産業技術硏究(江原大學校 産業技術硏究所 論文集) 第 15輯 (1995)
Research Report. The Institute of Industrial Technology, Kangwon Nat'l Univ., Vol. 15, 1995

캐쉬를 사용한 RAID5 상에서 작은 쓰기 문제의 성능 분석

최 황 규 · 서 주 하* · 이 승택**

Performance Analysis of the Small Write Problem on the Cached RAID5

Hwang-Kyu Choi · Ju-Ha Seo* · Seung-Taek Lee**

ABSTRACT

In this paper we evaluate the performance of the cached RAID5 which uses the parity cache and the data cache to overcome the small write problem. From the result of the simulation study we show that the cached RAID5 provides performance improvement with reasonable cache size.

1. 서 론

최근의 반도체 기술의 발달과 이에 따른 VLSI 기술의 급속한 발달에 따라 CPU와 메모리의 성능은 지수 적으로 향상되어 왔다. 마이크로프로세서의 성능 향상은 많은 영역에서 컴퓨터의 활용 영역을 크게 넓혀 놓았다. 특히 비디오(video), 멀티미디어(multimedia) 같은 영상처리 응용 분야에서의 활용과 컴퓨터를 이용한 디자인(design), 대용량의 데이터베이스 시스템(very large database system) 운영에서의 활용은 괄목할 만하다.

이러한 컴퓨터 활용 분야의 확대는 대용량의 데이터를 위한 고속, 대용량의 보조기억장치(secondary storage device)를 필요로 하게 되었다. 그러나 보조기억장치인 디스크는 기계적인 원인으로 인하여 그 성능 향상에 있어 많은 기술적인 어려움을 가지고 있다. 근래의 디스크의 성능 향상은 저장용량의 증가를 위주로 이루어졌으며, 디스크 억세스 속도(disk access time)면에서의 성능은 매년 10%정도로 느리게 향상되었다. 이에 따라 새로운 보조기억 시스템에 대한 연구가 요구되었으며, 기존의 SLED(Single Large Expensive Disks)의 성능 향상 한계를 극복하고자 하는 노력이 계속 되어 왔다[1][2][8]. 그 중에 하나가 디스크 배열(disk array)에 대한 연구이다. 디스크 배열은 용량이 적고 비용이 저렴한 독립적인 다중 디스크의 집합으로 단일의 대용량, 고성능의 논리적인 디스크를 구성한다. 디스크 배열은 보조기억장치 시장에서 급속도로 성장하고 있는데, 이는 SLED에 비하여 충분한 대역폭(bandwidth)과, 상대적으로 낮은 가격에 기인한다.

강원대학교 컴퓨터공학과 조교수
* 강원대학교 컴퓨터공학과 부교수
** 한국전산원 연구원

디스크 배열의 기본 원리는 데이터를 여러 개의 디스크에 분산(stripe)시킴으로써 큰 데이터 억세스일 경우에는 높은 데이터 전송률(data transfer rate)을, 그리고 작은 데이터 억세스일 경우에는 높은 입출력률(I/O rate)을 기대할 수 있다. 그러나 디스크 배열에서도 탐색시간과 회전지연시간은 기존의 디스크 시스템에 비하여 나아지지 않았고 이로 인하여 최악의 경우, 즉 데이터 베이스(database) 처리나 트랜잭션(transaction) 처리와 같이 소량의 입출력 요구가 병렬성 없이 일정 간격으로 요구될 경우는 기존의 디스크 시스템보다 많은 오버해드(overhead)가 발생하고 이로 인하여 더 낮은 성능을 보일 수도 있다. 또한 단순한 디스크의 병렬 구조에서는 신뢰성(reliability)의 문제가 제기되는데 디스크의 개수에 반비례하여 MTTF(Mean-Time-To-Failure)가 감소하게 되는 것이 그 원인이다. 이를 해결하기 위한 방안으로 에러 복구 코드(error-correction code)를 데이터에 추가시키는 새로운 형태의 디스크 배열이 제안되었는데, 이것이 RAID(Redundant Arrays of Inexpensive Disks)이다.

