주문형 오디오 시스템을 위한 웹 캐시 구조의 설계 및 평가

(Design and Evaluation of a Web Cache Architecture for Audio-On-Demand Systems)

이 태 원 † 심 마 로 † 배 진 욱 †† 이 석 호 †††
(Taewon Lee) (Maro Shim) (Jinuk Bae) (Sukho Lee)

요 약 인터넷을 통해 멀티미디어 데이타를 서비스하는 주문형 오디오 서비스(AOD, Audio On Demand)와 같은 시스템에서는 기존의 운영체제가 반복적으로 요청되는 데이타를 효과적으로 처리하지 못하고 있다. 본 논문에서는 웹 캐시(Web Cache) 구조를 제안한다. 이것은 과거 요청들과의 시간 간격 정보를 기초로 가까운 미래에 다시 요청될 곡들을 효과적으로 예측하고 웹 캐시에 유지하므로서 효율적인 서비스를 제공하도록 한다. 웹 캐시의 교체 전략으로는 LFRR(Least Frequently Requested Recently)을 제안한다. LFRR은 가까운 미래에 다시 요청될 확률이 적은 곡을 교체한다. 어느 한 곡이 다시 요청될 확률은 과거 요청들과의 시간 간격의 평균값이 작을수록 높게 된다. 제안된 웹 캐시의 이점은 디스크 액세스 횟수를 현저하게 줄일 수 있고 한정된 자원으로 더 많은 수의 동시 사용자를 지원할 수 있다는 것이다. 실제로 운영되고 있는 AOD 사이트의 요청 자료를 이용하여 제안된 웹 캐시를 시뮬레이션한 결과 높은 성능 향상을 얻을 수 있음을 보였다.

Abstract In the on-demand services like AOD(Audio On Demand) over the internet, existing operating systems cannot serve repeatedly requested data efficiently. This paper proposes a web cache architecture. It predicts the songs to be requested in near future, based on the intervals between the requests in the past on the same song and keeps the songs in the web cache. For the replacement strategy of the web cache, LFRR(Least Frequently Requested Recently) is proposed. LFRR replaces the song that has less probability to be requested in near future. The average of the intervals between the requests in the past and the new request is used as the probability of the requests. It is more likely to be requested in near future as the average is less. The web cache decreases the number of disk access extremely, and support to serve more users with restricted resources. From the simulation result based on the data at the AOD site currently operating, it is shown that the high performance enhancement is achieved.

1. 서 론

네트워크 속도가 향상되면서 인터넷을 통해 멀티미디어 데이타를 서비스하는 주문형(On Demand) 서비스가 생겨나고 있다. 대표적인 것으로 주문형 오디오 서비스(AOD, Audio On Demand)와 주문형 비디오 서비스(VOD, Video On Demand)를 들 수 있다.

이러한 서비스는 인터넷이 널리 알려지기 시작했던 몇 년 전만 해도 생각하지 못했던 종류의 서비스이다. 초기의 시스템과 네트워크는 텍스트나 이미지로 구성된 웹 페이지를 염두에 두고 구성이 되었다. 이미지와 텍스트로 구성된 웹 페이지는 시간의 지연이 있더라도 보는 데는 큰 문제가 없다. 반면에 주문형 오디오 서비스는 서비스되는 데이타가 시간적으로 연속된 흐름으로 구성

† 비 회 원 : 서울대학교 컴퓨터공학과
warrior@db.snu.ac.kr
maro@snucom.snu.ac.kr
†† 학생회원 : 서울대학교 컴퓨터공학과
oblody@db.snu.ac.kr
††† 종신회원 : 서울대학교 컴퓨터공학과 교수
shlee@comp.snu.ac.kr
논문접수 : 1999년 8월 9일
심사완료 : 2000년 2월 24일

되어 있어서 전송의 지연으로 인해 연속성이 유지되지 못할 경우 사용자들은 그 내용을 제대로 이해할 수가 없다. 그렇기 때문에 데이타가 사용되어야 할 시점에 사용자에게 도달할 수 있도록 실시간성이 보장되어야 한다.

주문형 오디오 서비스에서는 다른 곡에 비해 더 많이 요청되는 인기곡이 존재한다. 똑같은 곡이 반복적으로 요청되는 경우 불필요한 디스크 I/O 문제가 발생한다. 기존의 시스템에서는 버퍼 캐시 등을 이용해 이런 문제를 개선하려고 하고 있지만 주문형 서비스의 특성이 고려되지 않아 비효율적인 면이 많다.

