ABRN:주문형 멀티미디어 데이터베이스 서비스 시스템을 위한 버퍼 교체 알고리즘

정 광 철[†] · 박 웅 규[††]

요 약

본 논문은 시간 변화에 따라 편중된 자료 접근을 갖는 주문형 멀티미디어 데이터베이스 서비스 시스템에서 버퍼를 교체하는 방법에 관하여 연구한 것이다. AOD(audio-on-demand)와 VOD(video-on-demand) 서비스를 지원하는 멀티미디어 데이터베이스 시스템에서의 객체 접근은 인기있는 객체들에 편중되어 있으며 이 편중된 자료 접근 형태는 시간에 따라 변하는 특성을 가지고 있다. 기존의 LRU와 LFU 버퍼 교체 방법은 이러한 상황에는 적합하지 않다. 따라서 본 논문은 시간 변화에 따라 편중된 자료 접근을 갖는 주문형 멀티미디어 데이터베이스 서비스 시스템에서 고성능의 서비스를 위하여 신경망을 이용한 버퍼 교체 방법(ABRN: Adaptive Buffer Replacement using Neural Networks)을 제안하였다. 제안된 버퍼 교체 알고리즘이 이용한 신경망의 주요 역할은 요구된 객체를 핫셋(hot set)과 콜드셋(cold set)으로 분류하는 것이다. 핫셋은 상당한 인기가 있어서 편중된 자료 접근을 갖는 객체들의 집합이고 콜드셋은 별로 인기가 없어서 임의 접근을 갖는 객체들의 집합이다. 요구된 객체를 분류하기 위하여, 이 객체의 과거 시간 간격 값들은 신경망을 통과하게 된다. ABRN은 LFU와 LRU의 장점을 이용하기 위하여 버퍼를 LFU 영역과 LRU 영역으로 나눈다. LFU 영역은 핫셋으로 분류된 객체들을 포함하고 LFU 방법으로 버퍼를 관리하며 LRU 영역은 콜드셋으로 분류된 객체들을 포함하고 LRU 방법으로 버퍼를 관리한다. 제안된 알고리즘인 ABRN의 성능을 LFU, LRU, 그리고 LRU의 변형인 LRU-k와 비교하기 위하여 시간에 따라 변하는 편중된 자료 접근을 갖는 실제의 서비스 집계 정보를 이용하여 시뮬레이션하였다. 시뮬레이션 결과 ABRN이 다른 버퍼 교체 방법보다 성능이 우수한 것으로 나타났다. 신경망을 기반으로 한 ABRN은 가상 기억 장치 시스템의 페이지 교체 및 프리페칭과 같은 분야에도 효과적으로 이용될 수 있다.

ABRN: An Adaptive Buffer Replacement for On-Demand Multimedia Database Service Systems

Kwang Chul Jung[†] · Ung-Kyu Park[††]

ABSTRACT

In this paper, we address the problem of how to replace buffers in multimedia database systems with time-varying skewed data access. The access pattern in the multimedia database system to support audio-on-demand and video-on-demand services is generally skewed with a few popular objects. In addition the access pattern of the skewed objects has a time-varying property. In such situations, our analysis indicates that conventional LRU(Least Recently Used) and LFU(Least Frequently Used) schemes for buffer replacement are not suited. We propose a new buffer replacement algorithm(ABRN: Adaptive Buffer Replacement using Neural

† 정 회 원:한국전자통신연구소 데이타베이스연구실
†† 정 회 원:서원대학교 전자계산학과
논문접수:1996년 10월 4일, 심사완료:1996년 11월 20일

Networks) using a neural network for multimedia database systems with time-varying skewed data access. The major role of our neural network classifies multimedia objects into two classes : a hot set frequently accessed with great popularity and a cold set randomly accessed with low popularity. For the classification, the inter-arrival time values of sample objects are employed to train the neural network. Our algorithm partitions buffers into two regions to combine the best properties of LRU and LFU. One region, which contains the 핫셋 objects, is managed by LFU replacement and the other region, which contains the cold set objects, is managed by LRU replacement. We performed simulation experiments in an actual environment with time-varying skewed data access to compare our algorithm to LRU, LFU, and LRU-k which is a variation of LRU. Simulation results indicate that our proposed algorithm provides better performance as compared to the other algorithms. Good performance of the neural network-based replacement scheme means that this new approach can be also suited as an alternative to the existing page replacement and prefetching algorithms in virtual memory systems.

1. 개 요

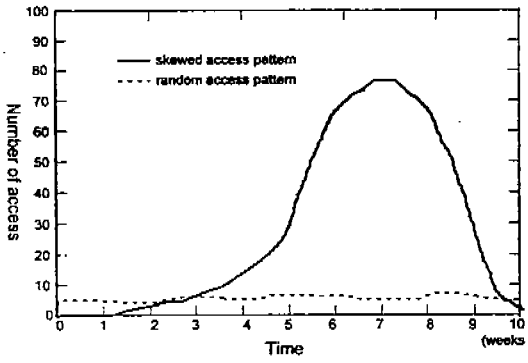
지난 10년 동안 컴퓨터와 통신 기술은 눈부신 발전을 이룩하였다. 이들 기술의 진보는 이미지, 비디오, 그리고 오디오와 같은 멀티미디어 정보를 네트웍을 통해 쉽게 접근할 수 있도록 하였다. 특히, AOD(audio-on-demand)와 VOD(video-on-demand)와 같은 주문형 서비스 시스템을 위한 멀티미디어 데이터베이스 시스템은 디지털 라이브러리 정보 시스템으로부터 오락 기술에 이르기까지 여러 응용 영역의 필수 분야로 대두되었다. 일반적으로 멀티미디어 데이터베이스 시스템은 멀티미디어 데이터베이스 서버와 고성능의 네트웍으로 연결된 다수의 클라이언트들로 구성된다. 클라이언트는 멀티미디어 객체를 실연(playback)하기 위하여 서버로부터 객체를 조회한다[4].

멀티미디어 정보는 대용량의 저장 공간을 요구하기 때문에, 멀티미디어 객체는 일반적으로 RAID (redundant arrays of inexpensive disks)와 같은 디스크 어레이 또는 CD-ROM 쥬크 박스(juke box)와 같은 광자기 저장 장치에 저장된다[2, 4]. 프로세서의 속도는 지속적으로 증가하는 반면, 보조 기억장치의 느린 I/O 속도는 멀티미디어 데이터베이스 시스템의 속도를 저하시키는 주 원인이 되어 왔다. 이 문제를 극복하기 위하여 대부분의 멀티미디어 데이터베이스 시스템은 주기억 장치의 일부를 캐쉬 버퍼로 사용한다. 또한 광자기 디스크는 하드 디스크보다 속도가 더 느리기 때문에, 광자기 디스크로부터의 평균 검색 시간을 줄이기 위하여 하드 디스크를 캐쉬 버퍼로 사용하기도 한다. 멀티미디어 데이터량은 매우 많고 시스템이 갖는 캐쉬 버퍼의 크기는 제한되기 때문에 적중률을 증가시키기 위한 버퍼 교체 방법은 신중하게 설계되어야 한다.

