두 데이타를 네 개의 디스크로 이루어진 디스크 배열에 저장하는 경우 두 개의 디스크들에는 각각 A와 B 원래 데이타를 저장하고, 다른 두 개 디스크에는 A+B, A+2B로 결정된 중복 데이타를 저장한다. 그림 1의 (a)는 원래 데이타 (A, B)에서 중복 데이타를 구하는 방법을 보여 주고 있다. 코딩 방법은 매트릭스로 표현되어 있다. 만일 A 데이타를 저장하고 있는 디스크와 A+B를 저장하고 있는 디스크 등 두 개의 디스크가 고장이 날 경우 (그림 1에서는 디스크 고장이 X로 표시되었다.), 먼저 중복 데이타 코딩시 사용한 매트릭스를 이용하여 그의 역(inverse) 매트릭스를 구한다. 그 다음 접근 가능한 두 디스크에서 얻은 데이타에 계산된 역 매트릭스를 가하면 손상된 데이타를 복구할 수 있음을 그림 1(b) (c)에서 보여 주고 있다.

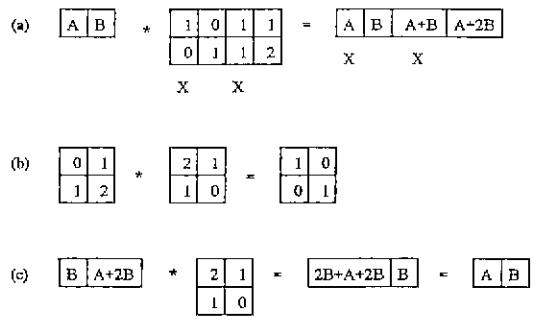

그림 1 에러 수정 과정: (a) 중복 데이타 생성 방법 및 디스크 고장 발생 (b) 데이타 복구를 위한 중복 데이타 생성 매트릭스의 역매트릭스 도출 (c) 잔류 데이타로부터 손실 데이타 복구

두 개 이상의 디스크 고장을 허용하기 위해 두 개 이상의 에러 발생에 대해서도 수정 능력을 가지는 full-2, 2d-parity, full-3, additive-3 등 다양한 코드들이 제안되었다[11]. 그러나 이런 일반적인 에러 수정 코드를 디스크 배열에 직접 적용할 경우 몇 개의 디스크들은 중복 데이타만을 저장하는 데 전적으로 사용되어야 한다. 즉, 체크 디스크를 데이타 디스크와 구별하여 따로 두어야 한다.

디스크 배열이 사용한 중복 데이타의 구성 및 배치에 따라 허용되는 고장 디스크 갯수가 정해짐은 이미 설명한 바와 같다. 디스크 고장이 발생하는 경우 이로 인해 접근 불가능한 데이타를 고장나지 않은 디스크 상에 복구하는 과정이 필요하다. 이를 위해 항상 시스템내에 대기시켜 놓은 몇 개의 디스크들을 hot spare 혹은 standby 디스크라 하며 hot spare 디스크 갯수와 신뢰도 척도로 사용하는 데이타 손실까지의 평균 시간(MTTDL: Mean Time To Data Loss)과의 관계는 그림 2에 나타나 있다. 그림에서 보여주듯이 hot spare 디스크 수가 없을 경우에 비해 있는 경우의 MTTDL 수준은 order of magnitude가 증가함을 알 수 있다. 이 때 hot spare 디스크 갯수에는 거의 영향을 받지 않음을 아울러 보여

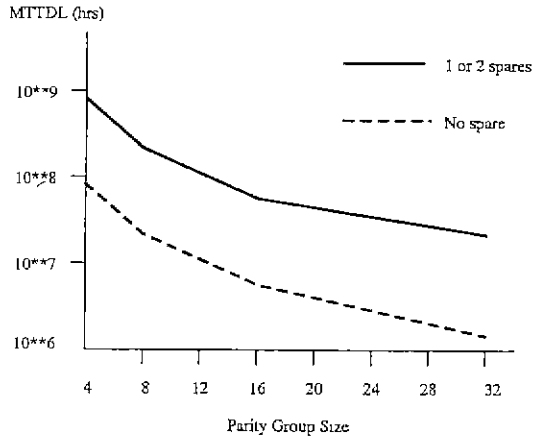

그림 2 Hot spare 디스크 갯수와 MITTDL

준다.

2.2 운영 모드

디스크 배열은 운영상의 상태에 따라 다음과 같은 여러 종류의 운영 모드들로 나눌 수 있다.

- 정상 모드(normal mode)
- 고장 모드(degraded mode)
- 구축 모드(rebuild mode)
- 감소 능력 모드(normal and reduced capacity mode)
- 재구성 모드(copyback mode)

먼저 정상 모드는 대스크 고장이 발생하지 않은 일반적인 상황하에서 이루어지는 모드이며, 디스크 고장이 발생한 상태의 수행 환경인 경우 고장 모드라 한다. 고장 모드에서는 고장난 디스크로의 입출력 요구들 중 쓰기 요구는 그와 관련한 중복 데이타를 갱신해 주어야 하는 요구로 변화되며, 읽기 요구는 그와 관련한 중복 데이타 및 정상 데이타를 읽는 요구로 변화된다. 따라서, 고장 모드에서의 성능은 정상 모드에 비해 현저하게 감소한다. 고장 모드 하에서 이용가능한 hot spare 디스크가 있다면 구축 모드로 이행한다. 구축 모드란 고장난 디스크 상에 있던 데이타들을 나머지 정상 디스크에 저장되어 있는 데이타를 이용하여 생성하여 hot spare 디스크로 저장하는 동작이 개시된 후의 수행 상태를 말한다. 감소 능력 모드는 hot spare 디스크로의 데이타 복구가 완전히 끝이 났을 경우를 말하며, 정상 모드에서와 같은 성능을 제공할 수 있으나 hot spare 디스크를 사용해 버렸다는 의미에서 감소능력으로 표현한다. 마지막으로 재구성 모드는 hot spare 디스크에 복구된 데이타를 고장난 디스크를 대체한 새로운 디스크로 재정하는 동작이 시작된 후의 상태를 말한다. 새로운 디스크로의 데이타 저장이 완전히 끝나면 다시 원래의 정상 모드로 복귀한다. 그림 3은 각 수행 모드와 성능과의 일반적인 관계를 보여 준다.

