지연 최소화를 위한 디스크-메모리 혼용 디스크 구조

이남규○ 한탁돈
연세대학교 컴퓨터과학과 미디어 시스템 연구실
{nklee,hantack}@kurene.yonsei.ac.kr

A Disk-Memory Hybrid Disk Architecture for Minimizing Latency

Namkyu Lee Tackdon Han
Media system Lab., Dept. of Computer Science, Yonsei University

요 약

이 논문에서는 폭넓게 사용되지만 컴퓨터의 메모리 계층 구조상에서 병목지점으로 알려진 하드디스크의 획기적인 성능향상을 위해서 메모리 시스템이 내장된 새로운 형태의 디스크 구조를 제안한다. 제안하는 디스크 구조에서는 디스크에 메모리를 혼용하여 사용함으로써 디스크 응답 시간을 크게 줄이고, 입출력을 빠르게 처리할 수 있다. 64MB까지의 디스크 메모리를 탑재한 경우 두 가지 실제 트레이스에 의한 시뮬레이션 결과 20여명이 사용하는 공유 시스템의 작업부하에서는 최대 80% 정도의 히트율을 통하여 최대 1/2, 그리고 개인용 시스템의 경우 최대 85% 가량의 히트율을 통해 1/5 수준으로 응답시간을 단축할 수 있었다. 앞으로 디스크에 단순히 메모리를 추가하는데 그치지 않고 데이터 블록의 배치 방법, 데이터 분산 배분 방법, 보관정책, 선인출 방법 등을 이용하면 추가된 디스크 메모리의 효용을 극대할 수 있다.

1. 연구 배경

컴퓨터에서의 지연을 살펴보면 Alpha 21164 시스템의 예에서 나타나듯이 캐쉬의 접근 시간은 레벨 3 외부 캐쉬일지라도 단지 수십 ns 이고 메인 메모리 접근 시간은 수백 ns이다[1]. 반면에 일반적으로 하드디스크(이하 디스크) 접근 시간은 수십 ms(mili second)단위로 큰 차이를 보인다. 따라서 사용자들은 디스크 접근이 일어 날 때마다 현저한 속도 차를 느끼고 있다. 그리고 현재 기술 발전 추세를 보면 디스크의 저장 용량은 크게 증가하고 있지만 디스크 접근 속도에 대한 개선은 느리게 진행되고 있으므로 메모리 계층구조상의 개선이 필요하다.

사용자의 디스크 접근 형태에 대한 분석 자료에 의하면 디스크에 대한 참조 요구는 메모리나 캐쉬를 사용하는 원리와 마찬가지로 국부적으로 일어난다. Staelin[2]과 Ferguson[3]는 연구 결과에서 입출력 요구 특성으로 80:20 규칙을 찾아냈다. 즉 전체 입출력 요구의 80%이상이 단지 20%의 디스크 공간에 국한되고 있고, 심지어 50% 이상의 입출력 요구가 단지 1% ~ 3%의 디스크 공간에 국한되어 나타나고 있는 현상을 보였다. 그리고 Wilkes의 연구결과 논문[7]에 의하면 24시간 동안 일반적으로 디스크에 대한 입출력요구는 평균적으로 전체 저장 공간의 2.6% ~ 6.7%, 최대 전체 저장 공간의 16% ~ 34% 만을 접근하는 것으로 나타났다. 그러므로 예를 들어 1GB 디스크를 기준으로 6.7%의 공간이 접근된다면 단지 67MB의 메모리만 디스크 내부에 포함되어 있다면 전체 입출력 요구를 메모리에서 처리할 수 있을 것으로 기대된다. 주기억장치의 대용량 화일 캐쉬는 디스크에 대한 읽기 요구를 효과적으로 줄일 수 있고, 쓰기 요구를 줄이기 위해서 DCD(disk caching disk)[4] 시스템과 같은 연구도 진행되었지만 Solid State Disks(SSDs)[5]에서와 같이 디스크 접근 속도를 근본적으로 개선하기는 미흡하다. 그리고 SSD나 플래시 메모리가 디스크를 대체하기엔 가격이 너무 비싸기 때문에 소규모 휴대용 컴퓨터에서조차 동전 크기 정도의 IBM 마이크로드라이브[6] 와 같은 디스크를 사용하는 추세를 보이고 있다. 따라서 컴퓨터가 최적의 성능을 나타내기 위해서는 디스크 자체의 구조 개선이 필연적이며, 현재 개발된 기술을 서로 융합시키면서 새로운 디스크 구조를 설계해야 한다.