본 논문에서는 주문형 서비스의 특성이 고려되지 않은 기존의 시스템을 사용하는 주문형 오디오 서비스에서 성능 향상을 가져오기 위해 웹 캐시 구조를 제안하고 웹 캐시에서 사용하는 교체 전략을 제시한다. 웹 캐시 구조를 통해서 불필요한 디스크 액세스 작업 등을 최소화해서 응답 시간을 줄이고 같은 자원으로 더 많은 사용자에게 곡을 서비스할 수 있음을 실험을 통해 보였다.

본 논문의 구성은 다음과 같다.

먼저 2장에서는 여러 가지 캐시 교체 전략 등 기존 연구에 대해 살펴본다. 3장에서는 주문형 오디오 서비스의 구조에 대해서 설명을 한다. 여기서는 실험에서 사용한 AOD 서비스에 기반해서 구조를 설명하고 있다. 4장에서는 기존의 운영체제 캐시에 대해 살펴보고, 여기서 사용하는 캐시 교체 전략이 주문형 오디오 서비스와 같은 주문형 서비스에서의 부적합성을 설명한다. 이를 개선하기 위한 웹 캐시 구조에 대해 설명하고 웹 캐시를 위한 교체 전략과 기대되는 성능 향상에 대해 설명한다. 5장에서는 실제로 주문형 오디오 서비스 상에서 이루어진 사용자들의 요청을 본 논문에서 제안한 웹 캐시를 이용해 구현했을 때 어느 정도의 성능 향상이 있을 수 있는지를 실험을 통해 보인다. 마지막으로 6장에서는 결론을 내린다.

2. 관련 연구

주문형 오디오 서비스나 주문형 비디오 서비스와 같은 주문형 서비스에서 성능을 향상시키기 위한 연구는 주문형 오디오 서비스에 대해서는 전혀 이루어지지가 않았고, 영상과 오디오를 함께 다루는 주문형 비디오 서비스 쪽에서 주로 이루어졌다. 이는 오디오가 비디오에 포함된 하나의 요소이기 때문에 그 관심도가 적었던 이유라고 볼 수 있다. 주문형 비디오 서비스에서 성능 향상에 대한 연구는 크게 멀티미디어 저장 장치 면에서의 성능 향상과 전송 상에서의 성능 향상으로 나누어 볼 수가 있다. 오디오에 대해서는 특정 곡에 대해 요청이 집중되는 경우에 그 부하를 분산시키는 것에 대한 연구와 요청이 집중될 곡을 미리 예측하여 효과적으로 서비스하는 방법에 대한 연구가 필요하다.

주문형 비디오 서비스에 대해서는 이런 연구가 많이 수행되어 왔는데, 대표적으로 Cohen[1]은 특정 영화에 대해 요청이 집중되는 것을 막기 위해 디스크 배열(disk array)을 이용한다. 디스크 배열은 G개의 그룹으로 나누어 지고, 한 영화를 세그먼트(segment)라는 단위로 나누어 세그먼트를 전체 각 그룹에 라운드-로빈(round-robin) 방식으로 저장한다. 하나의 그룹 내에서는 각 디스크에 균등한 크기로 저장한다. 이렇게 해서 각 디스크에 효과적으로 요청을 분산시킬 수 있다. 그러나 이것은 디스크 배열이라는 별도의 기기를 필요로 한다. Shenoy[2]는 파일 서버에서 효과적으로 비디오 데이타를 분산시켜 저장하는 것에 대해 연구했다. 이것은 미리 데이타를 부하가 적게 걸리도록 분산시켜 놓는 것인데, 오디오 데이타의 경우에는 그 크기가 매우 작아 분산시키는 데서 발생하는 부하가 더 클 수도 있다.

멀티미디어 데이타의 경우 데이타가 소비되기 전에 클라이언트에 도착하는 것을 보장해야 한다. Chen[3]은 멀티미디어 서비스가 만족시켜야 하는 문제들에 대해 제시하고 있다. 서비스가 끊기지 않기 위해서 필요한 요구 조건을 제시하고 있는데, 이는 주문형 오디오 서비스에서도 적용되어야 한다. 한정된 버퍼를 이용해 멀티미디어 데이타를 연속적으로 서비스해 주는 방법에 대한 연구도 있다. Dan[4]은 비디오 서버에서 버퍼링과 캐시에 대한 문제를 다루고 있다. 주문형 오디오 서비스에서도 버퍼링이 중요한데, 주문형 비디오 서비스에서 이루어진 연구는 하나의 비디오도 모두 버퍼링을 할 수 없기 때문에, 비디오 데이타 중 어느 부분을 어떻게 버퍼링을 해야 하는지를 다룬다. 그러나 주문형 오디오 서비스에서는 어떤 곡들이 자주 요청되고, 이를 어떻게 버퍼링해야 하는지가 연구되어야 한다. 주문형 비디오 서비스 서버에서 동적으로 버퍼를 관리하는 기법에 대해 제시하고 있는 연구도 있다[5]. 비디오 데이타와 오디오 데이타의 크기 차이로 버퍼 관리 기법 상에서 고려해야 할 문제가 다르지만 아직까지는 비디오 데이타에 대한 연구만이 이루어지고 있다.

