인터넷 상의 연속 미디어를 위한 Proxy Caching 기법

임은지[U] 최태욱 박성호 정기동
부산대학교 전자계산학과
{ejlim, tuchoi, shpark, kdchung}@melon.cs.pusan.ac.kr

Proxy Caching Mechanism for Continuous Media Data on the Internet

Eun-Ji Lim[U] Tae-Uk Choi Seung-Ho Park Ki-Dong Chung
Dept. of Computer Science, Pusan National University

요 약

WWW의 성장으로 인터넷은 과부하 상태에 이르렀으며, 현재 인터넷 상에는 오디오나 비디오와 같은 연속미디어 데이터가 급격히 증가하는 추세에 있다. 본 논문은 인터넷상의 연속미디어 데이터의 효율적인 서비스를 위한 프락시 캐쉬 기법을 제안한다. 연속미디어 스트림의 앞부분부터 점진적으로 캐쉬하고, 스트림의 캐쉬 길이를 유동적으로 변화시켜서 인기도에 따른 사용공간의 차별화를 수행한다. 그리하여 캐쉬의 제한된 저장공간을 보다 효율적으로 사용하고, 클라이언트의 서비스 지연시간을 최소화 시킬 수 있다. 그리고, 실험을 통하여 다른 알고리즘과의 성능을 비교 측정한다.

1. 서론

오늘날 인터넷의 사용자가 증가함에 따라 서버의 과부하, 네트워크의 혼잡(congestion), 클라이언트에 대한 응답 지연(latency)등의 문제가 야기되었다.

그 해결책의 하나로 프락시 캐쉬(proxy cache)에 관한 연구가 수행되었다[1,2]. 즉, 인터넷 상에서 자주 접근되는 데이터를 지역적으로 클라이언트에서 가까운 위치에 있는 캐쉬에 저장해 둠으로써 클라이언트가 겪게되는 지연을 확연히 줄이고 원격 서버로의 접근수를 상당량 줄여서 서버의 부하와 네트워크 트래픽을 감소시킨다.

한편, 현재 인터넷상에는 오디오와 비디오와 같은 연속 미디어 데이터가 급속히 증가하고, VOD, 전자도서관, 원격강의 등과 같은 멀티미디어 응용(application)이 널리 보급되고 있다. 이러한 연속미디어는 전통적인 데이터에 비하여 상대적으로 큰 크기를 가지므로 객체 전체를 저장하는 기존의 웹 캐쉬 기법을 그대로 적용하기에 부적합하다.

따라서 본 논문에서는 연속미디어에 적합한 프락시 캐쉬기법을 제안한다. 연속미디어의 큰 크기를 고려하여 객체 전체를 캐쉬에 저장하기 보다는 그것의 일부분을 저장하고, 캐쉬에서의 저장 크기를 유동적으로 변화시켜서, 데이터의 인기에 따라 디스크 사용 공간을 차별화시킨다. 또한, 스트림의 앞부분 우선적인 캐쉬 방식으로 클라이언트의 초기지연시간(startup latency)을 최소로 줄인다. 프락시 캐쉬 재배치 정책에 있어서도 연속미디어에 대한 접근 특성을 이용하여 그에 적합한 방법을 제안한다. 제안하는 캐쉬기법과 재배치 기법의 수행결과로 캐쉬를 안정화된 상태로 머물게하는 것이 목표이다. 즉, 캐쉬상의 스트림들의 저장 길이가 스트림의 인기도에 비례하는 형태로 캐쉬되어 있게 한다.

* 본 논문은 1998~1999년 한국 학술진흥재단의 공모과제 연구비에 의하여 연구되었음.

2. 관련연구

2.1 Web 캐쉬

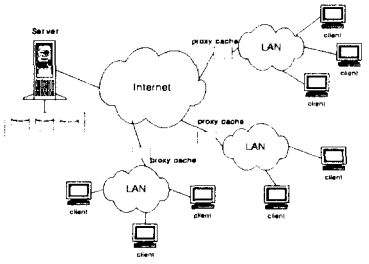
현재의 Web 캐쉬에 관한 연구들은 주로 Web 캐쉬의 재배치 알고리즘에 초점을 맞추고 있다. 즉, 객체들에 대한 접근 정보를 이용하여 가장 저장효율이 낮은 객체를 Web 캐쉬에서 제거하고 저장효율이 높은 객체를 저장하여 Web 캐쉬의 제한된 자원을 효과적으로 사용한다. 보편적으로 널리 사용되는 접근 정보로는 접근 빈도(frequency), 접근의 최근성(recency), 객체의 크기를 들 수 있고, 알고리즘은 LRU, LFU, SIZE, LRU-SIZE, LRU-MIN, LRFU 등을 대표적인 예로 들 수 있다[5]. 그러나 이러한 기법들은 text와 image와 같이 비교적 크기가 작은 전통적인 데이터를 위한 객체 단위의 캐쉬 기법이므로 상대적으로 대용량인 연속미디어에 적용하기는 부적합하다.