실제의 AOD와 VOD 시스템에서, 상당히 인기 있는 일부분의 객체는 다른 객체들에 비해 앞으로도 더 많이 조회될 확률이 높다[2]. 접근 형태를 분석한 결과, 주문형 멀티미디어 서비스 시스템에서 객체는 두 개의 구룹으로 분류된다. 하나는 앞으로도 계속 자주 조회될 객체들이 존재한다는 것(핫셋)과 다른 하나는 앞으로 덜 조회될 객체들이 존재한다는 것(콜드셋)이다. 상당한 인기를 갖는 객체의 접근 형태는 시간이 지남에 따라 빠른 속도로 접근 회수가 증가하는 상승 단계와 접근 회수가 점차 줄어드는 소멸 단계로 나타난다. 즉, 시간이 지남에 따라 인기있는 객체들은 핫셋에서 콜드셋으로 천이한다. 이것은 시간이 지남에 따라 자료 접근의 편중 정도와 편중된 객체의 수가 변함을 의미한다. 이런 접근 형태를 시변환적 편중 자료 접근(time-varying skewed data access)이라 한다. 또한 AOD나 VOD와 같은 주문형 멀티미디어 서비스 환경에서 인기 없는 객체는 가끔씩 요구되는 임의 자료 접근(random data access) 형태를 갖는다. 그러므로 주문형 멀티미디어 서비스 환경에는 시변환적 편중 자료 접근과 임의 자료 접근이 양존한다. (그림 1)은 주문형 멀티미디어 서비스 환경에서 나타나는 자료 접근 형태의 한 예를 보인 것이다.

본 논문은 시변환적 편중 자료 접근과 임의 자료 접근이 양존하는 주문형 멀티미디어 데이터베이스 서비스 시스템에서 어떻게 캐쉬 버퍼를 교체할 것인지에 관하여 연구한 것이다. 그동안 데이터베이스 버퍼링, 프로세서 캐싱, 가상 메모리 페이징 등을 위하여 다양한 버퍼 교체 알고리즘이 연구되어 왔다[1, 5,

6, 8, 9, 10]. FIFO(first-in first-out), LRU(least recently used), LFU(least frequently used), 그리고 LRU의 변형인 LRU-k와 같은 기존 버퍼 교체 방법이 여러 분야에서 이용되어 왔다. 본 논문에서 분석한 AOD와 VOD의 자료 접근 양태에 앞의 알고리즘들을 적용하면, 자료 접근의 편중 정도가 시간에 따라 변하는 환경에서 기존 버퍼 교체 방법들은 어떤 경우에는 좋은 결과가 나타나지만 상황이 바뀌면 상당히 나쁜 결과를 초래한다는 사실을 관찰하였다. 실험에 의하면, LRU는 시간 국부성(temporal locality)을 갖는 비편중된 자료 접근 분포 상황에서는 다른 방법보다 좋은 성능을 보인 반면, LFU는 편중된 자료 접근 분포 상황에서 다른 방법보다 좋은 성능을 나타내었다. 따라서 LRU와 LFU는 편중된 자료 접근과 임의 자료 접근이 동시에 존재하는 주문형 멀티미디어 서비스 시스템에는 적합하지 않다. 또한 LRU-k도 LRU를 기본으로 하기 때문에 편중된 자료 접근인 경우에는 부적합한 것으로 나타났다.



(그림 1) 주문형 멀티미디어 환경에서 자료 접근 형태의 예
(Fig. 1) Example of data access pattern in on-demand multimedia environment

본 논문은 LFU와 LRU의 좋은 특성을 결합하는 새로운 버퍼 교체 방법을 제안한다. 이 버퍼 교체 방법은 캐쉬 버퍼를 2개 영역으로 나눈다. 한 영역은 LFU 영역으로서 핫셋 객체들을 위해 이용되고 다른 한 영역은 LRU 영역으로서 콜드셋 객체들을 위해 이용된다. 기본적으로 LFU 영역은 LFU 버퍼 교체 방법이 적용되고 LRU 영역은 LRU 버퍼 교체 방법이 적용된다. 제안된 버퍼 교체 방법을 위한 신경망의 주요 역할은 요구된 객체를 편중 자료 접근 형태를 갖는 객체 그룹인 핫셋과 임의 자료 접근 형태를 갖는 객체 그룹인 콜드셋으로 분류하는 것이다. 이를 위하여 요구된 객체의 과거 도착 시간 간격(inter-arrival time) 값들은 신경망을 경유하게 된다.

본 논문은 제안된 버퍼 교체 방법의 성능을 측정하기 위하여 LFU, LRU, 그리고 LRU-k와 비교하였다. 성능 분석을 위해 실제로 인터넷으로 한국 가요를 서비스하고 있는 AOD 시스템의 서비스 정보를 이용하였다. 시뮬레이션 결과 제안된 알고리즘이 다른 알고리즘에 비해 성능이 우수한 것으로 나타났다.

본 논문은 다음과 같이 구성되었다. 2장에서 주문형 멀티미디어 데이터베이스 서비스 시스템의 구조와 이것의 환경에 관하여 기술하였다. 3장에서는 제안된 알고리즘에 이용될 신경망을 정의하고 4장에서 새로운 버퍼 교체 알고리즘을 제안하였다. 5장에서는 시뮬레이션을 통해 성능을 분석하였으며 6장에서 결론을 기술하였다.

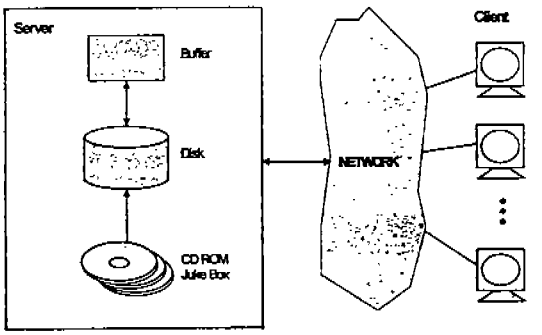
2. 주문형 멀티미디어 서비스 시스템 환경

이 장에서는 본 논문이 적용될 주문형 멀티미디어 서비스 시스템의 개략적인 구조와 저장 장치 구조에 관해서 살펴본다.

