연속미디어 파일 시스템의 버퍼 캐시에서 데이터 참조 유형의 고려

조 경 운[†] · 류 연 승[††] · 고 건[†††]

요 약

연속미디어 파일을 위한 버퍼 캐시 기법들은 연속 미디어의 순차적 접근만을 고려하고 반복 참조는 고려하지 않았다. 그러나, 외국어 영상 학습의 경우 사용자가 어떤 장면을 반복 구간으로 설정하면 자동으로 수회 반복 상영하는 기능이 있을 수 있다. 본 논문에서는 순차 참조와 반복 참조가 혼재하는 연속미디어 파일 시스템을 위한 새로운 버퍼 캐시 기법을 제안한다. 제안한 기법은 파일의 참조 유형을 탐지하고 파일 별로 적절한 교체 정책을 적용하여 버퍼 캐시 적중률을 높인다.

Considering Data Reference Pattern in Buffer Cache for Continuous Media File System

KyungWoon Cho[†] · YeonSeung Ryu[††] · Kern Koh[†††]

ABSTRACT

Previous buffer cache schemes for continuous media file system only exploited the sequentiality of continuous media accesses and didn't consider looping references. However, in some video applications like foreign language learning, users mark the scene as loop area and then application automatically playbacks the scene several times. In this paper, we propose a novel buffer cache scheme for continuous media file system that sequential and looping references exist together. Proposed scheme increases the cache hit ratio by detecting reference pattern of files and appling an appropriate replacement policy to each file.

키워드 : 연속 미디어 파일 시스템(Continuous Media File System), 참조 유형(Reference Pattern), 버퍼 캐시(Buffer Cache), 캐시 적중률(Cache Hit Ratio)

1. 서 론

1.1 연구동기

컴퓨터와 통신 기술의 발전은 초고속 인터넷을 통하여 비디오나 오디오와 같은 연속 미디어를 전송하는 응용분야를 가능하게 하고 있다. 비디오나 오디오 같은 데이터를 저장하고 서비스하는 연속미디어 파일 시스템은 처리해야 하는 데이터 크기가 크고 비교적 오랜 시간 동안 연속적으로 데이터를 전송해야 하는 특징을 가지고 있어서 일반 파일 시스템과 다르다[1-3].

파일 시스템에서의 버퍼 캐시는 디스크로부터 읽은 데이터를 임시로 저장시키고 미래의 요청에 대응하게 함으로써 파일 시스템의 성능을 향상시키는 목적을 가진다. 일반적으로 디스크의 접근 시간보다 메모리의 접근 시간이 더 빠르기 때문에 데이터를 디스크 대신에 메모리에서 읽을 수 있다면 데이터에 대한 응답시간을 빠르게 할 수 있다. 하지만 버퍼 공간이 제한적이기 때문에 제한된 버퍼를 이용하여 캐시 적중률(hit ratio)을 높이려는 버퍼 캐시 기법들이 연구되어 졌다[4-11].

연속미디어 파일 서버의 경우도 제한된 자원을 사용하여 많은 스트림을 서비스하기 위해서는 효율적인 버퍼 캐시 관리를 통하여 디스크 입출력을 줄이는 것이 매우 중요하다. 특히, 최근 메모리 가격이 급격히 떨어지고 용량은 커지고 있으므로 메모리를 효율적으로 사용한다면 경제적일 수 있다.

연속미디어 파일 시스템을 위한 여러 버퍼 캐시 기법들이 연구되어 왔다[12-14]. 이러한 버퍼 캐시 기법들은 데이터의 순차적인 접근에 착안한 기법들이 대부분이다. 예를 들어, 어떤 스트림이 어떤 비디오 파일을 참조하게 되면 파일의 처음부터 끝까지 순차적으로 참조한다. 따라서, 한번 참조한 데이터를 다시 참조하지 않으며, 대신에 이 스트림보다 늦게 같은 비디오 파일을 요청한 다른 스트림이 참조

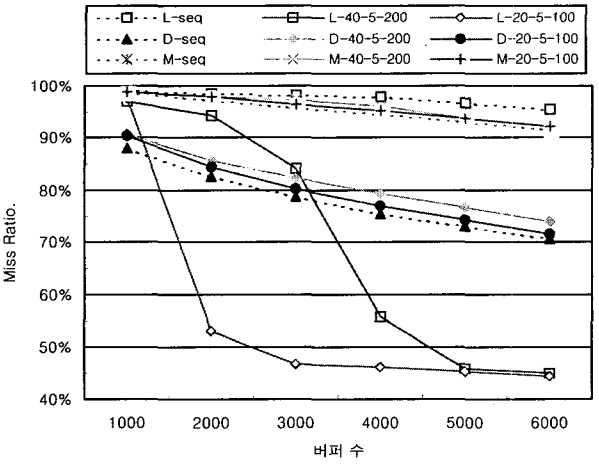
† 정 회 원 : 서울대학교 대학원 전기컴퓨터공학부
†† 정 회 원 : 한림대학교 정보통신공학부 교수
††† 정 회 원 : 서울대학교 전기컴퓨터공학부 교수
논문접수 : 2001년 10월 16일, 심사완료 : 2002년 5월 15일

할 수는 있다. 연구된 기법들은 이러한 성질을 이용하여 선행 스트림이 참조한 데이터를 캐시에 저장시키고 후행 스트림이 참조하게 하는 방식으로서 가정하고 있는 순차 참조 방식에서는 매우 좋은 캐시 적중률을 보인다.

하지만, 한 스트림이 하나의 연속미디어 파일 내에서 데이터를 반복 참조하는 경우를 고려한 연구는 거의 없다. 연속미디어 파일 서버에서 데이터가 반복 참조되는 경우로는 인터넷을 통하여 비디오와 오디오를 사용하여 원격 교육을 하는 예를 들 수 있다. 특히, 외국어 교육의 경우, 원격지의 학생이 대화 장면을 반복 학습하기 위해서 반복 구간의 시작과 끝을 지정하면 응용 프로그램이 반복 구간을 자동으로 일정 횟수를 반복하는 예가 있다.

기존의 연속미디어 파일 시스템의 버퍼 캐시 기법들은 순차적인 데이터 참조 특성을 주로 고려하였고 반복 참조가 있는 경우는 다루지 않았다. 본 논문에서는 반복 참조가 있는 연속미디어 파일 시스템에서의 기존의 버퍼 캐시 정책의 성능을 알아보고 성능 향상을 위한 새 방법을 연구하고자 하였다.

