중복 패리티 이미지 제거를 통한 RAID 5에서의 패리티 로깅 기법 성능 개선

길 준 호[†] 이 민 영[††] 이 진 호[†††] 박 명 순[††††]

요 약

RAID 5는 높은 성능과 신뢰성을 가지고 다양한 응용 분야에서 사용되지만 쓰기 작업시 성능이 저하되는 문제를 가지고 있다. 이러한 문제를 완화하기 위하여 패리티 로깅(parity logging) 기법이 제시된 바있다. 패리티 로깅 기법은 디스크로의 작은 크기의 임의의 블럭 단위 접근들을 큰 크기의 순차적인 트랙 단위 접근으로 변환하여 디스크 처리 시간을 줄이는 방법이다. 본 논문에서는 패리티 로깅 기법에서 디스크 버퍼의 패리티 이미지 중복 저장을 제거하여 패리티 갱신 오버헤드를 줄이는 방법을 설명하고 그에 대한 간략한 분석 모델과 시뮬레이션 결과를 제시하였다.

Performance Improvement of Parity Logging Scheme in RAID 5 by Eliminating Redundant Parity Image

Jun-Ho Ghil,[†] Min-Young Lee,[††] Jin-Ho Lee,[†††] and Myong-Soon Park[††††]

ABSTRACT

While RAID 5 disk arrays offer advantages of performance and reliability on wide variety of applications, they suffer from the performance degradation with write jobs. To reduce the effect of this problem, parity logging scheme was suggested. This scheme reduces disk busy time by transforming many small random accesses into large sequential accesses. This paper presents a new parity logging technique which prevents parity update image from redundant storing in the disk buffer and reduces parity maintenance overhead. A analytical model and several simulation results of our proposed scheme are presented.

1. 서 론

프로세서 등 전반적인 컴퓨터 하드웨어 핵심 요소들의 성능이 급속히 향상되고 있는 반면 입출력 장치의 성능은 이를 뒷받침하지 못하고 있다. 특히 디스크 저장 장치는 기계적인 부분에 많이 의존하기 때문에 다른 하드웨어 장치와의 성능 격차가 계속하여 벌어질 것으로 예측되고 있어 이에 대한 대책이 절실히 요구되고 있다. 현재 대용량 디스크의 대표적인 추세인 RAID (Redundant Arrays of Inexpensive Disks) 기술[1, 2, 3]은 적은 가격으로 대역폭, 처리율, 응답 시간, 오류 허용 등 여러 면에서 좋은 성능을 보이고 있으며 이미 상용화되어 보급되고 있다.

RAID는 구성 방법에 따라 여러 종류로 분류될 수 있으며 각각 그 특성이 다르다. 이들 중 RAID 5는 블럭 단위로 데이타를 처리하며 패리티 블럭을 각 디스크로 분산시켜 디스크 접근 병목 현상을 완화하는 가법으로, 현재 가장 널리 사용되고 있는 RAID 구성 방법이다. RAID 5는 소량의 읽기 요청과 대량의 읽기, 쓰기 요청에는 매우 효율적이지만 소량의 쓰기 요청인 경우는 패리티를 계산하기 위해서 읽고 수정하고 기록하는 과정(read-and-modify)을 거치기 때문에 좋

† 정 회 원 : 국방전산소 전산지원과 체계관리
†† 준 회 원 : 고려대학교 전산과학과 석사과정
††† 정 회 원 : 고려대학교 전산과학과 박사과정
†††† 정 회 원 : 고려대학교 전산과학과 교수

논문접수 : 1994년 9월 8일, 심사완료 : 1995년 6월 16일.

은 성능을 보이지 못한다[3, 4, 5, 6, 7]. 실제로 대용량 디스크를 사용하는 대표적인 응용 분야인 OLTP(Online-Transaction-Processing) 분야의 경우 대부분의 작업이 작은 읽기/쓰기임을 고려한다면 작은 쓰기 문제는 RAID 5의 활용에 결정적인 단점이 될 수 있다.

패리티 로깅(parity logging)기법[8]은 RAID 5에서 작은 쓰기 문제를 해결하기 위한 여러 방안 중 하나이다. 패리티 로깅 기법은 디스크에서 쓰기 작업이 일어날 때 패리티 유지를 위해서 작은 크기로 빈번히 디스크를 접근하지 않고 이들을 버퍼에 누적시킨 후 큰 크기로 순차적 저장하는 방법을 이용하여 디스크 처리 시간을 줄인다. 본 논문에서는 패리티 로깅 기법 사용시 디스크 버퍼에서 중복하여 존재하는 패리티 이미지를 제거하여 디스크 접근 횟수를 줄이고 패리티 갱신 오버헤드를 줄이는 방안을 제시한다. 본 논문의 2장에서는 RAID 5에서의 작은 쓰기 문제와 기존에 제안된 패리티 로깅 기법 등 기존 관련 연구 내용을 설명한다. 3장에서는 기존 패리티 로깅 기법의 문제점과 중복 패리티 이미지 제거를 고려한 새로운 패리티 로깅 기법을 설명하고 분석 모델을 제시한다. 4장에서는 제안된 새로운 패리티 로깅 기법에 대한 시뮬레이션 결과를 제시하고 5장에서는 결론 및 향후 연구를 제시한다.

2. 기존 연구

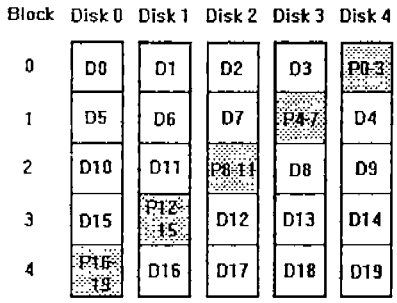
본 장에서는 RAID 5에서의 작은 데이타 쓰기 문제에 대해 설명하고 이에 대한 대표적인 방안인 디스크 캐쉬, 동적 패리티와 데이타, 패리티 로깅 기법 등에 대해 설명한다. 특히 본 논문에서 개선하고자 하는 패리티 로깅 기법의 분석 모델에 대해 자세히 설명한다.