본 논문에서는 디스크 성능 개선을 위한 한가지 방법으로 기존의 디스크에 메모리 시스템을 내장시킴으로써 궁극적으로 사용자들에게 신속한 응답을 주는 것을 목적으로 하였다. 더욱이 디스크에 단순히 메모리를 추가하는데 그치지 않고 효과적인 데이터 블록의 배치 방법, 실제 디스크와 디스크 내부 메모리 사이의 데이터 분산 배분 방법, 보관정책, 선인출 방법 등을 이용하면 추가된 메모리의 효용을 극대할 수 있다. 연구 배경에 이어 본 논문의 2장에서는 디스크-메모리 혼용 디스크 모델에 대해 설명하고, 3장에서는 간단한 성능 평가 결과에 대해 기술한다. 그리고 4장에서는 추가적인 성능 향상 방안을 설명하고, 마지막장에서 결론을 맺는다.

2. 디스크 메모리 혼용 디스크 모델

제안하는 메모리 혼용 디스크 모델은 기존의 하드 디스크에 메모리(DRAM) 시스템을 내부에 포함하는 구조이다. 디스크-메모리 혼용 디스크 구조는 크게 다음과 같이 제어 로직인 VDCL(Virtual Disk Control Logic), 디스크 내부 메모리 시스템 (disk memory), 디스크 미디어, 기존의 디스크 캐쉬로 구성된다. 일반적인 디스크에도 디스크 버퍼, 트랙 캐쉬 등의 이름으로 소규모 메모리, 대게 이중 포트 SRAM을 사용하고 있다. 32KB ~ 1MB의 크기에 지나지 않는 작은 크기이므로 한개의 트랙 전체를 읽어 오기 위한 버퍼 공간이나 선읽기(read ahead) 기능을 지원하기 위한 제한된 기능만을 지원한다. 디스크-메모리 통합운용에 있어서 메모리를 디스크에 두는 것보다 그 만큼의 메모리를 주기억장치로 확장시키는 것이 낫다고 말할 수도 있지만, 실제 주기억장치에는 물리적 공간에 제약이 따르고,

* 이 논문은 1998년도 연세대학교 학술연구비의 지원에 의하여 이루어진 것임.

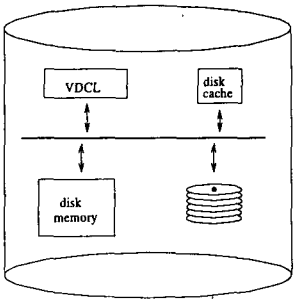

그림 1: 디스크-메모리 혼용 디스크 모델

경우에 따라서는 기억 공간의 크기가 커지면 유지 관리하기 위한 오버헤드가 더 필요하게 된다. 따라서 디스크 메모리는 장착되는 디스크 단위로 메모리를 확장시킬 수 있으므로 확장성 면에서 문제가 없다. 그림 2는 디스크 내부 로직 보드를 나타내는 그림으로 ASIC 기술의 발달로 인하여 그 크기가 기존의 절반만을 차지하는 단계에 이르렀으므로 여분의 공간에 메모리 시스템을 구성할 수 있다.

디스크 메모리에는 디스크 미디어에 저장되어 있는 화일중에서 참조 빈도가 높은 화일을 저장함으로써 참조의 지역성 원리를 이용할 수 있다. 디스크 읽기 요구시에는 일반적인 디스크에서와 같이 탐색(seek)을 시작하고, 동시에 디스크 메모리를 검색하게 된다. 디스크 메모리에서 히트가 되면 탐색 동작은 취소되고 디스크 메모리의 데이터가 전송된다. 쓰기 요구시에는 일단 디스크 미디어까지 쓰지 않고 디스크 메모리에 쓰기가 수행된 후 곧바로 쓰기 완료가 수행되었음을 호스트에 알린다. 그리고 디스크가 쉬고 있는 동안(idle)에 주기적으로 동기화가 일어난다. 트레이스 데이터의 분석결과 디스크가 동작을 수행하지 않고 쉬는 기간이 90%를 상회하기 때문에 신뢰성을 위한 수정된 데이터의 동기 시간은 충분하다. 일정량의 메모리를 사용하고 참조의 지역성을 이용하여 일부 블록을 저장한다면 최소한 메모리 사용량만큼의 성능 향상을 얻을 수 있다. 또한 주어진 디스크 메모리를 효과적으로 운용한다면 적은 양의 메모리로도 성능 향상을 극대화 할 수 있다. 디스크 메모리의 효과적 운용 방법으로는 지연을 숨길 수 있을 정도의 데이터를 미리 저장해 놓고 있는 최적 크기 캐싱과 개별 사용자 또는 모든 사용자들의 접근 히스토리, 또는 가장 빈번히 접근되는 블록에 의한 선인출 기법이 필요하다. 제안한 디스크 모델의 디스크 응답 시간을 T_R이라 한다면, T_R은 디스크 플레터에 대한 접근 시간(T_{dp}), 디스크 메모리 접근 시간(T_{dm}), 디스크 메모리 히트율(H), 큐에서의 대기 시간(T_q), 콘트롤러 오버헤드(T_c)를 이용하여 다음과 같은 식으로 나타낼 수 있다.