RAID는 에러복구 코드와 데이터의 분산 방법에 따라 다섯 개의 단계가 등장하였으며 이를 더욱 향상시키기 위하여 데이터와 패리티 블록(parity block)의 위치 결정, 분산단위 크기 등의 연구로 전반적인 입출력 성능을 높이기 위한 노력이 계속되고 있다. RAID의 5 레벨 중에서 가장 일반적인 형태는 RAID5로서 데이터의 신뢰성을 높이기 위하여 낮은 가격의 패리티 인코딩(parity encoding)을 사용한다. 데이터를 모든 디스크에 분산시켜 큰 억세스일 때 높은 대역폭으로 인출(fetch)할 수 있게 하였고, 이에 대한 패리티를 분산시킴으로써 특정 디스크의 집중(hot spot)을 제거하였다. RAID5 디스크 배열은 성능과 데이터 신뢰성의 장점으로 인하여 다양한 응용에서 활용될 수 있다.

그러나 RAID5는 대부분이 작은 데이터 쓰기(small data write)로 구성된 부하량(workload)에서는 과중한 디스크 억세스로 인하여 처리량(throughput)이 급격히 감소한다는 단점을 가진다[5]. 작은 데이터 쓰기일 경우 기존의 데이터 읽기, 기존의 패리티 읽기, 새로운 데이터 쓰기, 새로운 패리티 쓰기의 4번의 디스크 억세스가 발생한다. 따라서 작은 쓰기가 많이 발생하는 OLTP(On-Line Transaction Processing) 시스템에서는 성능이 크게 악화된다. 실제 보조기억장치 시장은 작은 데이터 쓰기 문제에 큰 영향을 받는다는 것을 감안하면 해결해야 할 중요한 문제인 것이다.

따라서 본 논문에서는 RAID5의 문제인 작은 쓰기 문제를 해결하기 위하여 비휘발성 메모리를 패리티 캐쉬와 데이터 캐쉬로 사용하여 디스크의 억세스 횟수를 줄임으로써 입출력 요구에 대한 평균응답시간을 향상시켰다. 시뮬레이션에 의한 성능 분석의 결과로 캐쉬를 사용한 RAID5가 작은 쓰기 문제에 대해서 우수한 성능 향상을 이룰 수 있음을 보인다.

2. RAID5의 작은 쓰기 문제

2.1 RAID5 개요

RAID는 에러복구 코드와 데이터의 분산 방법에 따라 다섯 개의 단계가 등장하였으며 RAID의 5 레벨 중에서 가장 일반적인 형태는 그림 1과 같은 RAID5로서 데이터의 신뢰성을 높이기 위하여 낮은 가격의 패리티 인코딩(parity encoding)을 사용한다. 데이터를 모든 디스크에 분산시켜 큰 억세스일 때 높은 대역폭으로 인출(fetch)할 수 있게 하였고, 이에 대한 패리티를 분산시킴으로써 특정 디스크의 집중(hot spot)을 제거하였다. 이로 인하여 RAID5는 높은 전송률과 부하의 균형분배(load balancing)를 제공하여 작은 읽기(small

data read), 큰 읽기(large data read), 큰 쓰기(large data write)일 경우 좋은 성능을

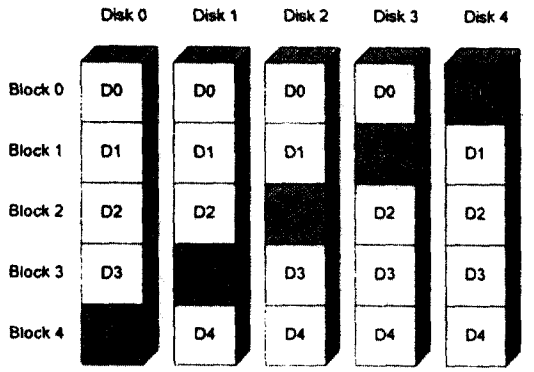

Figure 1. The RAID5.

타나낸다. 그러나 RAID5는 대부분이 작은 데이터 쓰기(data write)로 구성된 부하에서는 과중한 디스크 억세스로 인하여 처리량(throughput)이 급격히 감소한다는 단점을 가진다. 작은 데이터 쓰기일 경우 기존의 데이터 읽기(old data read), 기존의 패리티 읽기(old parity read), 새로운 데이터 쓰기(new data write), 그리고 새로운 패리티 쓰기(new parity write) 등 4번의 디스크 억세스가 발생한다. 따라서 작은 쓰기가 많이 발생하는 OLTP(On-Line Transaction Processing) 시스템에서는 성능이 크게 악화된다. 실제 보조기억장치 시장은 작은 데이터 쓰기 문제에 큰 영향을 받는다는 것을 감안하면 해결해야 할 중요한 문제이다.