(그림 1)은 연속 미디어 파일에 대해 반복 참조가 있을 때 대표적인 버퍼 캐시 기법인 LRU, DISTANCE, MRU를 실험하여 구한 캐시 미적중률(miss ratio)이다. 자세한 실험 방법과 결과는 3장에서 다룬다. 범례에서 L은 LRU, D는 DISTANCE, M은 MRU를 나타내고 seq는 순차 참조만 있는 경우이다. 숫자 40-5-200은 파일에 대한 루프 파라미터로서 루프의 평균 길이가 40초, 루프의 평균 반복 횟수 5회, 평균 루프간 간격이 200초임을 의미한다. 이것은 루프가 평균 200초마다 발생하고 루프의 길이가 평균 40초이고 평균 5회 반복되는 것을 뜻한다.



(그림 1) 반복 참조가 있는 경우 LRU, DISTANCE, MRU 기법의 성능

순차 참조만 있는 경우에는 DISTANCE의 성능이 제일 우수하며, LRU와 MRU는 90%이상의 미적중률을 보여 매우 좋지 않다. 그러나, 반복 참조가 있는 경우, LRU는 버퍼 수가 어느정도 많아지면 DISTANCE 보다 좋은 성능을 보이고 있다. (그림 1)을 보면, 반복 참조가 있는 경우 루프 파라미터(루프 길이, 루프간 간격, 루프 횟수)와 버퍼 수에 따라 LRU와 DISTANCE의 성능이 영향받고 있음을 알 수 있다.

1.2 관련 연구

연속미디어 파일 시스템에서 버퍼에 관한 많은 연구가 있다. 이 연구에는 디스크 스케줄링과 연관되어 요구 버퍼를 계산하고 예약하는 연구[14-19], 버퍼 사용률을 높이거나 요구 버퍼 크기를 줄이는 연구[20, 21], 데이터 선반입(prefetch) 연구[22], 버퍼 캐시의 교체 기법에 관한 연구[12-14] 등이 있다. 본 논문에서는 버퍼 캐시의 교체 기법에 관한 연구를 다루며 여기서는 이와 관련한 기존 연구를 살펴본다.

버퍼 캐시의 성능은 버퍼 교체 정책에 달려있다. 최적의 버퍼 교체 기법은 버퍼 용량이 정해져있을 때 캐시 적중률을 최대화할 수 있도록 데이터를 교체하는 기법이다. 최적의 교체 기법은 가장 오랜 시간동안 참조하지 않을 데이터를 교체하는 방법이다. 하지만 미래의 참조에 대한 지식이 필요하기 때문에 현실적으로 구현이 어렵다. 따라서, 최적의 교체 기법에 근사하는 교체 기법들이 연구되어 왔다. 전통적으로 이러한 연구들로는 LRU(Least Recently Used)나 MRU (Most Recently Used), LFU(Least Frequently Used) 등이 대표적이다[8-11]. LRU는 가장 오래동안 참조되지 않은 데이터가 앞으로도 오랜 시간 동안 참조되지 않을 것이라는 정책을 사용하여 버퍼를 교체하는 방법이다. MRU는 최근에 참조한 데이터가, LFU는 참조 횟수가 적은 데이터가 앞으로 오랜 시간 동안 참조되지 않을 것이라는 정책을 사용한다.

데이터에 대한 참조 유형에는 크게 순차(sequential) 참조, 반복(looping) 참조, 시간 국부성(temporal localized) 참조, 확률적(probabilistic) 참조가 있다. 순차 참조는 모든 데이터가 순차적으로 참조되며 한번 참조한 데이터가 다시 참조하지 않는 참조 유형을 말한다. 반복 참조는 정기적인 간격으로 반복해서 데이터가 참조되는 참조 유형을 말한다. 시간 국부성 참조는 최근에 참조되는 블록이 가까운 미래에 참조되는 참조 유형을 말한다. 시간 국부성 참조의 경우 LRU가 최적(optimal)에 가까운 버퍼 교체 정책으로 알려져 있다[4]. 확률적 참조는 독립 참조 모델(Independence Reference Model)에 의해 데이터가 참조되는 참조 유형이다. 이때, 각 블록 i는 확률 p_i를 가지고 확률에 의해 독립적으로 참조된다.

최근에는 참조 유형(pattern)의 정보를 이용하여 참조 유형에 따라서 버퍼 교체 기법을 적용하는 방법도 연구되었다[23-25]. 참조 유형을 알아내기 위한 방법으로는 사용자 수준에서 참조 유형의 정보를 제공하는 방법[23]과 수행 중에 참조 유형을 발견하는 방법이 제안되었다[24].

연속미디어 파일 시스템을 위해서는 인터벌 캐싱(Interval Caching) 기법[13, 14]과 BASIC/DISTANCE 기법[12]등이 연구되었다. 이 두 기법은 연속미디어 파일의 데이터들이 순차적으로 접근되는 특성을 이용하고 있다.

BASIC 기법은 현재 서비스 중인 스트림에 의해 가장 오

랜 시간동안 참조되지 않을 블록을 가진 버퍼를 교체하는 기법이다. 이 기법은 스트림마다 상태 정보를 유지하고 매 서비스 사이클마다 스트림의 상태 정보를 이용하여 모든 버퍼에 대하여 다음에 참조될 시간을 계산한다. 이 방법은 최적의 버퍼 교체 기법에 근사하는 성능을 보여주지만, 버퍼마다 미래의 참조 시간 값을 서비스 사이클마다 계산해야 하므로 실행 오버헤드가 큰 문제가 있다. 그래서 구현이 간단한 DISTANCE 기법을 고안하였다. DISTANCE 기법에서는 어떤 한 스트림의 distance를 후행 스트림간의 데이터 블록 개수로 정의한다. 만일 후행 스트림이 없다면 distance는 최대값을 가진다. distance 값은 새로운 스트림이 도착하거나 서비스 중인 스트림이 끝날 때에 계산한다. DISTANCE 기법은 스트림이 사용하고 반환한 버퍼를 스트림 별로 MRU 리스트로서 관리하고, 버퍼를 교체할 때 distance가 가장 큰 스트림의 MRU 리스트에서 버퍼를 선정한다. 이 방법은 distance가 작은 스트림이 참조한 블록을 버퍼에 오래 유지시킴으로서 후행 스트림이 참조할 가능성을 높여준다.