교체 전략에 대한 연구에서는 미래를 더 정확히 예측하기 위한 다양한 기법들이 제시되어 왔다. FIFO는 가장 먼저 캐시에 들어온 데이타를 가장 먼저 교체하는 방법을 사용한다. 가장 대표적인 교체 기법인 LRU는

가장 오래 전에 사용된 데이타를 캐시에서 교체하는 방법이고, LFU는 가장 적은 회수로 요청된 데이타를 교체하는 방법이다. LRU-Threshold[6]는 LRU와 비슷하지만 특정 크기 이상의 데이타는 전혀 캐시에 저장하지 않는다. 이외에도 Lowest-Latency-First[7]은 전송 지연 시간이 가장 적은 데이타부터 교체해서 평균 전송 지연 시간을 최소화하는 방법이다. 최근에는 웹 상에서 범용적으로 사용할 수 있는 캐시에 대한 연구가 진행되어 왔다. 웹에서는 요청되는 데이타의 크기가 일정하지 않기 때문에 기존의 LRU를 확장한 SLRU, PSS[8]와 같은 교체 전략을 제안하고 있다. SLRU에서는 캐시에 있는 각 데이타에 대해 그 데이타가 요청된 후에 다른 데이타에 대해 몇 번의 요청이 발생했는지를 나타내는 dynamic frequency라는 값을 정의한다. 캐시에 존재하지 않는 데이타가 새로 요청된 경우 데이타가 들어갈만큼의 데이타를 교체하되 dynamic frequency의 합이 최소가 되는 데이타들을 캐시로부터 교체한다.

3. 주문형 오디오 서비스의 구조

일반적으로 웹 서비스는 클라이언트 - 웹 서버 - 데이타베이스 서버의 3-tier 방식의 구조를 가진다. 클라이언트가 웹 서버에게 데이타를 요청하면 웹 서버는 클라이언트가 원하는 데이타를 데이타베이스 서버에게 요청해서 클라이언트에게 되돌려주는 구조를 가진다.

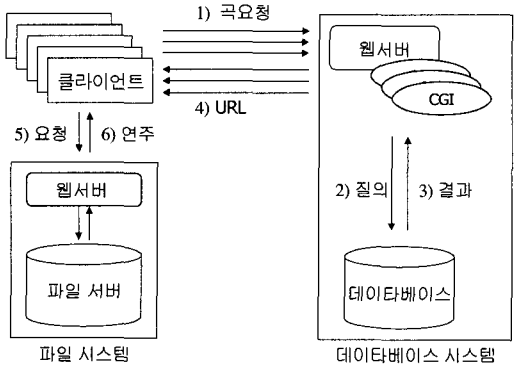

그림 1 주문형 오디오 서비스의 구조

현재까지 인터넷상에 구현되어 있는 주문형 오디오 서비스의 일반적인 구조는 3-tier 구조에 기반해서 웹 서버와 데이타베이스 서버, 그리고 파일 시스템을 연결하여 멀티미디어 파일을 저장하고 서비스해 준다. 오디오 데이타에 대해 효과적으로 검색을 지원하는 멀티미디어 데이타베이스가 아직 없기 때문에 주문형 오디오 서비스는 실제 데이타를 화일 시스템에 저장해 놓고, 요청이 들어오면 화일 시스템으로부터 서비스를 해 주는 구조를 가진다. 구조를 도시하면 [그림 1]과 같다[7].

주문형 오디오 서비스에서 일어나는 작업의 전체적인 흐름은 다음과 같다.

① 클라이언트가 곡을 요청.
② 웹 서버의 적절한 CGI 프로그램이 수행.
③ CGI가 데이타베이스에게 곡에 대한 정보를 질의.
④ 질의 수행 결과를 클라이언트에게 되돌려 줌.
⑤ 클라이언트는 URL 정보를 이용해 화일 서버에게 실제 데이타를 요청.
⑥ 화일 서버의 곡 요청을 받은 웹 서버는 화일 시스템으로부터 곡을 읽어들여 오디오 플레이어에게 돌려준다.

주문형 오디오 서비스의 구조를 보면 크게 곡에 대한 정보를 제공하는 데이타베이스 서버 부분과 실제 곡을 제공하는 화일 시스템 부분의 두 부분으로 나눌 수 있다.