응용에 따라서는 각 모드 하에서 반드시 지원해야 하는 최소 성능이 정의될 수 있으므로 디스크 배열을 설계할 때 이에 대한 고려가 있어야 한다. 최근 들어 고장 모드하에서 성능이 정상 모드에 못지않게 중요하게 인식되어 이에 대한 연구가 활발하다. Reddy는[32]에서 고장 모드하의 각 디스크 부하들을 균등화할 수 있도록 데이타 배치를 함으로써 고장 모드하의 성능을 개선하는 방법을 제안하고 있다. 고장 모드의 성능을 분석하기 위해서는 데이타 복구 방법에 대한 연구가 선행되어야 한다. [25]에서는 Baseline Copy 방법, Rebuild with Redirection of Reads 방법, Piggy-backing Rebuild 방법 등 세 가지를 제안하였다. 세 가지 방법들은 공통적으로 고장난 디스크로의 쓰기 요구들은 모두 hot

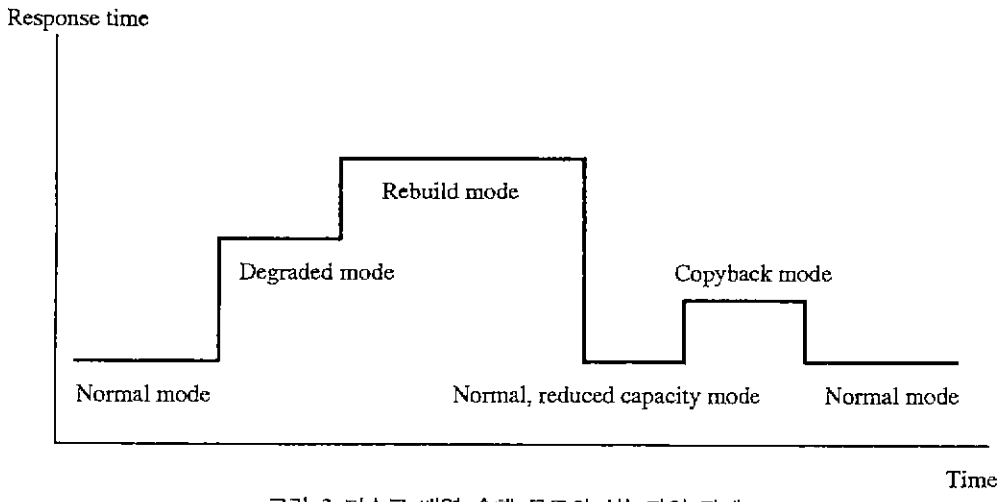

그림 3 디스크 배열 수행 모드와 성능과의 관계

spare 디스크로 보내어 처리되도록 한다. 이렇게 함으로써 정상 디스크로부터 데이타를 읽어서 복구해야할 데이타 양을 감소시킨다. Baseline Copy 방법은 고장난 디스크에 있던 데이타에 대해 처음부터 끝까지 순차적으로 복구 및 저장을 반복 수행하는 것이며, 이 때 고장난 디스크로의 쓰기 요구에 의해 이미 hot spare 디스크에 복구되어 있는 부분은 생략한다. Rebuild with Redirection of Reads 방법은 고장 모드하에서 읽기 요구를 처리할 때, 그 요구에 해당되는 데이타가 이미 hot spare 디스크에 복구되어 있는 경우 hot spare 디스크에서 그 읽기 요구를 처리하도록 하거나 또는 정상 디스크로부터 필요한 데이타를 읽어 데이타를 복구한 후 처리하도록 한다. 이 두 경우의 선택은 현재 어떤 디스크에서 병목 현상이 발생하고 있는 지에 따라 결정한다. 마지막 Piggy-backing Rebuild 방법은 복구된 데이타들이 버퍼 영역에 위치할 때(입출력 요구들의 처리 후 데이타는 버퍼에 얼마간 잔류한다.) 그 버퍼 영역을 마치 수정된 것 처럼 표시해 둠으로써 추후 그 버퍼 영역이 전환될 때 자동으로 hot spare 디스크로 저장되도록 하는 것이다.

2.3 데이타 및 패리티 배치

디스크 배열은 여러 디스크들로 데이타를 분산 배치(striping)시킴으로써 입출력 요구간의 병렬성을 도출하고 그 병렬성을 이용하여 성능을 개선시키는 것이다. 이때, 분산 배치시키는 기본 단위 즉 배치 단위(striping unit)의 크기에 따라 입출력 요구들에 나타나는 병렬성도 다양하게 나타난다. 예를 들어, RAID level 3의 경우 배치 단위는 bit 혹은 byte이며, RAID level 4와 5의 경우 sector, track, 혹은 cylinder 단위이다. 배치 단위가 bit 혹은 byte인 경우는 하나의 입출력 요구를 처리하기 위해 모든 디스크들이 참가해야 하므로 한 요구 내에서 병렬성이 나타나며, 배치 단위가 sector 이상의 경우는 입출력 요구의 크기에 따라 한 요구 내의 병렬성으로 나타나기도 하고, 또한 여러 요구들 간의 병렬성으로 나타나기도 한다. 하나의 요구내의 병렬성을 이용할 경우 서로 동기화된 디스크를 이용하여 디스크 배열을 구성하며, 여러 요구들 간의 병렬성을 이용하는 데는 비동기적 즉 서로 독립적인 디스크들을 이용하여 구성한다.

또한 신뢰도 향상을 위해 에러 수정 코드를 적용할 경우 중복 데이타만을 위해 따로 체크 디스크를 두어야 함은 전술한 바와 같다. 이 경우 데이타 쓰기 요구시 체크 디스크 상에 부하가 집중되어 병목 현상이 발생하며, 따라서 시스템 성능에 많은 영향을 미친다. 이를 위해 패리티 코드를 직접 적용하여 중복 데이타 저장을 위해 체크 디스크가 따로 유지되는 RAID level 4와는 달리 RAID level 5에서는 중복 데이타를 모든 디스크들에 균일하게 분포시킨다. Lee는 RAID 구조에 대하여 [20]에서 Flat-Left-Symmetric, Asymmetric, Symmetric, Extended-Left-Symmetric 등 여러 종류의 데이타 배치 방법을 제안하고(그림 4 참조), 작은 크기의 읽기 및 쓰기 요구에 대한 성능은 데이타 배치와 거의 상관이 없음을 보였으며, 또한 입출력 부하에 대해 정확하게 모델링할 수 없을 경우 일반적으로 Left-Symmetric 배치 방법이 대부분 좋은 성능을 제공함을 보이고 있다.

Gray와 Walker는 [13]에서 parity striping 방법을 제안하였다. 이 parity striping 방법은 패리티 정보는 각 디스크들에 분산 배치하지만 데이타는 분산 배치하지 않게 하는 것으로, throughput 측면으로는 미러링과 동등한 성능을 제공하며 아울러 cost/GB 측면에서는 RAID level 5와 비슷한 결과를 보였다[5,13].

