스트리밍 미디어의 캐쉬 서버를 위한 재배치 정책의 설계와 성능분석

임은지°, 정성인
한국전자통신연구원

Design and Performance Evaluation of Replication Policy For Streaming Media Cache Server

EunJi Lim, SungIn Jung
ETRI

요 약

본 논문은 인터넷 상에서 서비스되는 스트리밍 미디어를 캐쉬 서버에서 캐싱할 때 적용할 수 있는 재배치 정책에 관한 것이다. 스트리밍 미디어의 특성에 적합한 재배치 기준을 제시하고, 그에 따라 미디어 데이터를 캐싱하고 재배치하는 방법을 제안한다. 또한, 제안한 재배치 정책과 기존의 알려진 방법들에 대한 성능 비교 분석을 수행 한다.

1. 서론

과거에 인터넷을 통해 이용되는 컨텐트는 텍스트나 이미지에 불과했지만, 현재는 누구나 음악과 동영상과 같은 양질의 컨텐트를 요구한다. 그런데, 이러한 멀티미디어 데이터는 그 크기가 크고, 연속성이 보장되어야 한다는 점에서 양질의 서비스를 제공하기에 있어서 어려움이 오래 전부터 인식되고 있다.

이러한 스트리밍 미디어의 서비스를 위해서 하드웨어, 응용소프트웨어, 압축기술, 전송 프로토콜 등의 여러 분야에서의 연구가 진행되었는데, 본 논문에서는 특히, 스트리밍 미디어를 캐싱하기 위한 캐쉬 서버에 대해서 다루고자 한다.

캐쉬 서버는 스트리밍 미디어를 제공하는 원격서버와 서비스를 제공 받는 클라이언트의 사이에 위치하여 미디어 데이타를 캐싱하고 클라이언트에게 직접 데이터를 전달하는 기능을 수행한다. 캐쉬 서버를 이용하면 서버의 부하를 감소시키고, 네트워크 상의 트래픽을 감소시키고, 클라이언트로 전송하는 지연시간과 지터를 줄일 수 있다. 그러나, 캐쉬 서버는 디스크라는 제한된 자원을 이용하므로 이런 제한된 자원을 효율적으로 사용하기 위한 재배치 정책이 필요하다. 또한, 스트리밍 데이터는 이산(atomic) 데이터와는 다른 고유의 특성을 지니고 있으므로, 그 특유의 특성을 고려한 재배치 정책이 사용될 필요가 있다.

본 논문에서는 스트리밍 미디어의 특성에 적합한 재배치 기준을 제시하고, 그것을 이용한 캐쉬 재배치 정책을 설계한다. 그리고 제안되는 방법과 기존에 알려진 방법들의 성능을 비교 분석 한다.

2. 관련연구

2.1 캐싱 위치에 따른 분류

일반적으로 캐싱은 서버측, 클라이언트측, 그리고 네트워크 상에서 각각 수행될 수 있다. 서버측에서 수행되는 캐싱은 서버의 메인 메모리에서 수행하는 것[1]을 말하는데 이것은 서버의 디스크 입출력 회수를 감소시켜서 서버의 부하를 줄일 수 있으나 네트워크 트래픽을 감소시키지는 못한다. 클라이언트에서 수행되는 캐싱은 브라우저에 built-in 된 지역 캐쉬에서 수행되므로 어플리케이션 레벨 캐싱이라고도 하는데[2], 이것은 한 클라이언트에게 같은 데이터가 반복적으로 전송되는 것을 방지할 수 있으나, 이웃한 클라이언트가 같은 서버로부터 같은 데이터를 요구했을 경우에 그 데이터가 반복 전송되는 것을 방지하지 못한다. 네트워크 상에서 수행되는 캐싱은 프락시 캐쉬(proxy cache)에 의해 수행되는 것을 말한다[3]. 프락시에 캐싱된 데이터는 한 지역에 속한 모든 클라이언트에게 전송될 수 있으므로, 같은 지역에 있는 클라이언트에게 같은 데이터가 서버로부터 반복적으로 전송되지 않도록 한다. 따라서 서버와 프락시 서버 사이의 네트워크 상의 트래픽을 감소시키고, 서버의 부하와 클라이언트의 전송지연도 감소시킬 수 있다.