$$T_R = T_{dm} \times H + T_{dp} \times (1 - H) + T_q + T_c$$

3. 성능 평가

제안하는 디스크-메모리 혼용 기술을 이용한 고성능 디스크의 성능을 예측하기 위해 시뮬레이션을 이용한 성능평가를 수행하였다. 시뮬레이션을 위해 사용한 시뮬레이터는 미시간 대학에서 개발한 DiskSim 시뮬레이터[8]를 기반으로 하여 제안한 모델의 디스크 구조를 반영하도록 수정하여 이용하였다. 시뮬레이터에서 사용한 디스크 모델은 Segate ST41601N 1.37GB 이다. 또한, 효과적인 시뮬레이션을 위하여 휴렛패커드 연구실에서 생성한 두 가지 형태의 실제 트레이스 데이터를 이용하였다. 그 중 한가지는 cello 시스템에서 생성한 트레이스로 이 시스템은 20여명의 연구원이 공유하는 시분할 시스템으로 일반적인 연구목적으로 사용되는 화일 에디팅, 컴파일, 메일, 시뮬레이션등의 일을 수행하는 특징을 가지고 있었다. 그리고 다른 한가지는 hplajw 시스템으로써 개인용 워크스테이션이다[7]. 시뮬레이션에 의한 성능 평가의 척도로는 평균 디스크 응답 시간과 디스크 메모리에서의 히트율이 사용되었다. 응답 시간 내에는 큐에서의 대기시간, 디스크 접근시간이 추가로 나타내었으며, 디스크 메모리의 히트율에는 읽기, 쓰기, 그리고 종합을 별도로 생성하였다.

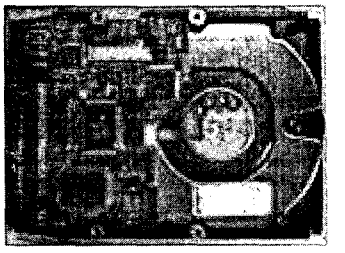

그림 2: 전형적인 실제 하드디스크 모습(IBM Deskstar 16GP DTTA 350640 6.4GB)

그림 3과 그림 4는 이를 동안의 Cello 트레이스 데이터를 이용하고, 디스크 메모리의 크기를 증가시키면서 평균 응답시간과 디스크 메모리에서의 블록 히트율을 측정한 결과이다. 그림에서 나타나듯이 디스크 접근 시간을 줄임으로 인하여 큐에서의 대기시간이 크게 줄어드는 것을 알 수 있고 그 결과 평균 응답시간을 크게 줄일 수 있었다. 디스크 메모리가 크지 않을 경우에는 쓰기에 의한 히트율이 크고, 읽기에 의한 히트율은 미미한 것을 알 수 있다. 쓰기 데이터는 주기적으로 동기화가 이루어짐을 알 수 있고, 읽기 데이터의 경우 디스크 메모리의 크기가 작을 경우 효과가 거의 없고, 크기가 커짐에 따라 메인 메모리 화일 시스템에서 교체된 페이지의 재요구에 따른 히트가 발생됨을 예측할 수 있다. 일주일 동안의 트레이스 데이터를 이용한 시뮬레이션 결과, hplajw와 같은 개인용 시스템의 경우 그림 5와 그림 6에서와 같이 작업부하가 낮기 때문에 16MB 정도의 디스크 메모리를 사용하더라도 좋은 효과를 볼 수 있을 것으로 예측된다.