2.2 RAID5 성능 분석

디스크 배열에서의 해석적 분석 방법은 다른 억세스가 없다고 가정하고 디스크 시스템의 평균 사용 시의 파라미터를 바탕으로 하여 디스크의 읽기와 쓰기를 수식으로 표현하는 방법이다. 해석하는 과정에는 디스크 에러가 발생하지 않는다고 가정한다. 그리고 패리티의 계산에 소요되는 시간을 영으로 하기 위하여 프로세서와 채널(chennel), 콘트롤러의 속도는 무한하다고 가정한다. 디스크 억세스 요구가 발생할 때 디스크 암은 임의의 위치에 존재한다고 가정하고 디스크 탐색시간을 구한다. 디스크 탐색시간은 실린더가 움직여야 할 거리에 의존하며 다음의 식으로 표현된다.

$$\text{Seek Time} = 2.0 + 0.01 \cdot \text{dist} + 0.46 \cdot \sqrt{dist}$$

RAID5에서 읽을 분산단위의 개수를 S라고 하고 이들이 분산되어 있는 디스크의 개수를 A라 하면 이는 다음식과 같다.

$$S = (request\ size)/(block\ size)$$

$$A = \min(S, N+1)$$

A개의 디스크에서 모두 탐색하여야 하므로 이들 디스크의 탐색시간의 최대 값이다. 탐색이 완료되면 평균적으로 회전시간의 1/2인 회전지연시간을 거쳐 *A(transfet rate)* 전송률로 데이터를 읽어들인다. 따라서 RAID5에서 읽기의 응답시간은 다음의 식과 같다.

$$RAID5\ read\ time = seek(A) + rotate/2 + (request\ size)/A(transfer\ rate)$$

A개의 디스크가 전송에 참여하므로 디스크가 데이터 전송에 서비스하는 총 시간은 응답시간의 A배이다.

$$RAID\ read\ time = A(seek(A) + rotate/2 + (request\ size)/A(transfer\ rate))$$

디스크 쓰기일 경우는 패리트 디스크에 대한 억세스가 추가되므로 A'를 다시 정의한다. 즉 $A' = \min(S+1, N+1)$이다. RAID5에서의 쓰기 요청은 A'개의 디스크에 모두 탐색해야 하며 이들의 최대 탐색시간을 *seek(A')*이라 하며 응답시간과 디스크 서비스 시간은 다음식과 같다.

$$RAID5\ write\ time = seek(A') + 1.5rotate + (request\ size)/A'(transfer\ rate)$$

$$RAIDwrite\ time = A'(seek(A) + 1.5rotate + (request\ size)/A'(transfer\ rate))$$

위의 두 식에서 보는 바와 같이 RAID5의 쓰기에서는 매번의 디스크 쓰기 요구에 대

하여 패리티를 수정하기 위해 읽기-수정-쓰기(read-modify-write)의 과정이 추가되므로 특히 적은 양의 쓰기 요구일 경우에는 비효율적인 특성을 나타낸다. 이처럼 한 번의 디스크 쓰기 요구에 대하여 기존의 데이터와 기존의 패리티를 읽고 새로운 데이터와 새로운 패리티를 써야 하며 이는 RAID5에서 큰 단점으로 작용한다. 작은 쓰기일 경우 이 단점을 작은 쓰기 문제(small write problem)라 하며 이를 해결하려는 노력이 계속되어 왔다[4][6][7].

2.3 패리티 로깅

RAID5에서는 실제 디스크 대부분의 사용 환경이라 할 수 있는 OLTP환경에서 지나친 디스크 억세스로 성능이 저하된다. 이것에 착안하여 패리티 로깅(parity logging)은 여러 개의 적은 양의 데이터 쓰기를 모아 큰 데이터 쓰기로 전환함으로써 전체적

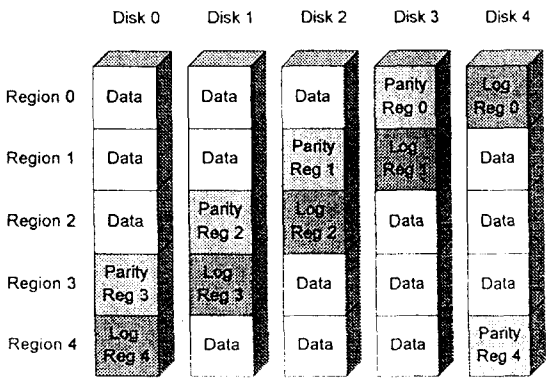

Figure 2. The parity logging.

