KSIAM IT series Vol.10, No.1, 95-105, 2006

버퍼와 디스크 캐시 사이의 중복 캐싱을 제한하는 효율적인 알고리즘

(An Efficient Algorithm for Restriction on Duplication Caching between Buffer and Disk Caches)

정 수 목
(Soo-Mok Jung)

요 약

기계적인 동작에 기반을 둔 디스크의 동작 특성 때문에 디스크의 속도는 처리기에 비하여 매우 느리다. 반도체기술의 발전으로 처리기의 속도 향상이 비약적으로 이루어지는 반면 디스크의 속도향상은 기계적인 동작특성 때문에 속도향상이 매우 제한적으로 이루어지고 있다. 따라서 컴퓨터시스템 전체의 성능향상을 이루이기 위하여 병목현상을 일으키는 디스크 입출력시스템의 성능을 개선할 수 있는 연구가 필수적이다. 처리기와 I/O subsystem의 속도차이를 해결하기 위한 하나의 기법으로 버퍼 캐시와 디스크 캐시를 두는 기법들이 사용되고 있다. 본 논문에서는 버퍼 캐시와 디스크 캐시사이에 디스크 블록 중복을 제한하고, 재 참조 될 가능성이 높은 디스크 블록을 오랫동안 캐시에 유지하게 함으로 캐시 hit ratio를 높여 디스크 접근을 줄이고 신속하게 처리기에 디스크 블록을 서비스하여 시스템의 성능을 개선하는 효율적인 버퍼 캐시 및 디스크 캐시 관리기법을 제안하였고 시뮬레이션을 통하여 제안된 기법의 성능을 평가하였다.

Abstract

The speed of hard disk which is based on mechanical operation is more slow than processor. The growth of processor speed is rapid by semiconductor technology, but the growth of disk speed which is based on mechanical operation is not enough. Buffer cache in main memory and disk cache in disk controller have been used in computer system to solve the speed gap between processor and I/O subsystem. In this paper, an efficient buffer cache and disk cache management scheme was proposed to restrict duplicated disk block between buffer cache and disk cache. The performance of the proposed algorithm was evaluated by simulation.

Key words : Buffer cache, Disk cache, Disk Block

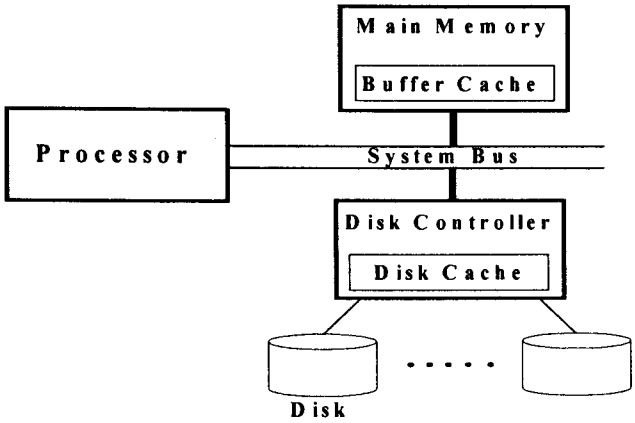
1. 서 론

디스크에 대한 접근(access)은 기계적인 동작에 의하여 이루어짐으로 속도가 대단히 느리다. 디스크 접근시간(access time)은 크게 3 부분으로 이루어진다. 데이터가 저장된 트랙(track)으로 헤드가 이동하는데 소요되는 탐색시간(seek time), 접근하고자 하는 데이터를 저장하고 있는 섹터가 헤드 밑으로 이동하는데 소요되는 시간인 회전지연시간(rotational delay time), 섹터에 저장된 데이터가 헤드 밑을 통과하는 전송시간(transfer time)이 있다. 이러한 동작은 모두 기계적인 동작을 기반으로 하고 있기 때문에 디스크의 성능향상은 매우 제한적이나 프로세스의 성능향상은 반도체 기술의 발전으로 매우 급속히 향상되고 있어 프로세스와 디스크가 주를 이루는 I/O 서브시스템과의 속도차가 갈수록 심화 되고 있다. 이러한 문제를 해결하기 위하여 DRAM으로 구성되는 디스크 캐시를 사용하고 있는데 대표적인 예가 UNIX에서 관리하는 버퍼 캐시(buffer cache)이다.[1,2] 버퍼 캐시에는 최근에 참조된 디스크 블록들이 유지되어 있기 때문에 프로세스가 디스크블록을 재사용할 때 디스크에 접근하지 않고 버퍼 캐시에서 해당 디스크블록을 read하게 되고, 버퍼 캐시 내에 해당 블록이 존재하지 않는 경우에만 디스크를 접근하여 데이터 블록을 읽고, 버퍼 캐시에 저장한 후 처리기에 서비스된다. 따라서 버퍼 캐시는 속도가 매우 느린 디스크로부터 디스크 블록을 읽는 부하를 크게 줄일 수 있다.[3]