Menon 등은 [24]에서 데이타 및 패리티 정보들이 고정된 위치에 항상 저장되는 것이 아니라 동적으로 변화하는 floating parity and data placement라는 방법을 제안하였다. 이것은 동적으로 변화하는 위치 정보를 계속 유지해야 하므로 디렉토리 유지 및 디스크 저장 공간상의 효율은 감소하지만 쓰기 요구의 최적 위치를 결정할 수 있어 추후 설명될 쓰기 오버헤드를 크게 줄일 수 있다.

두 개 이상의 디스크 고장 허용을 위해서 일반적인 에러 수정 코드를 적용하는 경우는 [28]에서 제안한 방법을 이용하여 중복 데이타를 분

P0	0	1	2	3
4	P1	5	6	7
8	9	P2	10	11
12	13	14	P3	15
16	17	18	19	P4

Right-Asymmetric

P0	0	1	2	3
7	P1	4	5	6
10	11	P2	8	9
13	14	15	P3	12
16	17	18	19	P4

Right-Symmetric

0	1	2	3	P0
10	11	P2	13	4
P4	21	12	23	14
20	31	22	P6	24
30	P8	32	33	34

0	1	2	3	P0
4	5	6	P1	7
8	9	P2	10	11
12	P3	13	14	15
P4	16	17	18	19

Left-Asymmetric

0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

Left-Symmetric

5	6	7	P1	9
15	P3	17	8	19
25	16	27	18	P5
35	26	P7	28	29
P9	36	37	38	39

Extended-Left-Symmetric

그림 4 디스크 갯수가 5일 경우 데이타 배치 방법

산 배치시킬 수 있다. 그림 5는 디스크 갯수가 7이고 중복 데이타를 전체 디스크 용량 중 33.3 %만을 이용하여 분산 저장한 예를 보여 준다. 여기서 D는 데이타 블럭을 나타내며, 그의 subscript는 해당 데이타가 속한 패리티 그룹 index를 나타낸다. 따라서 각 데이타 블럭은 두 개의 서로 다른 패리티 그룹에 속하고 있음을 알 수 있다.

디스크 배열에서 패리티같은 중복 정보들을 일관성있게 유지하기 위해 쓰기 요구시 항상 그와 관련한 중복 정보에 대한 갱신이 반드시 수반되어야 하는 부가적인 작업이 있다. 일반적으로 이것을 쓰기 오버헤드(overhead)라 말하며, 디스크 배열에서의 각 디스크들이 동기적인지 비동기적인지에 따라 다른 형태로 나타난다. 예를 들어, 하나의 입출력 요구를 처리하기 위해 모든 디스크들이 동기적으로 참여하는 RAID level 3의 경우에는 운영 체제에서 처리되는 최소 블럭 크기, 즉 논리적 블럭 크기에 관련되어 나타난다. 즉, 각 디스크의 최소 처리 크기는 대부분의경우 512 byte로 주어지는 데, 만약 RAID level 3에서 사용하는 디스크 수가 5이면 한번에 입출력 처리해야 하는 크기는 2 Kbyte가 된다. 만일 운영체제에서 사용하는 논리적 블럭 크기가 2 Kbyte보다 작은 512 byte라면, 한 논리적 블럭에 해당하는 작은 크기의 쓰기 요구를 처리할 경우 디스크 배열의 기본 입출력 단위는 2 Kbyte가 되어야 하므로 512 byte를 제외한 나머지

P_0	P_1	P_2	P_3	P_4	P_5	P_6
D_{23}	D_{34}	D_{45}	D_{56}	D_{60}	D_{01}	D_{12}
D_{14}	D_{25}	D_{36}	D_{40}	D_{51}	D_{62}	D_{03}

그림 5 디스크 갯수=7, 중복 데이타 저장 허용량=전체 용량의 33.3%의 경우 RM2 방법에 의한 데이타 배치

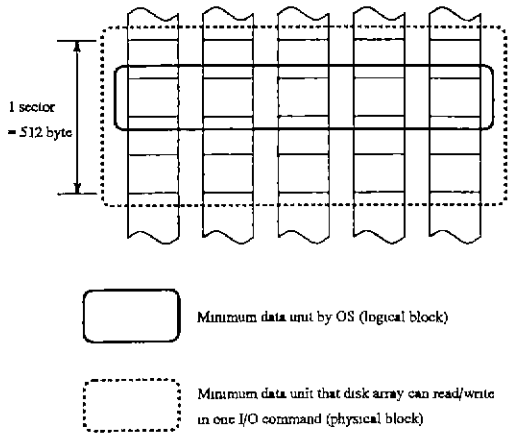

그림 6 RAID level 3의 경우 쓰기 오버헤드

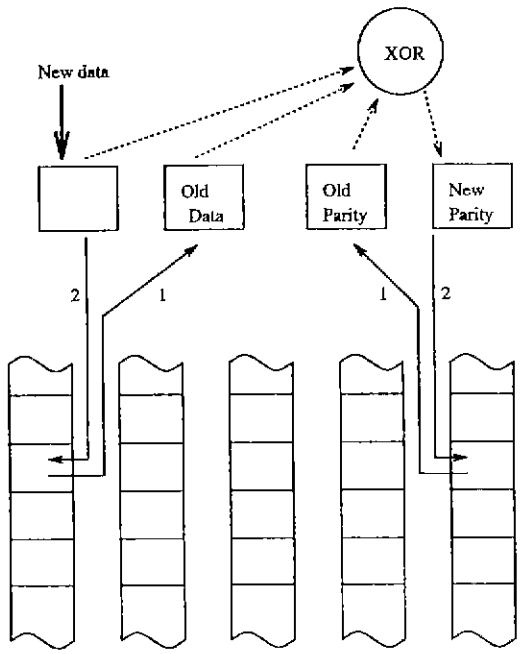

그림 7 RAID level 5 쓰기 오버헤드

부분을 읽어 들인 후 원래의 쓰기 요구를 2 Kbyte 크기의 쓰기 요구로 전환하여 처리해야 한다. 따라서, 부가적인 읽기 요구가 발생한다(그림 6 참고[14]). 이 문제는 근본적으로 운영체제의 논리적 블럭 크기와 디스크 배열의 한번에 처리하는 입출력 크기와의 불일치에 의한 것이므로, 이를 해결하기 위해서는 운영체제의 논리적 블럭 크기를 증가시켜야 한다. 그러나 운영체제에서 사용하는 논리적 블럭 크기는 상한점이 있으므로 (예를 들어, BSD UNIX나 Net Ware 3.11의 경우는 64 Kbyte로 제한된다.), 이 경우는 버퍼 캐쉬를 적절히 운영하여 다소간 해결할 수 있다.

