++디스크 드라이브 레벨에서 힌트정보를 이용한 디스크 캐쉬 운영 방안

조재동, 장태무

동 국 대 학 교 컴 퓨 터 공 학 과

Disk Cache Operating Strategy Using Hints in Disk Drive

Jaedong Cho, Taemu Chang

Dept. of Computer Engineering, Dongguk University

요 약

마이크로 프로세서의 동작 속도와 디스크 액세스 속도의 성능 차이는 컴퓨터 시스템의 성능을 제한하는 중요한 요인 주의 하나로 지적되고 있다. 이러한 격차를 줄이는 기술로 디스크 캐쉬의 운영이 연구되어 왔고 디스크 캐쉬 성능 개선 방법으로 선인출이 널리 연구되어 왔다. 본 논문에서는 디스크 드라이브 상에 구현된 캐쉬에서 디스크 요청에 대한 성격적 유형을 힌트로 이용한 선인출 적용 방법을 제안하고, 제안된 방법의 유효성은 시뮬레이션 방식으로 입증하였으며 적응적으로 변경된 선인출 적용 방법이 성능의 개선을 이룰 수 있음을 보였다.

1. 서론.

마이크로프로세서와 메모리의 동작 속도는 급격히 빨라지고 있는 반면 디스크 액세스 속도는 크게 증가를 보이지 않고 있다. 이러한 주요 시스템 구성 요소 사이의 성능 차이는 컴퓨터 시스템의 성능을 제한하는 중요한 요인 중의 하나로 지적되고 있어 이들간의 처리 속도의 격차를 줄이는 기술이 요구되게 되었다. 이러한 목적으로 디스크 캐쉬를 운영하는 것이 하나의 방법이 되어 왔다.

디스크 캐쉬에서의 성능 개선 요인은 최근에 접근한 자료를 유지하여 읽기 시간을 줄여 주는 것과, 쓰기 동작에 대하여 디스크에 전송하는 것을 지연시켜 쓰기 시간을 줄여 주는 것으로 요약할 수 있다. 그 외에도 순차적인 접근을 위한 선인출(prefetch) 동작, 요청 큐(request queue)를 두어 디스크 요청의 순서를 변경하는 스케쥴링(scheduling) 등으로 성능을 개선할 수 있다.[1,2] 그러나 디스크 캐쉬를 보다 잘 활용하려면 디스크 내의 내용[3]과 접근 양식[4]에 기반한 운영 방법이 바람직하다. 예를 들어 선인출 방법이 순차적인 디스크 접근에 효율적인 방법이 되지만, 이에 의하여 반드시 성능이 개선되는 것은 아니다[5, 6].

최근 하드웨어 기술의 발달로 지능형 디스크 드라이브를 구축하는 것이 가능하게 되었다. 본 논문에서는 디스크 드라이브 단계의 캐쉬에서 보다 효율적인 캐쉬 운영을 위하여 시스템으로부터 힌트 정보를 얻어 이를 활용하는 방안을 제시한다.

2. 관련연구

디스크 성능을 개선하기 위한 캐쉬에서 최근 접근된 데이터의 유지와, 선인출 그리고 스케쥴링 동작은 서로가 성능에 영향을 미치므로 통합되어 운영되어야 한다. 이를 위하여 파일 캐쉬 수준에서 캐쉬와 선인출, 스케쥴링 동작을 통합 운영하는 방안[5, 7]이 연구되어 있다. 또 효과적인 선인출을 위해 애플리케이션에서 힌트 정보를 구하는 방안[6]도 소개되었다. 만일 미래의 디스크 접근 형태를 알 수 있다면 이러한 선인출 동작은 실행 시간을 상당히 개선할 수 있다.[5, 6, 7] 그러나 이러한 캐쉬를 파일 캐쉬 수준에서 구현하는 경우 적중율(hit ratio)면에서 유리한 측면은 있으나 운영체제에 부담을 주며, 운영 체제를 RAID 특성 등 디스크 장치 특성에 맞게 수정해야 하며, 다중 프로세서 시스템의 경우 커널마다 따로 캐쉬를 운영해야 하는 부담이 있다. 본 논문에서는 디스크 드라이브 수준에서 실제 디스크 내용과 접근 형태를 운영 체제에게서 힌트

정보로 얻어 캐쉬내의 데이터 유지, 선인출, 쓰기 지연 기법을 통합하여 운영함으로써 효율성을 높이고, 이를 시뮬레이션 방법으로 실험하여 그 결과를 보이고자 한다.