주문형 오디오 서비스에서는 인기 있는 특정 곡이 자주 요청되는 현상이 발생한다. 이로 인해 서버에서는 불필요한 작업이 반복적으로 수행된다. 데이타베이스 서버에서는 똑같은 질의를 여러 번 수행하게 되고 화일 서버에서는 디스크의 같은 부분을 반복적으로 접근하게 된다. 이에 대해 상용 데이타베이스 서버[9]에서는 디스크의 캐시처럼 최근에 요청되었던 질의와 그 결과를 유지하여 같은 질의가 요청되면 효과적으로 처리를 하고 있다. 화일 서버는 최근에 요청된 디스크 페이지를 버퍼 캐시에 보관해서 문제를 해결한다.

주문형 오디오 서비스에서 데이타베이스는 기존의 응용과 다른 점이 없지만, 화일 서버에서는 일반적인 응용보다 수십~수백 배의 크기에 이르는 요청이 일반적으로 발생하기 때문에 기존의 디스크 페이지 교체 기법들이 적합한지를 점검해 보아야 한다. 4장에서는 주문형 오디오 서비스와 같은 대용량 데이타를 사용하는 응용에서 LRU 등 기존의 교체 기법의 부적합성을 설명하고 이에 대한 개선으로 본 논문에서 제안하고 있는 웹 캐시 구조에 대해 설명하겠다.

4. 웹 캐시

기존의 운영체제는 버퍼 캐시에 존재하지 않는 디스크 페이지가 요청되었을 때 어느 한 교체 기법에 따라 버퍼에 있는 어느 한 페이지를 교체한다. 특별히 LRU 교체 기법에서는 버퍼에 존재하지 않는 디스크 페이지가 요청되면 버퍼에 존재하는 페이지 중 가장 오래 접

근되지 않은 페이지를 교체해서 자주 요청되는 페이지를 항상 버퍼에 유지하도록 한다.

디스크의 페이지 크기를 $Size_p$, 버퍼 캐시의 크기를 $Size_c$라고 하면, 버퍼에 유지될 수 있는 페이지의 수는 $\lfloor Size_c/Size_p \rfloor$이다. 디스크의 페이지는 보통 8~16KB 정도의 크기를 가진다. 어느 한 페이지 A가 가장 오랫동안 요청되지 않은 페이지가 되어 버퍼로부터 교체되어 나가려면 페이지 A가 재요청되기 전에 최소한 $\lfloor Size_c/Size_p \rfloor$의 상이한 페이지가 요청되어야만 가능하다.

주문형 오디오 서비스에서 곡 하나의 크기를 $Size_{song}$라고 하면 버퍼 캐시에 유지할 수 있는 곡의 수는, $N_c = \lfloor Size_c/Size_{song} \rfloor$이다. 또 디스크에서 주문형 오디오 서비스에 사용되는 곡이 차지하는 전체 공간을 $Size_D$라고 하자. 그러면 약 $N_D = \lfloor Size_D/Size_{song} \rfloor$의 곡이 디스크에 존재할 수 있다.

주문형 오디오 서비스에서는 특정 인기 있는 곡이 반복적으로 요청되는 특징이 있다. 그렇지만 이런 곡 외에도 요청 빈도가 적기는 하나 여러 사용자에 의해 다양한 곡이 요청된다. 특정 곡 A가 인기가 있어서 자주 요청된다고 하면 A는 버퍼에 존재할 것이다. A가 계속해서 버퍼에 존재하려면 서로 다른 N_c개의 곡이 요청되기 전에 A가 재요청되어야 한다. 그렇지 않은 경우 A는 교체 전략에 의해 버퍼로부터 교체된다.

그런데 버퍼 캐시에 존재하는 곡의 수, N_c는 디스크에 존재하는 전체 곡의 수, N_D에 비해 매우 작다. 이런 상황에서 만일 A라는 곡이 자주 요청된다고 해도 $N_D(>> N_c)$개의 많은 곡에 대해서 요청이 발생하기 때문에 자주 요청되지 않는 곡으로 인해 교체될 확률이 매우 높다. 즉, 기존의 운영체제에서는 주문형 오디오 서비스와 같은 응용의 특성을 고려하지 않은 범용적인 교체 기법을 사용하기 때문에 이런 문제가 발생하며 따라서 큰 효과를 얻을 수 없다.