모든 디스크들이 비동기적으로 수행 가능한 RAID level 4와 5에서는 그림 7에서 보여 주는 바와 같이, 작은 크기의 쓰기 요구를 하나 처리하기 위해 옛 데이타 읽기, 옛 패리티 읽기, 새로운 데이타 쓰기, 새로운 패리티 쓰기 등 네 번의 입출력 요구를 처리해야 하는 부가적인 요구들이 수반된다. 쓰기의 경우 부가적으로 발생한 입출력 요구들에 대한 처리가 모두 끝이 난 후에야 비로소 원래의 쓰기 요구에 대한 처리가 끝이 났다고 말할 수 있다. 따라서, 디스크들이 비동기적으로 수행되는 이런 상황에서도 쓰기 요구를 처리하기 위해 발생된 요구들은 근본적으로 동기적인 특성을 지니고 있음을 알 수 있다. 이러한 쓰기 오버 헤드 문제를 해결하기 위해 일반적으로 비휘발성(non-volatile) RAM을 이용하여 쓰기 요구를 지연시키고 아울러 동기적인 특성을 완화시키는 방법을 사용한다. 비휘발성 RAM을 사용함으로써 신뢰도에 영향을 주지 않으면서도 쓰기 오버헤드에 대한 영향을 감소시킬 수 있다.

2.4 여분 공간(Sparing) 구성

디스크 배열에서 복구된 데이타를 저장하기 위해 일정한 여분 공간을 두어야 한다. 이러한 여분 공간을 위해 hot spare라 부르는 특정 디스크를 따로 둘 수도 있고, 디스크 배열의 각 디스크당 얼마간의 공간을 이러한 공간으로 사용할 수 있게 하여 구성할 수도 있다. RAID 시스템은 level에 관계없이 여유 공간을 hot spare 디스크 상에서 지원하게 되어 있다. 여유 공간을 어떻게 지원할 것인가에 따라 다음과 같은 세 가지의 방법이 제안되었다.

- 고정 여유 공간(Dedicated sparing)
- 분산 여유 공간(Distributed sparing)[24]
- 패리티 여유 공간(Parity sparing)[31]