3.새로운 디스크 캐쉬 운영방안

3.1 운영 전략

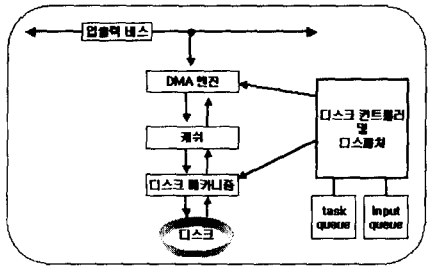
본 논문에서의 디스크 캐쉬 운영 방안의 새로운 점은 크게 다음의 두 가지로 정리할 수 있다. 첫 번 째는 선인출 여부와 적절한 크기를 정하는 것이다. 선인출은 미래에 요구될 데이터를 미리 캐쉬에 가져 오는 것이므로 필연적으로 선인출 동작 이전에 새로운 블록을 할당 받아야 한다. 따라서 참조되어야 할 캐쉬 내의 데이터가 교체될 가능성이 있다. [5] 에서 논의 되었듯이 선인출로 인하여 캐쉬 안에 부적절한 블록 할당으로 시스템 성능이 떨어 질 수도 있다.

본 논문에서는 이 문제를 해결하기 위해서 현재의 디스크 요청에 대한 성격적 유형을 디스크 요청 매개변수로 넘겨 준다고 가정한다. [3] 에 의하면 파일의 확장자만 관찰하여도 평균 크기, 요청 간격, 생명 주기의 차이를 관찰할 수 있다. 디스크 캐쉬는 이 힌트를 이용하여 선인출 블록의 크기를 조정하여 선인출 블록과 캐쉬 블록의 충돌 문제를 완화한다.

두 번 째는 독립된 읽기 요청이 뒤섞이는 것을 막기 위하여 캐쉬를 세그먼트(segment)로 나누어 운영하는 것이다. 이를 위하여 운영 체제로부터 구별될 수 있는 프로세스 식별자를 매개 변수로 얻어 온다고 가정한다. 이렇게 하여 상호 관련이 없는 자료가 캐쉬 내에 들어 가는 것을 막을 수 있다.

3.2 디스크 드라이브 모델

본 논문에서 사용하는 디스크 드라이브 모델은 [그림 1]과 같다. 즉 기록과 디스크 위치를 찾는 역할을 담당하는 디스크 메커니즘, 입출력 버스 인터페이스 측에서 실제 전송을 담당하는 DMA 엔진, 디스크 캐쉬 및 전반적인 운영 프로그램을 실행하는 마이크로 프로세서 등이 그 구성 요소들이다.



[그림 1]. 디스크 드라이브 모델

디스크 메커니즘은 디스크 컨트롤러가 행한 요청에 대하여 필요한 데이터가 있는 위치에서 데이터를 읽어 캐쉬에 이를 전송하거나 캐쉬에 있는 데이터를 받는 작업을 한다. 디스크 컨트롤러는 입력 큐(input queue) 에 들어 있는 요청들에 대하여 데이터 읽기/쓰기 요청 및 선인출을 위한 요청을 다시 만들어 타스크 큐(task queue) 에 넣는다. 입력 큐에 있는 요청들은 스케쥴링 알고리즘에 의하여 최적의 접근 시간이 되도록 재배치될 수 있다. 타스크 큐에 있는 작업은 순차적으로 디스크 메커니즘과 DMA 엔진을 위한 요청으로 디스패치된다.

3.3 운영 알고리즘

우선 디스크 캐쉬 컨트롤러가 운영체제로부터 받는 힌트 정보는 [표1]과 같다.

[표1] 선인출을 위한 힌트 정보

내용	설명
프로세스 식별자	입력 스트림이 섞이지 않게 하기 위함
파일 유형	시스템(페이징 또는 스왑, 메타데이터 등), 확장자에 따른 분류

[표1]의 정보는 캐쉬 세그먼트와 선인출 크기를 정하는 참고 자료가 된다.