인 성능을 향상시킨다[6]. 그림 2와 같이 패리티 로깅에서는 디스크 배열을 데이터영역(data region), 패리티 영역(parity region), 로그 영역(log region)으로 나누고 디스크 콘트롤러에 비휘발성 버퍼를 두어 작은 쓰기에서 발생한 패리티 수정 정보를 저장한다. 패리티 수정 정보를 위한 비휘발성 버퍼가 다 차면 디스크의 로그 영역으로 블록단위의 효율적인 디스크 쓰기를 하며, 패리티 버퍼를 비운다. 그리고 디스크에서 로그 영역이 다 차면 로그 영역 내의 패리티 수정 정보와 구 패리티(out-of-date parity) 데이터를 버퍼로 읽어 들여 새로운 패리티 데이터를 만든다. 새로 만들어진 패리티 데이터는 패리티 영역에 대용량의 쓰기를 하고 로그 영역은 다음의 패리티 로그를 위하여 비운다. 결과적으로 패리티 로깅에서는 작은 쓰기일 때 네번의 디스크 억세스 시간이 있어야 되는 것을 두번의 디스크 읽기와 패리티 로그 정보를 유지하는데 필요한 시간으로 줄일 수 있다.

3. 캐쉬를 사용한 RAID5

캐쉬를 사용한 RAID5는 그림 3에서와 같이 작은 쓰기 문제를 해결하기 위하여 디스크 콘트롤러상에 캐쉬를 사용하여 디스크 배열의 성능을 향상시키려는 것이다. 이는 데이터 또는 패리티를 위한 비휘발성 메모리(NVS:Non-Volatile Storage)를 사용하여 디스크 억세스를 줄이므로 호스트(host) 입장에서는 데이터 요구의 응답시간이 줄어드는 것이다. 호스트로부터 받은 쓰기 요청은 디스크 콘트롤러상에 있는 비휘발성 버퍼에 저장되고 이 시점에서 호스트에 데이터 쓰기 완료 신호를 보낸다. 본 논문에서는 RAID5 시스템의 디스크 컨트롤러에 데이터 캐쉬와 패리티 캐쉬를 적용하여 작은 쓰기를 해결하였으며, 이들 캐쉬의 적당한 크기에 대하여 연구하였다.

3.1 빠른 쓰기(fast write)

디스크 콘트롤러에 캐쉬를 두고 호스트로부터의 쓰기 요구 시에 디스크로 직접 억세스 하지 않고 데이터 캐쉬에 존재하는지를 확인한다. 만약 캐쉬 안에 데이터가 존재할 경우는 이 내용을 버퍼로 읽어 들

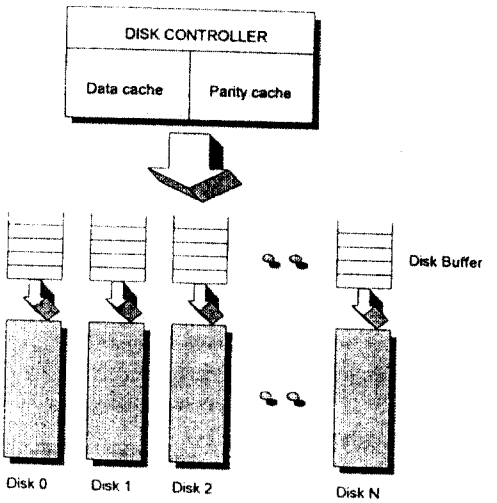

Figure 3. The cached RAID5.

이며 캐쉬를 수정하고, 존재하지 않을 경우는 디스크에서 데이터를 읽어 들이고 캐쉬에 저장한다. 다음 과정으로는 기존의 패리티를 읽어서 수정하여야 하는데 수정할 패리티가 패리티 캐쉬에 존재할 경우는 이것을 구데이터와 신데이터를 XOR한 것에 XOR하여 그 결과를 다시 캐쉬에 저장한다. 수정할 내용이 패리티 캐쉬 내에 존재하지 않는다면 패리티 디스크로부터 기존의 내용을 읽어 들여 구데이터와 구데이터와 XOR한 것에 XOR연산을 하여 캐쉬에 저장한다.