버퍼 캐시와는 별도로 Disk controller에 디스크 캐시와 I/O처리기를 가지고 있는 시스템이 개발 되었는데 이러한 시스템에서는 처리기의 간섭 없이 처리기와 병행하여 입출력이 수행될 수 있다. 따라서 이러한 시스템에서는 Disk controller 내에 있는 디스크 캐시는 디스크 시스템의 성능을 향상 시킨다.[4,5]

주기억장치내에 버퍼 캐시와 Disk controller내에 디스크 캐시를 가지는 시스템은 그림1과 같이 구성된다. 이러한 시스템에서 특정 디스크 블록에 대하여 처리기가 입출력을 요구하는 경우 UNIX 운영체제는 주기억장치내에 유지되고 있는 버퍼 캐시를 조사하여 해당 디스크 블록의 존재유무를 검사한다. 해당 디스크 블록이 버퍼 캐시에 존재하는 경우에는 버퍼 캐시내의 디스크 블록을 사용하게 되어 디스크에 접근하지 않게 됨으로 시스템의 성능향상이 이루어진다.

요청된 디스크 블록이 버퍼 캐시에 존재하지 않는 경우에는 UNIX 운영체제가 Disk controller 에게 해당 디스크 블록을 요청하게 되고, Disk controller는 요청된 디스크 블록이 디스크 캐시에 존재하는지를 조사한다. 요청된 디스크 블록이 디스크 캐시에 존재하는 경우에는 요청된 디스크 블록을 버퍼 캐시에 저장한 후 처리기에 서비스하여 사용자 프로세스가 사용하게 되고, 디스크 캐시에 존재하지 않는 경우에는 Disk controller는 디스크 입출력을 수행하게 된다. 따라서 Disk controller 내에 있는 디스크 캐시는 디스크 블록들을 캐싱(caching)하고 있어 Disk controller에 의한 저속의 디스크 입출력 횟수를 감소시켜 시스템 성능향상에 기여하게 된다.



〔그림 1〕 Buffer Cache와 Disk Cache를 가지는 시스템

버퍼 캐시와 디스크 캐시를 동시에 가지는 시스템에서 다수의 디스크 블록들이 버퍼 캐시와 디스크 캐시에 중복되어 존재함으로 캐시 활용이 효율적이지 못하게 되어 시스템의 성능이 저하된다[6]. 이러한 문제점을 해결하기 위하여 버퍼 캐시와 디스크 캐시의 디스크 블록 중복 유지를 제한하여 시스템의 성능향상을 이루는 효율적인 캐시 관리기법들이 연구되고 있다. [7-14]

본 논문에서는 버퍼 캐시와 디스크 캐시의 디스크 블록 중복을 효과적으로 제한하여 시스템의 성능을 향상시키는 효율적인 알고리즘을 설계하고 이를 시뮬레이션을 통하여 성능을 평가하였다.

