연속미디어 스트림의 재생량에 기반한 proxy caching 기법

임은지[0] 최태욱 박성호 정기동
부산대학교 전자계산학과
{ejlim, tuchoi, shpark, kdchung}@melon.cs.pusan.ac.kr

Quantity-based Proxy Caching Policy for Continuous Media Streams

Eun-Ji Lim[0] Tae-Uk Choi Seung-Ho Park Ki-Dong Chung
Dept. of Computer Science, Pusan National University

요 약

인터넷의 사용이 일반화되어 WWW 기반 서비스와 사용자가 급증함에 따라서 서버의 과부하, 네트워크의 혼잡, 사용자에 대한 응답 지연 등의 문제가 심각하게 나타나고 있다. 또한, 현재 인터넷 상에는 오디오나 비디오와 같은 연속미디어 데이터가 급격히 증가하는 추세에 있다. 본 논문은 인터넷상의 연속미디어 객체의 일부분 또는 전체를 캐쉬하는 프락시 캐쉬 기법을 제안한다. 제안하는 기법은 객체의 인기도에 따라서 캐쉬할 최적의 데이터 양을 결정하고, 결정된 양만큼의 앞부분 데이터를 캐쉬한다. 또한, 본 논문은 연속미디어 데이터의 특성을 고려하여, 각 객체에 대한 클라이언트의 재생량에 기반한 인기도 측정 방법을 제안한다. 마지막으로, 실험을 통하여 제안하는 기법의 성능을 평가한 결과, 제안한 캐쉬 기법이 BHR면에서는 다른 알고리즘과 비슷하였으나, 전송 지연과 재배치 횟수면에서 다른 알고리즘들에 비하여 최고 2배 이상 성능이 우수하였다. 재생량을 이용한 인기도 측정법도 접근빈도를 이용한 경우보다 성능이 우수하였다.

1. 서론

최근 인터넷의 사용이 일반화되고 VOD와 같은 멀티미디어 서비스가 널리 확산되면서 인터넷상의 연속미디어 데이터의 양이 급증하고 있다[1]. 이로 인한 문제점은 서버의 부하, 네트워크의 혼잡, 클라이언트에 대한 응답지연 등으로 나타난다.

프락시 캐쉬(proxy cache)[2,3]를 사용하면, 최근에 자주 요구된 데이터를 캐쉬에 저장하여 클라이언트의 요구가 발생했을 경우에 서버에 접근하지 않고도 직접 캐쉬에서 전송한다. 따라서, 서버의 부하와 클라이언트의 전송지연, 서버와 프락시 서버 사이의 네트워크 상의 트래픽이 감소 된다.

지금까지 연구된 프락시 캐쉬 기법은 텍스트나 이미지와 같은 이산미디어를 위한 기법이고, 이것은 오디오나 비디오와 같은 연속미디어의 캐쉬에 적용시키기에 부적합하다. 연속미디어는 이산미디어에 비하여 대용량이고, 전송에서 높은 대역폭을 요구하며, 실시간으로 서비스 된다. 따라서, 연속미디어를 프락시 서버에 저장하기 위해서는 이러한 미디어의 특성을 고려한 새로운 캐쉬 기법이 필요하다.

본 논문은 연속미디어의 인기도 분포와 캐쉬되는 데이터 양의 분포를 같게 하는 캐쉬 기법[7]을 개선시킨 Popularit-based Prefix Caching(PPC)에 대해 설명하고, PPC의 성능 향상을 위해 연속미디어의 특성을 고려한 인기도 측정법을 제안한다. 즉, 연속미디어의 재생 데이터 양과 접근의 최근성을 반영하여 인기도를 측정하고, 측정된 인기도에 기반하여 각 객체의 캐쉬할 데이터의 적정 양을 결정한다.

논문의 구성은 다음과 같다. 논문의 2장에서 관련연구를 살펴보고, 3장에서는 제안하는 캐쉬 기법과 인기도 측정 방법을 자세히 서술하며, 4장에서 실험을 통하여 성능평가를 하고, 마지막 5장에서 결론 및 향후 연구과제를 기술한다.

2. 관련연구

2.1 연속미디어 데이터의 프락시 캐쉬에 관한 연구

최근 들어 웹 문서를 위한 캐쉬 뿐만 아니라 연속미디어 데이터를 위한 프락시 캐쉬에 관한 연구가 많이 진행되고 있다[4,5,6]

[4]은 멀티미디어 스트림의 앞부분 일정 양만을 캐쉬하여 클라이언트의 초기 지연을 줄이고 workahead smoothing을 수행하는 기법을 제안했고, [5]은 interval caching의 개념을 프락시 캐쉬에 도입하여 프락시의 저장 공간와 디스크 대역폭을 고려하는 RBC(resource-based caching)을 제안하였다. [6]는 이질적인 네트워크 환경을 고려한 video staging이라는 기법을 제안하였다.

