2001 년 한국정보처리학회 춘계 학술발표논문집 제 8 권 제 1 호

인터넷 환경에서의 비디오서버 캐슁 알고리즘

임일명*, 육현규*, 박성순**, 박명순*
*고려대학교 컴퓨터학과
**안양대학교 컴퓨터학과
e-mail : alias@ilab.korea.ac.kr

Continuous Media Caching in Video Servers in Internet

Il-Myong Yim*, Hyun-Gyoo Yook*, Sung-Soon Park**, Myong-soon Park*
* Dept. of Computer Science, Korea University
** Dept. of Computer Science and Engineering, Anyang University

요 약

네트워크 기술의 발전으로, 인터넷을 통한 주문형 비디오 서비스가 가능해지고 있다. 주문형 비디오 시스템은 대용량 데이터를 실시간에 전송할 필요가 있기 때문에 디스크 전송대역폭은 매우 중요한 자원이다. 동시 다수에게 끊긴 현상이 없는 원활한 스트림 서비스를 하기 위해서는 매우 큰 디스크 대역폭이 필요하다. 비디오 스트림 데이터를 캐슁하여 디스크 대역폭을 절약하면, 한정된 디스크 대역폭에서 더 많은 스트림을 제공할 수 있다. 논문은 보다 향상된 비디오 스트림 캐슁 알고리즘을 제안한다. 제안된 알고리즘은 기존의 방법들 보다 더 많은 스트림을 제공하고, 스트림 재생의 지연을 줄인다.

1. 서론

시간이 갈수록 인터넷을 통한 서비스는 계속 증가하고, 사람들의 생활에 없어서는 안될 요소가 되어가고 있다. 그 중에서 멀티미디어 서비스는 매우 중요한 부분으로 자리잡고 있다.

인터넷을 통한 화상회의나 주문형 비디오 서비스 등을 실현하기 위해서 네트워크 기반 기술의 발달과 영상 압축 기술, 비디오 스트리밍 기술 등의 지속적인 연구가 이루어지고 있다.

주문형 비디오 시스템은 대용량 데이터를 동시에 여러 사용자에게 실시간으로 보내야 한다. 이러한 시스템에서, 네트워크 대역폭과 디스크 대역폭은 매우 중요한 자원이다. 비디오 데이터의 캐쉬를 통해서, 네트워크 대역폭이나 디스크 대역폭을 줄일 수 있다.

비디오 데이터는 다른 응용 프로그램의 데이터에 비해서, 매우 크고, 클라이언트에게 실시간에 전송되어야 하는 특징이 있다. 또한, 데이터 파일을 앞에서부터 뒤로 연속적으로 읽어나가는 특징이 있다. 이러한 특징들 때문에 LRU 와 같은 전통적인, 파일 시스템의 캐슁 방법은 비디오 데이터의 캐슁에는 적합하지 않다.

이전에, 비디오 서버를 위한 여러 캐슁 알고리즘이 소개되었다[7][8]. 첫번째로, 같은 비디오에 대한 서비스 요구들의 시간적 지역성을 이용한 인터벌 캐슁(Interval Caching)이 있다[5]. 같은 스트림을 재생하는 연속된 스트림 사이의 시간차 만큼의 데이터를 캐슁하는 방법이다. 동일한 비디오 데이터를 재생하는 두 개의 스트림에 대해서, 후행하는 스트림은 선행하는 스트림이 캐슁한 데이터블록을 캐쉬에서 읽어 들여 디스크 접근이 없이 스트림 서비스를 가능하게 하는 알고리즘이다. 한번 캐슁을 시작하면, 재생이 끝날 때까지 절대로 중단되지 않는 특징이 있다.

두번째로, 선점형 인터벌 캐슁(Preemptive but Safe Interval Caching)이 있다[2]. IC 의 캐슁을 중단할 수 없는 특징은, 캐슁중인 데이터 블록의 크기보다 시간차가 더 작은 스트림이 있는데도 불구하고, 캐슁할 수 없는 단점이 있다. 이 단점을 개선하여, 동일 디스크 대역폭과 캐쉬 메모리를 사용하여 더 많은 스트림을 제공하는 알고리즘이다.