이 과정을 마치면 호스트에 디스크 쓰기 완료 신호를 보내는데 이때 소요되는 시간은 데이터 캐쉬와 패리티 캐쉬에서 적중되었느냐에 좌우된다. 쓰기 할 데이터와 패리티가 둘 다 캐쉬에 있는 경우는 디스크를 전혀 억세스 하지 않아도 되며 데이터나 패리티중 하나만 캐쉬에 있을 경우는 한번의 디스크 읽기가 발생하며, 둘다 없을 경우는 두 번의 디스크 억세스가 발생한다. 그리고 여기에 캐쉬를 억세스 하는데 걸리는 시간이 더해진다. 캐쉬에 쓰여진 정보는 캐쉬의 교체(destage)시에 디스크에 쓰게 되는데 이 시기는 호스트로부터의 요구가 없을 경우에 교체 대상인 캐쉬의 정보들을 방출한다. 이는 연속적인 큰 데이터 쓰기이므로 효율적으로 이루어진다.

위와 같이 캐쉬를 사용하면 어느 경우에든지 RAID5에서의 네 번의 디스크 억세스보다 적은 디스크 억세스로 호스트의 쓰기 요구를 서비스한다. 그러나 캐쉬를 사용하는데 필요한 추가의 경비가 발생하므로 캐쉬의 크기와 성능의 향상간의 상관관계를 고려하여 캐쉬의 크기를 결정하여야 한다. 그리고 데이터 캐쉬와 패리티 캐쉬의 비율에 따른 특성의 변화도 고려하여야 한다.

3.2 캐쉬를 사용한 RAID5의 성능

RAID5에서 캐쉬는 데이터 캐쉬와 패리티 캐쉬로 나누어 고려할 수 있는데 서로간에 장단점을 가지고 있다. 데이터 캐쉬의 경우는 일반적인 디스크 캐쉬의 경우에서와 같이 시스템의 응용에 따라 그 효율이 결정된다. 예를 들어 트랜젝션 프로세싱(transaction processing)처럼 데이터를 자주 읽어 들이고 교체하며 다시 쓰기 하는 응용에서 더 나은 성능을 나타낸다. 새로운 쓰기에 대하여는 직접 디스크에 억세스 하지 않고 캐쉬에 쓰므로 RAID5에서 네번의 억세스를 세번으로 응답시간이 줄이고, 새로운 패리티를 생성하는데 필요한 구데이터가 데이터 캐쉬 안에 존재할 경우(cache hit)에는 두번의 억세스로 줄어든다. 그러나 데이터 캐쉬일 경우는 전체 데이터의 양이 크므로 성능 향상을 위하여 캐쉬의 크기도 커야 된다. 이에 반하여 패리티 캐쉬는 여러 개의 논리적으로 연속된 디스크 섹터들에 의하여 결정되므로 시간적(temporal)이고 공간적(spatial) 지역성(locality)이 있다. 패리티 캐쉬의 경우에서도 쓰기 하려는 데이터의 패리티가 캐쉬 내에 있으면 두번의 디스크 억세스를 줄일 수 있고 캐쉬 내에 없을 경우(cache miss)일 경우는 한번의 디스크 억세스를 줄일 수 있다. 패리티 캐쉬는 데이터 캐쉬에 비하여 적은 양으로 구현할 수 있다. 예로 22개의 디스크를 갖는

RAID5에서는 데이터의 정보가 패리티의 정보보다 21배가 크므로 캐쉬의 크기도 훨씬 커야 하기 때문이다.

캐쉬의 방출정책(destage policy)은 LRU(Least-Recently-Used)를 사용하는데 효율적인 방출을 위하여 같은 트랙 혹은 같은 실린더의 블록을 묶어서 방출한다. 이는 임의의 작은 디스크 쓰기를 순차적인 큰 디스크 쓰기로 바꾸는 역할을 한다.

4. 시뮬레이션

본 장에서는 캐쉬를 이용한 RAID5 시스템을 시뮬레이션을 통해 분석하고 작은 쓰기 문제를 디스크 억세스 횟수를 줄임으로써 효율적으로 해결함을 보인다. 시뮬레이션은 시뮬레이션 전용 언어인 SMPL[9]을 이용하였으며 RAID5와 캐쉬를 이용한 RAID5를 비교함으로써 성능 향상에 대하여 알아본다.

시뮬레이션에서는 작은 쓰기의 초당 요구의 개수(IO/sec)를 변화시키면서 이에 따른 표준응답시간을 구하는데, 시스템의 구성은 디스크 배열 컨트롤러, 데이터 캐쉬, 패리티 캐쉬, 디스크 요구버퍼(disk request buffer), 디스크로 이루어지며 이들간에는 이벤트를 발생시킴으로써 요구의 발생, 요구의 완료를 알린다.