4. 추가적 성능 개선 방안

제안한 디스크 메모리는 인터넷 접근 데이터, 읽기 블록의 우선 교체, 스왑 공간(swap space)으로의 이용, 히스토리에 의한 선인출, 확장 화일 시스템과 같은 용도로 이용함으로써 추가적으로 획기적인 성능 개선을 할 수 있다.

- 인터넷 접근 데이터 : 인터넷이 각광을 받으면서 최근 네트워크를 통하여 접근하는 데이터의 양이 크게 증가하고 있다. 이러한 데이터의 특성은 읽기 데이터가 대부분이고, 데이터의 크기가 큰 멀티미디어 데이터이므로 주기억장치 캐싱 대신 디스크 메모리에 저장하여 이용한다.
- 읽기 블록의 우선 교체 : 쓰기에 의한 블록인 경우와는 달리 읽기에 의한 블록인 경우 디스크 메모리 내의 블록은 대부분 주기억장치 내에 있게 된다. 그러므로 읽기에 의한 블록을 MRU(Most Recently Used) 교체 기법을 적용하여 우선적으로 교체한다.
- 스왑 공간 : 디스크 메모리는 스왑 공간으로도 유용하게 사용될 수 있다. UNIX와 같은 운영체제의 경우 일반적으로 디스크에 주기억장치의 세배 가량의 스왑 공간으로 확보해 놓는다. 그러므로 문맥교환시(context switching) 프로세스 스왑을 신속하게 하기 위하여 일정 공간을 이러한 용도로 사용할 수 있다.
- 히스토리에 의한 선인출 : 개인용 컴퓨터 사용자들을 위해서는 사용자 입출력 요구 히스토리에 의한 선인출 방식이 사용될 수 있다. 사용자 데이터 접근 히스토리를 기록하기 위한 데이터 구조가 디스크 메모리에 구성된다. 히스토리 정보를 이용하여 앞으로 참조 가능성이 높은 블록을 선인출하여 디스크 메모리에 위치시킨다.
- 확장 화일 시스템 : 주기억장치 화일 시스템에서 데이터 블록을 유지하는데 사용되는 화일 캐쉬에서 LRU(Least Recently Used) 기법에 의해 방출되는 블록의 저장 공간으로 사용된다. 읽기의 경우 주기억장치로부터 전송 받아 올 필요는 없고 디스크가 쉬는 동

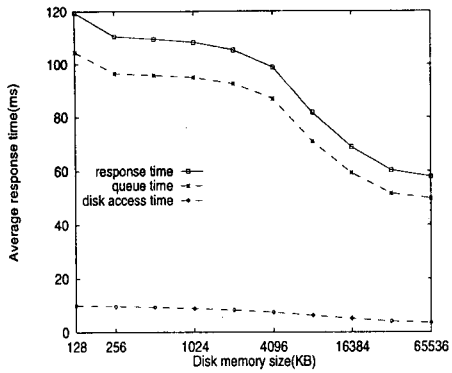

그림 3: 디스크-메모리 혼용 디스크 평균 응답시간(cello)

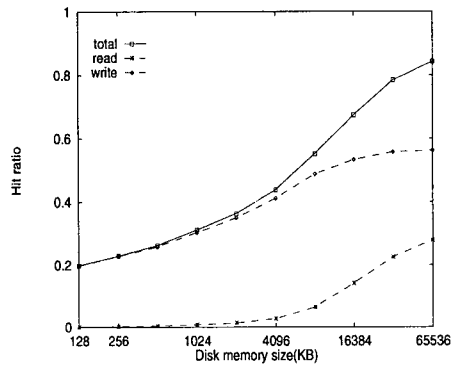

그림 4: 디스크-메모리 혼용 디스크 히트율(cello)

안 선인출하는 방법을 사용한다.