Dan 등은 비디오 파일 서버를 위한 캐시 방법으로 인터벌 캐싱 기법을 제안하였다[13]. 인터벌 캐싱은 선행 스트림이 읽은 데이터를 버퍼에 유지시켜 후행 스트림이 사용하게 하는 기법이다. 인터벌이란 같은 비디오 객체를 참조하면서 연속된 두 스트림 간의 데이터 블록으로 정의된다. 연속적인 두 스트림 간의 인터벌이 캐시되었다면 후행 스트림은 선행 스트림이 읽은 데이터를 버퍼에서 읽게 된다. 인터벌 캐싱 기법은 한정된 버퍼 공간에 보다 많은 수의 연속 스트림을 유지시켜 캐시 적중률을 최대로 하는 정책을 사용한다. 이를 위해 인터벌이 작은 스트림부터 캐시하고, 새로 도착한 스트림에 의해 만들어진 인터벌이 현재 캐시되어 있는 인터벌보다 작을 때 이를 캐시하기 위한 버퍼 공간이 부족하다면 인터벌이 가장 긴 것을 캐시에서 제거하는 교체기법을 사용한다.

DISTANCE 기법과 인터벌 캐싱 기법은 다음과 같은 점에서 매우 유사하다. 첫째, 멀티미디어 파일의 참조는 순차적이라고 가정한다. 둘째, 같은 멀티미디어 파일을 참조하는 연속적인 두 스트림 간의 데이터 블록을 캐시한다. 세째, 두 스트림간의 인터벌(또는 distance)이 작은 것의 블록을 버퍼에 유지시킴으로서 버퍼 적중률을 높인다.

1.3 공헌 내용

연속 미디어 파일 시스템의 기존 버퍼 캐시 기법들은 파일의 참조가 순차적이라는 특징을 이용하고 있다. 그러나, 이런 기법들은 반복 참조가 있을 경우에는 적합하지 않음을 (그림 1)에서 알 수 있다. 따라서, 순차 참조와 반복 참조가 혼재하는 파일 시스템에서는 새로운 버퍼 캐시 기법이 연구되어야 한다. 본 연구에서는 연속 미디어 파일 별로 참조 유형이 다른 응용 분야에서 캐시 성능을 향상시키는 버퍼 캐시 기법을 연구하였다. 제안한 기법은 파일 별로 참조 유형을 탐지하고 참조 유형에 따라 적합한 버퍼 캐시 기법을 적용하여 캐시 적중률을 높일 수 있다.

본 논문의 구성은 다음과 같다. 2장에서 새로운 버퍼 관리 기법인 ABM을 설명한다. 3장에서는 시뮬레이션을 통해 기존 버퍼 교체 기법과 ABM 기법의 성능을 알아본다. 결론과 향후 연구를 4장에서 기술한다.

2. ABM(Adaptive Buffer replacement for continuous Media file system) 기법

2.1 시스템 모델

연속미디어 파일 서버는 다수의 연속미디어 파일을 저장하고 있으며 고객의 요청에 따라 연속미디어 데이터를 고객에게 전송한다. 고객이 파일의 시청을 요청하면 파일의 첫 번째 블록부터 순차적으로 참조하게 된다. 또한, 고객은 파일을 시청하는 도중에 반복 구간을 설정하고 구간을 여러번 반복 시청할 수 있다. 고객이 반복 시청을 요청하므로 시스템은 파일의 반복 참조에 대한 정보를 알 수 있다.

N개의 블록으로 되어 있는 파일은 블록 1부터 블록 N까지 블록 번호가 있다. 고객이 참조하는 파일의 블록 번호들의 나열을 참조 열(reference string)이라 한다. 예를 들어, 어떤 고객이 파일을 참조하면서 만드는 참조 열이 {1, 2, **3, 4, 3, 4**, 5, 6, **7, 8, 7, 8, 7, 8**, 9, 10, …}이라 하자. 이 참조 열에서 블록 {3, 4}는 2회 반복 참조되었고, 블록 {7, 8}은 3회 반복 참조되었다. 반복 참조된 블록 구간을 루프라고 한다. 루프의 블록 개수를 루프 길이라고 하고 루프를 반복 참조한 횟수를 루프의 횟수라고 한다. 루프 {3, 4}는 루프 길이가 2이고 루프 횟수는 2이다. 루프와 루프 사이에 순차 구간이 있으면 이 순차 구간의 블록 개수를 루프간 간격이라고 한다. 참조 열의 예에서 두 루프간 간격은 2이다. (루프 길이, 루프 횟수, 루프간 간격)을 루프 파라미터라고 부르겠다.

서버는 고객이 요청하는 상영률을 보장해야 한다. 고객 i는 상영률 r_i로 데이터를 요청한다고 하자. 본 논문에서는 한 서비스 사이클 T 동안 각 고객들이 요청하는 $r_i * T$ 데이터를 디스크에서 읽어 버퍼에 저장하면 다음 사이클에서 각 고객에게 전송된다고 가정한다[1, 2]. 이를 위해서 이중 버퍼(double buffer)를 사용한다. 버퍼 공간은 n_B개의 버퍼로 구성된다. 각 버퍼의 크기는 d로서 같은 크기이다. 디스크에서 읽어오는 블록의 크기는 버퍼의 크기와 같다고 가정한다.

시스템은 서비스 중인 모든 고객들의 데이터를 저장할 버퍼 공간이 있어야 한다. 새 고객이 도착하면 고객에게 할당할 버퍼가 충분한 지를 검사하는 수용 제어를 수행한다. 서비스 중인 고객의 수를 M이라 하자. 모든 고객들이 요청하는 데이터를 버퍼에 저장할 수 있어야 하고 이중 버퍼를 사용하므로,

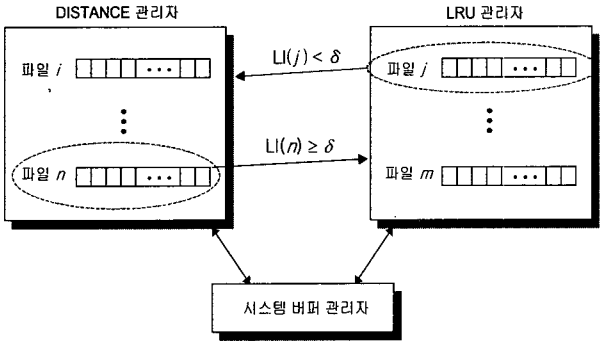
$$2T\sum_{i=1}^{M} r_i \le n_B \cdot d \qquad (1)$$

가 만족되어야 한다. 식 (1)로부터, M 고객들을 서비스하기 위해서 다음과 같은 버퍼가 있어야 한다.

$$\lceil 2\frac{T}{d}\sum_{i=1}^{M} r_i \rceil \quad (2)$$

서비스 사이클마다 서비스 중인 각 고객을 위해 다음 사이클에 필요한 블록을 버퍼에 읽어온다. 이때, 필요한 블록이 이미 버퍼에 있는 지를 검사하고 버퍼에 있다면 그 버퍼를 사용하고 버퍼에 없다면 디스크에서 해당 블록을 읽어오게 된다. 디스크에서 읽어온 블록을 저장할 가용(free) 버퍼를 선택하기 위해서 버퍼 교체를 수행한다.