먼저 고정 여유 공간 방법은 RAID처럼 가장 간단하게 hot spare 디스크를 따로 두어 그 디

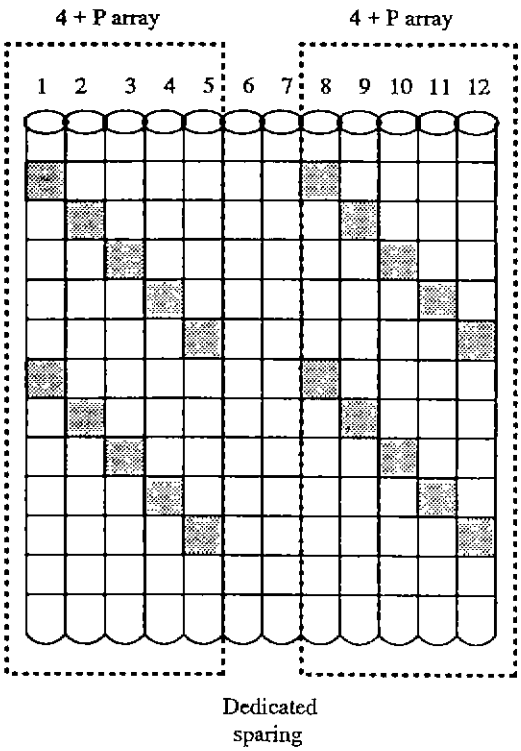

그림 8 고정 여유 공간 배치 방법: 디스크 수=12, 고정 여유 공간 디스크 수=2

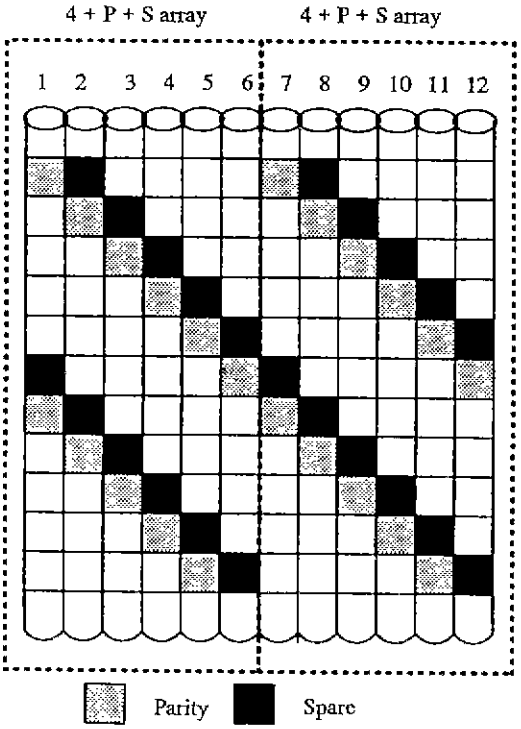

그림 9 분산 여유 공간 배치 방법: 디스크 수=12

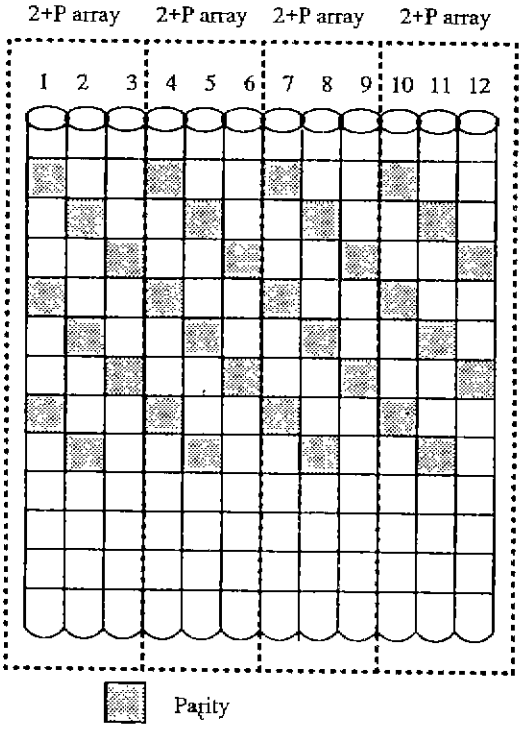

그림 10 패리티 여유 공간 배치 방법: 디스크 수=12

스크 상의 공간을 여유 공간으로 이용한다. 이 방법은 간단하지만 하나의 디스크를 완전히 대기시켜 놓아야 하는 문제점이 있다. 반면 분산 공간 방법은 디스크 배열의 각 디스크에서 부분적으로 동일한 크기의 여유 공간을 확보함으로써 분산된 여유 공간을 이용하는 방법이다. 이렇게 함으로써 정상 모드에서 대기 상태의 디스크를 두지 않아도 되므로, 모두 입출력 요구 처리에 사용될 수 있는 효과를 얻을 수 있다. 마지막으로 패리티 여유 공간 방법은 여유 공간을 따로 두지 않고 패리티 저장 공간을 필요시 여유 공간으로 이용하는 방법이다. 패리티 저장 공간을 여유 공간으로 만들기 위해서는 두 개의 패리티 공간을 사용하는 두 개의 패리티 그룹을 하나로 통합하여 하나의 패리티 공간만을 사용하게 함으로써 가능하다. 따라서 이 방법은 디스크 고장이 발생하면 패리티 그룹의 재조정이 필요하므로 운영상 복잡도는 그만큼 증가하나 디스크 공간을

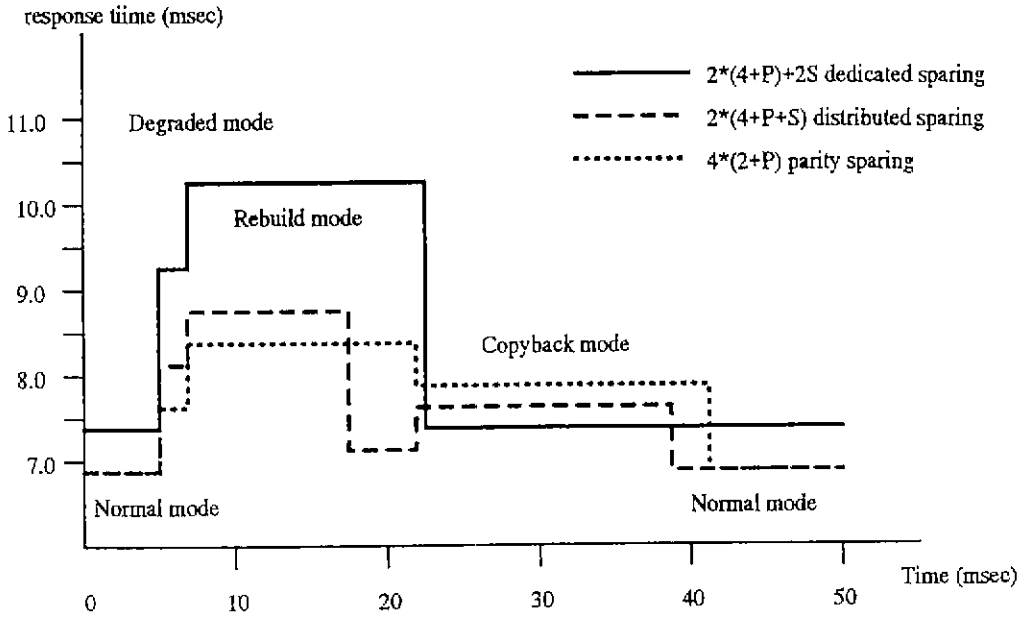

그림 11 여유 공간 배치에 따른 각 수행 모드하의 성능 비교

완전히 사용한다는 측면에서 많은 효용성을 지닌다. 그림 8, 9, 10은 위 세 가지 방법을 디스크 수가 12일 경우에 대해 각각 보여주고 있다. 같은 디스크 수라도 방법에 따라 에러 수정 그룹을 결정하는 것이 달라지므로 각기 다른 그룹 형태를 보인다. 그림 10의 경우 디스크 하나가 고장날 경우 해당 2+P array와 인접한 2+P array를 통합하여 하나의 $4+P$ array로 재구성하여 여유 공간을 확보한다. 그림 11은 수행 모드에 따라 전술한 여유 공간 확보 방법의 성능을 보여준다.

3. RAID의 확장 구조: RAID 5M

본 장에서는 RAID 5와 RAID 1 방식을 결합하여 간단하게 신뢰도를 증가시키는 RAID 5M을 제안하고, 그 신뢰도를 마코브 모델(markov model)을 이용해 비교하고 RAID 5M의 성능을 정상 동작, 디스크 고장시에 대해서 RAID 1과 RAID 5의 경우와 비교한다.

RAID 5M은 RAID 5의 각 디스크에 미러링을 부가하여 확장한 간단한 구조이다. 구성은 그림 12와 같다. d_1, d_2, …, d_5는 RAID 5 형태로 구성되며 d_i'는 d_i의 mirror image를 갖고 있는 디스크를 나타낸다.

따라서, RAID 1과 RAID 5는 하나의 디스크 고장 밖에는 허용하지 않지만 RAID 5M은 3개의 디스크 고장을 허용할 수 있다. 특별한 형태인 경우는 3개 이상이 되어도 복구할 수 있다[15].

3.1 신뢰도 분석

신뢰도 분석의 간단화를 위해 hot spare 디스크가 없다고 가정하며, 또한 디스크 고장 사건은 서로 독립적이며 평균값 λ를 갖는 포아송 프로세스라고 가정한다.

그림 13은 2N개의 디스크들로 구성된 RAID 1의 신뢰도를 보여주는 마코브 모델이다. 상태 n은 현재 동작하는 디스크의 갯수를 나타내며, 상태 F는 시스템이 더 이상 입출력 요구에 대한 처리를 할 수 없는 경우를 나타낸다. 이 때, 현재 k개의 디스크가 고장난 상태하에서 고장난 k개의 미러에 해당되는 디스크(peer 디스크)들 중에서 추가로 고장이 날 경우 시스템이 더 이상 동작할 수 없기 때문에, 상태(2N−k), $1 \le k \ge N$에서 상태 F로의 전이율(transition rate)은 kλ가 된다.