2.2 연속미디어의 캐쉬

연속미디어의 캐쉬은 주로 메모리 캐쉬의 개념에서 많이 연구가 되었다. 연속미디어 데이터의 대용량성으로 인하여 데이터의 일부분을 캐쉬하며, 접근의 연속성을 이용하여 같은 데이터에 대한 인접한 요구의 간격을 저장한다. 대표적인 것으로 Interval Caching과 Distance Caching[3,4]을 들 수 있다. 이러한 기법들은 메모리 기반의 캐쉬 기법이므로 대역폭이 제한되어 있는 디스크 기반의 프락시 캐쉬에 적용시키기는 부적합하다.

3. 연속미디어를 위한 프락시 캐쉬 기법

일반적으로 인터넷상의 프락시 캐쉬는 그림 1에서처럼 서버에서 클라이언트로의 경로상에서 클라이언트와 근접한 곳에 위치한다. 캐쉬는 연속미디어 데이터의 일부분 또는 전체를 저장하고 있다가 클라이언트의 요구가 들어올 경우 자신이 가진 데이터를 전송해준다.

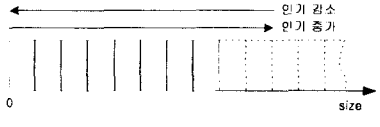


[그림 1] 인터넷 상의 proxy cache

3.1 프락시 캐쉬 기법

연속미디어 데이터는 그 크기가 크기 때문에 전통적인 데이터와 같이 객체단위로 캐쉬할 경우 적은 수의 객체만을 저장할 수 있어서 캐쉬 공간의 사용효율이 감소하게 되고, 그 재배치 비용 또한 상당히 커진다. 따라서 캐쉬에는 미디어 스트림의 일부분을 저장하되 초기부분부터 시작하여 점진적으로 캐쉬하는 방식을 사용하여 클라이언트에 대한 서비스 초기지연시간을 최소화시킨다. 또한 스트림의 인기도에 따라서 캐쉬된 길이를 변화시킨다. 즉, 인기가 증가하면 저장되는 길이를 점차적으로 늘려주고 인기가 감소하면 길이를 점점 줄인다. 인기가 매우 높은 스트림은 전체가 다 캐쉬될 수 있을 것이고, 인기가 매우 작은 것은 앞부분의 아주 작은 양만 캐쉬되거나, 아예 캐쉬에서 삭제 될 것이다. 그렇게 하여 각 스트림의 인기에 따라서 그것이 캐쉬에서 차지하는 공간을 차별화하는 효과를 얻을 수가 있다. 이 캐쉬 정책의 주된 목적은 클라이언트의 서비스 지연을 최소화하는 동시에, 캐쉬상의 스트림의 저장길이를 스트림의 인기도에 비례하는 상태로 유지시키는 것이다. 인기에 급격한 변화가 없을 경우에는 재배치 횟수가 감소하고 캐쉬는 안정화된 상태로 머물게 된다. 반면에 인기가 변화할 경우에는 유동적으로 스트림의 캐쉬된 크기를 조절하여 인기의 변화에 적응하게 된다. 이러한 캐쉬 정책에 따른 재배치를 수행하기 위하여 각 스트림을 같은 크기의 논리적인 블록으로 나누어서 그것을 단위로 하여 재배치를 수행한다.

그림 2는 이러한 캐쉬 정책에 따른 스트림의 저장 패턴을 보여준다.



[그림 2] 스트림의 인기에 따른 캐쉬 패턴

어떤 스트림에 대한 첫 번째 요구가 프락시에 도착하면 프락시는 스트림 전체를 서버로부터 패치(fetch)해와서 클라이언트에게 전송해주고, 스트림의 맨 첫 번째 블록을 캐쉬시킨다. 그 이후로 같은 스트림에 대한 요구가 들어오면 캐쉬되어 있는 데이터를 바로 클라이언트에게 전송해 주면서, 그와 동시에 뒷부분을 서버로부터 패치해온다. 이때에 현재 캐쉬되어 있는 블록의 바로 뒷 블록 하나를 더 캐쉬시켜서 캐쉬되어 있는 스트림의 저장길이를 늘려준다. 즉, 스트림이 요구될수록 그것의 블록이 하나 씩 더 저장되어 저장길이가 늘어난다. 이렇게 새로운 블록을 캐쉬시키려고 할 때 캐쉬에 여유 공간이 없다면 캐쉬 재배치를 수행하게 된다.