웹 캐시(Web Cache)는 운영체제의 버퍼 캐시와는 별도로 웹 서버에서 관리하는 버퍼용 메모리를 말한다. 주문형 오디오 서비스를 위한 버퍼를 별도로 관리함으로써 주문형 오디오 서비스의 특성을 고려한 교체 기법을 적용시킬 수 있고, 이로 인해 동일한 시스템 환경에서 더 효율적인 서비스가 가능하다.

기존 운영체제는 요청된 디스크 페이지가 버퍼에 존재하지 않고 버퍼 캐시가 꽉 차 있는 경우 교체 전략에 따라 항상 버퍼 캐시로부터 어느 한 페이지를 교체한다. 이런 경우에 주문형 오디오 서비스에서는 버퍼 캐시에 유지될 수 있는 곡의 수가 적기 때문에 자주 요청되는 곡이더라도 자주 요청되지 않는 곡에 의해 버퍼 캐시에서 교체되는 경우가 발생할 수 있다. 이렇게 자주 요청되는 곡이 부득이하게 교체되는 것을 방지하기 위해서는 곡에 대한 요청 패턴을 분석하여 자주 요청되는 곡을 가능한 한 캐시에 유지하는 전략을 사용해야 한다.

이를 위해 웹 캐시에서는 LFRR(Least Frequently Requested Recently)이라는 새로운 교체 전략을 고안했다. LFRR에서는 현재 웹 캐시에 유지하고 있는 곡 중 가까운 미래에 다시 요청될 가능성이 가장 적은 곡을 교체의 후보로 정한다.

버퍼 캐시나 웹 캐시에서 교체 전략을 사용하는 이유는 미래를 정확하게 예측하지 못하기 때문이다. 가까운 미래에 요청이 발생할 것을 미리 안다면 어떤 페이지를 교체해야 최적의 성능을 낼 수 있는지를 알 수 있지만, 그렇지 못한 상황에서는 과거에 대한 정보로부터 미래를 예측해야 한다. LFRR에서도 미래를 알 수 없기 때문에 대신 최근에 가장 자주 요청되지 않은 곡을 교체의 후보로 선택한다.

LFRR을 위해서는 먼저 최근에 가장 자주 요청되지 않은 곡의 명확한 정의를 내려야 할 필요가 있다. 자주 요청되었다는 것은 새로 곡이 요청되었을 때 그보다 과거에 일어난 요청에 대해 각 요청들과의 시간 간격의 평균이 작을수록 더 자주 요청되었다고 할 수 있다. 이를 구체적으로 설명하면 다음과 같다.

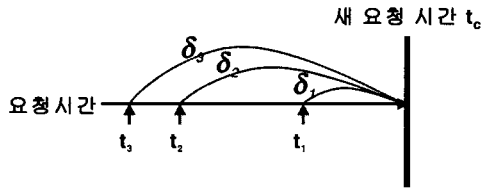

그림 2 요청 패턴의 예

[그림 2]는 새로운 요청이 들어온 시점을 기준으로 과거에 같은 곡을 요청했던 때와의 시간 간격을 표시한 것이다. 같은 곡에 대해 지난 요청들 사이의 간격을 통해 얼마나 자주 곡이 요청되고 있는지를 알 수 있다.

$f_n(id)$를 곡 id에 대해 새로운 요청과 과거 n개의 요청들 사이의 시간 간격(δ)의 평균, 즉

$$f_n(id) = \frac{\delta_1 + \delta_2 + \delta_3 + \cdots + \delta_n}{n} = \frac{(t_c - t_1) + (t_c - t_2) + (t_c - t_3) + \cdots + (t_c - t_n)}{n}$$

을 나타낸다고 정의하자. 가까운 미래에 다시 요청될 가

능성이 높은 곡은 최근에 자주 요청되었던 곡이다. $f_n(id)$를 이렇게 정의하면 최근에 자주 요청되었던 곡은 $f_n(id)$값이 더 작은 곡임을 알 수 있다.

다음은 LFRR 교체 전략의 알고리즘을 나타낸 것이다.

```
LFRR(A : 요청된 곡)
    if (웹 캐시에 A가 존재) then
        return HIT;
    else
        if (웹 캐시에 있는 곡보다 더 가까운 미래에 요청
         될 가능성이 큼) then
            if (웹 캐시가 full)
                최근에 가장 자주 요청되지 않은 곡
                과 교체;
                return MISS;
            else
                웹 캐시에 로드;
                 return MISS;
            fi
        else
            메모리로부터 직접 제공(웹 캐시에 로드하지
            않음)
            return MISS;
        fi
    fi
```

웹 캐시에서는 기존의 운영체제가 버퍼 캐시를 디스크의 페이지 단위로 관리하는 것과는 달리 곡 단위로 관리를 한다. 따라서 곡의 일부만이 버퍼 캐시에 남아 있는 일은 발생하지 않는다. 이렇게 하는 이유는 사용자들이 곡을 요청했을 때 끝까지 감상을 하는 경우가 많고, 곡을 페이지 단위로 관리한다면 한 곡 당 수 백 페이지를 관리해야 하는 부담이 있기 때문이다. 실제로 페이지 단위와 곡 단위로 관리했을 경우를 모두 실험한 결과, 차이를 보이지 않았다.