2.1 시스템 구조

VOD나 AOD, 그리고 멀티미디어 정보를 서비스하는 시스템은 네트워크를 통해서 멀티미디어 정보를 서버로부터 클라이언트로 전달하게 된다. VOD 서버는 보통 ATM 네트웍의 OC3 링크와 같은 고 대역폭의 ATM 망을 통해 클라이언트와 연결되고 AOD 서버나 그밖의 멀티미디어 서버는 인터넷을 이용하기도 한다. 서버는 대량의 멀티미디어 자료를 저장 관리하기 위하여 고성능의 컴퓨터 시스템, 특히 병렬처리 컴퓨터 시스템을 사용한다. 서버 시스템은 일반적으로 멀티미디어 데이터베이스 시스템을 이용하여 서비스 자료를 효율적으로 관리하기도 하지만 VOD나 AOD의 경우에는 이들 응용 분야에 맞는 전용의 비디오 서버 시스템이나 오디오 서버 시스템을 이용하기도 한다[11].

이들 주문형 멀티미디어 서비스 시스템이 대용량의 멀티미디어 자료를 고가의 하드 디스크에 모두 저장하여 관리하는 데는 많은 비용이 든다. 특히 검색 위주의 주문형 비디오 시스템이나 주문형 오디오 시스템의 경우 서비스의 다양성을 위해, 별로 인기없는 영화나 노래들도 장르 별로 갖추고 있어야 하는 경우가 대부분이다. 따라서 이들 모든 자료를 하드 디스크에 저장하여 서비스하는 것은 비용의 낭비를 초래하게 된다. 이를 위해 주문형 멀티미디어 서비스 시스템은 CD-ROM과 같은 값싸고 대용량인 저장 매체를 이용하여 비인기 자료를 저장하여 서비스하기도 한다. 이를 오프라인 미디어라고 한다. 즉 주문형 멀티미디어 서버 시스템은 오프라인 미디어-하드 디스크- 주기억 장치로 이어지는 저장 장치 계층 구조를 이용하여 멀티미디어 자료를 저장 관리한다[13]. (그림 2)는 주문형 멀티미디어 서비스 시스템의 구조를 보인 것이다.



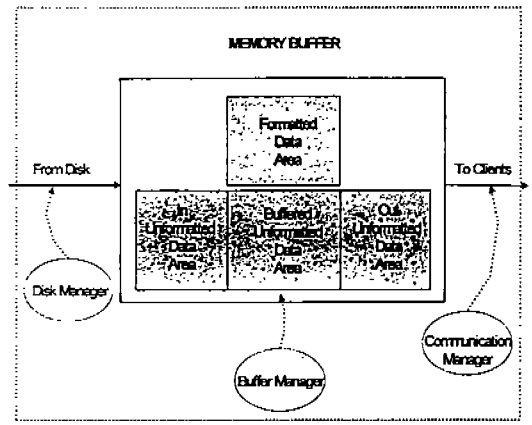
(그림 2) 주문형 멀티미디어 데이터베이스 서비스 서스템 구조

(Fig. 2) Architecture of on-demand multimedia service system

2.2 버퍼 관리

주문형 멀티미디어 서비스 시스템에 필요한 데이터베이스 관리 시스템은 비정형 자료(unformatted data)는 물론 정형 자료(formatted data)도 함께 관리한다. 비정형 자료는 그 크기가 수 킬로바이트에서 수 기가바이트까지 이르기 때문에 이를 하나의 버퍼 단위로 하는 것은 매우 어려운 실정이다. 특히 클라이언트에서 비정형 자료를 전달 받아 실시간으로 실연하기 위하여 서버는 논리적인 연속 객체 실연 단위(COPU: continuous object presentation unit)를 연속적으로 클라이언트에 전송하는 방법이 있어야 한다[12]. 이를 위하여 비정형 자료에 대하여는 특별한 버퍼 관리가 필요하다. 한편 정형 자료에 대해서도 버퍼 관리가 필요한데 이는 기존의 데이터베이스 관리 시스템이 하는 방법을 크게 벗어나지 않는다. 따라서 주문형 멀티미디어 서비스 시스템에 필요한 데이터베이스 관리 시스템은 비정형 자료와 정형 자료에 대한 버퍼 관리를 함께 해주어야 한다. (그림 3)은 이에 관한 개략적인 구조를 나타낸 것이다.

본 논문에서 우리는 앞에서 언급한 주문형 멀티미디어 데이터베이스 서비스 시스템의 성능 향상을 목적으로 객체를 재사용하기 위한 버퍼 교체 방법만을 고려한다. 시스템 환경에서 프로세서, I/O 버스, 그리고 이 자원들이 갖는 대용량성으로부터 유발되는 대역폭의 제한은 없다고 가정한다. 또한 멀티미디어 객체를 버퍼링하기 위한 단위와 이 객체를 실연하기 위한 일련의 버퍼 스트림을 통신망으로 전송하는 기술에 관하여는 논외로 한다. 즉 본 논문은 버퍼의 단위 및 전송 방법과는 상관없이 비정형 및 정형 자료에 모두 적용될 수 있는 버퍼 교체 방법만을 다루고자 한다. 또한 본 논문에서 제안한 버퍼 교체 방법은 주기억 장치 버퍼뿐만 아니라 오프라인 미디어 데이터를 위한 디스크 버퍼에도 적용될 수 있다.



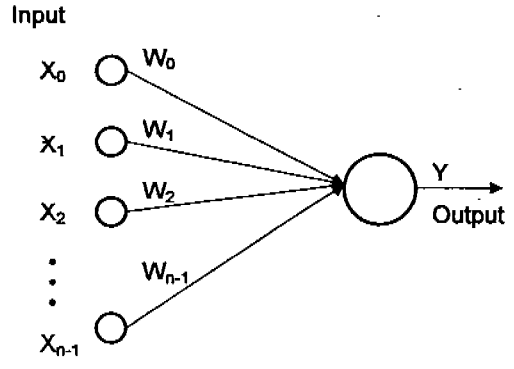
(그림 3) 주기억 장치 버퍼 구성

(Fig. 3) Memory buffer organization

3. 신경망을 이용한 접근 시간 예측

생물 신경망으로부터 유추된 인공 신경망은 비선형 병렬로 분산된 단순 처리 요소들의 밀집된 상호 연결을 통해 고성능을 일구어 내기 위한 하나의 정보 처리 구조이다. (그림 4)는 생물 신경을 단순화한 인공 신경 요소를 보여준 것이다. 한 요소의 출력 값 y는 $y=f(\sum_{i=0}^{N-1} W_i X_i - \theta)$ 공식에 의해 얻어진다. 신경망에서 가장 좋은 결과를 얻기 위해서는 지속적인 훈련과 학습 능력이 필요하다. 학습은 외부 입력 또는 요구된 응답에 따라서 가중치(W_i)를 수정해 나아가는 과정이다[7]. 가중치를 수정하는데 이용되는 공식을 학습 규칙이라 한다.