사용자는 비디오를 선택할 때 여러 비디오의 앞 부분을 본 후에 비디오를 선택하는 경향이 있다. 하지만, 기존의 연구들은 이러한 형태의 사용자의 요구에는 적합하지 않은 특징을 가지고 있다.

본 논문은, 주문형 비디오 서비스를 위한 개선된 캐쉥 알고리즘을 제안한다. 제안된 알고리즘은 일반적인 상황 외에, 사용자들이 비디오를 조금씩 재생해본 후 계속 볼지 여부를 결정하는 상황에서도 다른 알고리즘들에 비해서, 더 높은 성능을 보여준다.

논문은 나머지는 다음과 같이 구성되어있다. 2 절에서는 이전연구를 소개하고, 3 절에서는 제안된 캐쉥 알고리즘을 소개한다. 4 절에서는 시뮬레이션 수행과 결과를 분석하고, 4 절에서는 결론을 제시한다.

2. 이전연구

이 절에서, 두 가지 비디오 스트림 캐쉥 알고리즘을 설명한다. 첫번째는 인터벌 캐쉥이고(Interval Caching)이고, 두 번째는 선점형 인터벌 캐쉥(Preemptive but Safe Interval Caching) 이다. 이 두 가지 알고리즘은 끊김이 없는 스트림 서비스를 제공한다.

2.1. 인터벌 캐쉥 (Interval Caching)

같은 비디오에 대한 서비스 요구들의 시간적 지역성을 이용한 캐쉥이다. 같은 비디오를 재생하는 연속된 두개의 스트림이 읽고 있는 데이터 블록을 인터벌이라고 한다. 동일한 비디오 데이터를 재생하는 두개의 스트림에 대해서, 후행하는 스트림은 선행하는 스트림이 저장한 인터벌을 캐쉬에서 읽어 들여 디스크 접근이 없이 스트림 서비스를 가능하게 하는 알고리즘이다.

그러나 캐쉬 되어있는 인터벌은 비디오가 끝나거나 중지될 때 까지 절대로 다른 인터벌에 의해서 교체될 수 없다. 그림 1 은 인터벌 캐쉬의 동작방식을 보여준다.

(a) 만일 같은 비디오에서 서비스되고 있는 선행 스트림이 없을 경우 새로운 스트림(S_{11})의 승인에 의해서, 인터벌은 생성되지 않는다.

(b) 선행 스트림이 있을 경우, 새로운 인터벌(I_1) 이 생성된다. 인터벌의 길이는 선행 스트림과 후행 스트림의 시간차에 의해서, 결정된다. 그림 1(b)에서, 인터벌의 길이는 5 이다.

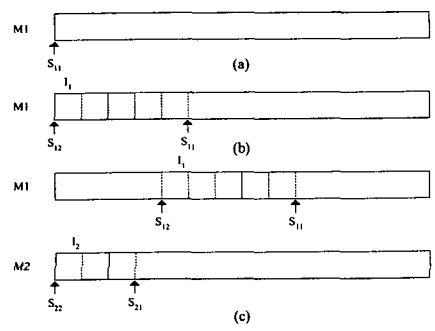

그림 1. 인터벌캐쉥 알고리즘

(c) 만일 캐쉬공간이 없다면, 캐쉬 저장된 인터벌(I_1) 보다 작은 크기를 가지고도 새로 생성된 인터벌(I_2)은 캐쉬에 저장될 수 없다..

디스크의 대역폭이나 버퍼의 저장공간에 여유가 있을 경우 새로운 스트림이 승인된다. 새로 만들어진 인터벌은 이용 가능한 캐쉬 공간이 인터벌의 크기보다 크거나 같으면, 캐쉬가 될 수 있다. 캐쉬에 사용된 공간은 선행 스트림이 종료되면 해제된다.