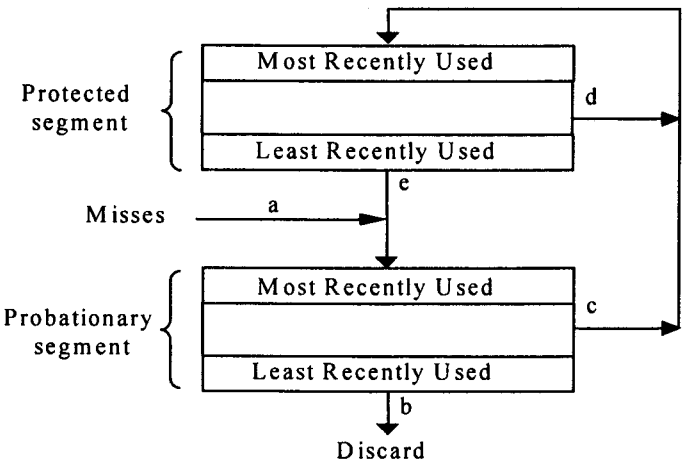
2. 관련 연구

대부분의 프로세스는 참조한 디스크 블록을 일정시간동안 반복적으로 참조하는 강한 지역성을 가진다. 사용자 프로세스에 의하여 디스크 블록이 참조되어 캐시에 저장된 후 일정한 시간 내에 두 번째 참조가 발생하면 캐시에서 제거되기 전까지 여러 번 계속적으로 사용되는 경향이 있고, 일정한 시간 내에 재 참조를 하지 않는 디스크 블록은 캐시에서 제거될 때 까지 재 참조를 하지 않는 경향이 있다.[9]

sLRU(segmented LRU)[12] 캐시 관리기법에서는 일정 시간 안에 블록에 대한 두 번째 참조가 발생하지 않으면 디스크 블록을 캐시에서 제거한다. 따라서 sLRU 캐시관리기법을 사용하면 순차참조와 같이 한번 사용된 뒤 다시는 참조되지 않는 디스크 블록이 계속 캐시로 적재되어 참조될 확률이 큰 다른 디스크 블록을 축출하는 현상을 방지하므로 캐시의 적중률을 높일 수 있다. 한번만 참조되는 디스크 블록과 재 참조되어 여러 번 반복적으로 참조될 가능성이 높은 디스크 블록을 구별하기 위하여 sLRU 캐시관리 기법에서는 그림 2와 같이 Probationary segment와 Protected segment로 나누어지고, sLRU 캐시의 세그먼트들은 단일 연결 리스트(single linked list)로 구성되어 Protected segment의 MRU end에서 임시 세그먼트 LRU end까지 연결된다. 캐시 미스(cache miss)에 의하여 데이터 블록이 캐시로 읽혀오면 임시 세그먼트의 MRU end에 추가된다.(a) 디스크 블록이 임시 세그먼트

내에 있고 캐시 히트가 발생할 경우에는 디스크 블록이 임시 세그먼트에서 제거되고 보호 세그먼트의 MRU end에 추가된다.(c) 이를 위하여 보호 세그먼트내의 LRU end에 있는 마지막 디스크 블록이 임시 세그먼트의 MRU end로 이동된다.(e)

따라서 Protected segment의 하나의 디스크 블록이 임시 세그먼트내의 하나의 디스크 블록과 교체된다. 디스크 블록이 보호 세그먼트 내에 있고 캐시 히트(cache hit)가 발생할 경우에는 디스크 블록이 보호 세그먼트의 MRU end로 이동하게 된다(d). 따라서 보호 세그먼트에 있는 디스크 블록은 적어도 한 번 이상은 접근된 것들이다. 캐시에서 하나의 데이터 블록이 버려질 경우에는 임시 세그먼트의 LRU end에 있는 하나의 데이터 블록이 버려지게(flush) 된다.(b) 임시 세그먼트와 금지 세그먼트를 구분하기 위하여 boundary pointer를 가지고 있다. 이 포인터는 일반적으로 리스트의 중간 부분을 가리키도록 set된다. 그리고 각 캐시라인은 1 bit flag를 가지게 되는데 이는 어느 세그먼트에 있는 지를 나타내는데 사용된다. 캐시 미스에 의해서 새로운 디스크 블록이 임시 세그먼트의 MRU end와 보호 세그먼트의 LRU end 사이에 삽입되고 포인터는 새로운 디스크 블록을 가리키도록 조정되고 flag는 임시 세그먼트에 있음을 나타내도록 세트된다. 이와 같이 캐시 미스에 의하여 임시 세그먼트에 새로운 디스크 블록이 삽입되는 경우 임시 세그먼트내의 디스크 블록은 새로운 디스크 블록을 위하여 캐시로부터 제거되어야 하는데 임시 세그먼트의 LRU end의 디스크 블록이 선택되어 버려지게(flush) 된다.(b) 이는 frequency-based LRU정책의 간단한 변형에 해당한다.