신경망에는 많은 모델이 있다[7]. 성공적인 신경망의 하나인 BPNN(backpropagation neural network)은 임의의 비선형 입/출력을 매핑하거나 비선형 좌표를 변환하는데 많이 이용된다. 전형적인 BPNN은 각각 1개씩의 입/출력 층(input/output layer)과 1개 이상의 감춤 층(hidden layer)으로 구성된다. 본 논문은 버퍼 교체 방법의 훈련과 학습을 위해 BPNN을 이용한다. 이용된 BPNN은 $k(k \geq 2)$개의 입력 포트와 1개의 출력 포트, 그리고 적어도 1개 이상의 숨김 층을 갖는다.



(그림 4) 신경망의 요소
(Fig. 4) Elements of neural network

제안된 버퍼 알고리즘에서 BPNN이 하는 일은 도착 시간 간격에 따른 과거와 미래 값 사이의 복잡한 관계성을 알아내는 것이다. $x_i(i=1, 2,...,t,..., \infty)$로 표시되는 연속된 시간 간격 사이에서, t 지점 이전의 값들이 관찰되었고 x_t가 예견되어야 할때, x_t 바로 이전의 k개의 자료 집합은 xt를 예측하는데 이용된다. 따라서,

$$f:(x_{t-k},...,x_{t-2}, x_{t-1}) \rightarrow \bar{x}_t \quad (1)$$

공식이 성립된다. $\bar{x}_t$는 x_t의 예측 값이다. k개의 자료는 입력 노드에 입력되고 x_t는 예견 값으로서 출력 노드에서 생성된다.

훈련 단계는 다음과 같이 이루어진다. 훈련 단계에서 $x_{t'}$의 예측값 바로 이전의 k개의 측정된 도착 시간 집합을 사용함으로써 BPNN의 출력 $\bar{x}_{t'}$는 공식 (2)에 의해 생성된다.

$$f:(x_{t'-k},...,x_{t'-2}, x_{t'-1}) \rightarrow \bar{x}_{t'} \quad (2)$$

훈련 견본을 최소화하고 예측의 정확성을 높이기 위하여 $x_{t'}$와 $\bar{x}_{t'}$ 사이의 오류 합산 값을 점차 줄여 나가는 오류 후전파 방법(error backpropagation method)[7]을 사용하여 망의 가중치를 수정한다. 즉 다음 예측 단계에서 이 수정된 가중치를 사용함으로써 $\bar{x}_{t'}$가 생성된다.

제안된 버퍼 교체 방법에서 신경망의 주요 역할은 자료 객체를 2개의 그룹 즉, 편중 자료 접근의 객체 그룹인 핫셋과 낮은 접근 확률의 임의 자료 접근 형태를 갖는 객체 그룹인 콜드셋으로 분류하는 것이다. 분류를 위하여, 견본 객체에 대한 최근의 과거 도착 시간 간격은 신경망을 통하게 된다. 즉 견본 객체의 최근 $k+1$ 개의 참조된 시간 값으로부터 계산된 k개의 도착 시간 간격 값은 입력 층의 k 포트에 적용된다. 이때 출력 포트의 응답 값은 다음의 도착 시간 간격을 예측하는 추정 값을 나타낸다. 예측된 값을 임계값과 비교하여 객체가 어느 그룹에 분류될 것인가를 결정한다. 만일 예측 값이 임계값보다 작으면(0에 근접하면) 객체를 핫셋으로 분류하고 임계값보다 크면(1에 근접하면) 콜드셋으로 분류한다.

4. ABRN : 버퍼 교체 방법

캐쉬 버퍼(cache buffer)는 자료를 저가에 대량으로 보관하기 위한 느린 속도의 저장 장치의 성능을 높이

는데 사용하는 빠른 속도의 저장 장치이다. 캐쉬 버퍼는 느린 저장 장치에 보관된 자료의 일부를 유지함으로써 빠른 저장 장치의 속도로 사용자 요구를 서비스할 수 있다. 캐쉬 버퍼 라인(cache buffer line)은 버퍼 관리를 위한 단위로서 연속된 자료 블럭이다. 이 장에서 우리는 기존의 LRU와 LFU 방법을 고찰하고 우리의 버퍼 교체 방법을 논의한다.

4.1 LRU와 LFU의 문제점

LRU는 버퍼 교체 방법에 있어서 가장 많이 사용하는 것 중 하나이다. 이 방법은 시간 국부성의 원리를 충실히 따르고 있으며 최근으로부터 가장 오랫동안 접근되지 않은 버퍼 라인은 가까운 미래에도 접근할 가능성이 적다는 가정하에 그 버퍼 라인을 교체하는 방법이다. 그러나 LRU는 버퍼 교체를 결정할 시기에 버퍼 라인의 접근 회수를 고려하지 않기 때문에, 상대적으로 자주 접근되는 버퍼 라인과 자주 접근되지 않는 버퍼 라인을 구분할 수가 없다[5, 8]. 편중된 자료 접근 환경에서 LRU가 갖는 문제점은 캐쉬 버퍼가 한번 접근된 버퍼 라인으로 가득찰 가능성이 있고 앞으로도 계속 접근 가능성이 있는 버퍼 라인을 캐쉬 버퍼로부터 몰아낼 가능성이 크다는 것이다. 이것은 버퍼 라인이 오직 한번 접근되는 경우가 작업의 대부분을 차지하고 캐쉬의 크기가 상대적으로 작은 경우에는 버퍼 교체가 자주 발생함으로써 시스템의 성능이 저하되는 원인이 된다.

LFU는 계속되는 참조 가능성을 예측하기 위하여 과거 접근 기록을 이용한다. 모든 버퍼 라인의 참조 회수는 유지되어 버퍼 교체가 필요한 경우에 가장 적게 접근된 버퍼 라인을 교체한다. 그러나 어떤 버퍼 라인이 한동안 많이 접근되었으나 당분간 다시는 이용되지 않는 상황이 존재할 경우, 이 버퍼 라인은 많은 참조 회수를 가지고 있기 때문에 더 이상 사용되지 않음에도 불구하고 계속 버퍼에 남아 있게 된다. 이와 같은 현상을 캐쉬 공해(cache pollution)라 한다[5]. 시변환적 편중 자료 접근 형태를 갖는 실제 상황에서 시간이 지남에 따라 인기 있는 객체들은 핫셋에서 콜드셋으로 변한다. 이 상황에서 LFU는 캐쉬 공해를 유발하여 성능을 크게 떨어뜨린다.