(그림 2)는 제안한 ABM의 구조이다.



(그림 2) ABM의 구조

ABM은 시스템 버퍼 관리자, DISTANCE 관리자, LRU 관리자로 구성되어 있다. 시스템 버퍼 관리자는 시스템의 전체 버퍼를 관리한다. 시스템 버퍼 관리자는 버퍼 자원에 대한 수용 제어(admission control)를 담당한다. 즉, 새로 도착하는 고객에게 버퍼를 할당할 수 있는지를 결정한다. 시스템 버퍼 관리자는 파일 별로 반복 참조 정보, 참조하고 있는 고객의 수, 파일의 블록을 가진 버퍼 수 등의 정보를 관리한다. 이 정보들을 이용하여 파일 별로 버퍼 교체 정책을 동적으로 결정한다. DISTANCE 관리자는 시스템 버퍼 관리자에 의해 자신에게 속하게 된 파일들의 버퍼를 DISTANCE 정책으로 관리한다. DISTANCE 알고리즘은 [12]에서 제안한 방법과 같다. DISTANCE는 연속미디어 파일의 순차적 참조에 적합한 버퍼 교체 정책으로 알려져 있다. LRU 관리자는 시스템 버퍼 관리자에 의해 자신에게 속한 파일들의 버퍼를 LRU 정책으로 관리한다. LRU는 시간 국부성 참조 유형에서 최적에 가까운 버퍼 교체 정책으로 알려져 있다[4].

2.2 버퍼 관리 방법

파일이 처음에 참조되기 시작하면 DISTANCE 정책으로 관리한다. 시스템 버퍼 관리자는 정해진 주기마다 파일의 루프 참조 지수(LI : Looping reference Indicator)를 계산하여 파일별로 적절한 버퍼 정책을 적용한다. 시스템 버퍼 관리자는 파일의 버퍼 정책을 결정하기 위해서 버퍼 정책 제어값 (buffer policy control value), δ를 가지고 있다. 파일의 LI가 δ보다 같거나 크게되면 그 파일은 LRU로 관리한다. 역으로 파일의 LI가 δ보다 작게되면 DISTANCE로 관리한다. δ가 작을수록 더 많은 파일들이 LRU로 관리되고, 역으로 δ가 클수록 더 많은 파일들이 DISTANCE로 관리된다.

이제 파일의 LI를 정의한다. 제안한 LI는 다음과 같은 성질을 이용한다. 이 성질들은 3장의 실험 결과에서도 볼 수 있다. 첫째, 파일에 루프가 자주 생긴다면 파일을 LRU로 관리하는 것이 DISTANCE로 관리하는 것보다 캐시 적중률이 좋아진다. 둘째, 파일의 루프들이 루프 길이가 짧을수록 파일을 LRU로 관리하는 것이 캐시 적중률이 좋아진다. 셋째, 버퍼가 충분하지 않다면 루프가 많은 파일을 LRU로 관리해도 캐시 적중률이 좋아지지 않을 수 있다.

고객 s가 파일 i를 참조하는 중에 시간 t에서 참조한 파일의 논리적 블록 번호를 $N_i(R_t(s))$라고 하자. 시간 t에서 파일 i를 참조 중인 고객들의 집합을 $S_t(i)$로 나타내고 집합의 구성요소의 개수를 $|S_t(i)|$로 나타내자. 시간 t에서 파일 i를 순차 참조하는 고객의 집합을 $SR_t(i)=\{s|N_i(R_t(s))=N_i(R_{t-1}(s))+1, s\in S_t(i)\}$로 정의한다. 시간 t에서 파일 i를 순차 참조하지 않는 고객들, 즉 반복 참조를 하는 고객들의 집합을 $SR_t(i)^c$으로 정의한다.

버퍼 공간 중에서 시간 t에서 파일 i의 블록을 가지고 있는 버퍼의 수를 $B_t(i)$로 나타내자. 고객이 파일을 반복 참조할 때 반복 참조를 요청하므로 루프 길이를 알 수 있다. 시간 t에서 $s\in SR_t(i)^c$일 때 현재 루프 길이를 $LT_t(s)$라고 하자. 파일 i의 루프 중에서 루프 길이 즉, 루프 내 블록의 수가 $s\in SR_t(i)^c$인 고객 s에 할당된 평균 버퍼 수보다 작다면 이 루프를 유효 루프 (effective loop)라고 정의한다. $ELR_t(i)=s|LT_t(s)\le\frac{B_t(i)}{|S_t(i)|}, s\in SR_t(i)^c$를 시간 t에서 유효 루프를 가진 고객들의 집합이라 하자. 이것은 루프로 인하여 참조되는 블록들이 버퍼 공간에 존재할 가능성이 높은 고객들의 집합이다.

파일 i의 LI를 다음과 같이 정의한다.

$$LI_t(i)=\frac{\sum_{i=t-\theta}^{t}|ELR_t(i)|}{\sum_{i=t-\theta}^{t}|SR_t(i)|+\sum_{i=t-\theta}^{t}|ELR_t(i)|} \quad (3)$$

여기서 θ는 LI를 계산하는 주기의 값이다. 이 값은 파일들의 버퍼 정책의 결정 주기이다.

시스템 버퍼 관리자는 DISTANCE 관리자와 LRU 관리자에게 할당할 버퍼양을 제어한다. 처음에 시스템의 버퍼는 모두 DISTANCE 관리자에 할당된다. 이후에 DISTANCE 관리자로 관리되던 파일의 정책이 LRU로 전환되면 그 파일의 블록을 가진 버퍼는 LRU 관리자에 할당된다. 역으로 LRU 관리자에 의해 관리되던 파일의 정책이 DISTANCE

로 전환되면 그 파일의 블록을 가진 버퍼는 DISTANCE 관리자에 할당된다.