2.1 RAID 5의 작은 데이타 쓰기 문제

(그림 1)은 전형적인 RAID 5 구성을 나타낸다. 디스크에서 새로운 데이타 쓰기에 대한 패리티 유지 방법은 이전 데이타(D_{old})와 새로운 데이타(D_{new})와 이전 패리티(P_{old})를 XOR한 값이다. 새로운 패리티(P_{new})를 유지하는 식은 다음과 같다[1].

$$P_{new} = D_{old} \text{ XOR } D_{new} \text{ XOR } P_{old}$$

이와 같은 작업을 수행하기 위해서는 디스크에서 작은 쓰기 때마다 새로운 패리티를 유지하기 위해 이전 데이타와 패리티를 읽기 위한 2번의 디스크 읽기 접근과 새로운 데이타와 패리티를 저장하기 위한 2번의 디스크 쓰기 접근 등 총 4번의 디스크 접근이 필요하다는 부담이 존재하며 이를 작은 데이타 쓰기 문제(small write problem)라 한다[2, 3, 8, 9].



(그림 1) 전형적인 RAID 5 구조
(Fig. 1) Typical RAID 5 structure

2.2 작은 데이타 쓰기 문제를 줄이기 위한 기존 연구

2.2.1 디스크 캐쉬(Disk cache)

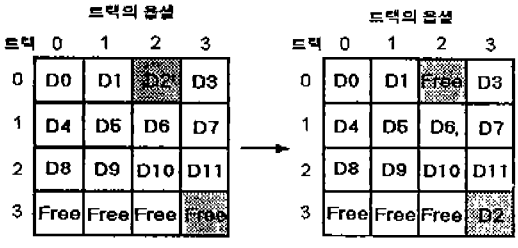
디스크 캐쉬 기법[10]은 RAID 5에서 작은 데이타 쓰기 문제에 대해 여러 측면에서 이득을 얻을 수 있다. 먼저 현재 쓰기에 대한 데이타나 패리티가 디스크 캐쉬에 존재할 경우 읽기 접근 횟수를 줄일 수 있다. 또한, 쓰기 작업을 지연시키는 쓰기 버퍼의 역할을 수행하여 빠른 응답이 가능하도록 하는 등 여러 장점이 있다[10, 11]. 또한 디스크에 저장할 때 시스템 특성에 맞는 효율이 좋은 스케줄링 기법을 사용하여 디스크의 탐색 시간(seek time)을 줄여 빠른 응답 시간과 시스템의 성능 향상을 가져올 수 있다[12].

2.2.2 동적 패리티와 데이타(Floating parity and data)

동적 패리티와 데이타 디스크 배열 기법[13]

은 하나의 실린더마다 빈 공간의 트랙을 유지하여 디스크에서 임의의 블럭을 갱신할 때 새로 쓰여질 데이타를 저장하는데 사용한다. 디스크 컨트롤러 같은 실린더에서 이전 데이타가 저장된 블럭에서 위치적으로 가장 가까운 블럭 즉 회전 지연 시간(rotational delay time)이 가장 적은 빈 공간의 블럭을 찾아 그곳에 새로운 데이타를 저장하고, 메모리에 유지되고 있는 매핑 테이블(mapping table)을 갱신한다. 단, 매핑 테이블을 유지해야 하는 어려움과 별도의 메모리 비용이 드는 단점을 가진다.

(그림 2)는 동적 패리티와 데이타 기법을 설명하기 위한 그림이다. 블럭 D2에 저장된 이전 데이타를 갱신할 때 컨트롤러 실린더에서 블럭 D2에 가장 가까운 빈 공간의 블럭을 찾는다. 이 경우 트랙 0에서 옵셋 2에 해당하는 데이타 블럭 D2에 대해서 트랙 3에서 옵셋 3이 가장 가까운 회전 지연 시간을 갖는 빈 공간이라할 때, 이 자리에 새로운 데이타를 저장하고 이전 데이타가 저장된 D2 블럭을 빈 공간으로 만들며 매핑 테이블을 갱신한다. 이 기법은 RAID 5에서 디스크를 접근하여 데이타를 읽고 수정하고 기록하는 과정에서 불필요한 디스크 회전에 따른 지연 시간을 줄일 수 있다.



(그림 2) 동적 패리티와 데이타 기법에서 데이타 쓰기 방법
(Fig. 2) Data write method in floating parity and data

2.2.3 패리티 로깅(Parity logging) 기법

패리티 로깅 기법[8]은 OLTP작업에서 RAID 5의 성능 개선을 주목적으로 제안된 기법으로 트랙 단위 크기를 가진 버퍼와 로그(log) 디스크를 추가로 사용한다. 전형적인 RAID 5는 작은 크기의 데이타 쓰기 작업에서 패리티 계산을 위하여 여러 번 디스크를 접근하기 때문에 많은 찾기 시간과 회전 지연 시간이 소비된다. 이에 반해 패리티 로깅 기법이 적용될 경우 디스크의 임의 위치에 대한 작은 크기의 데이타 쓰기 작업들을 버퍼에 누적시켜 두고, 후에 이를 큰 크기의 데이타로서 디스크에 순차적으로 접근하도록 바꾸어 디스크 사용 시간을 줄이려는 방법이다. 단, 쓰기 작업이 지연된 패리티를 저장해 두기 위한 버퍼와 로그 디스크의 추가 비용이 드는 단점을 가진다. 2.3절에서 패리티 로깅 기법에 대한 내용을 자세히 설명한다.

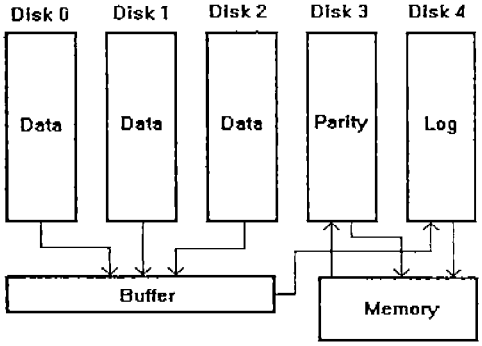
2.3 기존의 패리티 로깅 기법

2.3.1 패리티 처리 방법

패리티 로깅 기법은 RAID 5에서 작은 크기의 데이타 쓰기 작업에 대한 비용을 줄이기 위한 기법이다. (그림 3)은 패리티 로깅 기법에서 패리티를 계산하는 과정을 설명하기 위한 그림으로 설명의 편의상 RAID 5가 아닌 RAID 4상에서의 패리티 유지 방법을 나타내고 있다. 패리티 로깅 기법에서 패리티를 유지하는 방법은 디스크에 작은 크기의 데이타 쓰기를 수행하기 전에 디스크에서 이전 데이타를 읽어 오고 그 자리에 데이타 갱신을 위한 새로운 데이타를 저장한다. 그리고 읽어온 이전 데이타와 새로 저장된 데이타를 XOR 시킨 후 그 결과 값인 패리티 변환 이미지를 버퍼에 기록한다. 버퍼가 다 차면 패리티 변환 이미지들을 로그 디스크에 기록한다. 단, 로그 디스크라는 이름에서 알 수 있듯이 새로 저장되는 변환 이미지는 이전에 기록한 바로 다음 위치부터 순차적으로 기록된다.