데이터베이스 페이지 버퍼링을 위한 LRU-k 페이지 교체 알고리즘[8]은 LRU의 변형이며 이것은 I/O를 위한 캐쉬 버퍼에서 사용되기도 한다. 각 페이지에 대하여 이 알고리즘은 최근 k 번의 참조 시간을 포함하는 과거 기록 테이블을 유지 및 관리한다. 캐쉬는 LRU 리스트로 구현되고 캐쉬 버퍼 라인을 교체하기 위한 결정은 과거 기록 테이블을 기반으로 한다. 그러나 LRU-k는 LRU를 기초로 하기 때문에, 이것의 성능은 LRU와 마찬가지로 편중된 자료 접근 형태의 경우 LFU의 성능보다 나쁘다.

위에서 논의한 바에 따르면, 기존의 LRU와 LFU 버퍼 교체 방법은 시간에 따라 가변적인 편중 자료 접근과 낮은 자료 접근 확률의 임의 자료 접근이 공존하는 주문형 멀티미디어 데이터베이스 서비스 시스템의 버퍼 교체 알고리즘으로는 적합하지 않음을 알 수 있다.

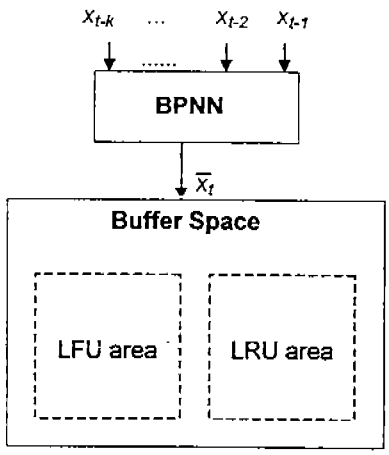
4.2 ABRN : 신경망을 이용한 버퍼 교체

이 장은 상당한 인기를 갖고 그 인기도가 시간에 따라 가변적인 객체를 포함하는 주문형 멀티미디어 데이터베이스 서비스 시스템을 위하여 신경망을 이용한 새로운 버퍼 교체 알고리즘인 ABRN (Adaptive Buffer Replacement using Neural networks)을 제시한다. 이 알고리즘은 4.1절에서 분석한 결과에 따라 LRU와 LFU의 장점을 취하고 그것들의 단점을 보완하기 위하여 편중된 자료 접근에는 LFU 방법을 적용하고 임의 자료 접근에는 LRU 방법을 적용하는 것이다. ABRN은 LRU와 LFU를 각각 적용하기 위하여 버퍼를 2개 영역으로 나눈다. LFU 영역은 핫셋 객체를 포함하고 이 영역의 객체들은 LFU 알고리즘에 의해 관리된다. LRU 영역은 콜드셋 객체를 포함하고 이 영역의 객체들은 LRU 알고리즘에 의해 관리된다.

ABRN은 요구된 객체가 편중된 자료 접근을 갖는지 아니면 임의 자료 접근을 갖는 지를 알아내기 위해여 신경망(BPNN)을 이용한다. 즉, 신경망은 객체를 핫셋 또는 콜드셋으로 분류하는 역할을 한다. 핫셋은 상당한 인기가 있어서 편중된 자료 접근을 갖는 객체들의 집합이고 콜드셋은 별로 인기가 없어서 임의 접근을 갖는 객체들의 집합이다. (그림 5)는 ABRN이 작동하는 개략적인 구조를 나타낸 것이다.

ABRN은 다음의 3가지 주요 버퍼 관리 규칙을 기반으로 한다.

규칙 1:(신경망을 이용하여 객체를 분류)



(그림 5) ABRN 동작 구조
(Fig. 5) Operational structure of ABRN

객체가 접근되었을 때, 이 객체의 최근부터의 마지막 k개의 과거 도착 시간 간격은 입력 층의 k 포트에 적용된다. 이때 출력 포트의 응답 값은 이 객체에 대한 다음 번의 도착 시간 간격을 예측하는 값으로 이용된다. 만일 출력 값이 임계 값보다 작으면 요구된 객체를 핫셋으로 분류하고 그렇지 않으면 콜드셋으로 분류한다.

규칙 2:(적중의 경우:요구된 객체가 캐쉬 버퍼에 존재할 때)

적중된 객체에 대하여 규칙 1의 결과 값에 따라 이 객체가 LRU 또는 LFU 중 어느 영역에 놓일지를 재분류한다. 즉, 신경망으로부터 예측된 차기 접근 확률에 따라 접근된 객체가 콜드셋에서 핫셋으로 또는 핫셋에서 콜드셋으로 바뀌는 경우에, 이 객체를 각각에 해당하는 버퍼 영역(LFU 또는 LRU)으로 갱신한다.

규칙 3:(실패의 경우:요구된 객체가 캐쉬 버퍼에 존재하지 않을 때)

경우 1:(프리 버퍼가 존재하는 경우)

요구된 객체에 프리 버퍼를 할당하고 규칙 1의 결과가 핫셋으로 분류되었으면 이 객체를 LFU 버퍼 영역에 포함시키고 콜드셋으로 분류되었으면 LRU 버퍼 영역에 포함시킨다.

경우 2:(프리 버퍼가 존재하지 않고 요구된 객체가 핫셋으로 분류된 경우)

우선, LFU 알고리즘을 이용하여 교체될 후보 객체를 LFU 버퍼 영역에서 선정한다. 그런 다음, LRU 알고리즘을 이용하여 LFU 버퍼 영역으로부터 선정된 후보 객체와 LRU 버퍼 영역에 존재하는 모든 객체들을 함께 경합시켜 최종적으로 교체될 희생자를 선정한다. LFU 버퍼 영역에서 선정된 희생자를 다시 한번 LRU 버퍼 영역의 객체와 최종 희생자를 선정하기 위하여 경합시키는 이유는 핫셋에서 콜드셋으로 천이하는 객체는, 적은 회수이긴 하지만 가까운 미래에 접근될 확률이 크기 때문이다.

경우 3:(프리 버퍼가 존재하지 않고 요구된 객체가 콜드셋으로 분류된 경우)

LRU 영역에 존재하는 기존의 객체들 중에서 LRU 버퍼 교체 알고리즘에 의해 희생자를 선정하고 요구된 객체를 버퍼에 넣는다.

앞에서 기술한 규칙을 기반으로 하여 알고리즘을 표현하면 (그림 6)과 같다. ABRN 알고리즘에서 사용되는 기호들을 다음과 같이 정의한다.

o_i : 접근 요구된 객체

BUFFER = fBUFFER ∪ rBUFFER : 버퍼에 존재하는 객체들의 집합(LFU 버퍼 영역에 존재하는 객체 집합(fBUFFER)과 LRU 버퍼 영역에 존재하는 객체 집합(rBUFFER)의 합으로 구성)

H_{set} : 핫셋 그룹

C_{set} : 콜드셋 그룹

$Xo_i = \{xo_i \mid 〈=i〈=k\}$: 객체 o_i에 대한 k개의 과거 도착 시간 간격 값

xo_i : 접근 요구된 객체 o_i의 신경망을 통해 예측된 도착 시간 간격 값

T_0 : 도착 시간 간격의 임계값

ABRN 버퍼 교체 알고리즘은 다음의 특징을 가진다.