2.2 캐쉬 척도

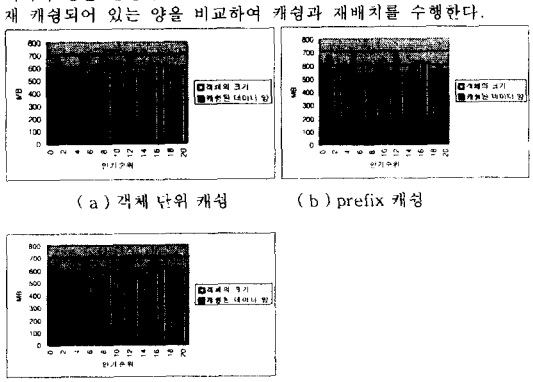
일반적으로 캐쉬에서 가장 많이 사용되는 캐쉬 척도는 접근 빈도(access frequency)와 접근의 최근성(access recency)이다. 접근 빈도를 사용하는 대표적인 알고리즘은 LFU이고 접근의 최근성을 사용하는 것은 LRU 이다[5]. 웹 캐쉬에서는 재배치 단위의 크기가 다르다는 점을 고려하여 문서의 크기를 재배치 척도로 고려하기도 하는데 SIZE가 여기에 속한다[3]. 하지만 이렇게 특정한 한 가지 척도만을 사용하는 것은 그다지 바람직하지 못하므로 여러 요소들을 복합적으로 적용시키는 정책들도 많이 연구되었다. LRU-MIN, GD-SIZE, LRV와 같은 것들이 이에 포함된다[3].

3. 캐쉬 정책

그림 1 은 연속미디어 데이터에 객체 단위 캐쉬과 prefix 캐쉬, 그리고 인기도 기반의 prefix 캐쉬을 각각 적용했을 경우에 각 객체의 인기 순위에 따라 캐쉬되는 데이터 양을 예를 들어서 나타낸 것이다. (a)에서 보듯이, 연속미디어 데이터는 크기가 크기 때문에 객체 단위로 캐쉬할 경우에 인기가 높은 소수의 객체들만 저장할 수 있다. 따라서 저장 공간을 비효율적으로 사용하게 되고 소수의 캐쉬된 데이터를 제외한 나머지 데이터에 대한 전송의 초기 지연 시간이 크다. (b)는 각 객체의 앞부분 일정량의 데이터만을 캐쉬하는 prefix 캐쉬의 경우이다. 이것은 모든 데이터에 대한 전송의 초기 지연 시간을 감소시키지만, 인기가 낮아서 거의 요구되지 않는 비디오에 대해서도 인기가 매우 높은 비디오와 동일한 저장공간을 할당함으로써 저장공간의 낭비를 초래한다. (c)는 이 두 가지 기법의 단점을 보완한 것으로서, 비디오의 인기도 분포와 캐쉬되는 데이터 양의 분포를 같게 한 것이다. 이것은 가능한 많은 객체에 대한 전송의 초기 지연 시간을 감소시키면서, 서버로부터 전송되는 데이터 양을 최소화 한다.

본 논문에서는 연속미디어를 위한 프락시 캐쉬 기법인 PPC(Popularity-based Prefix Caching)를 제안한다. 이것은 각 연속미디어 객체의 전체 또는 부분을 캐쉬하는 것으로써, 객체의 인기도

분포와 캐쉬되는 데이터 양의 분포를 같게 하는 것이다. 즉, 모든 객체들의 인기도를 계산하고 그것에 기반하여 각 객체의 캐쉬될 최적의 데이터 양을 결정하게 된다. 그리고, 계산된 최적의 데이터 양과 현재 캐쉬되어 있는 양을 비교하여 캐쉬과 재배치를 수행한다.



(a) 객체 단위 캐쉬 (b) prefix 캐쉬

(c) 인기도 기반의 prefix 캐쉬

그림 1 : 데이터의 인기 순위에 대한 캐쉬되는 데이터 양

PPC의 장점은 다음과 같다.