RAID 5는 오직 하나의 디스크 고장만 허용하고 2번째 고장부터는 바로 시스템의 고장으로

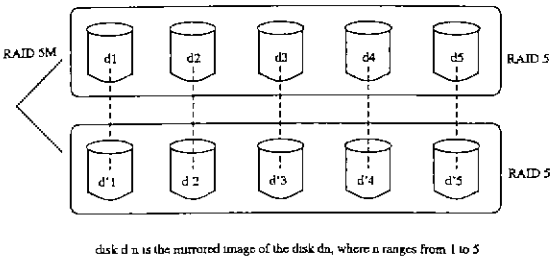

그림 12 10개의 디스크로 구성된 RAID 5M

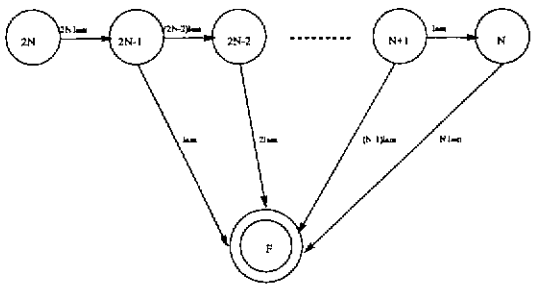

그림 13 RAID 1의 신뢰도 Markov 모델: lam=λ

이어지므로 신뢰도 모델은 그림 14와 같이 주어진다.

그림 15의 각 상태(x, y)에서 x는 서로 peer 관계에 있는 디스크가 둘 다 망가진 경우의 수를 나타내며 y는 동작하고 있는 디스크의 갯수를 나타내며 F는 시스템이 망가진 상태를 나타낸다. RAID 5M은 x가 2 미만인 모든 상태에서 동작할 수 있다. 상태 전이율의 계산은 RAID 1에서 한

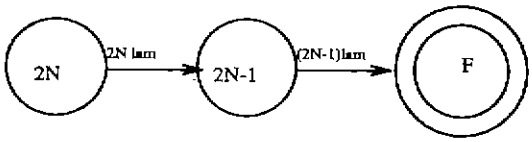

그림 14 RAID 5의 신뢰도 Markov 모델: lam=λ

방법과 동일하다.

각 모델으 신뢰도를 λ=1/30000로 하고 HARP [1]를 이용해 마코브 모델을 분석한 후 특정 디스크 갯수에 대해 신뢰도를 구했다.

그림 16은 8개의 디스크들로 이루어진 각 디스크 배열 시스템의 신뢰도를 나타내고 있다. RAID 1은 기본적으로 하나의 디스크 고장만 허용하지만 2개 이상의 디스크 고장이 발생해도 동작할 수 있는 경우가 있으므로 RAID 5에 비해 신뢰도가 우수하다. RAID 5 M은 3개의 디스크 고장을 허용하며 4개 이상의 디스크 고장이 발생해도 동작할 수 있는 경우가 많으므로 가장 신뢰도가 뛰어남을 볼 수 있다.

3.2 성능 분석

R개의 작은 읽기 요구와 W개의 작은 쓰기 요구가 디스크 배열 전체로 동일하게 분산된다고 가정하자. 여기에서 작은 요구란 하나의 데이타 배치 단위(striping unit) 크기의 요구를 말한다. R과 W는 디스크 배열 외부로부터 오는 것이고 실제 이들 요구는 개별 디스크들로 여러 개의

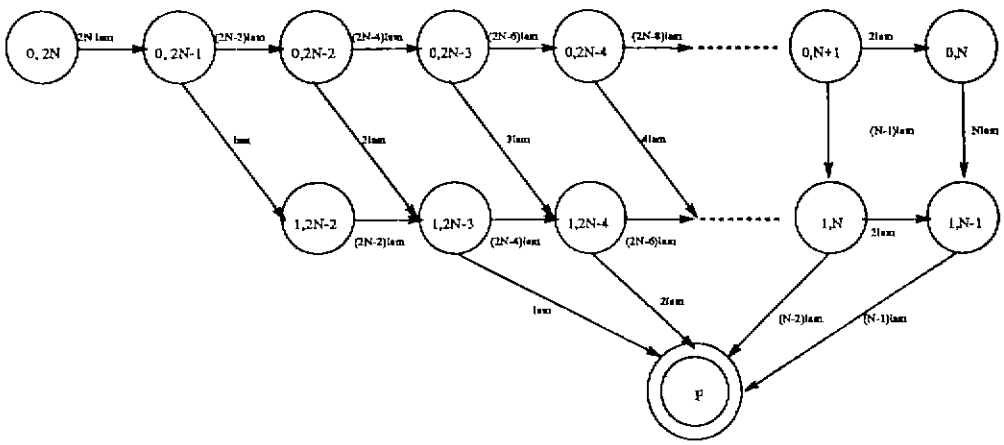

그림 15 RAID 5M의 신뢰도 Markov 모델: lam=λ

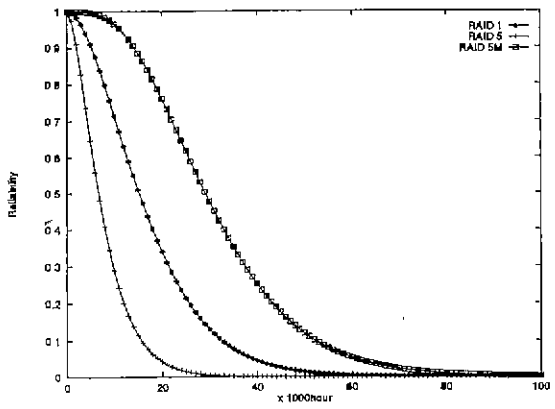

그림 16 RAID 1, RAID 5, RAID 5M의 신뢰도 비교: 디스크 수=8

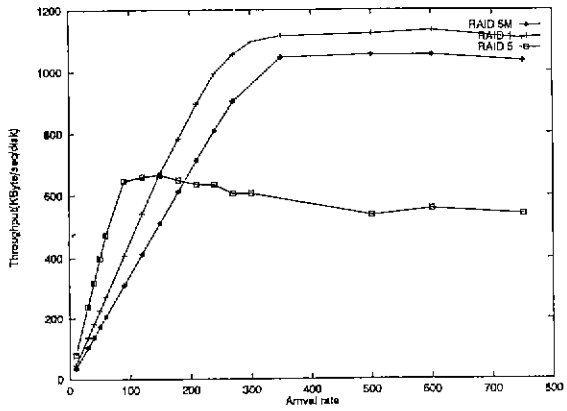

그림 17. 고장 상태에서 RAID 1, RAID 5, RAID5M의 성능 비교: 읽기 100%

읽기 또는 쓰기 요구들이 만들어진다. 다음과 같이 정상인 경우와 고장인 경우로 나누어 RAID 1, RAID 5, RAID 5M에 부가되는 각각의 부하를 구할 수 있다.