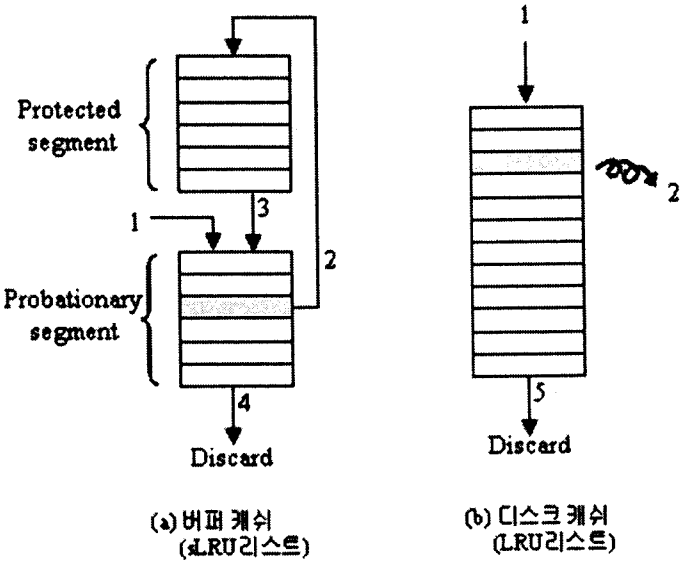
〔그림 2〕 sLRU 캐시의 논리적인 흐름도

디스크 캐시와 버퍼 캐시를 가지는 시스템에서 프로그램의 참조 패턴에 의한 디스크 블록 분배 방법이 제안되었다〔13〕. 이 기법에서는 (그림 3)과 같이 버퍼 캐시는 sLRU 캐시이고 디스크 캐시는 단순 연결리스트로 구성되어 있다. 처음으로 참조되어 캐시로 올라온 디스크 블록은 버퍼 캐시의 임시 세그먼트와 디스크 캐시로 동시에 진입한다(그림 3a와 b에서 화살표 1). 블록이 임시 세그먼트에서 머무르는 시간(Δ_t)안에 두 번째 비연관 참조가 발생하면 이 블록은 보호 세그먼트로 이동하고(그림 3a에서 화살표 2). 디스크 캐시에서는 제거된다

(그림 3b에서 화살표 2).

Δ_t동안에 비연관 참조가 발생하지 아니하면 버퍼 캐시내의 디스크 블록은 임시 세그먼트의 MRU end에서 LRU end로 이동되어 결국에는 버퍼 캐시에서 제거된다.(4) 그러나 디스크 캐시에서는 MRU end에서 LRU end로 이동하고 LRU end에 도착 후 캐시 미스에 의하여 다른 디스크 블록이 디스크에서 읽혀져 오면 LRU end에 있던 디스크 블록이 제거된다.(5) Δ_t는 임시 세그먼트의 크기와 디스크 블록의 캐시 진입율에 의해 정해진다.

이 기법에서는 첫째, 처음으로 참조된 디스크 블록이 버퍼 캐시의 임시 세그먼트 MRU end와 디스크 캐시의 MRU end에 함께 삽입된다. 이때 재 참조되어 보호 세그먼트로 진입하기 전까지, 혹은 재 참조되지 않아 임시 세그먼트에서 제거 될 때까지 디스크 블록은 버퍼 캐시와 디스크 캐시에 동시에 존재하게 되는 단점이 있다. 둘째, 디스크로부터 참조된 디스크 블록이 일정한 시간(Δ_t)안에 참조되지 않으면 향후 더 이상 참조될 가능성이 적음에도 디스크 캐시에 상당한 기간 동안 유지되는 단점이 있다.



(그림 3) sLRU기법을 적용한 버퍼 캐시와 디스크 캐시의 동작

3. 디스크 블록 중복저장을 제한하는 효율적인 알고리즘

버퍼 캐시와 디스크 캐시를 가지는 그림3과 같은 sLRU캐시 관리기법에서는 운영체제가 요청한 디스크 블록을 Disk controller가 디스크로부터 읽어 디스크 캐시에 저장하고, 버퍼 캐시에 전달되어 저장된 후 처리기에 서비스 되어 사용자 프로세스가 수행된다. 그림 3과 같은 sLRU 캐시 관리기법에서는 다수의 디스크 블록이 버퍼 캐시와 디스크 캐시 사이에 중복되어 저장된다.

본 논문에서는 사용자 프로세스에 의하여 디스크 블록이 참조되어 캐시에 저장된 후 일정한 시간 내에 두 번째 참조가 발생하면 캐시에서 제거되기 전까지 여러 번 계속적으로 사용되고 일정한 시간 내에 재 참조를 하지 않는 디스크 블록은 캐시에서 제거될 때 까지 재 참조를

하지 않는 경향이 강한 특성[9]을 이용하여 버퍼 캐시와 디스크 캐시사이 디스크 블록 중복을 최소화함으로 버퍼 캐시와 디스크 캐시를 가지는 시스템의 성능을 향상시키는 기법인 그림 4와 같은 디스크 블록 중복 캐싱을 제한하는 알고리즘을 제안하였다.
제안된 알고리즘의 동작은 다음과 같다.