2.2 웹 캐싱

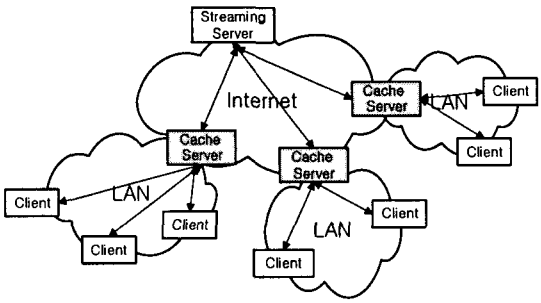
웹 캐싱이란 텍스트나 이미지와 같은 데이타를 포함하는 웹 문서에 대한 캐싱을 말한다. 이것은 문서 단위로 캐싱이 수행되며 관련 연구들은 주로 웹 캐쉬의 재배치 알고리즘에 초점을 맞추고 있다. 웹 문서들에 대한 과거의 접근 정보를 이용하여 재배치를 수행하는데 주로 이용되는 정보로는

대표적인 알고리즘은 LRU, LFU, SIZE, LRU-SIZE, LRU-MIN 등이 있다[4].

2.3 스트리밍 미디어의 캐싱

스트리밍 미디어의 캐싱에 관한 연구는 전통적인 웹 캐쉬가 가진 결함을 보안하고 스트리밍 미디어를 효율적으로 서비스하기 위한 기술들을 제공한다. 이것은 서버의 부하를 감소시키고, 네트워크 트래픽을 감소시키고, 클라이언트로 전송하는 지연시간과 지터를 줄임으로써 서비스의 질을 향상 시킬 수 있다. 본 논문에서의 연구도 이 분야에 속한다.

3. 스트리밍 미디어를 위한 캐쉬 서버



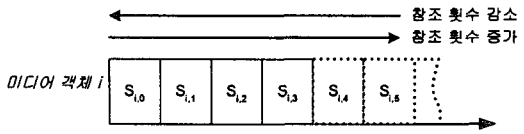
[그림 1] 인터넷 상의 캐쉬 서버

일반적으로 인터넷 상에서의 캐쉬 서버는 그림 1에서처럼 서버와 클라이언트 사이에서 클라이언트와 근접한 곳에 위치한다. 클라이언트가 서버로 데이터에 대한 요청을 보내면 캐쉬 서버는 요청을 받아서 자신이 저장하고 있는 데이터인지 아닌지를 검사한다. 캐싱되어 있는 데이터이면 직접 전송해주고, 그렇지 않으면 서버로 요청을 보내고 데이터를 받아서 디스크에 저장함과 동시에 클라이언트에게도 전송해준다. 스트리밍 미디어를 위한 캐쉬 서버일 경우에는 데이터 파일의 일부분 또는 전체를 캐싱할 수 있으며 만일 일부분을 캐싱하고 있을 경우에는 캐싱된 부분을 먼저 클라이언트에게 전송하고 캐싱되지 않은 부분을 서버로부터 요청하여 클라이언트에게 전달해준다.

3.1 스트리밍 미디어를 위한 캐싱 기법

일반적으로 비디오나 오디오와 같은 멀티미디어 데이터는 전통적인 텍스트나 이미지 데이터보다 크기가 매우 크기 때문에 객체 단위로 캐싱하는 방법은 바람직하지 않다. 따라서 미디어 객체의 일부분을 캐싱할 수 있어야 한다.

본 논문에서 제안하는 방식은 미디어 객체를 일정 크기의 세그먼트 단위로 캐싱하는 것이다. 미디어 객체가 참조되면 맨 앞의 세그먼트부터 캐싱시키고, 더 많이 참조될수록 점점 더 뒤의 세그먼트를 추가로 캐싱시킨다. 그림 2에 이런 캐싱 패턴을 나타내었다.



[그림 2] 미디어 객체의 캐쉬 패턴

추가의 세그먼트를 캐쉬하려할 때 캐쉬에 여유 공간이 없으면 캐쉬 재배치를 수행한다.