가. 먼저 정상 모드하에서 경우

· RAID 1 : 디스크의 갯수에 관계없이 R+2W개의 요구가 발생한다.

· RAID 5 : 디스크의 갯수에 관계없이 R+4W개의 요구가 발생한다.

· RAID 5M : 디스크의 갯수에 관계없이 R+6W개의 요구가 발생한다.

나. 고장 모드하에서의 경우

· RAID 1 : 디스크의 갯수가 2N 이라고 한다면

$$R+2W-\frac{W}{N} \tag{1}$$

의 요구가 발생한다.

· RAID 5 : 디스크의 갯수가 N이라고 하면 읽기의 경우 발생하는 요구의 갯수는 $R+\frac{(N-2)R}{N}$이고, 쓰기의 경우 $4W+\frac{(N-8)W}{N}$이 발생한다. 결국

$$R+4W+\frac{(N-2)R}{N}+\frac{(N-8)W}{N} \tag{2}$$

의 요구가 발생한다.

· RAID 5M : 계산의 편의상 디스크의 갯수가 2N이라고 하면 읽기의 경우 R개의 요구가 발생한다. 쓰기의 경우 $6W-\frac{2W}{N}$ 개의 요구가 발생한다. 결국

$$R+6W-\frac{2W}{N} \tag{3}$$

개의 요구가 발생한다.

시뮬레이션을 통해 확인한 결과는 앞서 분석한 부하 모델이 적합함을 보여준다. 자세한 결과 분석 및 시뮬레이션 결과는 [15]를 참조하기 바라며, 여기에서는 고장 모드하에서의 성능을 읽기 100%인 입출력 부하에 대한 결과만을 그림 17에서 보인다. 이 경우, 식 1, 식 3에 따라 RAID 1과 RAID 5M의 경우는 읽기의 경우에 고장으로 인해 시스템에 추가되는 부하는 없다. RAID 5의 경우는 식 2에 따라 3/4R의 추가 부하가 생긴다. 따라서 RAID 1과 RAID 5M이 RAID 5의 1.75배 정도의 성능을 보인다. RAID 1, 5M의 경우 디스크 갯수의 감소로 인해 고장전에 비해 성능 감소가 생기며, RAID 5의 경우는 디스크 갯수의 감소와 추가 부하로 고장전에 비해 거의 2배 가까운 성능 감소가 발생한다.

따라서, 극도의 신뢰도를 필요로 하는 응용의 경우 본질적으로 하나의 디스크 고장만 허용하는 RAID 1을 선택하는 것 보다는 2개의 부가적인

디스크 도입으로 3개의 디스크 고장을 허용하는 RAID 5M을 선택하는 것이 바람직하다. 디스크의 갯수가 충분히 크다고(N≫W,R) 가정한다면 고장시 RAID 1과 RAID 5M의 경우 시스템 전체의 부하는 변하지 않는다라고 간주할 수 있다. 고장 발생시 RAID 5의 경우 읽기 요구를 처리하기 위해서는 100%의 부하가 부가되며, 쓰기 요구를 위해서는 25%의 부하가 부가된다. 따라서 RAID 1과 RAID 5M이 고장시 성능 감소율이 RAID 5에 비해 낮음을 알 수 있다. 그러나 2개의 패리티 디스크 추가로 인해 쓰기의 경우 RAID 5M이 RAID 1, RAID 5에 비해 성능이 뒤떨어지는 것을 볼 수 있다. 그러므로 RAID 5M은 읽기 요구가 대부분이며 극도의 신뢰도를 요구하는 응용에 적합하다[15].

4. 결 론

본 논문에서는 향후 컴퓨터 시스템의 핵심적인 요소로 인식되는 입출력 부분의 성능 향상을 디스크 배열을 이용하여 해결하고자 했을 때 나타날 수 있는 문제점들과 그에 따른 제안된 방법들에 대해 살펴보았다. 또한 기존의 디스크 배열 구조를 간단하게 확장하여 신뢰도를 증가시킬 수 있는 RAID 5M을 제안하고 성능 및 신뢰도를 기존의 RAID 1과 RAID 5와 비교 분석하였다. 향후 입출력 시스템의 중요성이 점점 증대되어 감에 따라 디스크 배열에 관한 연구는 더욱 활발할 것으로 예상된다.

참고문헌

[1] Bavuso, S. J., J. B Dugan and K. S. Trivedi, "Analysis of Typical Fault-Tolerant Architecture Using Harp," *IEEE Trans.. Rel.*, Vol. R-36, No. 2, March 1987.

[2] D. Bitton and J. Gray, "Disk Shadowing," *Proc. 14th VLDB Conf*. 1988, pp. 331~338.

[3] J. P. Buzen, "I/O Subsystem Architecture," *Proc. IEEE,* Vol. 63, No. 6, June 1975.

[4] L.-F. Cabera dnd D. D. E. Long, "Swift: Using Distributed Disk Striping to Provide High I/O Data Rates," *USENIS Computing Systems,* Vol. 4, No. 4, Fall 1991, pp. 405~436.

[5] S. Chen and D. Towsley, "The Design and Evaluation of RAID 5 and Parity Striping Disk Array Architectures," *J. Parallel and Distributed Computing,* Vol. 17, No. 1, Jan. 1993, pp. 58~74.

[6] Thomas, W. Crockett, "File Concepts for Parallel I/O." Proc. ACM089791-341-8, 1989, pp. 574~579.

[7] P. C. Dibble, "A Parallel Interleaved File System," *Tech Rep. 334,* Univ. Rochester, March 1990.

[8] R. J. Feiertag and E. I. Organick, "The Multics Input/Output System," *Proc. 3rd ACM Symp. Operating System Principles,* 1971, pp. 35~41.

[9] H. C. French, T. W. Pratt, and M. Das, "Performance Measurement of a Parallel Input/Output System for the Intel iPSC/2 Hypercube," *Proc ACM 089791-392-2,* 1991, pp. 178~187.

[10] R. M. Geist and S. Daniel, "A Continum of Disk Scheduling Algorithms," *ACM Trans. Computer Systems,* Vol. 5, No. 1, Feb. 1987, pp. 77~92.

[11] G. A. Givson, L. Hellerstein, R. M. Karp, R. H. Katz and D. A. Patterson, "Coding Techniques for Handling Failures in Large Disk Arrays," *Techincal Report UCB CSD 88-477,* Univ. California at Berkeley, 1988.

[12] G. A. Gibson, "Redundant Disk Arrays: Reliable, Parallel Secondary Storage," Tech. Rep. UCB CSD91-613, Dec. 1990.