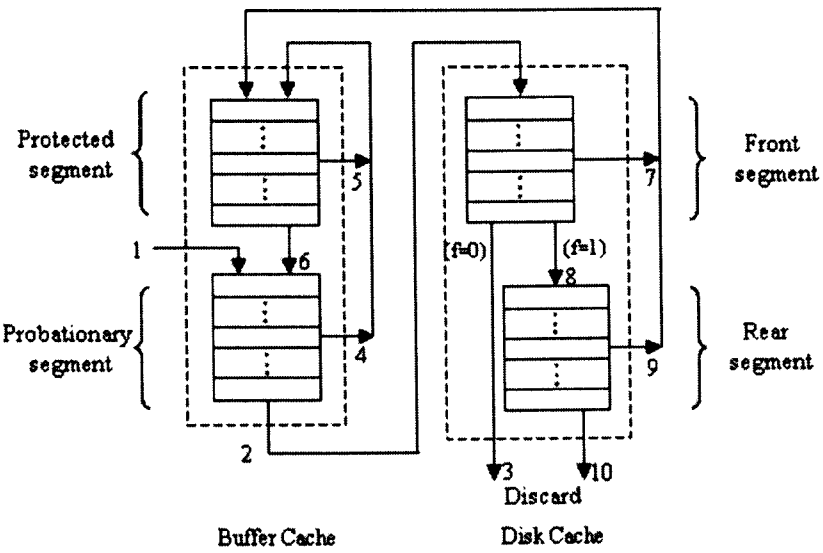
1. 프로세스가 디스크 블록을 요청
2. if(Protected segment에 존재?) 디스크 블록을 Protected segment LRU end로 이동
3. else if(Probationary segment에 존재?) {디스크 블록을 Protected segment MRU end로 이동(플래그 값을 1로 세트 f=1), Protected segment LRU end의 디스크 블록을 Probationary segment MRU end로 이동}
4. else if(Front segment에 존재?) {디스크 블록을 Protected segment MRU end로 이동(플래그 값을 1로 세트 f=1), Protected segment LRU end의 디스크 블록을 Probationary MRU end로 이동, Probationary LRU end의 디스크 블록을 Front segment의 MRU end로 이동}
5. else if(Rear segment에 존재?) {디스크 블록을 Protected segment MRU end로 이동(플래그 값을 1로 세트 f=1)
   if(Front segment LRU end의 디스크 블록의 f==0?) {Front segment LRU end의 디스크 블록을 discard, Probationary segment LRU end의 디스크 블록을 Front segment MRU end로 이동}
   else {Rear segment LRU end의 디스크 블록을 discard, Front segment LRU end의 디스크 블록을 Rear segment의 MRU end로 이동, Probationary segment LRU end의 디스크블록을 Front segment MRU end로 이동, Protected segment LRU end의 디스크 블록을 Probationary MRU end로 이동}
6. else { // 디스크 접근
7. if(Front segment LRU end의 디스크 블록 f==0?){Front segment LRU end의 디스크 블록 discard, Probationary segment LRU end의 디스크 블록을 Front segment MRU end로 이동}
8. else {Rear segment LRU end의 디스크 블록 discard, Front segment LRU end의 디스크 블록을 Rear segment MRU end로 이동, Probationary segment LRU end의 디스크 블록을 Front segment MRU end로 이동}
9. }
10. 디스크에서 읽은, 요청된 디스크 블록(f=0)을 Probationary segment MRU end에 저장
11 . goto 1.

버퍼 캐시와 디스크 캐시가 그림 4와 같이 구성되어 전역 리스트를 구성한다. 디스크 블록이 디스크에서 읽혀져 버퍼 캐시에 저장될 때는 플래그 값이 0으로 세트된다(f=0). 디스크 블록이 전역리스트에 존재하는 동안 재 참조 될 때에는 플래그 값이 1로 세트된다.

따라서 플래그 값이 1인 재 참조된 디스크 블록은 그림 4와 같이 구성된 전역 리스트에서 오랫동안 머무르면서 반복적으로 재 참조 될 때 디스크 접근 없이 처리기에 곧바로 서비스되어 프로세스가 수행되게 된다.