DISTANCE와 LRU 관리자는 시스템 버퍼 관리자에게 버퍼를 요청할 수 있다. 어느 한쪽 관리자 A로부터 버퍼를 요청 받은 시스템 버퍼 관리자는 상대방 관리자 B에게 버퍼를 요청할 수 있는 지를 검사한다. 시스템 버퍼 관리자는 관리자별 버퍼양의 결정을 위해서 각 관리자에게 할당되어 있는 버퍼 수 대비 그 관리자에 의해 서비스 받고 있는 고객의 수의 비율을 이용한다. 기본 아이디어는 서비스하는 고객 수가 많다면 버퍼를 더 할당하여 버퍼에서 서비스될 고객의 수를 늘려서 버퍼 적중률을 높이는 것이다. 버퍼를 요청한 관리자 A의 고객의 수가 할당받은 버퍼에 비해 많다면 상대방 관리자 B에게 버퍼를 요청한다. 버퍼를 요청 받은 상대방 관리자 B는 자신의 정책에 의해 교체할 버퍼를 선택하여 시스템 버퍼 관리자에게 전달하고, 시스템 버퍼 관리자는 그 버퍼를 관리자 A에게 전달한다.

DISTANCE와 LRU 관리자가 버퍼를 요청하는 시기는 주기적으로 수행할 수 있으며 본 논문에서는 매 서비스 주기마다 요청한다고 가정한다. 수행 주기에 관한 자세한 논의는 다루지 않는다.

ABM의 성능은 버퍼 정책 제어값 δ, LI 계산 주기, 관리자 별 버퍼 할당 정책, 버퍼 재할당 주기 등에 좌우된다. 본 논문은 이 중에서 LI와 δ를 이용한 버퍼 성능의 제어를 연구하며, LI 계산 주기, 향상된 관리자 별 버퍼 할당 정책, 버퍼 할당 주기에 관한 연구는 향후 과제로 남겨둔다.

3. 실 험

3.1 실험 방법

본 연구에서 가정하는 시스템의 실험을 위해 인공적으로 미디어 파일에 대한 참조 열을 만들었다. 파일에 대한 참조 열은 순차 참조와 반복 참조에 의해 만들어지는 블록 열이다.

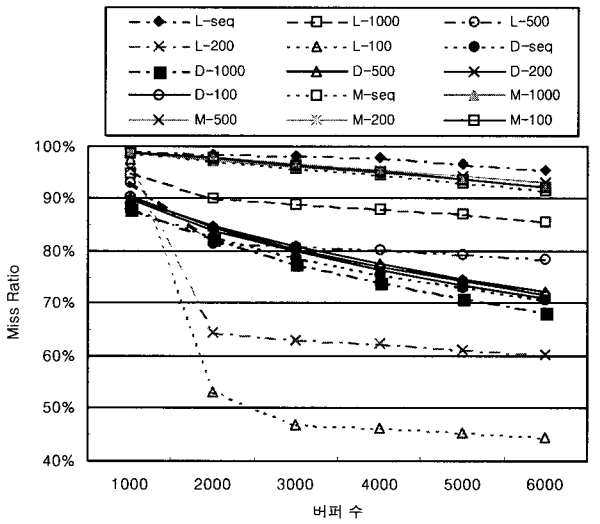
고객은 임의(random)로 도착하며, 고객의 도착 간격(interarrival time)은 지수 분포를 따른다. 고객의 평균 도착 간격은 100초를 사용하였다. 고객이 도착하면 파일을 선택하고 그 파일에 대한 참조 열을 생성한다. 참조 열을 생성할 때 시뮬레이터에 정의된 루프 파라미터, 즉 루프 길이, 루프 횟수, 루프간 간격의 값을 이용하여 참조 열을 만든다. 서비스 라운드마다 각 고객은 파일의 참조 열을 이용하여 파일의 블록을 접근하게 된다. 서비스 라운드는 1초이고 서비스 라운드마다 한 블록을 참조하는 것으로 가정하였다. 파일의 개수는 총 20개이며 모든 파일의 길이는 60분으로 하였다. 고객의 파일 선택은 파일의 인기도를 반영하기 위하여 인수가 0.271인 Zipf 분포함수를 사용하였다[12, 13, 26]. 각 실험은 8시간동안 수행하였으며 블록 요청 수와 캐시 적중 수를 측정하였다.

3.2 LRU, DISTANCE, MRU의 성능 비교

파일에 대한 반복 참조가 있을 때 LRU, DISTANCE 및 MRU의 성능을 알아보았다. 먼저, 루프간 간격의 영향을 실험하였다. 실험에서 사용한 시스템 파라미터는 <표 1>과 같다.

〈표 1〉 루프간 간격을 변화시킨 실험의 파라미터

시스템 파라미터	값
평균 루프 길이	20 초
루프 횟수	5회
평균 루프간 간격	100, 200, 500, 1,000초
버퍼 수	1,000~6,000

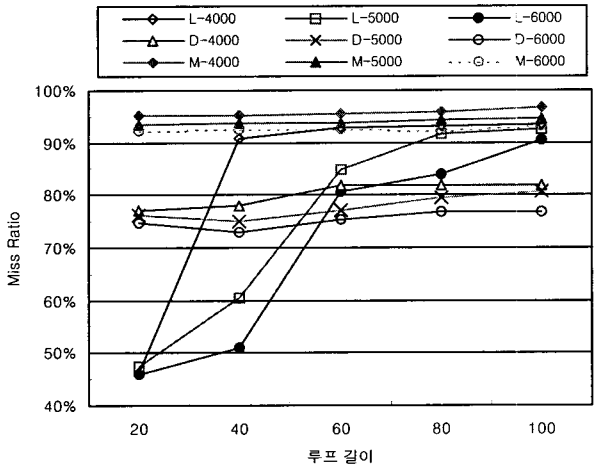


(그림 3) 루프간 간격에 대한 버퍼 정책들의 성능

실험 결과는 (그림 3)과 같다. (그림 3)의 범례에서 L은 LRU, D는 DISTANCE, M은 MRU를 뜻하며, 숫자는 루프간 간격을 의미한다. 이 실험을 통해 다음과 같은 몇 가지 사항을 알 수 있다. 첫째, 버퍼의 수가 증가할수록 캐시 성능이 좋아진다. 둘째, 순차 참조만 있는 경우 LRU와 MRU 정책의 성능은 매우 좋지 않다. 두 정책의 캐시 미적중률(miss ratio)이 90~99%에 이르고 있다. DISTANCE가 LRU와 MRU보다 우수하다. 예로써 버퍼 수가 6,000일 때 LRU와 MRU의 캐시 미적중률은 90% 이상이고 DISTANCE는 71%이므로 약 20% 이상 성능이 좋은 것을 알 수 있다. 따라서, 연속미디어 파일의 순차 참조에는 DISTANCE 기법이 좋다고 볼 수 있다. 셋째, 반복 참조와 순차 참조가 혼재되어 있는 경우에 MRU의 성능은 순차 참조만 있는 경우와 비슷하다. DISTANCE의 경우, 루프간 간격에 따라 약 7~8%의 차이를 보인다. LRU의 경우, 루프간 간격이 작아질수록 성능이 좋아진다. 버퍼 수가 6,000일 때 루프간 간격이 500과 1,000이면 DISTANCE가 LRU보다 우수하지만 루프간 간격이 200과 100일때는 LRU가 더 우수하다. 루프간 간격이 100초일 때 LRU가 DISTANCE보다 약 30%가 우수한