쓰기 작업이 반복되어 로그 디스크가 다 채워지거나 또는 디스크 처리 시간에 여유가 있을 때 등 필요한 시점마다 패리티 갱신 작업을 수행한다. 먼저, 패리티 디스크에 있는 이전 패리티 값을 메모리로 읽어들인다. 그 후 로그 디스크에 있는 패리티 변환 이미지들을 순차적으로 읽어들이면서 각각에 해당되는 패리티 값과 XOR 연산을 수행한다. 로그 디스크에 기록된 내용을 모두 처리한 후 최종 결과를 메모리에 저장된 새로 계산된 패리티 값들을 패리티 디스크에 저장한다.

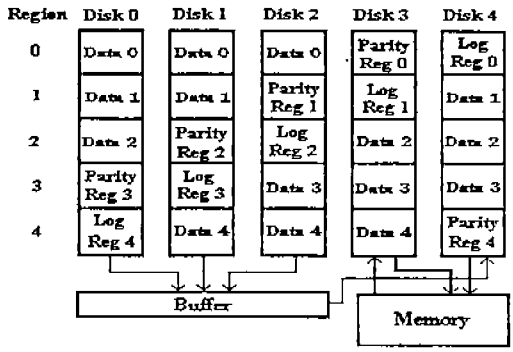
만약 데이타 디스크에 오류가 발생한다면 로그와 패리티 디스크로 잃어버린 데이타를 재구성하며 로그 또는 패리티 디스크에 오류가 발생한다면 해당 데이타들로부터 로그나 패리티 값을 재구성할 수 있다.



(그림 3) RAID 4에서의 패리티 로깅 기법
(Fig. 3) Parity logging scheme in RAID 4

2.3.2 패리티 로깅 기법 모델 분석

기존 패리티 로깅 기법을 분석하기 위한 모델로 (그림 4)와 같은 RAID 5에서의 패리티 로깅 기법을 가정한다. 패리티 갱신을 위해 모든 패리티를 메모리로 읽어들이는 것은 용량 문제로 인하여 현실적으로 불가능하기 때문에 디스크를 몇 개의 지역(region)으로 나누고 지역 단위별로 패리티와 로그를 처리하도록 한다. 이를 위해 각 지역마다 별도의 버퍼를 가지고 있으며 특정 지역에 해당하는 버퍼가 다 차면 지역 단위로 패리티 계산 과정을 수행한다. (그림 4)에서 디스크, 버퍼, 메모리간에 나타난 선들은 지역 0에 대한 데이타 쓰기 작업의 패리티 계산 과정 진행을 보여준다.



(그림 4) RAID 5에서의 패리티 로깅 기법
(Fig. 4) Parity logging scheme in RAID 5

본 절에서 설명할 패리티 로깅 기법의 분석 모델은 작은 크기의 데이타 쓰기 작업에 대해서 패리티를 유지하는데 필요한 디스크 처리 시간에 대한 모델이다. 디스크 처리 시간은 하나의 쓰기 작업 수행을 위해 사용되는 모든 디스크들의 사용 시간 합을 의미하며 응답 시간과는 다르다. 분석 모델에서 사용되는 디스크 처리 시간은 디스크 배열을 구성하고 있는 각 디스크에서 소비되는 처리 시간의 합으로 표현된다. 모델에서 사용되는 매개 변수는 〈표 1〉과 같다.

〈표 1〉 분석 모델에서 사용되는 매개 변수
〈Table 1〉 Parameters used in analysis model

S	평균 찾기 시간
R	평균 회전 지연 시간 (1/2 디스크 회전 시간)
H	헤드 변경 시간
M	단일 트랙 찾기 시간
T	실린더당 트랙 수
D	트랙당 데이타 블럭 수
B	한 블럭당 섹터 수
K	지역당 버퍼 크기의 트랙 수
Cd	지역당 데이타의 실린더 수
Cr	지역당 로그의 실린더 수
Cp	지역당 패리티의 실린더 수

기존의 패리티 로깅 기법에서 패리티를 유지하는데 걸리는 디스크 처리 시간은 패리티 계산을 수행하는 각 과정의 디스크 처리 시간의 합으로 나타낸다. 패리티를 계산하는 과정에서 먼저 이전 데이타를 읽고 새로운 데이타를 기록하는데 걸리는 평균적인 디스크 처리 시간($t_{read/write}$)은 다음과 같다.

$$t_{read\ write}=(S+R)+2RB/D+(2R-2RB/D)+2RB/D$$

(S+R)은 처음 디스크에 저장된 데이타를 읽기 위해서 헤드가 위치하는데 걸리는 평균적인 찾기 시간과 회전 지연 시간이다. 두 번째 항의 2RB/D는 데이타 갱신을 위해서 디스크에 저장된 이전 데이타 한 블럭을 읽는데 걸리는 시간이다. (2R−2RB/D)은 이전 데이타가 저장된 위

치에 새로운 데이타를 저장하기 위하여 헤드가 기다리는 회전 지연 시간이다. 마지막 항의 2RB/D는 새로운 데이타를 기록하는데 걸리는 시간이다. 본 모델에서 데이타를 읽는 시간과 쓰는 시간은 같다고 가정하였다. 위 식을 간단히 정리하면 (식 1)과 같다.

$$t_{read/write}=S+(3+2B/D)R \quad \cdots\cdots\cdots \text{(식 1)}$$

지역별로 버퍼에 저장된 패리티 이미지를 로그 디스크에 저장하는데 걸리는 디스크 처리 시간($t_{log-store}$)은 다음과 같이 표현된다. 각 지역마다 유지하는 버퍼는 K 트랙의 크기이며 로그 디스크를 접근할 때마다 KD개 블럭의 버퍼 이미지를 로그 디스크에 저장한다. 버퍼의 크기인 K 트랙은 같은 실린더에 저장된다고 가정한다.