- 가능한 많은 수의 객체에 대한 초기지연시간 감소
- 서버에서 전송되는 데이터 양의 최소화
- 객체의 인기에 따라서 할당되는 캐쉬 공간 차별화

PPC의 수행을 위한 인기도 측정 방법과 측정된 인기도에 기반하여 캐쉬과 재배치를 수행하는 부분을 자세히 서술하겠다.

3.1 데이터의 재생량에 기반한 인기도 측정법

전통적인 미디어의 캐쉬에서 일반적으로 고려되는 사항은 접근 빈도와 접근의 최근성이다. 그러나, 연속미디어 데이터는 streaming mode로 전송 되므로 전통적인 미디어에서 사용하는 " 접근 빈도" 의 개념보다는 " 접근 정도" 의 개념을 사용하는 것이 적합하다. 예를 들어 두 비디오가 단위시간동안에 한 번씩 요구되었는데, 한 비디오는 1분 동안 재생되고 다른 것은 10분 동안 재생되었다면 두 비디오의 접근 빈도는 같지만 접근 정도는 다르다. 또한 후자의 인기도가 더 높다고 봐야 바람직할 것이다.

본 논문에서는 이런 점에 착안하여 접근의 정도를 나타내는 데이터의 재생량과 접근의 최근성을 반영하는 인기도 측정 방법을 제안한다. 그림 2는 현재 시간이 t일 때, $[t-\Delta, t]$ 구간 동안에 재생된 총 데이터 양을 측정하는 것을 나타낸 그림이다. $S_{i,j}$는 현재 시간 t에 재생이 진행 중이거나, 주어진 구간에서 재생이 종료된 스트림으로서, 비디오 i, $i \in [i..N]$ 를 클라이언트 j가 재생하는 스트림이다. 실선으로 표시된 부분은 각 $S_{i,j}$에서 주어진 구간에 포함되는 부분만을 나타낸 것이고, $DATA^t_{i,j}$ 는 실선 구간동안에 클라이언트 j가 비디오 i에 대해 재생한 데이터 양이다. 주어진 구간에서 비디오 i를 재생한 클라이언트의 수를 c_i라 했을 때, 구간 $[t-\Delta, t]$에서 비디오 i의 총 재생량 $DATA^t_i$는 다음과 같다.

$$DATA^t_i = \sum_{j=0}^{c_i} DATA^t_{i,j}$$

프락시는 각 비디오의 가중치를 둔 평균 재생량인 D^t_i 를 다음과 같이 계산하여 유지한다.

$$D^t_i = \alpha DATA^t_i + (1-\alpha) D^{t-\Delta}_i,\ 0 < \alpha \le 1$$

여기에서 α는 최근 정보에 대한 가중치 값으로써, NOD와 같이 생명주기가 짧은 데이터에 대해서는 α값을 크게 하고, 상대적으로 생명주기가 긴 데이터에 대해서는 α값을 작게 한다. 이렇게 최근정보에 대해 가중치를 둠으로써 접근의 최근성을 반영한 인기도를 구할 수 있다.

계산된 D^t_i 값을 이용하여 시간 t에서 각 비디오 i, $i \in [i..N]$ 의 인기도 P^t_i 를 다음과 같이 계산한다.

$$P^t_i = \frac{D^t_i}{\sum_{k=0}^{N} D^t_k}$$

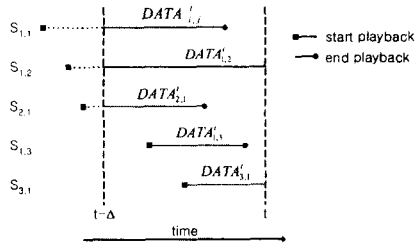


그림 2 : 구간 $[t-\Delta, t]$에서 각 비디오의 재생량

3.2 인기도에 기반한 캐쉬 정책

PPC는 각 연속미디어 객체의 인기도 분포와 객체의 캐쉬되는 데이터 양의 분포를 같게 한다. 그렇게 하기 위하여 계산된 인기도 값을 이용하여 각 객체의 캐쉬할 최적의 데이터 양을 계산한다.

그러나, 단순히 3.1절에서 계산된 인기도의 비율로 캐쉬할 데이터 양을 계산할 경우, 아주 인기가 높은 것은 그 객체의 길이보다 더 많이 할당 받고, 인기가 낮은 객체들은 공간을 거의 할당 받지 못한다. 따라서 최적의 양을 결정할 때는 인기가 높은 객체부터 계산한다. 그리고 나머지 객체들에 대해서는 남은 객체들 사이에서의 인기도 비율과 남은 캐쉬 공간의 크기를 이용하여 최적의 양을 계산한다.