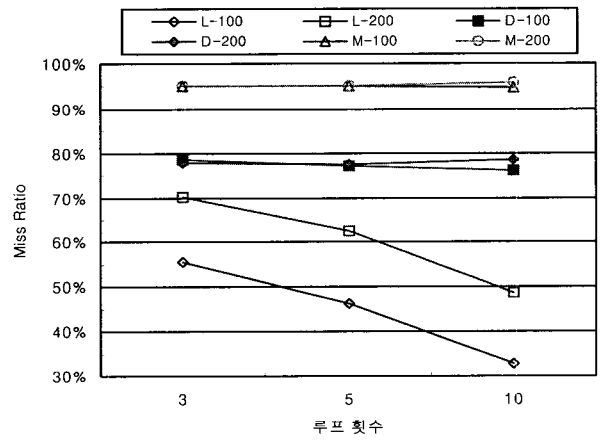
것을 알 수 있다. 이와 같이 루프가 자주 발생할수록 LRU가 더 우수한 것은 반복 참조와 순차 참조가 혼재하면서 시간 국부성 참조 효과가 생기기 때문으로 볼 수 있다.

다음으로 평균 루프 길이가 미치는 영향을 알아보았다. 평균 루프간 간격은 100초를 사용하고 루프 횟수는 5회를 사용하였다. 버퍼 수는 4,000, 5,000, 6,000개이다. (그림 4)는 평균 루프 길이를 20, 40, 60, 80, 100초로 변화를 주어 측정한 캐시 미적중률이다. (그림 4)의 범례에서 숫자는 버퍼 수이다.



(그림 4) 루프 길이에 대한 버퍼 정책들의 성능

LRU의 경우 루프 길이가 길어질수록 캐시 미적중률이 증가하고 있다. 이것은 루프 길이가 길어지면 루프 내에서 순차 참조의 효과가 커지게 되며 또한 버퍼가 긴 루프를 포함하지 못하기 때문이다. 버퍼 수가 커질수록 긴 루프를 포함할 가능성이 높아져 캐시 미적중률이 낮아지게 된다. DISTANCE의 경우는 루프 길이에 큰 영향을 받지 않고 있다. (그림 4)에서 버퍼 수가 6,000개일 때, 루프 길이가 60보다 크면 DISTANCE가 우수하며, 루프 길이가 60보다 작으면 LRU가 우수한 것을 알 수 있다. 또, 루프 길이가 40일 때 버퍼 수가 4,000이면 DISTANCE가 LRU보다 우수하지만 버퍼 수가 5,000과 6,000이면 LRU가 더 우수하다. 이것은 버퍼 수가 늘어나면서 루프의 블록들을 버퍼에서 포함하게 되었기 때문이다.



(그림 5) 루프 횟수에 대한 버퍼 정책들의 성능

다음으로 루프 횟수가 미치는 영향을 알아보았다. (그림 5)는 루프 횟수를 3, 5, 10회로 변화를 주어 측정한 캐시 미적중률이다. 평균 루프 길이는 20초이고 버퍼 수는 4,000개이다. 그림의 범례에서 숫자는 평균 루프간 간격이다.

(그림 5)에서 LRU는 루프 횟수가 많아지면 캐시 성능이 좋아지는 것을 알 수 있다.

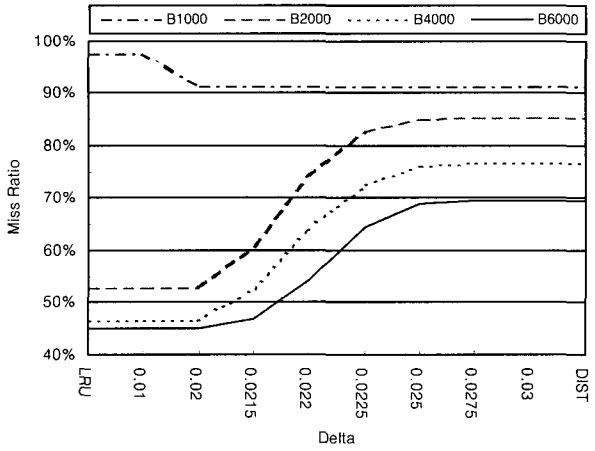
3.3 ABM의 성능

파일마다 참조 유형을 다르게 하고 ABM의 성능을 알아보았다. 20개의 파일을 1부터 20까지 번호를 붙이고 파일마다 평균 루프 도착률을 다르게 하였다. 실험에서 사용한 파일의 참조 유형이 <표 2>에 나와있다. 모든 참조 유형에서 평균 루프 길이는 20초, 루프 횟수는 5회를 사용하였다. 파일의 루프 참조 인수 LI는 20초마다 계산하여 파일의 버퍼 정책을 결정하였고, 버퍼 정책 제어값 δ를 변화시키면서 캐시 성능을 측정하였다.

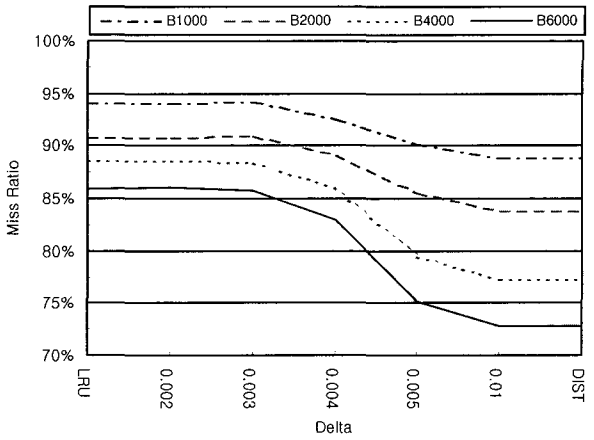
(그림 6)은 참조 유형 별로 ABM의 버퍼 성능 결과를 보이고 있다. 여기에서 ABM은 버퍼 정책 제어값 δ에 따라 캐시 미적중률을 보였으며, LRU와 DISTANCE의 캐시 미적중률과 비교하고 있다. 그림의 범례에서 숫자는 버퍼의 수이다. LDT 1과 LDT 2는 각각 모든 파일이 반복 참조가 많은 경우와 모든 파일이 순차 참조가 많은 경우이다. 이 경우에는 ABM의 성능이 LRU나 DISTANCE 사이에 존재하게 된다. LDT 3과 LDT 4의 경우, 순차 참조가 많은 파일과 반복 참조가 많은 파일이 혼재하고 있어서 버퍼 정책 제어값을 잘 정하면 ABM의 성능을 제일 좋게 할 수 있다. 예를 들어, LDT 3의 경우 버퍼가 6,000개일 때 버퍼 정책 제어값이 0.01이면 ABM의 성능이 제일 좋아진다.