- 신경망을 이용하여 다음 번의 도착 시간 간격을 예측함으로써 객체를 핫셋과 콜드셋으로 정확하게 분류한다.
- LRU와 LFU 알고리즘의 장점을 이용하며 이를 동적으로 운용한다. LRU와 LFU 영역의 크기는 객체의 접근 형태에 따라 동적으로 변한다. 그러나 캐쉬 버퍼의 전체 크기는 항상 일정하다. 만일 버퍼 대부분이 핫셋으로 분류된 객체로 채워

져 있으면 버퍼 교체는 LFU 알고리즘에 의해 운영되고 반대의 경우에는 LRU 알고리즘이 적용된다.

- ABRN은 시간 변화에 따라 편중되는 자료 접근 형태에 잘 적용 된다. 객체가 핫셋에서 콜드셋으로 또는 콜드셋에서 핫셋으로 변화하는 경우, 적중율을 증가시키기 위해 각각의 경우에 따라 적절히 LRU 또는 LFU 버퍼 교체 알고리즘을 적용한다.
- ABRN은 미래의 예측을 자기 자신의 정보에 의존한다. 즉 신경망을 훈련한 후에는 더 이상 외부로 부터의 보조 정보가 필요하지 않다.

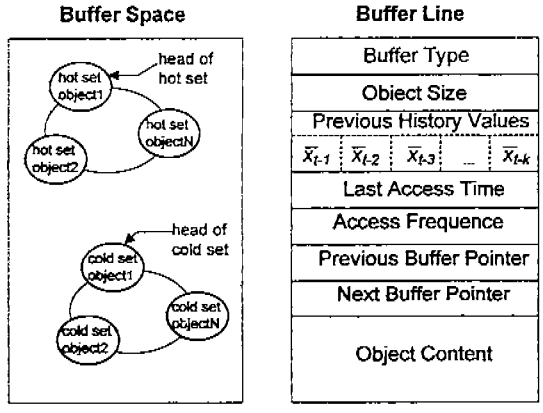
```
ABRN Algorithm

ABRNmanager(o_i): Pointer to o_i
begin
 Xo_i = GetAccessHistoryInformation(o_i);
 xo_i = GetInterArrivalTimeUsingBPNN(Xo_i);
 if xo_i <= T_0 then
 //predicted inter-arrival time value is less than a threhold value
    put o_i in H_set;
 else
    put o_i in C_set;
 if o_i ∈ BUFFER then  //Hit case
    update buffer, type;  //hot set or cold set
 else  //miss case
    begin
       if o_i ∈ H_set then //o_i is classfied into hot set
          if there exists in a free buffer then
             //put the requested object in LFU area
             put o_i in fBUFFER;
          else
             begin
                select a candidate victim from LFU area;
                select a final victim between LRU objects and
                a candidate victim using LRU sheme;
                expel victim from LFU area and put o_i in it;
             end
       else //o_i is classfied into cold set
          if there exists in a free buffer then
             //put the requested object in LRU area
             put o_i in rBUFFER;
          else
             begin
                select victim from LRU area;
                expel victim from LRU area and put o_i in it;
             end
    end
 return buffer address of o_i
end
```

(그림 6) ABRN 알고리즘
(Fig. 6) ABRN Algorithm

4.3 구 현

여기에서는 ABRN을 구현하기 위하여 고려해야 할 점에 관해서 기술한다. (그림 7)은 버퍼의 구조와 버퍼 라인이 갖는 정보를 표현한 것이다. 버퍼는 핫셋으로 분류된 객체 그룹과 콜드셋으로 분류된 객체 그룹을 포함한다. 각각의 그룹은 객체들이 서로 양방향으로 연결되고 각 그룹에 대한 헤드 포인터를 가지고 있다. 헤드 포인터는 접근된 객체가 핫셋 또는 콜드셋으로 분류되었을 때 이 객체가 적용될 그룹을 찾아가는데 이용된다. 각각의 그룹을 구성하는 객체들을 양방향으로 연결하여 원을 이루게 한 것은 목표 객체를 쉽게 찾도록 하기 위해서다. ABRN이 갖는 버퍼 영역은 핫셋과 콜드셋 영역이 양분되어 독립적으로 존재하는 것이 아니고, 두 영역이 물리적으로 혼재되어 있다. 그러나 버퍼 영역은 각 그룹을 구성하는 객체들을 원형으로 연결하기 때문에 논리적으로 양분되어 있다. 그리고 각 그룹의 영역은 동적으로 증감한다. 즉 시간이 지남에 따라 편중된 자료 접근 형태를 갖는 객체 수가 많으면 핫셋 영역이 콜드셋 영역에 비해 상대적으로 클 것이고 낮은 접근 확률의 임의 자료 접근 형태를 갖는 객체 수가 많으면 그 반대가 될 것이다.



(그림 7) 버퍼의 구조
(Fig. 7) Structures of Buffer

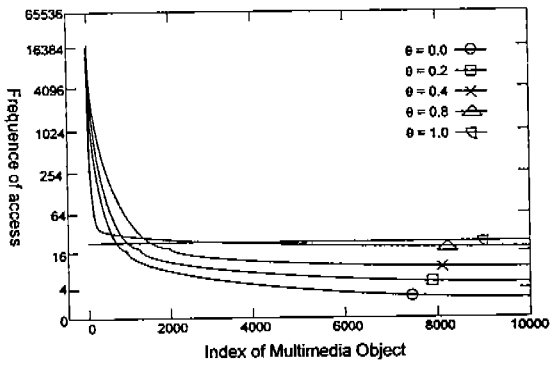
버퍼링 단위인 버퍼 라인은 신경망을 위한 정보, 핫셋 그룹과 콜드셋 그룹을 유지하기 위한 정보, 버퍼 교체 방법을 적용하기 위한 정보, 그리고 객체 자신의 정보로 구성된다. 첫째, 신경망을 위한 정보로는 과거 측정된 k개의 도착 시간 간격 값으로써, 새로 추가된 접근 시간 간격을 유지하기 위해 가장 오래된 값을 버리고 새로운 값을 접근 정보로 추가한다. 그리고 시스템이 적용되는 응용에 잘 맞도록 하기 위하여 과거 접근 시간 간격의 개수(k값)를 임의로 변경할

수 있도록 유연성을 부여한다. 둘째, 그룹을 유지하기 위한 정보로는 객체가 어느 그룹에 속하는 지를 나타내는 값과 각 그룹을 양방향 원형 리스트로 만들기 위한 두개의 포인터로 구성된다. 셋째, 버퍼 교체 방법을 적용하기 위한 정보로는 LFU 방법을 적용하기 위한 접근 회수를 기록한 값과 LRU 방법을 적용하기 위한 가장 최근의 접근 기록이 있다. 모든 버퍼 라인에 LFU와 LRU를 적용하기 위한 정보를 갖게 하는 이유는 시간이 지남에 따라 객체가 핫셋에서 콜드셋으로 천이하고 그 반대의 현상이 나타날 수도 있기 때문이다. 마지막으로 객체 자신의 정보로는 객체의 크기와 객체 내용이 있다.