제안한 알고리즘에서는 요청된 디스크 블록이 존재하는 위치가 아래의 4가지 경우가 있다.

첫 번째의 경우는 버퍼 캐시의 Probationary segment에 존재하는 경우이다. 이 경우는 디스크 블록이 디스크에서 처음 읽혀져 온 후 재 참조가 되지 않은 상태(f=0)에서 Probationary segment에 유지되는 경우와 디스크 블록이 디스크에서 처음 읽혀져 온 후, 한 번 이상의 재 참조가 발생하여 버퍼 캐시의 Protected segment의 MRU end에 이동되었다가 더 이상 참조가 발생하지 않아 버퍼 캐시의 Protected segment의 LRU end 에서 probationary segment의 MRU end로 이동한 후 Probationary segment에 유지되는 경우이다. 디스크 블록이 재 참조 되어 Protected segment MRU end로 진입하면 디스크 블록에 대한 1 bit flag가 1로 세트되고 해당 디스크 블록이 discard 될 때까지 유지된다.



〔그림 4〕 버퍼 캐시와 디스크 캐시사이의 디스크 블록 중복 캐싱을 제거하는 캐시관리기법

두 번째의 경우는 버퍼 캐시 내의 Protected segment에 존재하는 경우이다. 디스크 블록이 처음 읽혀져 온 후 최소한 한번 이상의 재 참조가 발생하여 버퍼 캐시의 Protected segment의 MRU end로 이동 후 Protected segment에 유지되는 경우이다.

세 번째의 경우는 디스크 캐시의 Front segment에 존재하는 경우이다. 한번이상 재 참조 되었던 디스크블록은 Protected segment와 Probational segment를 거쳐 Front segment로 이동 후 유지되는 경우이고 이때 해당 디스크 블록의 플래그 값은 1이다. 재

참조 되지 않은 디스크 블록은 플래그 값이 0이고 Probationary segment에서 Front segment로 이동 후 유지된다.

네 번째의 경우는 디스크 캐시의 Rear segment에 유지되는 경우이다. 요청된 디스크 블록이 한번 이상의 재 참조된 후 Protected segment에서 Probationary segment, Front segment를 거쳐 Rear segment에 유지되는 경우이다.

요청된 디스크 블록은 위의 4가지 영역에 유지 되고 프로세서에 의하여 디스크 블록이 요청될 경우 제안된 알고리즘은 버퍼 캐시와 디스크 캐시의 디스크 블록 중복을 제한하기 위하여 그림 4에 표시된 10가지의 동작을 수행하게 된다.

1: 디스크에서 읽은 디스크 블록을 버퍼 캐시의 Probationary segment의 MRU end에 저장(f=0)
2: 디스크 블록이 한번 또는 그 이상 사용된 후(f=0 for f=1) Probationary segment에서 Front segment로 이동
3. 디스크 블록이 재사용되지 않아(f=0) Probationary segment에서 Front segment로 이동한 후 Discard
4: 디스크 블록이 한번 또는 그 이상 사용된 후(f=0 or f=1) 처리기에 의하여 재사용 될 때 Probationary segment에서 Protected segment MRU end로 이동. 이 때 디스크 블록의 플래그는 1로 세트됨
5: 디스크 블록이 한번 이상 재사용되어(f=1) Protected segment에 유지되는 동안 처리기에 의하여 재사용될 때 Protected segment MRU end로 이동
6: 디스크 블록이 한번 이상 재 참조된 후(f=1) Protected segment에 유지되다가 Protected segment LRU end.에서 Probationary segment MRU end로 이동
7: 디스크 블록이 한번 이상 참조된 후(f=0 or f=1) Front segment에 유지되다가 처리기에 의하여 재사용되어 Protected segment MRU end로 이동. 이 때 요청된 디스크 블록의 플래그는 1로 세트됨
8: 디스크 블록이 한번 이상 재사용된 후(f=1) Front segment에 유지되다 Rear segment MRU end로 이동
9: 디스크 블록이 한번 이상 재사용된 후(f=1) Rear segment에 유지되는 동안 재참조되어 Protected segment MRU end로 이동
10: 디스크 블록이 한번 이상 재사용된 후(f=1) Rear segment에 유지되다가 더 이상 재참조 되지 않아 discard 됨