인터벌이 캐쉬에 저장되고 있는 후행 스트림은 디스크에 부하를 전혀 주지않고, 캐쉬에서만 필요한 데이터 블록을 얻을 수 있다. 이것은 캐쉬가 유지하는 인터벌의 수가 많을수록 더 많은 스트림의 제공이 가능하다는 뜻이다. 이것은 시스템 성능의 병목현상이 디스크 입출력 대역폭인 경우에 매우 중요하다. 불행하게도, 인터벌 캐쉥은 중요하지 않은 데이터 블록을 캐쉬에 저장하는 경우가 발생한다. 비디오와 같은 멀티미디어 데이터의 경우 2 시간정도의 길이에 달한다. 그래서, 중요하지 않는 인터벌이 오랜 기간동안 캐쉬에 머무르는 일이 발생한다. 이런 문제점을 해결하기 위해서, 선점 가능한 인터벌 캐쉥이 제안되었다.

2.2. 선점형 인터벌 캐쉥 (Preemptive but Safe Interval Caching)

각각의 스트림들은 그림 2 에 나와있는 상태 변이도를 따른다. 하나의 스트림 서비스가 승인되면, 서비스 그룹 상태에서 서비스된다. 인터벌이 캐쉬에 저장되기로 결정되면, 그 인터벌의 후행 스트림은 일시적 캐쉬상태에 들어간다. 인터벌의 길이가 1 초 라면, 그 인터벌을 가진 후행 스트림은 1 초가 지난 후에 캐쉬 상태가 된다. 일시적 상태의 스트림은 디스크에서 서비스된다. 그리고, 캐쉬된 상태에서는 캐쉬에서, 읽혀져 서비스된다. 스트림이 재생을 모두 마치면, 종료상태가 된다. 그림 2(a)를 보면, 인터벌 캐쉥에서는 선점이 불가능하기 때문에, 서비스 그룹 상태로 뒤편으로

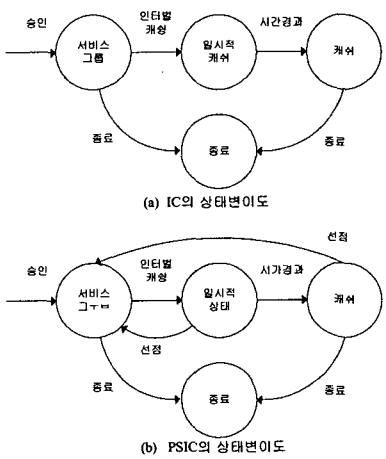

그림 2. 상태변이도

향한 화살표가 없다. 그러나, PSIC 에서는 선점이 자주 일어날 수 있다. 그림 3(b)에 나타나듯이 언제든지, 스트림의 캐쉬된 인터벌을 교체되어 서비스 그룹상태로 갈 수 있다.

IC 에서는, 새로운 인터벌을 저장할 정도로 이용가능한 캐쉬 공간의 크기가 적으면, 새로운 스트림은 다른 1 개 이상의 스트림이 종료되어 캐쉬에 여유공간이 생길 때 까지 기다려야만 했다. 그러나 PSIC 는 기다리는 대신에 다른 방법을 시도한다. PSIC 는 크기가 새로운 인터벌의 크기보다 크고, 스트림의 끊김 현상이 일어나지 않을 캐쉬중인 인터벌을 찾으려고 시도한다. 스트림이 찾아지면, 찾아진 스트림은 서비스 그룹상태가 된다.

3. 제안방안

사용자는 비디오를 선택할 때 여러 비디오의 앞 부분을 본 후에 비디오를 선택하는 경향이 있다. 하지만, 기존의 연구들은 이러한 형태의 사용자의 요구에는 적합하지 않은 특징을 가지고 있다.