버퍼 교체를 위해 많은 정보를 유지하는 것이 저장 장치의 낭비와 처리 시간에 부하가 걸리는 원인이 될 수 있다. 그러나 본 논문이 버퍼 교체의 대상을 멀티미디어 자료로 한정하므로, 정보 유지에 따른 저장 장치와 처리 시간의 비용은 멀티미디어 자료의 그것들에 비해서 아주 적은 양이므로 무시해도 된다.

5. 성능 분석

본 논문은 제안된 버퍼 교체 알고리즘의 성능 측정을 위해 추적 유도 시뮬레이션(trace-driven simulation) 방법을 사용하였으며, 이 방법으로 LRU, LFU, 그리고 LRU-k를 ABRN과 비교하였다. 성능을 비교하기 위하여, 본 논문은 참조 회수가 Zipf 분산을 갖는 자료와 실제 운영되고 있는 AOD 시스템의 서비스 통계 자료를 시뮬레이션 자료로 삼았다.



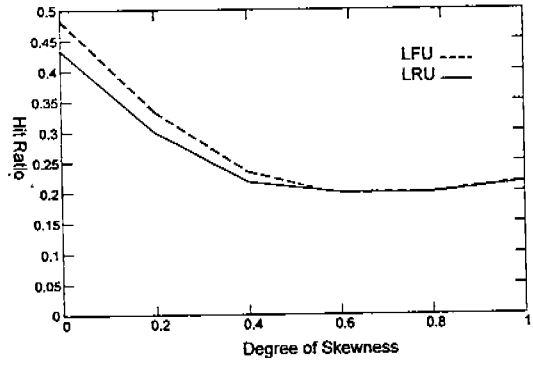
(그림 8) Zipf 분산 형태
(Fig. 8) Illustration of Zipf distribution

실제 데이터베이스 시스템에서는 자료 접근이 Zipf 분산[2, 3, 8]을 형성하는 것으로 알려졌기 때문에 본 논문은 첫번째 실험으로 Zipf 분산을 갖는 객체의 집합을 임의 접근하는 경우에 대하여 시뮬레이션을 수행하였다. 이 실험은 편중된 자료 접근 상황에서 LRU와 LFU의 성능을 분석하기 위해 필요하다. Zipf 분산에서 i번째 객체가 참조될 확률은 다음 공식으로부터 구할 수 있다.

$$pi = \frac{c}{i^{(1-\theta)}}, \qquad 1 \langle = i \langle = N,$$

$$where\ c = \frac{1}{H_N^{(1-\theta)}} \quad H_N^{(1-\theta)} = \sum_{i=1}^{N} \frac{1}{i^{(1-\theta)}}. \tag{3}$$

$\theta=1$일 경우에는 자료 접근에 대한 분포가 일정한 경우이고 $\theta=0$일 경우에는 자료 접근에 대한 분산이 고도로 편중된 경우이다. (그림 8)은 Zipf 분산을 개략적으로 보여 주고 있다. 일반적인 VOD 시스템의 자료 접근 형태는 $\theta=0.271$ 이하의 Zipf 분포를 갖는 것으로 알려져 있다[3]. 이 실험은 LRU와 LFU를 이용함에 있어서 자료 접근의 편중 정도에 따라 버퍼 성능에 미치는 영향을 고려하였다. 실험은 똑같은 상황에서 각각의 알고리즘에 대하여 10번 수행하였다. (그림 9)에 나타난 실험 결과는 10번 수행한 결과의 평균 값을 나타낸 것이다. 시뮬레이션 결과에 따르면 자료 접근의 편중 현상이 커질수록 LFU가 LRU보다 성능이 좋게 나타났다. 이것은 LRU 캐쉬 버퍼는 단지 한번 접근한 객체는 항상 버퍼 라인에 채워지므로, 재사용 가능성이 높은 객체가 버퍼로부터 밀려날



(그림 9) 기존 방법의 성능 측정 : 편중 정도 대 적중률
(Fig. 9) Performance evaluation of conventional methods

수 있기 때문이다. 반면에, LFU는 자주 참조되는 객체와 그렇지 않은 객체를 구별하는 특성을 가지므로 편중된 자료 접근에 적합하다. 편중의 정도가 작은 경우에 LRU와 LFU의 적중율에 차이가 나지 않는 것은 시간 국부성을 고려하지 않은 시뮬레이션 데이터의 임의 자료 접근 형태에 기인한다.

두번째 실험으로, ABRN과 다른 버퍼 교체 방법의 성능을 비교 분석하기 위하여 본 논문은 실제 상황에서 멀티미디어 객체의 편중된 자료 접근을 모형화하기 위하여 인터넷으로 서비스되고 있는 한국 가요의 접근 정보를 이용하였다. 이 AOD 시스템은 현재 약 310 곡을 서비스하고 있으며 최근 2 개월 동안 15000 회의 자료 접근이 있었다. 본 논문은 이 자료 접근 정보를 가지고 LRU, LFU, LRU-k와 ABRN의 성능을 평가하였다. 결과는 〈표 1〉과 같다. 표에 따르면 실험 결과는 제안된 알고리즘이 다른 알고리즘에 비해서 우수한 것으로 나타났다. 특히, 제안된 알고리즘은 버퍼 크기가 작을수록 다른 알고리즘에 비해 우수한 것을 알 수 있다. 이것은 제안된 알고리즘이 신경망을 통해 객체를 핫셋 또는 콜드셋으로 정확하게 분류하기 때문이다.

버퍼 라인의 갯수가 20일 때, 본 논문에서 제안한 알고리즘은 LRU-3와 LFU에 비해 각각 15.1%, 42.3% 정도 적중율이 높다. LFU 알고리즘은 버퍼 크기의 모든 경우에 대해서 다른 알고리즘에 비해 적중율이 가장 낮게 나타났다. 이것은 콜드셋 객체가 핫셋 객체로 천이될 때 실패율의 증가와, 또한 핫셋에서 콜드셋으로 천이될 때 미래에 자료 접근이 이루어지지 않는 객체가 버퍼에 존재함으로써 발생하는 적중율의 감소에 기인한다.