버퍼 캐시와 디스크 캐시의 디스크 블록 중복 캐싱을 제한하는 제안된 알고리즘은 버퍼 캐시와 디스크 캐시 사이에 디스크 블록의 중복을 제거하고 재사용 가능성이 높은 디스크 블록은 오랫동안 버퍼 캐시와 디스크 캐시에 유지하고 재사용가능성이 적은 디스크 블록들은 신속히 제거하도록 알고리즘이 구성되었다. 따라서 처리기가 디스크 블록을 요청할 경우 버퍼 캐시와 디스크 캐시에서 디스크 블록을 서비스하여 디스크 접근을 최소화함으로 평균 디스크 접근지연시간을 효과적으로 감소시켜 시스템의 성능이 향상된다.

4. 시뮬레이션 결과

본 논문에서의 시뮬레이션 환경은 다음과 같았다. 버퍼 캐시와 디스크 캐시의 크기가 각 20MB, 10MB이고 Protected segment와 Probationary segment의 크기는 15MB로 같고 Front segment와 Rear segment의 크기는 5MB로 동일한 것으로 가정하였다. 처리기에 의하여 500개의 디스크 블록이 요청에 대하여 수행하였고 디스크 블록의 크기는 1MB로 가정하였다. 캐시의 교체기법으로 LRU기법을 적용하였고 디스크 블록은 변경되지 않는 것으로 가정하여 디스크 블록 교체 시 디스크 블록을 디스크에 갱신하지 않는 것으로 하였다. 디스크 블록에 대한 참조횟수는 랜덤 함수에 의하여 생성되는 1부터 20까지의 값으로 하였고 재 참조 시까지의 디스크 블록 간 간격은 랜덤함수에 의하여 생성되는 1부터 30까지의 값으로 하였다.

제안된 알고리즘의 성능을 평가하기 위하여 버퍼 캐시내의 Protected segment와 Probationary segment의 hit ratio를 구하였고 디스크 캐시내의 Front segment와 Rear segment의 hit ratio를 구하였다. Hit ratio는 요청된 디스크 블록의 수와 해당 segment 내에서 디스크 블록이 존재하여 처리기에 서비스된 비율을 표시한다. Hit ratio가 높을수록 디스크 접근 없이 디스크 블록이 처리기에 서비스되어 프로세스가 신속하게 수행될 수 있다.

〈표1〉 에서 보는 바와 같이 제안된 기법은 버퍼 캐시와 디스크 캐시사이의 디스크 블록 중복을 매우 효율적으로 제한함으로 보다 많은 디스크 블록을 유지하고 재 참조 될 가능성이 높은 블록은 전역리스트를 순환하면서 오랫동안 캐시에 유지되도록 하고 재 참조 될 가능성이 없는 디스크 블록은 신속하게 제거되도록 구성되었기 때문에 버퍼 캐시와 디스크 캐시의 디스크블록 hit ratio가 증가하게 된다. 디스크 캐시의 hit ratio가 sLRU기법에 비하여 1.5배 증가 하였는데 이는 sLRU기법은 그림3과 같이 디스크 블록이 버퍼 캐시와 디스크 캐시에 중복되어 저장되고 있어있고, 제안된 기법은 그림4와 같이 디스크 블록 중복을 제한하고 캐시를 전역리스트로 구성하여 재 참조 가능성이 높은 디스크 블록을 디스크 캐시에 오랫동안 유지될 수 있도록 알고리즘이 구성되었기 때문이다. 따라서 처리기가 디스크 블록을 요청하는 경우 디스크에 대한 접근을 최소화하고 버퍼 캐시와 디스크 캐시에 유지되고 있는 요청된 디스크 블록을 서비스하여 프로세스가 수행될 수 있다. 따라서 버퍼 캐시와 디스크 캐시를 갖는 시스템에서 제안된 효율적인 캐시관리기법은 전체시스템의 병목현상을 일으키는 디스크 접근을 최소화 하여 컴퓨터시스템의 성능을 향상 시킨다.