이처럼 웹 캐시에서는 LFRR이라는 교체 전략을 통해 기존의 버퍼 캐시가 주문형 오디오 서비스를 효과적으로 지원하지 못했던 문제를 해결하고 있다. 다음 장에서는 주문형 오디오 서비스에서 웹 캐시를 사용했을 때 얻을 수 있는 성능 향상을 실험을 통해 평가하였다.

5. 실 험

실험은 실제로 운영되고 있는 주문형 오디오 서비스 사이트(http://aod.snu.ac.kr)를 대상으로 하였다[10]. 이 사이트는 사용자가 6,000명 정도이고 하루 평균 1,000명이 접속을 한다. 실제로 운영되는 사이트에서의 데이타를 가지고 웹 캐시를 시뮬레이션했기 때문에 그 결과는 매우 전형적인 환경에서의 결과라고 할 수 있다.

주문형 오디오 서비스 사이트는 4개의 화일 서버에 분산된 mp3 데이타들을 서비스한다. 이 중 하나의 서버에 존재하는 화일들에 대해 1998년 4월부터 1998년 7월까지 요청되었던 자료들을 토대로 시뮬레이션을 수행하였다. 자료는 웹 서버의 로그 데이타로부터 추출하였다.

웹 캐시의 크기는 80~200(MB)까지 40MB 단위로 변화시켜 가면서 실험을 하였다. 화일 서버에는 400여곡이 존재한다. 이 곡들의 평균 크기를 4MB라고 했을 때 웹 캐시에 들어갈 수 있는 곡의 수는 20~50개 정도이다. 웹 캐시의 크기를 80MB로 했을 경우 전체의 1/20 정도에 해당하는 곡을 관리할 수 있게 된다. 버퍼 캐시를 시뮬레이션하기 위해서 교체 기법으로는 LRU를 사용하였고, 교체 단위는 8KB의 페이지 단위로 하였다. 이는 기존 시스템에서의 캐시를 좀 더 정확하게 시뮬레이션하기 위한 목적이다.

5.1 운영체제 캐시와 웹 캐시의 비율에 따른 적중률 변화

이 부분에서는 웹 캐시를 사용할 때 버퍼 캐시만 사용한 기존의 운영체제에 비해 얻어지는 성능 향상에 대해 실험했다. 구체적으로 버퍼 캐시의 비율을 100%부터 0%까지 조금씩 낮춰가면서 웹 캐시와 병행하여 사용할 때 캐시 적중률이 어떻게 변화하는지를 실험하였다. 캐시의 크기는 80~200MB까지 40MB씩 변화시켰다. 실험 결과는 [그림 3]과 같다.

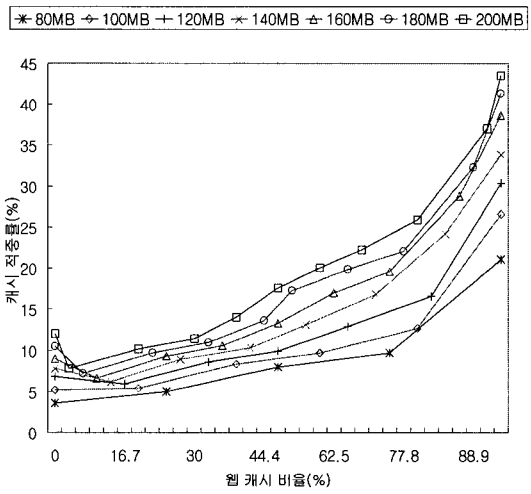

그림 3 운영체제 캐시와 웹 캐시 비율에 따른 적중률

[그림 3]을 분석해보면 웹 캐시의 비율이 증가할수록 전체적인 캐시 적중률이 증가되는 것을 알 수 있다. 웹 캐시만을 100% 사용했을 경우 일반 버퍼 캐시만을 사용한 경우에 비해서 4~5배의 적중률 증가를 보이고 있다. 그림에서 웹 캐시의 비율이 0%인 경우가 버퍼 캐시만을 사용하는 경우이다. 제일 아래쪽의 그래프가 캐시 메모리의 크기가 80MB인 경우를 나타내는데 버퍼 캐시만을 사용했을 때 캐시 적중률이 5%가 안 되는 것을 알 수 있다.