$$t_{log-store}=(S+R)+2RK+(K-1)H$$

(S+R)은 패리티 이미지를 저장하기 위하여 로그 디스크의 적당한 위치에 헤드를 위치시키는데 소비되는 평균적인 찾기 시간과 회전 지연 시간이다. 2RK는 버퍼가 패리티 이미지로 채워졌을 때 K 트랙 크기의 이미지를 로그 디스크에 저장하데 걸리는 시간이다. (K−1)H는 K 트랙을 저장하는데 걸리는 헤드 변경 시간(head switch time)이다. 헤드 변경 시간이 생기는 이유는 디스크에서 헤드가 동기적으로 움직이더라도 저장될 데이타의 빈 공간이 빈번한 쓰기 작업으로 인하여 임의 위치에 분포되기 때문이다. 따라서 한 번에 여러 헤드는 동시에 움직이지만 쓰기 작업을 동시에 하지 않고 한 헤드의 작업이 끝나면 다른 헤드의 작업을 시작한다. 위의 식을 간단히 정리하면 (식 2)와 같다.

$$t_{log-store}=S+(2K+1)R+(K-1)H \cdots \text{(식 2)}$$

패리티를 계산하는 과정은 로그 디스크와 패리티 디스크의 데이타를 읽어서 두 값을 XOR한 후 이 결과 값을 패리티 디스크에 저장하는 것이다. 패리티 계산을 위해서 로그 디스크 읽기와 패리티 디스크 읽기와 쓰기 작업이 이루어진다. 로그 디스크는 한 지역에 Cr 크기의 실린더를 가지며 로그 디스크에서 로그 데이타를 읽는데 걸리는 평균적인 디스크 처리 시간($t_{log-read}$)은 다음과 같다.

$$t_{log-read}=(S+R)+(2RT+(T-1)H)Cr+(Cr-1)M$$

(S+R)은 로그 데이타를 읽기 위해서 로그 디스크에서 읽을 데이타의 위치에 헤드를 위치시키는데 걸리는 평균적인 찾기 시간과 회전 지연 시간이다. (2RT+(T−1)H)Cr은 로그 디스크에서 한 지역 크기의 로그 데이타를 읽는데 걸리는 시간이다. Cr은 여러 개의 트랙으로 이루어져 있고 (2RT+(T−1)H)는 이중에서 한 트랙의 로그 데이타를 읽는데 걸리는 시간이다. (Cr−1)M은 Cr 크기의 실린더를 읽기 위해 소비되는 찾기 지연 시간이다. 따라서 한 지역의 로그 데이타를 읽는데 걸리는 디스크 처리 시간은 (식 3)과 같다.

$$t_{log-read}=S+(2TCr+1)R+(T-1)HCr+(Cr-1)M \quad \cdots\cdots\cdots\cdots \text{(식 3)}$$

다음은 패리티 디스크에서 패리티를 읽거나 쓰는데 걸리는 디스크 처리 시간을 알기 위한 과정이다. 패리티 디스크는 한 지역에서 Cp 크기의 실린더를 가지며 패리티 디스크에서 패리티를 읽는데 걸리는 평균적인 디스크 처리 시간(tparity_read/write)은 다음과 같다. 패리티 디스크에서 읽기와 쓰기 비용은 같은 값이다.

$$t_{parity-read/write}=(S+R)+(2RT+(T-1)H)Cp+(Cp-1)M$$

(2RT+(T−1)H)Cp는 패리티 디스크 한 지역에 있는 패리티를 읽는데 걸리는 시간이다. 이를 정리하면 패리티 디스크 한 지역에 있는 패리티를 읽는데 걸리는 평균적인 디스크 처리 시간은 (식 4)와 같다.

$$t_{parity-read/write}=S+(2TCp+1)R+(T-1)HCp+(Cp-1)M \quad \cdots\cdots\cdots\cdots \text{(식 4)}$$

결국, 기존의 패리티 로깅 기법에서 한 블럭의 데이타 쓰기 작업을 수행하는데 소요되는 평균적인 디스크 처리 시간(t_{ASWDBT}(Average Small Write Disk Busy Time))은 버퍼의 접근 시간 tbuffer을 포함하여 (식 5)와 같이 나타낸다.

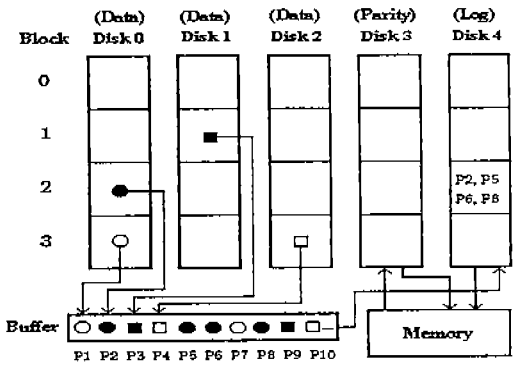
$$t_{ASWDBT}=t_{read/write}+(1/KD)t_{log-store}+(1/DTCd)[t_{log-read}+2*t_{parity-read/write}]+t_{buffer} \quad \cdots\cdots\cdots\cdots \text{(식 5)}$$

3. 중복 이미지 제거를 적용한 패리티 로깅 기법

본 장에서는 기존 패리티 로깅 기법에서 중복된 패리티 이미지가 존재하며, 이를 제거하여 성능을 향상시키고자 하는 방안에 대해 설명하고 본 기법의 디스크 처리 시간에 대한 분석 내용을 제시한다.