인터벌 캐싱은 주문형 비디오 시스템에서 연속적인 같은 비디오에 대한 서비스 요구들의 시간 지역성을 이용해서 디스크 입출력을 줄인다. PSIC 알고리즘 또한 인터벌 캐싱에 기초하고 있다. 그러나 IC 알고리즘은 비디오의 앞부분을 캐싱할 수 없다. 왜냐하면, 앞부분의 블록들이 클라이언트로 전송된 후에야 캐싱이 시작되기 때문이다. 따라서, 비디오의 앞부분은 반드시 디스크에서 반복되어 서비스 된다.

결과적으로, 인터벌 캐싱이 비디오 서버에서 수행될 때, 데이터 블록 억세스는 비디오데이터의 앞부분에 집중된다. 따라서, 아래와 같은 시스템과 알고리즘을 설계 했다.

그림 3 은 제안된 비디오 서비스 시스템을 보여준다. 시스템은 디스크 관리자, 통신 관리자, 메모리 관리자 등 여러 소프트웨어 구성요소 들로 이루어져있다. 입력버퍼, 인터벌 캐쉬버퍼, 출력버퍼가 논리적으로는 분리되어 있지만, 실제로는 물리적 메모리를 공유하고 있다.

알고리즘은 기본적으로 PSIC 에 기초하고, 비디오의

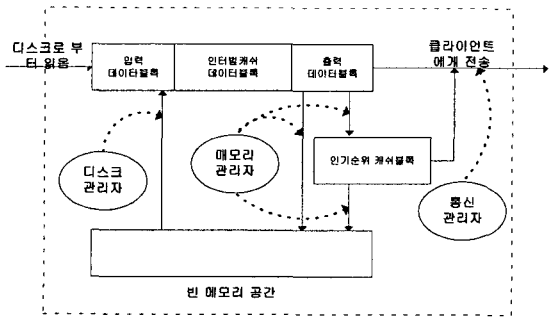

그림 3. 시스템의 구조

(a) 만일 한 비디오에 대한 요구가 발생하면, 선호도를 높여준다.
(b) 만일 비디오의 앞부분의 인기순위 캐쉬블록에 있으면, 읽어서 서비스한다.
(c) 만일 선행 같은 비디오에 대한 선행 스트림이 없다면, 인터벌을 만들지 않는다.
(d) 선행 스트림이 있으면, 인터벌을 만든다. 인터벌의 길이는 선행스트림과 후행 스트림 사이의 시간차로 정해진다.
(e) 캐쉬에 인터벌을 저장을 위한 충분한 공간이 있으면 승인하고, 없으면, 저절한다.
(f) 클라이언트로 전송된 데이터 블록은 선호도에 따라서 빈 메모리 공간이나, 인기순위 캐쉬블록으로 이동한다.

그림 4. 제안된 방안의 알고리즘

앞부분 캐싱을 위한 방법을 추가했다. 스트림 요구가 승인된 후에, 각각의 데이터 스트림 마다, 디스크 관리자는 선인출을 시작하고, 선인출된 데이터 블록을 입력 데이터 블록에 읽어넣는다. 통신관리자는 데이터 블록을 출력 데이터 블록으로 이동시키고, 네트워크 건너편의 대응되는 클라이언트 프로세스에게 전달한다. 캐싱을 하지 않을 경우, 하나의 블록은 전송이 완료된 다음 빈 메모리 블록으로 이동한다. 이런 경우, 스트림 마다 선인출을 위한 버퍼만 필요하다. 인터벌 캐싱의 경우, 메모리 공간에 선인출된 블록들을 유지할 수 있는 공간이 된다면 데이터가 후행하는 다른 스트림들에 의해서 사용될 수 있다.

전송이 완료되면, 메모리 관리자는 블록을 비디오 인기도와 비디오에서의 위치에 따라서 사용하지 않는 메모리 블록으로 보낼지 앞부분 데어터 캐쉬 블록으로 보낼지를 결정한다. 블록들의 이동은 데이터의 복사가 아닌 포인터의 교환만으로 이루어진다.