[13] J. Gray, B. Horst and M. Walker, "Parity Striping of Disc Arrays: Low-cost Reliable Storage with Acceptable Throughput," Tech. Rep. 90.2, Tandem Computers, Jan 1990.

[14] T. Imai and H. Eda, "WS, PC Makers Eye Enhanced Disk Arrays for Server Storage," Nikkei Electronics Asia, July 1993.

[15] D.-Y. Kang and C-I. Park, "RAID 5M: A Disk Array for High Reliability," *Tech. Rep. TR-SS-94-01,* POSTECH, Sept, 1994.

[16] K. Kannan, "The Design of a Mass Memory for a Database Computer," Proc. 5th ACM SIGARCH, Palo Alto, April 1978.

[17] M. Y. Kim, "Synchronized Disk Interleaving," *IEEE Trans. Computers,* Vol. C-35, No. 11, Nov. 1986.

[18] P. D. L. Koch, "Disk File Allocation Based on Buddy System,l *ACM Trans. Computer Systems.* Vol. 5, No. 4, Nolv. 1987, pp. 352~370.

[19] D. F. Kotz and C. Schlatter Ellis, "Prefetching in File Systems for MIMD Multiprocessors," *IEEE Trans. Parallel and Distributed Systems,* Vol. 1, No. 2, April 1990, pp. 218~230.

[20] Edward K. Lee, "Software and Performance Issues in the Implementation of a RAID Prototype,l *Tech Report UCB/CSD 90/573,* Univ. California at Berkeley, May 1990.

[21] M. Livny, S. Khoshafian, and H. Boral, "Multidisk Management Algorithms," *Proc. 1987 ACM SIGMETRICS,* May 1987.

[22] M. Malhotra and K. S. Trivedi, "Reliability Analysis of Redundant Arrays of Inexpensive Disks," *J. Parallel and Distributed Computing* Vol. 17, No. 1, Jan. 1993, pp. 146~151.

[23] M. K. McKusick, W. N. Joy, S. J. Leffler and R. S. Fabry, "A Fast File System for UNIX," *ACM Trans. Computer Systems,* Vol. 2, No. 3, Aug. 1984, pp. 181~197.

[24] J. Menon, J. Roche, and J. Kasson, "Floating Parity and Data Disk Arrays." *J. Parallel and Distributed Computing,* Vol. 17, No. 1, Jan. 1993, pp. 129~139.

[25] Richard R. Muntz, and John C. s. Lui, "Performance Analysis ot Disk Arrays Under Failuer," *Proc. 16th VLDB Conference,* Brisbone, Australia, 1990.

[26] U. Nagaraj, U. S. Shukla and A. Paulraj, kDesign and Evaluation of a High Performance File System for Message Passing Parallel Computers,l *Intl. Parallel Processing Symposium,* 1991, pp. 549~554.

[27] J. K. Ousterhout et al., "A Trace-Driven Analysis of the UNIX 4.2 BSD File System," ACM Operating System Review, Vol. 19, No. 5, Dec. 1985, pp. 15~24.

[28] C.-I. Park, "Distribution of Data/Parity to Enhance Reliability in Disk Arrays," Porc. Intl Conf. Parallel and Distributed Systems, Taiwan, Dec. 1993, pp. 535~539.

[29] David A. Patterson, Garth Gibson, and Randy H. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," *Proc. ACM SIGMOD Conference, Chicago, Illinois, June. 1988.*

[30] Paul A. Fishwick, "SIMPACK: Getting Started with Simulation and Programming in C and C++," *Tech Report TR92-022,* Univ. Florida, Gainsvile, 1992.

[31] A. L. Narasimha Reddy, "Parallel Input/Output Architectures for Multiprocessors," Ph. D. thesis, Univ. Illinois at Urbana Champaign, 1990.

[32] A. L. Narasimha Reddy, J. Chandy, and P. Banerjee, "Design and Evaluation of Gracefully Degradable Disk Arrays," *J Parallel and Distributed Computing,* Vol. 17, No. 1, Jan. 1993, pp. 28~40.

[33] K. Salem and H. Garcia-Molina, "Disk Striping," *Proc. 2nd IEEE Intl. Conf. Data Engineering.* 1986.

[34] M. I. Seltzer, P. M. Chen, and J. K. Ousterhout, "Disk Scheduling Revisited," *Proc. winter 1990 USENIX,* Washington D.C., 1990.

[35] A. J. Smith, "Disk Cache-Miss Ratio Analysis and Design Consideration,l *ACM Trans. Computer Systems,* Vol. 3, No. 3, Aug. 1985, pp. 161~203.

[36] M. Stonebraker, "Distributed RAID-A New Multiple Copy Algorithm,l Tech. Rep. UCB ERL M89-56, Electronics Research Lab. Univ. California Berkeley, May 1989.

[37] G. Weikum, P. Zabback, and P. Scheuermann, "Dynamic File Allocation in Disk Arrays," *Proc. ACM SIGMOD Conference,* Denver, May 1991.

[38] G. Weikum and P. Zabback, "Tuning of Striping Units in Disk Array Based File Systems," *Proc. 2nd IEEE Intl. Workshop on Research Issues in Data Engineering: Transaction and Query Processing,* Mission Palms, Arizona, Feb. 1992.

[39] S. Tomonaga and H Yokoda, "An Implementation of a Highly Reliable Parallel Disk System using Transputers," *Proc. 6th Transputer/Occam Intl Conf.,* June 1994.

박 찬 익

1983 서울대학교 전자공학과 학사
1985 KAIST 전기 및 전자공학과 석사
1988 KAIST 전기 및 전자공학과 박사
1989 ~현재 포항공대 전자계산학과 조교수
1991 ~1992 IBM, T. J. Watson연구소 초청 연구원
관심 분야 : 병렬처리, 운영체제, 고성능 입출력 시스템 구조, 병렬 실시간 시스템

강 득 윤

1993 금오공과대학교 전자공학과 학사
1993 ~현재 포항공대 전자계산학과 석사 과정
관심 분야 : 병렬 입출력 시스템 구조, 실시간 시스템

논 문 모 집

HCI '95 학술대회

- **대회일자** : 1995년 2월 16일~17일
- **대회장소** : 서울 KOEX
- **논문마감** : 1994년 12월 28일
- **논문제출** : 한국과학기술원 전산학과 서정연 교수
  대전시 유성구 구성동 373-1 (우 : 305-701)
  Tel : (042) 869-3534, Fax : (042) 869-3510
- **문 의 처** : 한국과학기술원 전산학과 인공지능연구센터
  Tel : (042) 869-3554, Fax : (042) 867-8700