3.1 기존 패리티 로깅 기법의 문제점

기존 패리티 로깅 기법에서 디스크의 같은 장소에 작은 크기의 데이타 쓰기 작업이 여러 번 일어날 경우 버퍼에는 쓰기가 일어난 순서대로 디스크의 저장 장소와 패리티 이미지가 기록되며, 후에 버퍼가 가득 찼을 때 로그 디스크에 기록된다. 이 때 디스크상의 같은 위치에 대한 패리티 이미지가 중복하여 존재할 수 있게 된다. (그림 5)는 기존의 패리티 로깅 기법에서 패리티 이미지의 중복 저장으로 생기는 문제점을 설명하기 위하여 (그림 4)에서 하나의 지역 부분만을 보인 그림이다. 하나의 지역은 여러 개의 블럭으로 구성되며 설명의 편의상 쓰기는 한 블럭 단위로 발생한다고 가정한다. 예를 들어, 디스크 0, 블럭 2에서 쓰기 작업이 여러 번 일어날 때, 이전 데이타와 저장할 새로운 데이타를 XOR 한 패리티 이미지들을 버퍼에 저장한다. 버퍼에 저장된 패리티 이미지 P2, P5, P6, P8은 로그 디스크로 보내지고, 디스크에 저장된 로그 데이타는 패리티 유지를 위해서 사용된다. 데이타 디스크에는 P8과 관련된 최근 데이타를 유지하는 반면, 버퍼에는 이전 패리티 이미지인 P2, P5, P6, P8가 모두 존재한다. 이들 중 P8을 제외한 나머지들은 모두 불필요한 이미지들이다. 이들처럼 불필요한 패리티 이미지들로 인해 버퍼가 차게 되는 시간 간격이 줄어들게 되며 이로 인해 로그 디스크로의 접근과 패리티 이미지 갱신 작업이 자주 발생하게 된다.



(그림 5) 패리티 로깅 기법에서 패리티 유지 방법
(Fig. 5) Parity management method in parity logging scheme

3.2 중복 패리티 이미지를 제거를 통한 패리티 로깅 기법 성능 개선

(그림 5)는 패리티 로깅 기법에서 중복 패리티 이미지를 제거하는 새로운 패리티 로깅 기법을 설명하기 위한 그림이다. (그림 5)에서 P2, P5, P6, P8들은 동일한 디스크 위치에 대한 쓰기 작업에 대한 패리티라 가정할 때 중복 패리티 이미지 제거란 최종적으로 유효한 값인 P8을 제외한 나머지 불필요한 정보인 P2, P5, P6을 제거하는 것이다. 즉 새로운 패리티 로깅 기법에서 버퍼에는 디스크의 같은 장소에서 일어나는 데이타 쓰기 작업 기록을 일어난 순서대로 저장하되 중복된 위치에 대한 쓰기가 발생할 경우 로그 디스크에 저장하기 전의 최종 쓰기에 대한 정보만 유지한다. 새로운 데이타에 대한 패리티 이미지를 버퍼에 기록하는 방법은 새로 저장한 데이타의 패리티 이미지와 버퍼에 이미 저장된 이미지를 XOR 한 후 이 값을 버퍼에 저장하는 것이다. 처음 버퍼에 P2가 기록되고, 디스크 0, 블럭 2에서 또 다른 작은 크기의 데이타 쓰기 작업이 수행될 때, 새로운 데이타에 대해서 얻어진 P5는 P2를 기록한 위치에 P2와 P5를 XOR한 이미지를 저장한다. 이러한 이미지 갱신 과정을 반복하여 P6와 P8의 이미지도 버퍼의 같은 위치에 저장된다. 새로운 패리티 로깅 기법은 기존의 패리티 로깅 기법과 같은 크기의 버퍼를 사용하지만 디스크의 같은 장소에서 일어나는 작업에 대하여 버퍼는 같은 위치에 이 데이타들을 저장하기 때문에 실질적으로 더 많은 양의 데이타를 저장하여 로그 디스크로 전송하도록 하며 결과적으로 로그 디스크를 접근하는 횟수를 줄일 수 있다. 또한 처리할 전체 작업에서 패리티 계산을 위한 데이타들이 줄어 들었기 때문에 패리티를 계산

하는데 걸리는 디스크 처리 시간을 줄일 수 있다.

3.3 새로운 패리티 로깅 기법의 분석

새로운 패리티 로깅 기법은 디스크의 같은 장소에서 작은 크기의 데이타 쓰기 작업이 중복되어 일어날 경우 버퍼에 저장되는 중복된 패리티 이미지를 제거하는 방법이다. 따라서 새로운 패리티 로깅 기법의 효율은 동일한 위치에 대한 쓰기 작업의 중복율과 중복된 패리티 이미지를 찾는 버퍼 탐색 시간이 중요한 변수로 작용하기 때문에 새로운 패리티 로깅 기법에 대한 분석을 위해 〈표 1〉에서 기술된 매개 변수들 외에 추가로 〈표 2〉의 매개 변수들을 사용하였다.

〈표 2〉 새로운 패리티 로깅 기법에서 사용하는 매개 변수
〈Table 2〉 Parameters used in new parity logging scheme

$T_{block-number}$	전체 작업(작업 부하)에 대한 데이타 디스크 접근 총 블럭 수
$B_{block-number}$	버퍼가 찰 때 버퍼에 저장된 블럭 수
$LD_{block-number}$	로그 디스크의 한 지역에 데이타가 찰 때 저장된 모든 블럭 수
$PD_{block-number}$	패리티 디스크의 한 지역에 데이타가 찰 때 저장된 모든 블럭 수
β	버퍼의 중복 저장 비율
Nb	로그 디스크 접근 횟수 (버퍼 데이타를 저장시) ($T_{block-number}$ / $B_{block-number}$)
Nr	로그 디스크 접근 횟수 (로그 데이타를 읽을 때) ($T_{block-number}$ / $LD_{block-number}$)
Np	패리티 디스크 접근 횟수 ($T_{block-number}$ / $PD_{block-number}$)

새로운 패리티 로깅 기법에서 패리티를 계산하는 과정은 기존의 패리티 로깅 기법의 패리티 계산 과정과 같다. 제안된 새로운 패리티 로깅 기법에서 데이타를 읽고 쓸 때 걸리는 디스크 처리 시간 $t'_{read/write}$은 $t_{read/write}$와 같다. 이는 데이타의 읽기와 쓰기 작업은 버퍼의 중복 저장과 무관하게 이루어지기 때문이다. 제안된 새로운 패리티 로깅 기법에서 작은 크기의 데이타 쓰기 작업에 대한 평균적인 디스크 처리 시간($t'_{read/write}$)은 (식 6)과 같다.