<표 2> 파일의 참조 유형 종류

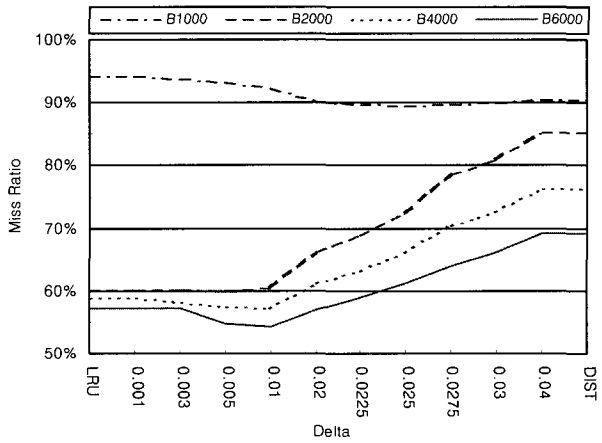
참조 유형	설 명
LDT 1	모든 파일의 루프 도착률을 작게 한다. 이를 위해서 파일을 참조하는 고객들의 평균 루프간 간격을 100으로 설정하였다.
LDT 2	모든 파일의 루프 도착률을 크게 한다. 이를 위해서 파일을 참조하는 고객들의 평균 루프간 간격을 1,000으로 설정하였다.
LDT 3	파일 번호가 커질수록 루프 도착률을 점차 낮아지게 한다. 이를 위해서 파일 i를 참조하는 고객들의 평균 루프간 간격 $IBL(i)=IBL(1)*1.2^{(i-1)}$으로 설정하였다. 예를 들어, 파일 1의 평균 루프간 간격이 50이면, 파일 2는 60, 파일 3은 72, 파일 20은 1,597이 된다.
LDT 4	파일을 두 그룹으로 나누고 앞 그룹은 높은 루프 도착률을 가지게 하고 뒷 그룹은 매우 낮은 루프 도착률을 가지게 한다. 파일 번호 i가 10보다 작거나 같으면 LDT1의 $IBL(i)$ 식을 사용하고 파일 번호 i가 11보다 크면 $IBL(i)=3,600/1.2^{(20-i)}$으로 설정하였다.



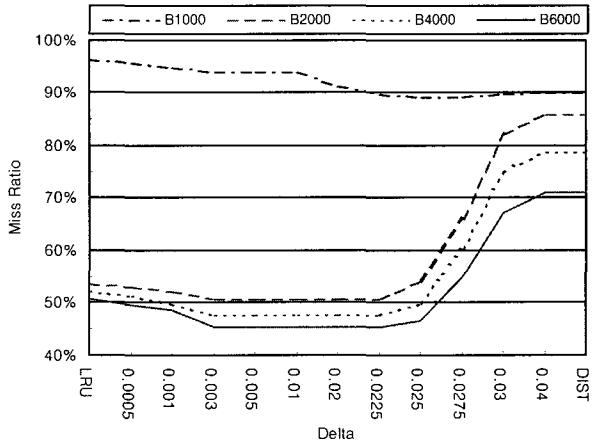
(가) LDT1의 결과



(나) LDT2의 결과



(다) LDT3의 결과



(라) LDT4의 결과

(그림 6) ABM의 실험 결과

4. 결론 및 향후 연구

비디오나 오디오 파일을 이용한 교육 시스템에서 학생들이 학습효과를 높이기 위해 파일의 부분들을 반복 참조할 수 있다. 본 논문은 연속미디어 파일 시스템에서 반복 참조가 있는 경우, 기존의 버퍼 캐시 기법들의 성능을 살펴보고 새로운 버퍼 관리 기법인 ABM을 제안하였다. 제안한 ABM은 파일별로 참조 유형을 고려하여 적절한 버퍼 정책을 적용한다. 이를 위해서 파일의 루프 참조 지수(LI)를 정의하였고 파일의 루프 참조 지수를 버퍼 정책 제어값(δ)과 비교하였다. 파일의 LI가 δ보다 크게되면 그 파일은 LRU로 관리한다. 역으로 파일의 LI가 δ보다 작게되면 DISTANCE로 관리한다. ABM은 δ를 조정하여 성능을 조절할 수 있으며 반복 참조가 많은 파일과 순차 참조이 많은 파일이 적절하게 혼재하는 경우에는 LRU나 DISTANCE보다 더 좋은 성능을 가져다 줄 수 있다.

실제 시스템에서는 적절한 δ의 값을 정하는 문제가 있다. 좋은 성능 결과를 얻으려면 수행 중에 δ를 조정하는 방법이 필요하다. δ의 조정 방법은 [27]에서 제시한 방법을 응용할 수 있다. 주기마다 그 주기에서 캐시 적중률을 계산하고 캐시 적중률이 향상되었는 지에 따라 δ를 조정한다. 예를 들어, 주기 i에서 캐시 적중률이 주기 i-1의 캐시 적중률보다 좋아졌다고 하자. 이때, 주기 i에서 δ값이 주기 i-1의 δ값보다 크다면, δ의 값을 증가시킨다. 역으로 주기 i에서 δ값이 주기 i-1의 δ값보다 작다면, δ의 값을 감소시킨다.

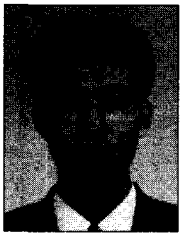
본 논문은 루프 참조 지수와 버퍼 정책 제어값을 이용한 버퍼 관리 방법을 연구하였다. 향후 과제로는 버퍼 정책 적용 주기, 버퍼 정책 관리자 별 버퍼 할당 방법 등이 남아있다. 또한, 실제 시스템의 워크로드를 구하여 성능을 실험하는 것도 과제로 남아있다.