캐쉬를 사용한 RAID5의 시뮬레이션을 위하여 사용된 디스크 배열의 구조와 인자들은 표 1에 나타내었다. 패리티 블록의 수정을 위한 연산시간과 캐쉬를 억세스 하는데 소요되는 시간은 무시하였는데 이는 디스크를 억세스 하는데 소요되는 시간에 비하여 이들의 시간이 상당히 적기 때문이다.

Table 1 Simulation parameters.

Workload	
Access size:	Fixed at 2KB
Alignment:	Fixed at 2KB
Write Ratio:	100%
Spatial Distribution:	Uniform over all data
Temporal Distribution:	Gaussian distribution
Array Parameter	
Stripe Unit:	Fixed at 2KB
Number of Disks:	22 spindle synchronized disks
Head Scheduling:	FIFO
Disk Parameters	
Geometry:	949 cyls, 14 heads, 48 sectors/track
Sector Size:	512 bytes
Revolution Time:	13.9 ms
Seek Time Model:	$2.0+0.01 \cdot dist+0.46 \cdot \sqrt{dist}$ 2 ms min, 12.5 ms avg, 25 ms max
Head Switch Time:	1.16 ms

4.1 시뮬레이션 모델

시뮬레이션에서 사용한 시스템의 흐름도는 그림 4와 같다. 디스크의 배열에 정규분포(normal distribution)의 시간 간격으로 쓰기 요구가 발생되며 이때의 시간을 요구의 도착시간(arrival time)으로 한다. 억세스 데이터 블록의 위치에 따라 패리티 블록의 위치가 결정되는데 이들은 각각 독립적으로 디스크에 저장된다. 시스템의 각 부분에서의 작업의 시작과 종료는 이벤트(event)로서 다음 부분에 전송된다. 데이터 캐쉬, 패리티 캐쉬, 디스크 버퍼 그리고 모든 디스크에서는 독립적으로 이벤트를 처리하는데 이는 디스크 배열에서 여러 개의 억세스 요구를 병렬로 처리함을 뜻한다. 각각의 디스크 배열 억세스 요구의 수행 완료는 데이터 읽기, 패리티 읽기, 데이터 쓰기, 패리티 쓰기가 모두 완료된 시점에 해당하며 요구의 도착시간으로부터 완료된 시점까지의 시간을 응답시간이라 한다. 그리고 정해진 개수만큼의 요구를 발생시켜서 이들이 모두 완료된 시간을 구하고 이를 요구의 개수로 나눔으로서 평균응답시간을 구하였다. RAID5에서의 디스크 배열의 평균응답

시간(average response time)은 억세스 요구의 평균도착시간에 따라 크게 차이가 발생한다. 즉 디스크 배열에서 처리할 수 있는 능력 이상을 억세스를 할 경우에 응답시간은 급속히 증가하게 된다. 캐쉬를 사용한 경우에는 하나의 억세스에 대한 응답시간이 향상되므로 억세스를 처리할 수 있는 능력 또한 향상된다. 이러한 특성을 관찰하기 위하여 IO/sec, 즉 초당 발생하는 억세스 요구의 개수를 50에서 300까지 변화시키면서 표준응답시간을 구하였다. 캐쉬를 사용할 경우는 타당한 결과를 얻기 위하여 시뮬레이션을 시작하기 전에 캐쉬의 내용을 같은 억세스 패턴으로 채웠다.

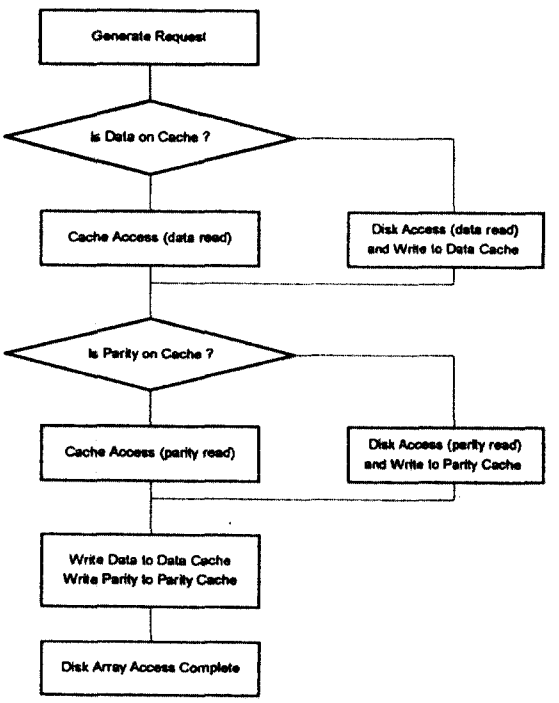

Figure 4. Simulation flowchart.