$$t'_{read/write}=t_{read/write}=S+(3+2B/D)R \quad \cdots \text{(식 6)}$$

버퍼는 K개의 트랙을 가지며 버퍼에서 중복 저장을 제거한 데이타를 로그 디스크에 저장하는데 걸리는 평균적인 디스크 처리 시간($t'_{log-store}$)은 다음과 같다.

$$t'_{log-store}=((1-\beta)Nb)/Nb*(S+(2K+1)R+(K-1)H)$$

$((1-\beta)Nb)/Nb$는 로그 디스크 접근 비율이다. 이는 버퍼에서 로그 디스크로 패리티 이미지를 저장할 때 버퍼의 중복 저장을 제거하여 줄어든 로그 디스크 접근 횟수를 버퍼의 중복 저장을 제거하지 않은 디스크 접근 횟수로 나눈 값이다. 이를 기존의 패리티 로깅 기법에서의 (식 2)와 관련하여 표현하면 버퍼 내용을 로그 디스크에 저장하는데 걸리는 평균적인 디스크 처리 시간은 (식 7)과 같다.

$$t'_{log-store}=(1-\beta)*t_{log-store} \quad \cdots\cdots\cdots \text{(식 7)}$$

(식 2)와 (식 7)을 비교하면 $t'_{log-store}$은 버퍼의 중복 저장율에 비례하여 $t_{log-store}$보다 시간이 줄어든다. 이는 (식 7)에서 처리할 전체 작업의 데이타 크기 대해서 디스크의 처리 시간에 포함되어 있는 S, R, H 시간이 에 비례하여 줄어들기 때문이다.

다음은 패리티 계산을 위해서 버퍼의 중복 저장을 제거하여 로그 디스크에서 메모리로 데이타를 읽어오는데 걸리는 평균적인 디스크 처리 시간(t'_{log_read})이다.

$$t'_{log-read}=((1-\beta)Nr)/Nr*((S+R)+(2RT+(T-1)H)Cr+(Cr-1)M)$$

$((1-\beta)Nr)/Nr$은 전체 처리 작업에 대해서 버퍼의 중복 저장을 제거하여 줄어든 로그 데이타를 읽기 위한 디스크 접근 횟수를 버퍼의 중복 저장을 고려하지 않은 디스크 접근 횟수로 나눈 값이다. 이를 (식 3)과 관련지으면 평균적인 디스크 처리 시간은 (식 8)과 같이 정리된다.

$$t'_{log-read}=(1-\beta)*t_{log-read} \cdots\cdots\cdots\cdots \text{(식 8)}$$

(식 3)과 (식 8)을 비교하면 $t'_{log-read}$은 $t_{log-read}$보다 버퍼의 중복 저장율에 비례하여 시간이 줄어든다. 이는 처리할 전체 작업의 데이타 크기에 대해서 디스크 처리 시간에 포함되어 있는 S, R, H, M 시간에 비례하여 줄어들기 때문이다.

패리티를 계산하는 과정에서 제안된 패리티 로깅 기법은 패리티 디스크의 읽기와 쓰기를 수행한다. 패리티 디스크에서 패리티를 읽거나 쓰는데 걸리는 디스크 처리 시간($t'_{parity-read/write}$)은 다음과 같다.

$$t'_{parity-read/write}=((1-\beta)Np)/Np*((S+R)+(2RT+(T-1)H)Cp+(Cp-1)M)$$

$((1-\beta)Np)/Np$은 전체 처리 작업에 대해서 버퍼의 중복 저장을 제거하여 줄어든 패리티 데이타를 읽기 위한 디스크 접근 횟수를 버퍼의 중복 저장을 고려하지 않은 디스크 접근 횟수로 나눈 값이다. 패리티 디스크에서 패리티를 읽거나 쓰는데 걸리는 평균적인 디스크 처리 시간을 (식 4)를 대입하여 정리하면 (식 9)와 같다.

$$t'_{parity-read/write}=(1-\beta)*t_{parity-read/write} \quad \text{(식 9)}$$

(식 4)와 (식 9)를 비교하면 $t'_{parity-read/write}$은 $t_{parity-read/write}$보다 시간이 줄어든다. 이는 전체 데이타 크기에 대해서 저장되는 패리티 데이타 수가 줄어들어 패리티 디스크를 읽는 횟수가 줄어들기 때문이다. 패리티 데이타를 읽거나 쓰는 데 걸리는 디스크 처리 시간에서 S, R, H, M 시간에 비례하여 줄어들기 때문에 패리티 갱신에 소비되는 평균적인 디스크 처리 시간이 줄어든다.

새로운 패리티 로깅 기법에서 한 블럭의 작은 크기의 데이타 쓰기 작업에서 패리티를 유지하는데 소비되는 평균적인 디스크 처리 시간(t'_{ASWDBT})은 버퍼의 접근 시간인 t_{buffer}와 버퍼의 탐색 시간인 t_{search}을 포함하여 (식 10)과 같이 나타낸다.

$$t'_{ASWDBT}=t'_{read/write}+(1/KD)t'_{log-store}+(1/DTCd)[t'_{log-read}+2*t'_{parity-read/write}]+t_{buffer}+t_{search} \quad \text{(식 10)}$$

결론적으로 (그림 4)의 모델에서 (식 5)와 (식 10)을 비교하면 중복 저장을 제거한 새로운 패리티 로깅 기법이 기존의 패리티 로깅 기법에서 패리티를 갱신하고 유지하는데 소비되는 오버헤드를 (식 11) 만큼 줄일 수 있다.

$$t_{ASWDBT}-t'_{ASWDBT}=(\beta/KD)*[t_{log-store}]+(\beta/DTCd)*[t_{log-read}+2*t_{parity-read/write}]-t_{search} \quad \text{(식 11)}$$

(식 11)에서 설명된 새로운 패리티 로깅 기법의 이득은 버퍼의 중복 저장율이 커질수록 증가하고 버퍼에서 데이타를 탐색하는 시간인 tsearch가 커질 수록 감소한다. 비록 버퍼에 중복 저장되는 데이타의 비율이 적다하더라도 새로운 패리티 로깅 기법을 이용하여 얻어진 패리티 처리 작업에 대한 이득이 버퍼에 저장된 이미지를 탐색하는데 소요되는 tsearch보다 크다면 본 논문에서 제안하는 새로운 패리티 로깅 기법은 기존의 기법보다 좋은 성능을 나타낼 수 있다.