객체 i의 전체 길이를 L_i, i의 캐쉬될 최적의 크기를 $\hat{s}_i$, 캐쉬의 전체 크기를 S라 하자. 모든 객체를 D^t_i 값에 따라 내림차순으로 정렬한 후, 다음 수식을 이용하여 수정된 인기도 값을 계산한다. $\hat{P}^t_i$는 i, $i+1, \ldots, N$ 의 객체 중에서 객체 i의 인기도 비율을 나타내는 값이다.

$$\hat{p}^t_i = \frac{D^t_i}{\sum_{k=i}^{N} D^t_k}$$

그리고 각 객체의 최적의 캐쉬할 데이터 양은 다음과 같이 계산된다.

$$\hat{s}_i = \min\{\hat{p}^t_i \times (S - \sum_{k=0}^{i-1} \hat{s}_k), L_i\}$$

프락시는 계산 오버헤드를 감소시키기 위하여 각 객체의 인기도와 최적의 캐쉬할 데이터 양을 주기적으로 계산하여 table로 유지한다. 그리고 각 객체에 대한 클라이언트의 요구가 발생하면 이미 계산되어 있는 값과 실제 캐쉬되어 있는 데이터 양을 비교하여 그 차이 값만큼 캐쉬 또는 재배치를 수행한다.

객체 i에 대한 요구가 발생했을 경우에 만일 $s_i < \hat{s}_i$ 이면 $(s_i - \hat{s}_i)$ 만큼을 더 캐쉬하고, $s_i \ge \hat{s}_i$ 이면 더 캐쉬하지 않는다.

재배치가 필요한 경우에는 최적의 데이터 양보다 더 많이 캐쉬되어 있는 객체의 초과되는 데이터 양 만큼을 삭제한다. 즉, $s_m > \hat{s}_m$ 인 객체 중의 하나의 뒷부분 $(s_i - \hat{s}_i)$을 삭제하고, 필요한 공간이 생길 때까지 이 과정을 반복한다.

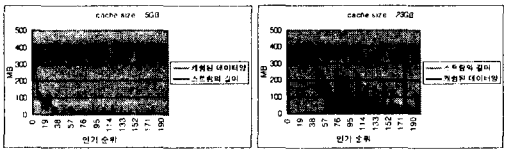
4. 실험

실험환경은 다음 표와 같고, 모든 클라이언트는 재생도중에 임의의 시점에서 재생을 중단할 수 있다고 가정했다.

Media object size	280MB ~ 420MB
Request distribution	Zipf distribution of θ = 0.27
Average request interarrival time	10 sec
Number of media object	200
Simulation duration	8000개의 사용자 요구를 처리하는 시간
프락시 서버와 클라이언트 사이의 전송 delay	100ms
서버과 프락시 서버 사이의 전송 delay	200ms
계산 주기	360sec
α	0.3

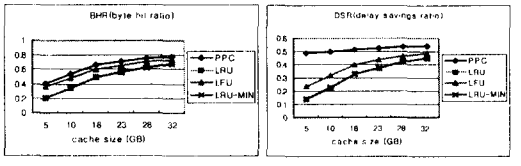
그림 3은 객체의 인기 순위에 따라 캐쉬된 데이터 양을 나타낸 것이다. (a)에서 보듯이, 캐쉬 크기가 작을 때는 인기가 높은 극소수의 객체만 전체가 저장되고 나머지는 앞부분 약간만 저장된다. 따라서, 캐쉬의 크기가 작더라도 대부분의 클라이언트들에 대한 초기지연시간은 감소한다. (b)와 같이 캐쉬크기가 더 클 경우에는 더 많은 객체가 전체 캐쉬되고, 상대적으로 인기가 낮은 객체도 앞부분이 더 많이 캐쉬된다.