5. 기대 효과 및 결론

현재의 디스크 구조를 근본적으로 개선하지 않고, 소프트웨어 기술만으로 획기적인 성능향상을 기대하기는 어렵다. 제안하는 디스크-메모리 혼용 디스크는 현재의 디스크 구조에 메모리 시스템을 추가하여 이용하는 구조이다. 사용자 데이터 접근에 대한 패턴 분석에 의하면 전체 디스크 공간의 일부분만이 사용되는 경향을 보이므로 디스크 내부에 디스크 메모리를 사용한다면 만족할 만한 성능 향상을 보일 수 있다. 64MB까지의 디스크 메모리를 탑재한 경우 두 가지 실제 트레이스에 의한 시뮬레이션 결과 20여명이 사용하는 공유 시스템의 작업부하에서는 최대 80% 정도의 히트율을 통하여 최대 1/2, 그리고 개인용 시스템의 경우 최대 85% 가량의 히트율을 통해 1/5 수준으로 응답시간을 단축할 수 있었다. 특히 주어진 크기의 디스크 메모리를 효과적으로 유지, 관리함으로써 보다 높은 디스크 성능 향상을 가져올 수 있다. 궁극적으로 제안된 디스크는 메모리만으로 이루어진 SSD에 근접한 접근 시간을 보일 것으로 기대된다. 실제 디스크-메모리 혼용 디스크를 개발한다고 할 때, 가격적인 측면에서 살펴보면 1999년 현재 64Mb DARM이 약 $9이므로 64MB 메모리 구성시 $72정도 소요되고, 그 외 일정량의 추가 비용이 필요할 것으로 보인다. 디스크 제조 업체에서는 가격에 매우 민감한 반응을 보이지만 일정한 추가 비용으로 고성능 디스크를 설계하여 제공한다면 사용자들의 만족감을 극대화 할 수 있다. 결론적으로 제안하는 디스크 모델에서는 디스크에 메모리 시스템을 혼용하여 사용함으로써 디스크 응답 시간을 크게 줄이고, 향후 제안하는 추가적 성능개선 방안을 적용함으로써 입출력을 빠르게 처리할 수 있을 것으로 기대된다.

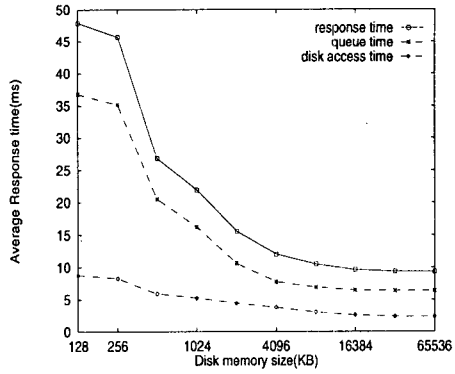

그림 5: 디스크-메모리 혼용 디스크의 평균 응답시간(hplajw)

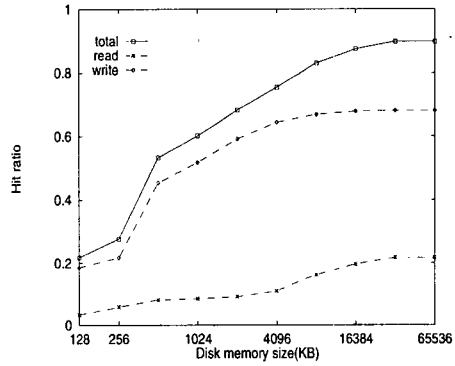

그림 6: 디스크-메모리 혼용 디스크의 히트율(hplajw)

참고 문헌

[1] D.A. Patterson,"A Case for Intelligent RAM: IRAM," IEEE Micro, April 1997.

[2] Staelin and Carl, "File access patterns," Technical report CS-TR-179-88, Princeton University, 1988.

[3] Ferguson and Paula, "Hot File analysis of FSTRACE I/O data," internal digital memorandum, 1989.

[4] Yiming Hu and Qing Yang, "A New Hierarchical Disk Architecture," IEEE Micro, pp64-76, november-december 1998.

[5] -,"Solid State Disks,"http://www.quantum.com/src/whitepapers/ssd", Quantum, 1998

[6] -, "IBM 170MB/340MB microdrives" http://www.storage.ibm.com/hardsoft/diskdrdl/prod/micro/170340/170spec.htm, September, 1998.

[7] C. Ruemmler and J. Wilkes, "UNIX disk access patterns," HP Lab. technical report, January 1993.

[8] G.R. Ganger, B. L. Worthington, and Y.N. Patt, "The DiskSim Simulation Environment Ver 1.0 Reference Manual," CSE-TR-358-98, Feburary 27, 1998.

[9] Garth A. Gibson, et al.,"A Cost-Effective, High-Bandwidth Storage Architecture", Proc. of the 8th Conf. on architectural Support for PL and OS, 1998.