4.2 결과 분석

먼저 RAID5를 데이터 캐쉬를 사용하지 않은 RAID5의 평균응답시간을 구하고 같은 환경 하에서 캐쉬의 크기를 변화시키면서 이때의 억세스에 대한 평균응답시간을 비교하였다. 캐쉬의 크기의 변화 방법은 먼저 전체 캐쉬의 크기를 고정시키고 데이터 캐쉬와 패리티 캐쉬의 비율을 변화시키면서 이때의 성능을 구하였다. 그림 5에서는 RAID5와 캐쉬를 사용한 RAID5의 디스크 배열의 평균응답시간을 나타낸 것이다. 캐쉬의 크기는 데이터 캐쉬가 10000블록이고 패리티 캐쉬는 300블록이다.

그림 5와 같이 RAID5에 비하여 캐쉬를 사용한 RAID5의 성능이 크게 향상됨을 알 수 있다. 이는 디스크로 억세스 하는 횟수를 줄임으로 해서 응답시간이 줄어들게 되며 IO/sec가 증가할수록 RAID5와 캐쉬를 사용한 RAID5의 응답시간 차이는 크게 증가한다. RAID5에서는 200 IO/sec 이상에서는 응답시간이 급격히 증가하여 높은 I/O처리율을 요구하는 시스템에서는 적합하지 않음을 알 수 있다.

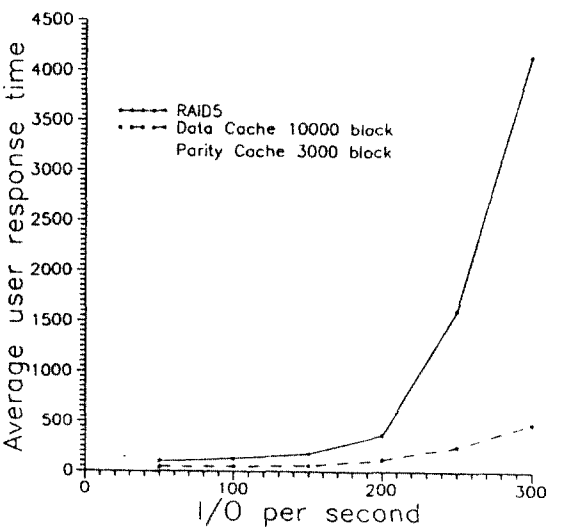

Figure 5. Response time.

데이터 캐쉬의 크기에 따른 디스크 배열의 성능의 변화를 알아보기 위하여 패리티 캐쉬 크기를 3000블록으로 고정시킨 상태에서 데이터 캐쉬를 변화시키면서 평균응

답시간을 구하였는데 이의 결과는 그림 6과 같다. 그림 6에서 보는 바와 같이 데이터 캐쉬의 크기가 커짐에 따라 표준응답시간은 향상된다. 그러나 캐쉬의 크기가 어느 정도 이상에서는 성능의 성능 향상 정도가 둔화됨을 알 수 있다. 데이터 캐쉬의 크기가 16000블록 이상에서는 캐쉬의 크기를 증가시켜도 성능 향상은 그리 크지 않다.

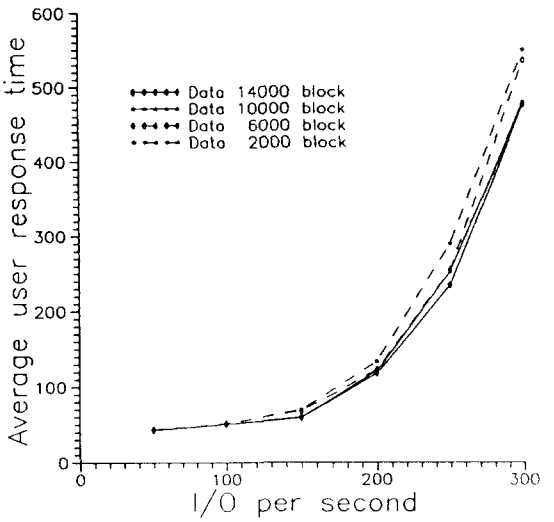

Figure 6. Data cache size variation for response time.