그림 4는 PPC와 다른 캐쉬 알고리즘의 성능을 비교측정 한 것이다. 여기에서 성능 척도로는 BHR(*byte hit ratio*), DSR(*delay savings ratio*), 재배치 횟수를 사용했다. PPC는 객체를 부분적으로 캐쉬하는 기법이므로 hit ratio를 측정하는 것은 부적합하다고 사료되어 BHR을 측정하였다. DSR은 캐쉬로부터 서비스 해 줌으로써 줄일 수 있는 지연 시간의 비율을 나타내는 값이다. 그림 4의 (a)에서 보듯이, PPC의 BHR값이 다른 알고리즘보다 높지만, 그 차이가 작고 LFU와 거의 비슷하다. 그러나 (b)를 보면 DSR면에서 PPC가 다른 알고리즘보다 월등히 우수함으로 알 수 있다. 특히, 캐쉬 크기가 작을 때는 2배 이상 성능이 향상되었다. 이것은 나머지 알고리즘들이 객체 단위의 캐쉬을 수행하므로 캐쉬크기가 작을 때는 인기가 높은 몇 개의 객체만 저장할 수 있어서 평균 지연시간이 높은 반면, PPC는 대부분의 객체의 앞부분을 약간씩 저장하므로 지연시간이 현저히 감소한다. (c)는 전체 실험 기간동안 발생한 재배치 횟수를 측정한 것이다. PPC를 제외한 나머지 알고리즘들은 요구된 데이터가 캐쉬에 없으면 재배치를 수행한다. 그러나, PPC는 요구된 객체가 최적의 양만큼 캐쉬되어 있다면 재배치를 수행하지 않는다. 따라서 재배치 횟수가 적게 나타난다.

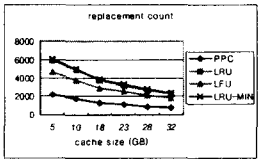


(a) 캐쉬 크기 : 5GB (b) 캐쉬 크기 : 23GB

그림 3 : 객체의 인기 순위에 따른 캐쉬된 데이터 양



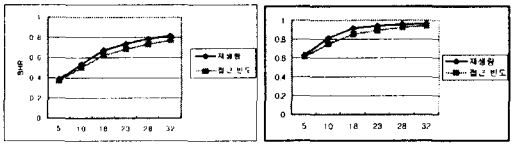
(a) BHR (b) DSR



(c) 재배치 횟수

그림 4 : 다른 캐쉬 알고리즘과의 비교

그림 5는 PPC의 캐쉬정책을 그대로 사용하되 인기도 측정방법에서 재생량을 이용했을 경우와 기존에 일반적으로 많이 사용하는 접근 빈도를 사용했을 경우를 비교한 것이다. (a)는 클라이언트가 임의의 시점에서 재생을 중단한다고 가정한 경우이고, (b)는 인기와 재생 시간을 비례하게 한 경우이다. 접근 빈도를 사용했을 경우보다 재생량을 사용했을 경우에 성능이 더 우수했으며, (a)에서 보다 (b)의 경우에 전체적으로 성능이 더욱 뛰어났다.



(a) random (b) 인기도에 따라

그림 5 : 클라이언트의 재생 종료 시점에 따른 BHR

5. 결론 및 향후 연구 과제

본 논문은 기존의 프락시 캐쉬 기법들이 연속미디어에 적용시키기에 부적합하다는 점에 착안하여, 연속미디어의 특성을 고려한 캐쉬 기법을 제안하고 실험을 통하여 성능을 검증하였다. 각 연속미디어 객체의 인기도를 이용한 캐쉬 정책을 제안하였으며, 미디어의 재생량에 기반한 새로운 인기도 측정 방법 또한 제안하였다. 실험을 통하여, 제안한 캐쉬 기법이 다른 알고리즘들에 비하여 BHR, DSR, 재배치 횟수 면에서 우수하고, 재생량에 기반한 인기도 측정 방법도 접근 빈도수를 이용하는 방법보다 성능이 좋음을 증명하였다.

향후에는, 트레이스 기반 모의 실험을 수행할 계획이며, 캐쉬되지 않은 뒷부분의 데이터를 서버로부터 전송할 때 전송 제어를 하는 것에 대한 연구도 필요하다.

6. 참고문헌

[1] G. A. Gibson, J. Witter, and J. Wilkes, " Storage and I/O Issues in Large-Scale Computing" , ACM Workshop on Strategic Directions in Computing Research, ACM Computing Surveys, 1996.

[2] 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.

[3] J. Wang, "A Survey of Web Caching Schemes for the Internet", Technical Report TR99-1747, Cornell University Department of Computer Science

[4] S. Sen, J Rexford and D. Towsley, "Proxy Prefix Caching for Multimedia Streams," In Proc. IEEE Infocom, March 1999.

[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] Y. Wang, Z.-L. Zhang, D. Du, and D. Su, "A Network- Conscious Approach to End-to-End video Delivery over Wide Area Networks Using Proxy Servers", In Proc. IEEE Infocom, April 1998.

[7] 박성호, 임은지, 최태욱, 정기동, " 인터넷상에서 NOD 서비스를 위한 연속미디어 전송 및 푸쉬-캐쉬 기법" , 한국정보처리학회 논문지 제7권 제6호.