3.2 재배치 정책

프락시 캐쉬에 새로운 블록을 저장하려고 할 때 여유 공간이 없으면 현재 캐쉬되어 있는 스트림들의 캐쉬 효율(utility)을 비교하여 효율이 가장 낮은 스트림을 희생(victim) 스트림으로 선택한다. 그리고 희생 스트림의 가장 마지막 블록을 캐쉬에서 삭제함으로써 캐쉬된 길이를 줄인다.

캐쉬의 자원은 제한 되어 있기 때문에 어떤 데이터를 캐쉬상에 유지할 때는 그것의 저장으로 얻을 수 있는 이익과 그것이 사용하는 캐쉬의 자원과의 상관관계를 항상 고려하여야 한다. 즉, 이익이 큰 것은 그만큼 많은 자원을 사용할 가치가 있고, 이익이 작은 것은 적은 양의 자원을 사용해야 한다. 여기에서 캐쉬의 자원은 저장공간으로 볼 수 있고, 데이터의 저장으로 얻을 수 있는 이익은 이 데이터를 캐쉬함으로써 캐쉬로부터 서비스 해줄 수 있는 총 전송 양으로 볼 수 있다. 데이터의 사용 자원에 대한 이익의 비율을 데이터의 캐쉬 효율이라고 하겠다. 따라서 다음과 같은 수식이 성립된다.

$$Caching\ Utility = \frac{Caching\ Benefit}{Used\ Resouce\ Amount} \quad (1)$$

여기에서 사용자원량은 스트림에 할당된 캐쉬공간, 즉, 스트림의 저장길이가 된다. 캐쉬이익은 사용하는 성능평가기준에 따라 달라진다. 여기에서 잠시 연속미디어의 데이터 특성에 대해 고려할 필요성이 있다. 연속미디어는 전통적인 데이터와는 달리 단순한 "접근"의 여부로 그 인기도를 평가할 수가 없다. 즉, 한번 접근할 때에 스트림의 전체를 재생할 수 있고, 아주 적은 부분만을 재생할 수도 있다. 따라서 이 논문에서는 캐쉬이익의 척도로서 스트림의 재생 데이터량(PlaybackBytes)을 사용할 것이다. 그래서 식은 다음과 같이 변형된다.

$$Caching\ Utility = \frac{\sum_{\Delta} PlaybackBytes}{CachedLength} \quad (2)$$

여기에서 PlaybackBytes는 한 클라이언트에 의한 한 스트림에 수행된 총 재생데이터량을 나타내고 Δ 는 고려 범위가 되는 시간 윈도우의 크기를 나타낸다.

프락시는 재배치 정책을 수행하기 위하여 각 스트림의 재생 데이터량에 대한 정보를 유지하여야 한다. 또한, 캐쉬되어 있는 모든 스트림들의 캐쉬효율값에 대한 정렬된 리스트를 유지한다. 새로운 데이터를 위한 여유공간이 없을 때, 리스트의 헤드가 희생 스트림(캐쉬 효율값이 가장 작은 것)이 되고 그것의 마지막 블록을 캐쉬에서 제거한다.

4. 성능 측정

실험은 큰 크기의 객체(360M)와 작은 크기의 객체(20M)에 대하여 각각 수행하였다. 성능평가요소로서 HR(hit ratio) 대신 BHR(byte hit ratio)를 사용하였고, 그 외에, 평균 초기지연시간, 재배치 횟수를 성능평가요소로 사용하였다. 캐쉬의 크기를 변화시키면서 이러한 요소들의 값을 측정하였다.