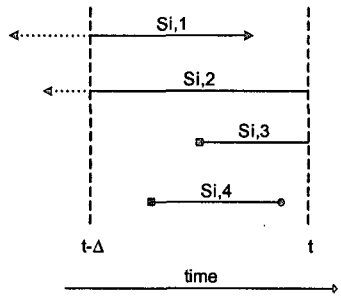
3.2 재배치 정책

그림 2에서 미디어 객체 i는 현재 세그먼트 Si,3까지 캐쉬되어 있다. 미디어 객체 i가 다음 번에 참조될 때 Si,4를 추가로 캐쉬시킬 것인지를 결정해야 한다. 그리고 캐쉬에 여유공간이 없다면 추가의 세그먼트를 캐쉬하기 위해 이미 캐쉬된 다른 미디어 객체의 세그먼트를 삭제해야 하는데, 어떤 것을 삭제할지도 결정해야 한다. 이러한, 결정 절차를 캐쉬 재배치라 한다. 그리고, Si,4 와 같이 캐쉬될 것인지 아닌지의 검사 대상이 되는 세그먼트를 후보 세그먼트라 하겠다. 또한, 캐쉬에서 삭제될 세그먼트를 희생 세그먼트라 하고, 그것의 미디어 객체를 희생 객체라 하겠다.

캐쉬 재배치를 위해서는 어떠한 재배치 척도가필요하다. 본 논문에서는 객체의 캐쉬 효율(Caching Utility)이라는 값을 캐쉬 재배치 척도로서 사용할 것을 제안한다. 캐쉬 효율은 어떤 미디어 객체를 캐쉬할 때 드는 비용에 대한 얻을 수 있는 이익으로 계산된다. 여기서 비용은 데이터가 캐쉬에서 차지하는 저장공간의 크기로 볼 수 있고, 이익은 캐쉬함으로써 캐쉬에서 적중될 수 있는 데이터량으로 볼 수 있다. 그런데, 여기서 잠시 스트리밍 미디어의 특성을 고려할 필요가 있다. 스트리밍 미디어는 전통적인 데이터와 달리 한번 접근할 때 미디어 객체의 일부분만이 재생될 수 있다. 따라서 본 논문에서는 캐쉬이익의 척도로서 재생 데이터량(PlaybackBytes)을 사용할 것이다. 이에 따라 캐쉬 효율은 다음과 같이 계산된다.

$$CachingUtility = \frac{\sum^{\Delta} PlaybackBytest}{CachedBytes}$$

$\sum^{\Delta} PlaybackBytes$ 는 특정 미디어 객체에 대해서 최근 Δ 시간 동안에 재생된 총 데이터 양을 나타낸다. 그림 3은 이 값을 측정하는 예를 보여준다.



[그림 3] Δ 시간 동안의 미디어 객체 i의 재생량

Si,1 ~ Si,4는 현재시간 t에 재생이 진행 중이거나 [t-Δ, t]에서 재생이 종료된 미디어 객체 i에 대한 스트림을 나타내고 실선으로 표시된 부분은 [t-Δ, t]에 포함되는 구간으로서 이 부분에서 재생된 데이터 양의 총합을 계산한다.

미디어 객체 i가 참조되면 후보 세그먼트를 캐쉬할지를 결정해야 한다. 그러기 위해서 먼저 미디어 객체 i의 캐쉬 효율 CU_i 값을 계산한다. 그리고 다른 캐쉬된 모든 미디어 객체의 캐쉬 효율값도 계산하여 그 중 최소의 캐쉬 효율(CUmin)을 갖는 희생 객체를 찾는다.

$$CU_{\min} = \min(CU_j)$$

CU_{min} 과 CU_i를 비교하여 아래와 같이 후보 세그먼트를 캐쉬할지를 결정한다.

case 1 : $CU_i \le CU_{\min}$

case 2: $CU_i > CU_{\min}$

case 1 일 경우에는 후보 세그먼트를 캐쉬하지 않는다. case 2 일 경우에는 후보 세그먼트를 캐쉬한다. 이때 후보 세그먼트를 캐쉬하기 위한 여유 공간이 없으면 캐쉬 효율 값이 가장 작은 희생 객체의 캐쉬된 세그먼트 중 가장 뒷부분인 희생 세그먼트를 캐쉬에서 삭제한다.