만일 모든 곡이 똑같은 확률로 요청된다고 하면 다음에 요청되는 곡이 캐시 메모리에 존재할 확률은 1/20, 즉 5%이다. 즉, 버퍼 캐시만을 사용했을 때의 적중률이 임의의 곡을 교체하는 전략보다도 비효율적이라는 것을 의미한다.

캐시의 적중률 증가가 동시 사용자 수에는 어떤 영향을 미치는지 알아보기 위해 로그 데이타를 토대로 동시 사용자 수를 비교하는 실험을 수행하였다. 실험은 디스크의 대역폭을 33Mb/sec로 설정하고 수행하였다. 웹 캐시만 사용했을 때와 버퍼 캐시만을 사용했을 때를 실험했다. 캐시의 크기는 이전의 실험과 같이 80~200MB까지 40MB씩 변화시켜 가며 실시하였다.

실험 결과를 살펴보면 웹 캐시를 이용할 때 동시 사용자 수가 버퍼 캐시만을 이용할 때 보다 약 15% 정도 증가하는 것을 볼 수가 있다. 버퍼 캐시만을 이용하면 80MB를 캐시 메모리로 사용할 때 최대 350명 정도의 동시 사용자를 수용할 수 있다는 결과를 얻었는데, 웹 캐시를 이용하면 50명 정도 더 많은 사용자를 수용할 수가 있다. 웹 캐시를 통해서 캐시 적중률을 향상시킬 수 있기 때문에 디스크의 대역폭을 좀 더 효율적으로 사용하게 되고 동시 사용자 수가 증가되는 결과를 가져오는 것이다.

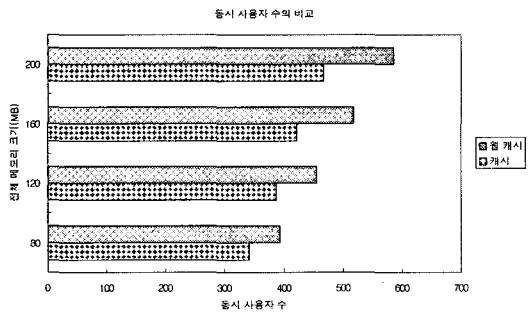

그림 4 동시 사용자 수의 비교

5.2 이상적인 캐시와 웹 캐시의 비교

본 논문에서 제안한 웹 캐시의 성능을 비교평가하기 위해 이상적인 캐시와 비교를 해 보았다. 이상적인 캐시[11]는 미래를 정확히 알고 그에 따라 forward-distance(미래에 다시 요청되는 시점까지의 시간)가 가장 큰 페이지를 교체하기 때문에 이보다 적중률이 높은 캐시는 존재할 수가 없다. 버퍼 캐시 크기는 역시 80~200MB까지 40MB 단위로 변화시켜 가며 실험을 하였다.

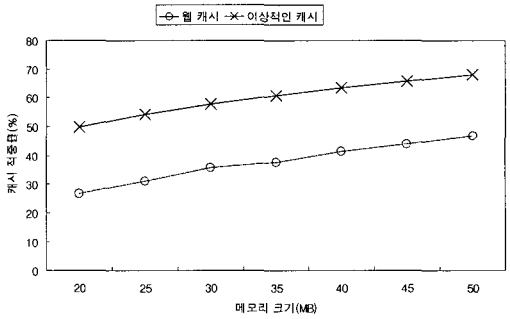

그림 5 웹 캐시와 이상적인 캐시의 비교

[그림 5]의 그래프에서 보면 웹 캐시가 이상적인 캐시의 50~60% 정도의 성능을 보이고 있다.

6. 결 론

본 논문에서는 기존의 웹 환경에서 주문형 오디오 서비스를 할 때 운영체제의 버퍼 캐시에 웹 캐시라는 새로운 구조를 도입하여 동일한 시스템 환경 하에서 전체적으로 효율을 높일 수 있는 구조를 제안하였다. 버퍼 캐시가 주문형 오디오 서비스에 부적합한 이유를 설명하고, 이를 개선하기 위한 웹 캐시 구조를 제안하였고, 웹 캐시에서 사용하는 효과적인 교체 기법인 LFRR을 제안하였다.

성능 평가를 위해 실제로 운영중인 AOD 사이트의 데이타를 이용해 웹 캐시를 기존의 버퍼 캐시와 병행하여 사용할 경우 캐시 적중률이 4~5배까지 높아지는 것을 실험을 통해 보였다. 또한 가장 이상적인 경우의 캐시를 정의하고 이상적인 캐시와 본 논문에서 제안한 웹 캐시와의 성능을 비교했다. 이 경우 이상적인 캐시의 50~60%에 해당하는 적중률을 보임을 확인했다.