4. 시뮬레이션 및 결과

본 장에서는 기존의 패리티 로깅 기법[8]과 본 논문에서 제안하는 새로운 패리티 로깅 기법의 성능을 비교하기 위해 수행한 시뮬레이션과 결과 분석 내용을 설명한다.

4.1 시뮬레이션 환경

기존의 패리티 로깅 기법과 본 논문에서 제안하는 새로운 패리티 로깅 기법의 성능을 비교하기 위해 시뮬레이션 언어인 SLAM을 이용하여 시뮬레이터를 구현하였다. 시뮬레이션에 적용된 기본적인 디

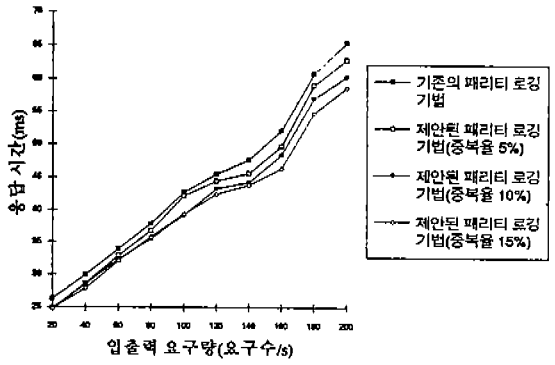
〈표 3〉 시뮬레이션에 사용된 매개 변수
〈Table 3〉 Parameters used in simulation

디스크 특성	1944실린더, 20헤드, 88섹터/트랙, 512바이트/섹터
찾기 시간 모델(S)	평균 11.0 ms
단일 트랙 찾기 시간(M)	2.0 ms
회전 지연 시간(R)	11.1 ms
배열의 디스크 수(N)	5 개
블럭 크기(B)	2KB(4 섹터)
버퍼의 크기(K)	400KB
실린더당 트랙 수(T)	9 트랙
지역당 실린더 수 (Cd=Cr=Cp)	23 개

스크 관련 매개 변수들은 주로 Fujitsu M2652H/S 5.25" 1.8 GB SCSI 디스크 드라이브의 사양을 참조하였다(〈표 3〉). OLTP 작업 환경을 고려하여 시뮬레이션에서의 모든 읽기, 쓰기 작업시 디스크에 접근하는 크기는 한 블럭으로 가정하였으며 디스크 캐쉬와 트랙 버퍼의 사용은 고려하지 않았다. 디스크 당 지역은 250개를 유지하며 디스크당 한 지역의 크기는 8MB이다.

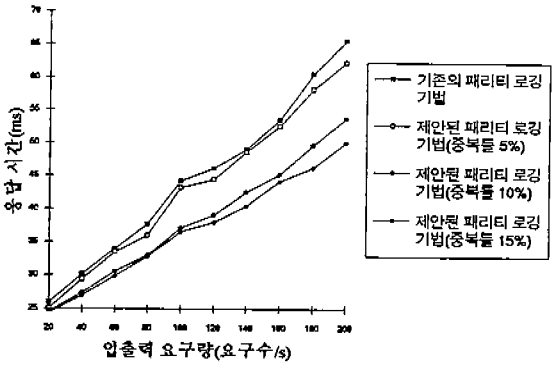
4.2 시뮬레이션 결과

(그림 6)은 버퍼의 중복율을 각각 5%, 10%, 15%로 가정하고 입출력 요구량(초당 I/O 요청수)을 20~200으로 변화시켜가면서 응답 시간의 변화를 비교한 그림이다. 단, 응답 시간에는 쓰기 요청들뿐 아니라 읽기 요청들까지 포함되어 있으며 읽기 요청대 쓰기 요청의 비율은 80:20이다. 전반적으로 I/O 비율이 커질수록 기존의 패리티 로깅 기법과 새로운 패리티 로깅 기법의 성능차가 조금씩 커진다. 또한 중복율이 증가될 수록 새로운 패리티 로깅 기법의 이득이 향상됨을 확인할 수 있다.

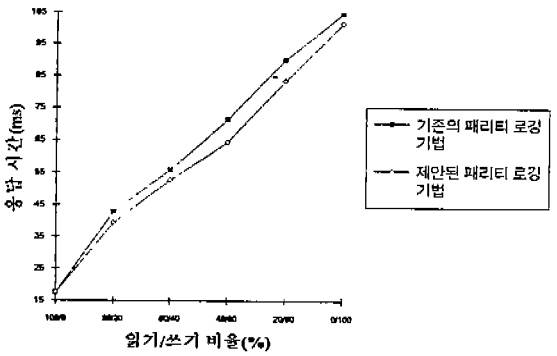


(그림 6) 입출력 요구량의 변화에 따른 응답 시간
(Fig. 6) Response time with various I/O request rate

(그림 7)은 읽기 요청과 쓰기 요청이 80:20의 비율로 혼합된 입출력 요청중에서 쓰기 요청에 대한 응답 시간을 입출력 요구량의 변화에 따른 차이를 보인 것이다. (그림 6)에서 보인 응답 시간의 변화보다 큰 변화의 폭을 보인다. 입출력 요구량이 커질수록 기존의 패리티 로깅 기법과 새로운 패리티 로깅 기법의 성능차가 (그림 6)에서보다 급격하게 커진다. 즉, 쓰기 요청의 경우 새로운 패리티 로깅 기법 사용의 이득이 커짐을 볼 수 있다.



(그림 7) 입출력 요구량의 변화에 따른 쓰기 요청 응답 시간
(Fig. 7) Write request response time with various I/O request rate



(그림 8) 읽기/쓰기 비율 변화에 따른 응답 시간
(Fig. 8) Response time with various read vs. write rate

(그림 8)은 읽기 요청대 쓰기 요청의 비율의 변화에 따른 디스크 응답 시간의 변화를 비교한 그림이다. 제안된 패리티 로깅 기법의 중복율은 10%로, 입출력 요구량은 초당 100개로 설정하고 쓰기 요청의 비율을 0%부터 20%씩 증가시켜가면서 측정하였다. RAID 5의 특성상 쓰기 작업의 비율이 많을 수록 응답 시간이 급격히 증가한다. 쓰기 작업의 비율이 높은 작업에서 새로운 패리티 로깅 기법이 좋은 성능을 보이지만 쓰기 작업의 수가 너무 많아지면 다시 이득이 줄어든다. 이는 쓰기 작업 수행시 기존의 데이타를 다시 읽는 작업이 수행되

므로 쓰기 작업의 양이 많아질수록 부가적인 읽기 작업 수가 증가되어 중복 패리티 이미지 제거를 통한 쓰기 작업에서의 이득이 조금씩 상쇄되어 발생하는 현상이다.