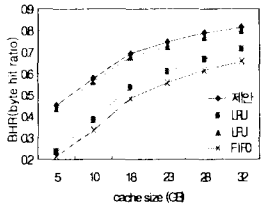
4.1 실험 인자

Media Length	360M/20M
total media	200개
Media Chosen	θ =0.27의 Zipf 분포
Request Interarrival time	10초/3초
실험측정시간	8000개의 사용자요구를 처리하는 시간

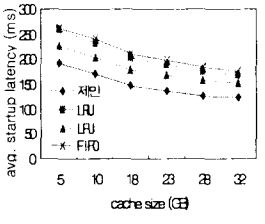
4.2 실험 결과

그림 3,4,5는 큰 크기의 객체에 대한 실험 결과를 보여준다. 그림 3을 보면 제안한 캐쉬기법을 사용했을 경우에 가장 BHR값이 높게 나타남을 알 수 있다. 이것은 캐쉬에서 서비스되는 데이터량이 많고, 따라서 서버로부터 전송되는 데이터량이 그만큼 적어져서 서버의 부하와 네트워크 트래픽이 감소할 수 있다는 것을 나타낸다. 그림 4에서는 제안하는 캐쉬기법이 스트림의 앞부분을 우선적으로 저장하는 방법을 사용함으로써 클라이언트가 겪게되는 초기지연을 확연히 줄였음을 보여준다. 프락시와 클라이언트 사이의 지연은 50ms, 프락시와 서버 사이의 지연은 100ms로 가정한 것이다. 그림 5는 재배치 횟수를 비교한 것이다. 본 논문에서 제안하는 기법을 사용할 경우, 어느 정도 시간이 지나면 캐쉬상의 스트림들의 저장길이는 그 데이터의 인기도에 비례하는 상태로 저장이 되어 캐쉬는 안정한 상태로 머물게 된다. 이러한 상태가 되면 인기도에 급격한 변화가 없는 한 스트림의 저장길이에 큰 변화가 없게 되고 캐쉬의 재배치 횟수도 감소하게 된다. 또한, 이 기법은 객체단위로 재배치를 수행하는 다른 알고리즘과 달리 논리적 블록단위로 재배치를 수행하므로, 총 재배치 데이터량을 고려한다면 훨씬 더 월등한 성능을 보임을 알 수 있다.

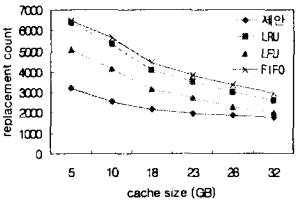
마지막, 그림 6을 보면, 작은 크기의 객체에 있어서도 객체단위의 캐쉬보다 제안한 부분 캐쉬이 더욱 높은 성능을 보임을 알 수 있다. 이것은 작은 크기일지라도 연속미디어 데이터일 경우 전통적인 데이터에 비하여 훨씬 큰 크기를 가지기 때문에 객체단위 캐쉬보다 부분 캐쉬이 더 적합하다는 것을 증명해준다.



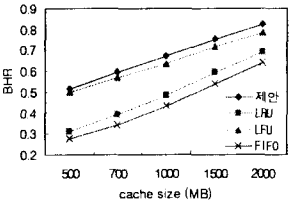
[그림 3]



[그림 4]



[그림 5] Cache 크기에 따른 재배치 횟수



[그림 6] 작은 크기 객체(20M), 평균 요구도착 시간간격=3초

5. 결론

본 논문은 인터넷 상의 연속미디어에 적합한 프락시 기반의 캐싱 기법을 제시하였다. 연속미디어 스트림의 앞부분 우선적인 캐쉉 방법과 데이터의 캐쉉 효율에 따라 저장길이를 유동적으로 변화시킴으로써 서비스 초기지연을 최소화하고, 캐쉬의 사용 효율을 상당량 증가시킬 수 있었다. 향후 연구과제로는 네트워크 상황에 대한 고려를 통하여 외부 네트워크 환경에 적응할 수 있는 캐쉉기법, 캐쉉되어 있지 않은 데이터를 서버에서 가져와서 전송할 때 사용하는 버퍼사용량에 대한 고려, 프락시 메타 데이터 구조에 관한 연구가 필요하다.

6. 참고 문헌

[1] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, K. J. Worrell, "A Hierarchical Internet Object Cache", In Proc. of 1996 Usenix Technical Conference, January 1996.

[2] A. Luotonen, K. Altis, "World Wide Web Proxies", In Proceedings of theFirst International Conference on the WWW, May 1994.

[3] A. Dan, D. Dias, R. Mukherjee, D. Sitaram, R. Tewari, "Buffering and Caching in Large-Scale Video Servers", In Proc. of IEEE COMPCON, March 1995.

[4] B. Ozden, R. Rastogi, A. Silberschatz, "Buffer replacement algorithms for multimedia storage systems", In Proc. of the International Conference on Multimedia Coomputing and Systems, June 1996.

[5] R. Tewari, H. M. Vin, A. Dan, D. Sitaram, "Resource-based caching for Web servers", In Proc. SPIC/ACM Conference on Multimedia Computing and Networking, January 1998.

[6] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams", In Proc. IEEE INFOCOM, March 1999.