아직 발생하지 않은 미래의 요청 패턴을 정확하게 예측하는 것은 어렵다. 실험에서보다 더 좋은 적중률을 얻기 위해서는 본 논문에서 제안한 교체 전략을 좀 더 다양화하는 연구가 필요하다. 이상적인 캐시와 같은 적중률을 보일 수는 없겠지만, 주문형 오디오 서비스의 특징

을 충분히 고려한 교체 전략으로 효율적인 웹 캐시를 설계할 수 있을 것이다.

참 고 문 헌

[1] Ariel Cohen, Walter A. Burkhard, P. Venkat Rangan, "Pipelined Disk Arrays for Digital Movie Retrieval," *Proceedings of ICMCS*, May 1995.

[2] P. Shenoy and H.M. Vin. "Efficient Striping Techniques for Multimedia File Servers," *Proceedings of the 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97)*, pp. 25~36, May 1997.

[3] H.J. Chen, T.D.C. Little Storage "Allocation Policies for Time-Dependent Multimedia Data," *IEEE Trans. on Knowledge and Data Engineering*, 1996.

[4] Asit Dan, Dan Dias, Rajat Mukherjee, Dinkar Sitaram, Renu Tewari "Buffering and Caching in Large-Scale Video Servers," *Compcon - Technologies for the Information Superhighway, Digest of Papers, IEEE Computer Society. Los Alamitos, CA, IEEE Computer Society Press*, pp. 217~224 (RC 19903 - January 4, 1995), 1995

[5] Y. S. Ryu, K. Koh, "A Dynamic Buffer Management Technique for a Video-on-Demand Server," *Proceedings of ICMCS(International Conference on Multimedia Computing and Systems*, Yokohama, Japan, Mar. 1996.

[6] M. Abrams, C.R. Standbridge, G.Abdula, S.Williams and E.A. Fox, "Caching Proxies ; Limitaions and Potentials," WWW-4, Boston Conference, Dec. 1995.

[7] 이 태원, 심 마로, 배 진욱, 이 석호, "Audio On Demand 를 위한 웹 캐시 구조", '98 가을 학술발표논문집(1), 제25권 제2호, pp. 295~297, 1998.10

[8] Charu Aggarwal, Joel L. Wolf, and Philip S. Yu, "Caching on the World Wide Web," *IEEE TKDE*, Jan/Feb, 1999.

[9] Oracle7 Server Concept Manual.

[10] http://aod.snu.ac.kr

[11] Aho, A. V., P.J.Denning, and J.D. Ullman, "Principles of Optimal Page Replacement," *Journal of the ACM, Vol. 18, No. 1*, pp. 80~93, Jan. 1971.

이 태 원

1997년 서울대학교 컴퓨터공학과 졸업(학사). 1999년 서울대학교 컴퓨터공학과 졸업(공학석사). 1999 ~ 현재 서울대학교 컴퓨터공학과 박사과정. 관심분야는 XML, 웹캐시 등임.

심 마 로

1993년 서울대학교 컴퓨터공학과 졸업(학사). 1995년 서울대학교 컴퓨터공학과 졸업(공학석사). 1995년 ~ 현재 서울대학교 컴퓨터공학과 박사과정. 관심분야는 병렬 데이타베이스 등임.

배 진 욱

1996년 서울대학교 컴퓨터공학과 졸업. 1998년 서울대학교 컴퓨터공학과 석사학위 취득. 1998년 3월 ~ 현재 서울대학교 컴퓨터공학과 박사과정. 관심분야는 데이타마이닝, 데이타웨어하우스, 공간인덱스

이 석 호

1964년 연세대학교 정치외교학과 졸업. 1975년, 1979년 미국 텍사스대학교 전산학 석사와 박사학위 취득. 1979년 ~ 1982년 한국과학원 전산학과 조교수. 1982년 ~ 1986년 한국정보과학회 논문편집위원장. 1986년 ~ 1988년 한국정보과학회 부회장. 1988년 ~ 1989년 미국 IBM T.J. Watson 연구소 객원교수. 1988년 ~ 1990년 데이터베이스연구회 운영위원장. 1989년 ~ 1991년 서울대학교 중앙교육연구전산원 원장. 1994년 한국정보과학회 회장. 1997년 ~ 현재 한국학술진흥재단 부설 첨단학술정보센터 소장. 1982년 ~ 현재 서울대학교 컴퓨터공학부 교수. 관심분야는 데이타베이스, 멀티미디어 데이타베이스