5. 결론 및 향후 연구

본 논문에서는 기존의 패리티 로깅 기법에서 디스크 버퍼의 중복 저장된 패리티 이미지를 제거하여 디스크 접근 횟수를 줄여 쓰기 작업에서 디스크 처리 시간을 줄일 수 있는 새로운 패리티 로깅 기법을 제안했다. 새로운 패리티 로깅 기법은 쓰기 작업의 중복 정도와 버퍼상에서 중복된 이미지를 탐색하는 시간에 따라 효율이 좌우된다. 시뮬레이션을 통해 중복율이 수% 이상을 보이는 작업들에 대해서는 일반적으로 새로운 패리티 로깅 기법에서 중복을 제거하여 얻어진 줄어든 디스크 처리 시간의 차이가 버퍼에 저장된 이미지를 탐색하는데 걸리는 시간보다 크다고 볼 수 있기 때문에 새로운 패리티 로깅 기법은 쓰기 작업에 대해 향상된 결과를 얻을 수 있다. 차후 연구로는 버퍼의 크기 변화와 중복된 패리티 이미지의 분포에 대한 관련 등을 고려해야 할 것이다. 현재 본 논문에서의 분석 및 시뮬레이션에서는 패리티 이미지 중복율에 따라 고르게 중복 이미지가 존재할 것으로 가정하였으므로 버퍼의 크기는 중복 정도에 영향을 미치지 않았다. 실제로는 수행되는 작업의 시간적, 공간적 지역성 등이 고려된다고 한다면 버퍼의 크기에 따라 중복율이 달라질 수 있으며 이러한 상황에서 새로운 패리티 로깅 기법의 모델링과 성능 평가가 필요하다. 또한, 버퍼가 커짐에 따라 중복된 이미지를 찾기 위한 탐색 시간이 무시되기 어려운 값이 될 수 있을 것이므로 이에 대해서도 고려되어야 할 것이다.

참 고 문 헌

[1] D. Patterson, G. Gibson, and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks(RAID)," *Proceedings of the ACM SIGMOD Conference*, pp. 109" 116, Jan. 1988.

[2] G. Gibson, "Redundant Disk Arrays: Reliable, Parallel Secondary Storage," *MIT Press*, 1992.

[3] R. Katz, G. Gibson, and D. Patterson, "Disk System Architecture for High Performance Computing," *Proceedings of the IEEE*, vol. 77, no. 12, Dec. 1989.

[4] D. Patterson. G. Gibson, and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks(RAID)," *Proceedings of the ACM SIGMOD Conference*, pp. 109-116, Jan. 1988.

[5] J. Menon and D. Mattson, "Performance of Disk Arrays in Transaction Processing Environments," *Proceedings of the 12th International Conferenmce on Distributed Computing Systems*, pp. 302-309, Jun. 1992.

[6] J. Menon and J. Kasson, "Methods for Improved Update Performance of Disk Arrays," *Proceedings of the Hawaii International Conference on System Sciences*, pp. 74-83, Jan. 1992.

[7] P. M. Chen and P. A. Patterson, "Storage Performance-Metrics and Benchmarks," *Proceedings of the IEEE*, pp. 1151-1165, vol. 81, no. 8, Aug. 1993.

[8] D. Stodolsky, G. Gibson, and M. Holland, "Parity Logging Overcoming the Small Write Problem in Redundant Disk Arrays," *Proceedings of the 20th Annual International Symposium on Computer Architecture*, pp. 64-75, May 1993.

[9] W. Burkhardt and J. Menon, "Disk Array Storage System Reliability," *Proceedings of the 23rd Annual International Symposium on Fault-Tolerant Computing, IEEE Computer Society*, pp. 432-441, Jul. 1993.

[10] J. Menon and J. Cortney, "The Architecture of a Fault-Tolerant Cached RAID Controller," *Computer Architecture News*, vol. 21, no. 2, pp. 76-86, May 1993.

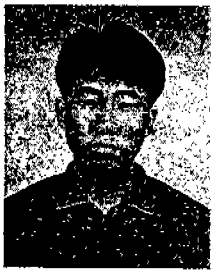
[11] A. Hospodor, "Hit Ratio of Caching Disk Buffers," *Spring Compcon 92*, pp. 427-432, Feb. 1992.

[12] M. Seltzer, P. Chen, and J. Ousterhout, "Disk Scheduling Revisited", *Proceedings of Winter 1990 USENIX Technical Conference*, USENIX Association, pp. 313-324, Jan. 1990.

[13] J. Menon, J. Roche, and J. Kasson, "Floating Parity and Data Disk Arrays," *Journal of Parallel and Distributed Computing*, vol. 17, pp. 129-139, Jan. 1993.

길 준 호

1989년 금오공과대학 전자계산기공학과 졸업(학사)
1995년 고려대학교 대학원 전산과학과(이학석사)
1995년~현재 국방전산소 전산지원과 체계관리담당
관심분야 : 컴퓨터 구조, 병렬처리임.



이 민 영

1994년 고려대학교 전산과학과 졸업(학사)
1994~현재 고려대학교 대학원 전산과학과 석사과정
관심분야 : 컴퓨터 구조, 병렬처리임.

이 진 호

1992년 고려대학교 전산과학과 졸업(학사)
1994년 고려대학교 대학원 전산과학과(이학석사)
1994년~현재 고려대학교 대학원 전산과학과 박사과정.
관심분야 : 컴퓨터 구조, 병렬처리임.

박 명 순

1975년 서울대학교 전자공학과 졸업(학사)
1982년 Utah 대학교 전기공학과(공학석사)
1985년 Iowa 대학교 전기전산기공학과(공학박사)
1975년~80년 국방과학연구소 연구원
1985년~87년 Marquette 대학교 조교수
1987년~88년 포항공과대학 전자전기공학과 조교수
1988년~현재 고려대학교 전산과학과 교수
관심분야 : 컴퓨터 구조, 운영체제, 컴파일러임.