〈표 1〉 시뮬레이션 결과 : 버퍼 크기에 따른 적중률
〈Table 1〉 Simulation result : hit ratio related buffer size

버퍼크기	LRU	LRU-2	LRU-3	LFU	ABRN
20	0.250	0.255	0.259	0.209	0.298
40	0.384	0.388	0.402	0.300	0.439
60	0.473	0.476	0.479	0.379	0.527
80	0.531	0.533	0.542	0.439	0.573
100	0.574	0.588	0.589	0.491	0.593

6. 결 론

본 논문은 시간에 따라 가변적인 인기를 갖는 객체를 포함하는 주문형 멀티미디어 서비스 시스템의 효과적인 서비스를 위하여 신경망을 이용한 새로운 버퍼 교체 알고리즘인 ABRN을 제안하였다. 주문형 시스템에 이용되는 멀티미디어 데이터베이스 시스템에서의 객체 접근 형태는 시간이 지남에 따라 편중된 자료 접근 형태와 낮은 접근 확률의 임의 자료 접근 형태가 양존한다. 따라서 본 논문에서 제안한 버퍼 교체 알고리즘은 이러한 접근 형태에서 최대한의 적중율을 높이기 위한 것이다. ABRN은 LRU와 LFU의 이점만을 이용하기 위하여 캐쉬 버퍼를 2개의 영역으로 분류하였다. 그리고 요구된 객체를 핫셋과 콜드셋으로 분류하기 위하여 신경망을 이용하였다. 실제로 운영되고 있는 AOD의 자료 접근 정보를 이용하여 시뮬레이션한 결과 LRU, LFU, LRU-k 등 다른 버퍼 교체 알고리즘에 비해 제안된 알고리즘의 성능이 우수한 것으로 나타났다.

ABRN 버퍼 교체 방법은 다음과 같은 특징을 갖는다.

- 다음 번의 도착 시간 간격을 예측하여 객체를 핫셋과 콜드셋으로 분류하기 위해 신경망을 이용한다.
- LRU와 LFU 알고리즘의 장점을 이용하며 신경망의 예측 결과에 따라 이들을 동적으로 운용한다.
- ABRN은 객체가 핫셋에서 콜드셋으로 또는 콜드셋에서 핫셋으로 천이하는 시변환적편중 자료 접근 형태에 잘 적용된다.
- ABRN은 신경망을 교육한 후에는 더 이상 외부로 부터의 보조 정보가 필요하지 않다.

ABRN은 주문형 멀티미디어 서비스 환경에서 데이터베이스 관리 시스템의 자료 저장 관리기에 포함되는 버퍼 관리기를 대상으로 하였다. 그러나 ABRN은 오프라인 미디어 서비스 환경의 디스크 기반 버퍼 관리, 가상 기억 장치의 페이지 교체 및 프리 패칭과 같은 분야도 잘 응용될 수 있다.

대부분의 멀티미디어 객체의 크기가 매우 클 수 있기 때문에 이것을 하나의 버퍼 단위로 삼는 것은 모든 응용 분야에 일반적으로 적용하기가 곤란하다. 이러한 상황에서 논리적으로 연속된 객체를 실시간으로 실연하기 위해서는 적절한 크기의 버퍼 단위를 고

려한 버퍼 교체 방법이 필요하다. 향후 연구는 멀티미디어 객체를 논리적인 단위로 나누어 캐싱하는 방법과 이렇게 함으로써 발생되는 멀티미디어 객체 접근에 따르는 문제를 해결하고자 한다.

참 고 문 헌

[1] R. Alonso, D. Barbara, H. Garcia-Molina, "Data Caching Issues in an Information Retrieval System", ACM Trans. Database Systems, Volume 15, pages 359-384, 1990.

[2] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching", ACM Multimedia, 1994.

[3] A. Dan, D. M. Dias, and P. S. Yu, "Buffer Analysis for a Data Sharing Environment with Skewed Data Access", IEEE Trans. Knowledge and Data Engineering, Volume 6, pages 331-337, 1995.

[4] D. J. Gemmell, et al., "Multimedia Storage Servers: A Tutorial", IEEE Computer, pages 40-49, 1995.

[5] R. Karedla, J. S. Love, and B. G. Wherry, "Caching Strategies to Improve Disk System Performance", IEEE Computer, pages 38-46, 1994.

[6] H. Khalid, "The Unconventional Replacement Algorithms", ACM Computer Architecture News, pages 20-26, 1996.

[7] M. M. Nelson and W. T. Carson, 'A Practical Guide to Neural Nets', Addison-Wesley, first edition, 1991.

[8] E. J. O'Neil, P. E. O'Neil, and G. Weikum, "The LRU-K Page Replacement Algorithm For Database Disk Buffering", In ACM SIGMOD Conference, pages 297-306, Washington DC, May 1985.

[9] A. J. Smith, "Disk Cache-Miss Ratio Analysis and Design Considerations", ACM Trans. Computer Systems, Volume 3, pages 161-203, 1985.

[10] D. Thie'baut, H. S. Stone, and J. L. Wolf, "Improving Disk Cache Hit-Ratios Through Cache Partitioning", IEEE Trans. Computers, Volume 41, pages 665-676, 1992.

[11] A. Dan, Daniel M, and R. Mukherjee, et. al, "Buffering and Caching in Large-Scale Video Servers", Proceedings of IEEE Data Engineering Conference, pages 217~224, 1995.

[12] F. Moser, A. Kraib, and W. Klas, "L/MRP: A Buffer Management Strategy for Interactive Continuous Data Flows in a Multimedia DBMS", Proceedings of the 21th VLDB Conference, Zurich, Switzerland, pages 275-286, 1995.

[13] 정광철, 김준 , 허대영, "객체지향 DBMS에서 오프라인 미디어의 지원", 한국정보처리학회 '96 춘계 학술 발표 논문집, 제3권 1호, pages 627~631, 1996.

정 광 철

1987년 인하대학교 전자계산학과 졸업(학사)
1989년 인하대학교 대학원 전자계산학과 졸업(이학석사)
1990년~1992년 LG소프트웨어 연구원
1993년~현재 한국전자통신연구소 연구원
관심분야: 객체지향 & 멀티미디어 DBMS, 디지털 라이브러리

박 응 규

1984년 서강대학교 전자공학과 졸업(학사)
1986년 한국과학기술원 전기 및 전자 공학과 졸업(공학 석사)
1995년 한국과학기술원 전기 및 전자 공학과 졸업(공학 박사)
1986년~1990년 한국전자통신연구소 연구원
1991년~현재 서원대학교 전자계산학과 조교수
관심분야: 객체지향&멀티미디어 DBMS, 병렬 DBMS