후에, 스트림에 대한 새로운 요구가 승인되었을 때, 인기순위 캐쉬 블록에 비디오의 앞부분데이터가 존재하면, 디스크에서 선인출 하지않고, 앞부분 데이터 캐쉬에서 서비스를 한다.

4. 실험결과

시뮬레이션에서, 서버와 클라이언트 네트워크 대역폭을 무한대로 가정을 하였다. 그렇기 때문에, 비디오 서버는 실제로 제작을 하고, 클라이언트는 비디오 서버와 하나의 하드웨어에서 동작하고, 네트워크가 아닌 함수 콜을 사용해서, 비디오 서버와 통신하도록 했다. 서버와 클라이언트 사이의 실제의 비디오 데이터의 이동은 없고, 화면에 보이지도 않도록 하여 비디오 서버의 성능에 영향을 미치지 않도록 했다. 클라이언트는 서버에게 단지 데이터 요구만 보내어 서버가 동작을 하도록 유도하는 것이다. 그에 따라서, 비디오 서버에 약간의 수정을 가했다.

서버의 운영체제는 Windows NT 을 사용하였다. Windows NT 의 디스크를 버퍼캐쉬를 사용하지 않고, 저 수준 입출력 API 를 사용하였다. Disk 는 80 G EIDE 디스크 1 개를 사용하였다.

캐쉬메모리는 256M 를 사용하였다. 이중 인터벌 캐쉥을 위해서 196M 를 사용하고, 인기순의 캐쉥을 위해서, 64M 를 사용하였다. MPEG 1 파일 하나의 길이는 대략 1G 를 사용하고 CBR 로 가정하였다. 1G 비디오 80 개를 저장함. Windows NT 운영체제 자체의 파일 시스템이 아닌 비디오 서비스를 위한 별도의 파일 시스템을 만들었다. 메타데이터는 별도로 보관하였다.

클라이언트는 시간당 2000 개의 스트림이 오도록 설정함. 비디오의 선택은 Zipfian 분산을 따르고, 비디오의 시작 요구는 포아송 분포를 따르게 했다[3]. 사용자는 평균적으로 비디오 5 개를 각각 10 분 정도 본 후, 선택하여 끝까지 보게했다.

디스크와 버퍼의 상태를 검사하고, 끊김이 일어나지 않을 경우에만 스트림 요구를 승인하도록 했다. 10 시간 동안 시스템을 가동 시켜 안정화 시킨 후에 측정을 하였다.

실험에서, 4 가지 캐쉥 알고리즘을 비교하였다. 인터벌 캐쉥을 하지 않을 경우, 인터벌 캐쉥을 할 경우, PSIC 를 사용할 경우 제안된 알고리즘을 사용할 경우, 4 가지를 비교하였다. 그림 5 는 PSIC 와 제안된 방안의 비디오 데이터에 내린 블록 읽기 요구의 분산 패턴을 보여주고 있다. PSIC 알고리즘은 요구의 40%가 영화의 앞부분 10%에 집중되어있다. 그리고, 제안된 방안에서는 요구의 13%가 영화의 앞부분 10% 정도에 위치하고 있다. 그리고, 전체적인 읽기 요구의 수도 제안방안이 PSIC 보다 적다.

그림 6 은 동시제공 비디오 스트림 수를 나타내고 있다. 알고리즘은 IC, PSIC, 제안된 알고리즘은 각각 인터벌 캐쉥을 사용하지 않았을 때보다 10% ,34% 그리고, 46% 많은 스트림을 제공하였다.