참 고 문 헌

[1] D. J. Gemmel, H. M. Vin, D. D. Kandler, P. V. Rangan, and L. A. Rowe, "Multimedia Storage Servers : A Tutorial," IEEE Computer, pp.40-49, May, 1995.

[2] B. Özden, R. Rastogi, and A. Silberschatz, "A Framework for the Storage and Retrieval of Continuous Media Data," Proc. of the IEEE International Conference on Multimedia

Computing and Systems, May, 1995.
[3] H. M. Vin and P. V. Rangan, "Designing a Multi-User HDTV Storage Server," IEEE Journal on Selected Areas in Communications, pp.153-164, Jan., 1993.
[4] Coffman, E. G. and P. J. Denning, Operating Systems Theory, Prentice-Hall, Englewood Cliffs, N. J., 1973.
[5] King, W. F., "Analysis of Paging Algorithms," In Proc. IFIP Congress, Ljublanjana, Yugoslavia, pp.485-490, Aug., 1971.
[6] Lang, T., C. Wood, and I. B. Fernandez, "Database Buffer Paging in Virtual Storage Systems," ACM Transactions on Database Systems, Vol.2, No.4, pp.339-351, Dec., 1977.
[7] Rao, G. S., "Performance Analysis of Cache Memories," Journal of the ACM, Vol.25, No.3, pp.378-395, Jul., 1978.
[8] A. Dan and D. Towsley, "An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes," ACM SIGMETRICS, May., 1990.
[9] H. T. Chou and D. J. DeWitt, "An Evaluation of Buffer Management Strategies for Relational Database Systems," Proc. of the Eleventh International Conference on Very Large Databases, pp.127-141, Aug., 1985.
[10] J. Robinson and M. Devarakonda, "Data Cache Management Using Frequency-Based Replacement," ACM SIGMETRICS, pp.134-142, 1990.
[11] E. J. O'Neil, P. E. O'Neil, and G. Weikum, "The LRU-K Page Replacement Algorithm for Database Disk Buffering," Proc. of the 1993 ACM SIGMOD Conference, pp.297-306, 1993.
[12] B. Özden, R. Rastogi, and A. Silberschatz, "Buffer Replacement Algorithms for Multimedia Storage Systems," Proc of IEEE International Conference on Multimedia Computing and Systems, Jun., 1996.
[13] A. Dan and D. Sitram, "Buffer Management Policy for an On-Demand Video Server," IBM Research Report, RC 19347, Yorktown Heights, NY. 1993.
[14] A. Dan and D. Sitram, "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Environments," IS&T SPIE Multimedia Computing and Networking Conference, Jan., 1996.
[15] A. Reddy and J. Wyllie, "Disk Scheduling in a Multimedia I/O System," Proc. of ACM Multimedia, pp.225-233, 1993.
[16] K. Wu and P. S. Yu, "Consumption-Based Buffer Management for Maximizing System Throughput of a Multimedia System," Proc. Of IEEE International Conference on Multimedia Computing and Systems, Jun., 1996.
[17] F. Tobagi, J. Pang, R. Baird, and M. Gang, "Streaming RAID-A Disk Array Management System for Video Files," Proc. Of ACM Multimedia, pp.383-400, Aug., 1993.
[18] M. Chen, D. Kandler, and P. S. Yu, "Optimization of the Grouped Sweeping Scheduling(gss) with Hetetogeneous Multimedia Streams," ACM Multimedia'93, pp.235-242, 1993.
[19] H. M. Vin, A. Goyal, and P. Goyal, "An Observation-Based Admission Control Algorithm for Multimedia Servers," Proc. of IEEE International Conference on Multimedia Computing and Systems, pp.234-243, May, 1994.
[20] E. Chang and H. Garcia-Molina, "Effective Memory Use in a Media Server," Proc. of the 23rd VLDB Conference, pp.496-505, Aug., 1997.
[21] Y. S. Ryu and K. Koh, "A Dynamic Buffer Management technique for Minimizing the Necessary Buffer Space in a Continuous Media Server," Proc. of the IEEE International Conference on Multimedia Computing and Systems, Jun., 1996.
[22] T. Raymond, and Y. Jinhai, "An Analysis of Buffer Sharing and Prefetching Techniques for Multimedia Systems," Multimedia Systems, pp.55-69, Jun., 1996.
[23] P. Cao, E. W. Felten, and K. Li, "Implementation and Performance of Application Controlled File Caching," Proc. of the 1st USENIX Symposium on Operating Systems Design and Implementation, pp.165-178, 1994.
[24] J. Choi, S. Noh, S. Min and Y. Cho, "An Implementation Study of a Detection-based Adaptive Block Replacement Scheme," 1999 USENIX Annual Technical Conference, ACM, pp.239-252, 1999.
[25] R. H. Patterson, G. A. Gibson, E. Ginting, D. Stodolsky, and J. Zelenka, "Informed Prefetching and Caching," Proc. of the 15th Symposium on Operating System Principles, pp. 1-16, 1995.
[26] Video Store Magazine, Dec. 1992.
[27] D. Lee, et al., "On the Existence of a Spectrum of Policies that Subsumes the Least Recently Used(LRU) and Least Frequently Used(LFU) Policies," Proc. of the 1999 ACM SIGMETRICS Conference, May, 1999.



조 경 운

e-mail : cezanne@oslab.snu.ac.kr
1995년 서울대학교 전산과학과 졸업(학사)
1997년 서울대학교 대학원 전산과학과 (이학석사)
1997년~현재 서울대학교 대학원 전기컴퓨터공학부 박사 과정
관심분야 : 멀티미디어 파일 시스템, 클러스터 시스템 등

류 연 승

e-mail : ysryu@hallym.ac.kr
1990년 서울대학교 계산통계학과 졸업 (학사)
1992년 서울대학교 대학원 전산과학과 (이학석사)
1996년 서울대학교 대학원 전산과학과 (이학박사)
1996년~2000년 삼성전자 선임연구원
2000년~현재 한림대학교 정보통신공학부 전임강사
관심분야 : 멀티미디어 시스템 소프트웨어, 멀티미디어 네트워크, 실시간 스케줄링, 성능평가 등

고 건

e-mail : kernkoh@june.snu.ac.kr
1974년 서울대학교 응용물리학과 졸업 (학사)
1979년 미국 버지니아 대학교 전산학 (공학석사)
1981년 미국 버지니아 대학교 전산학 (공학박사)
현재 서울대학교 전기컴퓨터 공학부 교수
관심분야 : 운영체제, 멀티미디어시스템, 클러스터 시스템, 웹캐쉬, 저전력 시스템