지금까지 설명한 캐쉬 효율을 이용하는 재배치 정책을 최소 캐쉬 효율(Smallest Caching Utility) 라 부르겠다.

4. 성능 평가

4.1 실험 환경

텍스트나 이미지와 같은 전통적인 데이터의 캐쉬 기법

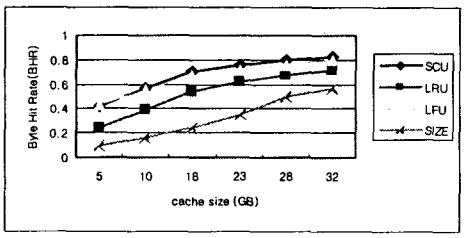
을 평가하는 척도로는 HR(Hit Ratio)를 많이 사용하였으나, 논문에서 제안하는 SCU는 스트리밍 미디어 객체의 일부분 또는 전체를 캐싱하는 기법이므로 HR 보다는 BHR(Byte Hit Ratio)를 사용하는 것이 바람직하다. 또한, 클라이언트에 대한 서비스 초기지연시간과 재배치 횟수를 측정하여 다른 재배치 정책들과 비교한다.

실험에서 가정하는 변수들은 다음과 같다.

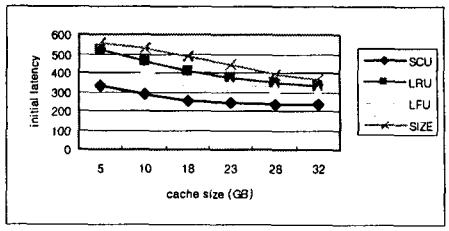
미디어 객체 크기	280MB ~ 420MB
요구 분포	Zipf distribution of $\theta = 0.27$
평균 요구 도착 간격	10 sec
미디어 객체 수	200
실험 시간	8000개의 사용자 요구를 처리하는 시간
캐쉬 서버와 클라이언트 사이의 전송 delay	100ms
서버과 캐쉬 서버 사이의 전송 delay	200ms
세그먼트의 크기	10MB

4.2 실험 결과

그림 4,5는 각각 BHR, 초기 지연 시간을 측정한 결과이다. SCU를 사용했을 때 LRU, LFU, SIZE 알고리즘을 사용한 경우에 비해서 BHR 이 가장 높게 나타났다. 이것은 캐쉬에서 적중되는 데이터량이 많고, 따라서 서버로부터 전송되는 데이터량이 그만큼 감소하여 서버의 부하와 네트워크 트래픽이 감소할 수 있음을 나타낸다. 그림 5에서는 SCU를 사용하면 클라이언트에 대한 초기지연시간을 확연히 줄일 수 있음을 보여준다. 이것은 스트리밍 미디어의 서비스에 있어서 아주 중요한 요소로 작용할 수 있다.



[그림 4] BHR



[그림 5] 초기 지연 시간

5. 결론

본 논문에서는 인터넷 상에서 스트리밍 미디어 시스템의 성능을 향상 시키기 위한 방법으로서 캐쉬 서버에 초점을 맞추었다. 특히, 캐쉬 서버에서 적용할 수 있는 캐쉬 재배치 정책을 제안하고, 성능을 분석하여 기존에 알려진 재배치 알고리즘들과 비교 분석을 하였다. 제안하는 재배치 정책은 스트리밍 미디어 객체의 일부분 또는 전체를 캐싱하는 방식으로서 미디어 객체의 앞부분을 우선적으로 캐싱하는 기법이다. 성능 분석 결과, LRU, LFU, SIZE 등의 기존에 잘 알려진 알고리즘들에 비하여 성능이 우수함을 알 수 있었다.

향후 연구 계획으로는 예측을 통한 데이터의 프리패칭과 네트워크 환경에 적응할 수 있는 캐싱기법 등이 있다.

[참고문헌]

[1] I. Tatarinov, V. Soloviev, A. Rousskov, "Static Caching in Web Servers", In Proc. of the 6th International Conference on Computer Communications and Networks (IC3N 97), Sept 1997.

[2] A. Bestavros, R. L. Carter, M. E. Crovella, "Application-Level Document Caching in the Internet", In Proc. of Workshop on Services and Distributed and Networked Environments, June 1995.

[3] 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.

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

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

[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