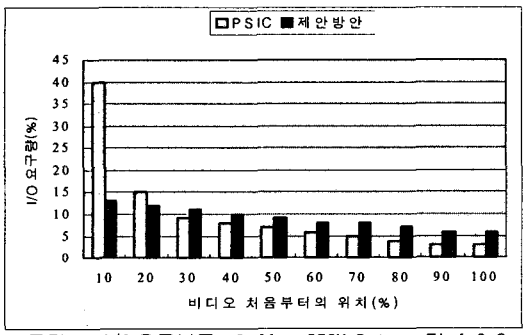

그림 5. I/O 요구분포, Buffer=256M Bytes, Zipf=0.2

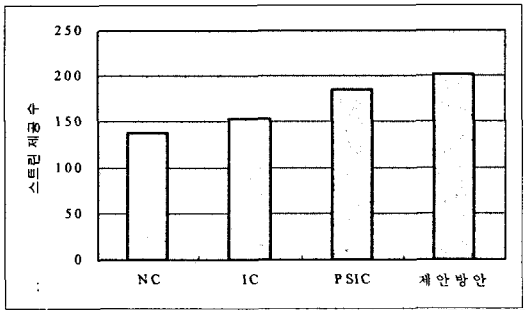

그림 6. 동시제공 스트림 수,Buffer=256MBytes,Zipf=0.2

5. 결론

멀티미디어 시스템의 성능을 높이는 다양한 방법이 존재한다. 디스크 입출력은 멀티미디어 시스템에서 가장 중요한 병목현상 지점이기 때문에, RAID 가 널리 쓰이고[1][4], 디스크 대역폭을 효율적으로 사용하기 위한 데이터 배열 전략과 스케쥴링 전략이 광범위하게 연구되어 왔다. 서버의 성능을 높이는 또 다른 방법은 캐쉥을 사용하는 것이다. 만일 서비스 되는 스트림 중에 매번 디스크에서 읽어오는 대신 캐쉬에서 읽어온다면 디스크 대역폭을 높이지 않고도, 시스템의 성능을 향상 시킬 수 있다. IC 과 PSIC 는 비디오 서버의 성능을 향상 시킬 수 있다. 그러나, 비디오의 앞부분을 절대로 캐쉥할 수 없는 단점이 있다.

이 논문에서, 우리는 비디오 데이터에 대한 블록 읽기 요구가 비디오의 초반부에 집중되는 특징을 이용해서 초반부를 캐쉥 할 수 있는 알고리즘을 제안하였고, 실험을 통하여 성능 평가를 하였다. 제안된 알고리즘은 절약한 디스크 대역폭을 이용해서, 보다 많은 스트림을 제공할 수 있었고, IC 와 PSIC 보다도 좋은 성능을 보였다.

6. 참고문헌

[1] S. A. Barnett and G. J. Anido. Performability of disk-array based video servers. ACM Multimedia Systems Journal, 6(1):60-74, Jan. 1998.

[2] KyungOh Lee, Jin B. Kwon and H. Y. Heom, Exploiting Caching for Realtime Multimedia Systems, pages 506-511,

[3] P. Cao and S. Irani. Cost-aware www proxy caching algorithms. In Proceedings of the USENIX Symposium on Internet Technologies and Systems, pages 193-206, Monterey, California, Dec. 1997.

[4] E. Chang and A. Zakhor. Cost analysis for vbr video servers. In IS&T/SPIE Int'l Symposium on Electronic Imaging: Science and Technology, page 29-31, California, Jan. 1996.

[5] A. Dan, R. M. Daniel M. Dias, D. Sitaram, and R. Tewari. Buffering and caching in large-scale video servers. In Compcon-Technologies for the Information Superhighway, page 217-224, Los Alamitos, CA, Jan. 1995.

[6] K. Lee and H. Y. Yeom. Deciding round length and striping unit size for multimedia servers. In Proceedings of the 4th International Workshop on Multimedia Information Systems(MIS'98), pages 33-44, Istanbul, Turkey, Sept. 1998.

[7] T. R. Ng and J. Yang. An analysis of buffer sharing and prefetching techniques for multimedia systems. ACM Multmedia systems, 4(2):55-69, Apr. 1996.

[8] B.Ozden, R. Rastogi, and A. Silberschatz. Disk striping in video server environments. In Proc. of IEEE Internation Conference on Multimedia Computing and Systems, pages 580-589, Hiroshima, Japan, June 1996.