〈표 1〉 버퍼 캐시와 디스크 캐시에서의 적중률

		hit ratio(%)	
		sLRU	제안된 기법
버퍼 캐시	Protected segment	25%	27%
	Probationary segment	38%	38%
디스크 캐시	Front segment	8%	9%
	Rear segment		11%

5. 결 론

본 논문에서는 버퍼 캐시와 디스크 캐시를 갖는 시스템에서 처리기가 디스크 블록을 요청할 때, 신속하게 서비스하기위하여 디스크 블록 중복을 제한하고 재사용될 가능성이 높은 디스크 블록을 캐시에 더 오래 동안 유지함으로 캐시 hit ratio를 향상시켜 평균 디스크 블록 접근 지연시간을 감소시키는 버퍼 캐시 및 디스크 캐시 관리기법을 제안하였다. 제안된 기법에 대한 시뮬레이션에서 기존의 sLRU기법에 비하여 제안된 알고리즘의 디스크 캐시의 hit ratio가 1.5배 증가하였다. 이는 sLRU기법에서는 디스크 블록 중복 저장이 발생하지만 제안된 알고리즘에서는 디스크 블록 중복저장을 제한하고 재 참조될 가능성이 높은 디스크 블록은 그림4와 같이 전역 리스트 내에 순환하면서 오랫동안 유지되도록 알고리즘이 설계되었기 때문이다. 제안된 캐시관리기법은 처리기가 디스크 블록을 요청할 때 버퍼 캐시와 디스크 캐시에서 디스크 블록을 서비스하여 디스크 접근이 최소화된다. 따라서 수행되는 프로세스는 I/O 수행 시 대기시간이 최소화 되어 컴퓨터 시스템의 성능이 향상된다.

참 고 문 헌

1. C. P. Grossman, "Cache-DASD storage design for improving system performance," IBM Systems Journal, vol. 24, no. 3/4, pp. 316-334. 1985.
2. P. J. Jalics and D. R. McIntyre, "Caching and Other Disk Access Avoidance Techniques on Personal Computers," Communications of the ACM, vol. 32, no. 2, pp. 246-255, Feb. 1989.
3. M. J Bach, "The Design of the UNIX Operating System," Prentice-Hall, Englewood Cliffs, NJ. 1986.
4. Data General Corporation, "Configuring and Managing a CLARiiON Disk-Array Storage System," 1994.
5. ETRI, "TICOM-91: Hardware Functional Specification Rev. 1.0," 1989.
6. B. McNutt, "I/O subsystem configurations for ESA: New roles for processor storage," IBM Systems Journal, vol. 32, no. 2, pp. 252-264, 1993.
7. D. M. Muntz and P. Honeyman, "Multi-level Caching in Distributed File Systems," Proceedings of the 1992 Winter USENIX Conference, pp. 305-313, 1992.
8. J. T. Robinson and N. V. Devarakonda, "Data Cache Management Using Frequency-Based Replacement," Proceedings of the ACM SIGMETRICS Conference, pp. 134-142, 1990.
9. E. J. O'Neil, P. E. O'Neil, and G. Weikum, "The LRU-K Page Replacement Algorithm for Database Disk Buffering," Proceedings of the

1993 ACM SIGMOD Conference, pp. 297-306, 1993.
10. P. Cao, E. W. Felton, and K. Li, "Application-Controlled File Caching Policies." Proceedings of the Summer 1994 USENIX Conference, pp. 171-182, Jun. 1994.
11. M. D. Dahlin, R. Y. Wang. T. E. Anderson, and D. A. Patterson, "Cooperative Caching: Using Remote Client Memory to Improve File System Performance," First Symposium on Operating Systems Design and Implementation, pp. 267-280, Nov. 1994.
12. R. Karedla, J. S. Love, and B. G. Wherry, "Caching Strategies to Improve Disk System Performance," Computer, vol. 27, no. 3, pp. 38-46, March 1994.
13. D. H. Lee, S. H. Noh, S. L. Min, and Y. K. Cho, "Efficient Cache Management Schemes for Reducing Duplication Caching between Buffer and Disk Caches," Journal of KISS(A), vol. 22, no. 10, october 1995.
14. S. M. Jung, "An Efficient Cache Management Scheme in a System which has Disk Cache and Buffer Cache," Sahmyook University Journal, vol. 40, February 2006.

정 수 목(Soo-Mok Jung)
· e-mail : jungsm@syu.ac.kr
· 1984년 경북대학교 전자공학과 학사졸업
· 1986년 경북대학교 전자공학과 석사 졸업
· 2002년 고려대학교 컴퓨터학과 박사졸업
· 2004년 Virginia Tech. 연구교수
· 1998년 ~ 현재 삼육대학교 컴퓨터학부 부교수
· 관심분야 : 멀티미디어, 컴퓨터시스템