그림 7에서는 일정한 캐쉬의 크기 하에서 데이터 캐쉬와 패리티 캐쉬간의 비율의 변화에 따른 디스크 배열의 표준응답시간을 보였다. 디스크 배열의 성능을 향상시키기 위하여 캐쉬의 크기를 무작정 증가시킬 수 없기 때문에 이들간의 비율을 고려하여 좀더 효율적인 캐쉬 시스템을 구현하여야 한다. 그림 7에서는 전체 캐쉬의 크기를 15000블록으로 정하고 데이터 캐쉬와 패리티 캐쉬간의 비율을 변화시켰는데 패리티 캐쉬의 비율을 크게 할수록 디스크 배열의 응답시간이 향상된다는 것을 알 수 있다. 이는 디스크 시스템에 억세스 하는 패턴이 시간적, 공간적 지역성이 있을 경우 패리티 영역에 대하여는 그 지역성이 더 증가하기

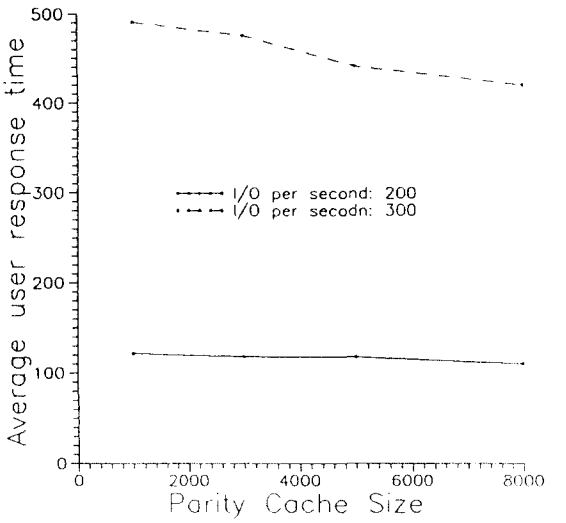

Figure 7. Parity cache size variation for response time.

때문이다.

5. 결 론

본 논문에서는 디스크 배열 형태 중에서 가장 널리 사용되는 RAID5에서의 작은 쓰기 문제의 해결 방안으로서 캐쉬를 사용한 RAID5의 성능 분석에 관하여 연구하였다. 디스크 배열 컨트롤러에 데이터 캐쉬와 패리티 캐쉬를 사용하였으며, 이로 인한 디스크 억세스 횟수의 감소에 따른 성능 향상에 대하여 연구하였다. 캐쉬의 신뢰성 향상을 위하여 NVS를 사용하였으며 효율적인 관리를 위하여 캐쉬의 크기와 디스크 배열 억세스의 평균응답시간의 관계에 대하여도 연구하였다. 또한 적당한 크기의 캐쉬에서 데이터 캐쉬와 패리티 캐쉬의 비율을 바꾸면서 성능을 측정하였으며 이와 같은 시뮬레이션을 통하여 캐쉬를 사용한 RAID5가 작은 쓰기 문제를 효율적으로 해결할 수 있음을 보였다.

참 고 문 헌

[1] M.Y.Kim "Sychronized disk interleaving," *IEEE Trans on computer*, Vol. c-35 no. 11. Nov 1986.

[2] K. Salem and H. Grcia Molina, "Disk striping," in *Proc. of the IEEE International Conference on Data Engineering*, pp. 336-342, Feb. 1986.

[3] Edward K. Lee and Randy H. Katz, "Performance Consequences of Parity Placement in Disk Arrays," in *Proc. of 4th ACM Arch. Supp. Prog. Lang. and Oper. Syst.*, Santa Clara, California, pp. 190-199, 1990.

[4] Jim Gray, Bob Horst, and Mark Walker, "Parity Striping of Disk Arrays: Low-Cost Reliable Storage with Acceptable Throughput," in *Proc. of VLDB Conference*, 1990.

[5] J.M. Menson and R.L. Mattson, "Performance of disk arrays in transaction processing environments," in *Proc. of the 12th International Conference on Distributed Computing Systems*, June 1992.

[6] Daniel Stodolsky, Grath Gibson, and Mark Holland, "Parity Logging : Overcoming the small write problem in redundant disk arrays," in *Proc. of the 20th International Symposium on Computer Architecture*, 1993.

[7] Jai Menon and Jim Cortney "The Architecture of a Fault-Tolerant Cached RAID Controller," in *Proc. of the 20th International Symposium on Computer Architecture*, 1993.

[8] Peter M, Edward K and Garth A. Gibson, "RAID: High-Performance, Reliable Secondary Storage," *ACM Computing Surves*, Vol. 26, No. 2, June 1994.

[9] M.H. MacDougall, *Simulating Computer Systems Techniques and Tools*, The MIT Press.