디스크 캐쉬 운영 알고리즘을 정리하여 가상 코드 형태로 보인 것이 [그림 2] 와 같다.

```
<Queue Task>
for all requests in input queue
    determine prefetching size;
    determine segment number;
    send the request to Task Queue;
<Caching Task>
if(any request in Task Queue)
    Replace clean block by LRU;
    if (no clean block) flush into disk;
    Process read or write request
<Idle Task>
if(idle time out or the half of cache is dirty)
    flush blocks into disk;
```

[그림 2] 캐쉬 운영 가상 코드

4.시뮬레이션 환경 및 실험 결과

본 논문에서 제시된 방법의 유효성을 입증하기 위하여 DiskSim 2.0 시뮬레이터[8]를 이용하였다. DiskSim 은 실행 시간 면에서 효율적이고 다양한 입출력 장치 환경에서 사용할 수 있는, 저장 장치 서브 시스템의 성능을 점검할 수 있는 입증된 시뮬레이터이다[9].
작업 부하의 특성과 시뮬레이션 변수는 각각 [표2] 및 [표3]과 같다.

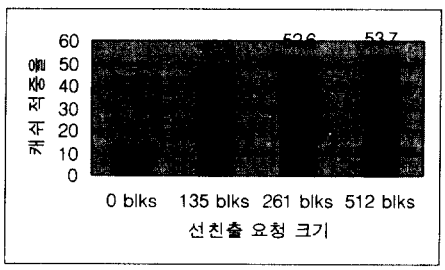
[표2]작업 부하의 특성

요청간격	평균 24.24 ns, 표준 편차 27.46 ms, chleo 간격 444.ms
요청크기	평균 4.9KB, 표준 편차 3.8KB, 최대 38 KB
읽기/쓰기 비율	0.66/0.34
요청회수	100,000

[표3] 시뮬레이션 변수

디스크 캐쉬	캐쉬 최대 요청 크기 : 512 블록 선인출 요청 크기: 최대 512 블록까지 가능
디스크 (HPC 3323A)	최대 용량 : 1.03 GB 실린더2982,표면 7, 섹터크기 512B,8존(zone) 평균 탐색시간 : 10 ms 헤드 회전 속도 : 5400rpm

선인출 크기가 여러 가지로 다르게 적용된 시뮬레이션의 결과는 [그림 3] 과 같다. 따라서 성능 개선 정도를 엄밀히 수치화하기는 어렵지만 성능의 개선이 있음은 확인할 수 있다.



[그림 3] 선인출 요청에 대한 캐쉬 적중율

5. 결론 및 향후 연구 방향

선인출이 디스크 캐쉬에 미치는 영향은 여러 연구에서 결론이 나 있다. 본 논문에서는 디스크 드라이브 상에 구현된 캐쉬에서 선인출 적용 방법을 다양하게 적응적으로 변경함으로써 성능의 개선을 이룰 수 잇음을 보였다. 향후 보다 정확한 드라이브 모델 상에서 실제 시스템의 입출력 추적 자료(trace dtata)를 이용하여 보다 정확한 시뮬레이션 결과를 얻는 방안을 모색하고 있다.

[참 고 문 헌]

[1] E. Shriver et al, "An Analytic Behavior Model for Disk Drives with Readahead and Request Buffering," Proc. SIGMETRICS'98, pp.182-191.

[2] C. Ruemmler et al, "An Introduction to Disk Drive Modeling," IEEE Computer, vol.27, no.3, Mar. 1994, pp.17-29.

[3] J. R. Doucer et al, "A Large-Scale Study of File-System Contents," Proc. SIGMETRICS'99, pp.59-70.

[4] C. Ruemmler et al, "UNIX Disk Access Patterns," USENIX 1993 Tech. Conf. Proc. Pp.405-420.

[5] P. Cao et al, "A Study of Integrated Prefetching and Caching Strategies," Proc. SIGMETRICS'95, pp.188-197.

[6] P. Cao et al, "Implementing and Performance of Integrated Application-Controlled File Caching, Prefetching, and Disk Scheduling", ACM Trans. on Computer Systems, vol. 14, No. 4, Nov. 1996, pp.311-343.

[7] A. Tomkins et al, "Informed Multi-Process Prefetching and Caching," Proc. SIGMETRICS'97, pp.100-114.

[8] G. R. Ganger et al, The DiskSim Simulation Environment Version 2.0 Manual, Dec. 1999.

[9] G. R. Ganger et al, "System-Oriented Evaluation of I/O Subsystem Performance," Tech. Report CSE-TR-243-95. Dept. of EECS, Univ. Of Michigan, June, 1995.