제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

# 멀티미디어를 위한 개선된 프락시 캐싱 관리 기법

홍병천*, 조경산**
단국대학교 정보컴퓨터학부
e-mail:*hongbc@dankook.ac.kr
**kscho@dankook.ac.kr

# An Improved Proxy Caching Management for Multimedia

Byung-Chun Hong*, Kyungsan Cho**
* **Div. of Information and Computer Science,
DanKook University

요 약

멀티미디어 특히 비디오 데이터는 대용량 파일 단위의 저장 특성을 가지며 사용자 접근 패턴이 기존의 웹 객체인 이미지 또는 텍스트와 다르기 때문에 기존의 웹 캐싱 정책은 멀티미디어 프락시 서버에는 적절하지 않다. 본 연구에서는 멀티미디어를 위한 프락시 서버의 캐쉬 히트율을 높이고 파일 단위 캐쉬 관리 정책을 세그먼트 관리에 적용하여 세그먼트 관리의 과부하를 줄이는 고정 크기 세그먼트 기법과 개선된 캐쉬 제거 기법을 제안하였다. 또한, 시뮬레이션을 통해 제안 기법의 성능 개선 정도를 분석 제시하였다.

## 1. 서론

웹(world wide web)의 급속한 성장으로 인터넷 이용자 수는 1999년에 이미 1억7천5백만을 넘어섰고, 웹 페이지의 수는 매일 약 일백만 개씩 증가하여 15억 개에 달하게 되었다[4]. 그러나 웹의 기하 급수적인 성장은 인터넷 사용자에 대한 접근지연과 네트워크 과부하라는 문제를 발생시켰다. 네트워크 접근 지연시간을 감소시키고 서버의 과부하를 감소시키기 위한 방법으로서 프락시 캐쉬 구조가 사용되었다.

웹에서 비디오나 오디오 같은 스트림의 전송과 요청이 증가함에 따라, 프락시 서버에서의 멀티미디어 데이터 캐싱이 보다 중요해졌다. 그러나, 멀티미디어 파일의 크기는 매우 크기 때문에 파일 전체를 캐싱하는 기존의 웹 서버 캐싱 기법을 멀티미디어 프록시 캐싱에 적용하는 것은 부적절하다. 예를들어, HTML 문서와 HTML 문서에 포함된 이미지 객체들 대부분은 수Kbtye 에서 수십Kbyte 정도의 크기인 반면에 멀티미디어 데이터는 미디어의 형태에 따라 수백Mbyte까지로 전형적인 웹 파일에 비해서 매우 크다[5]. 따라서 멀티미디어 객체를 전체로 캐싱하는 것은 프락시 캐시의 가용성을 낭비하게되어, 단지 몇 개의 멀티미디어 객체만으로 캐쉬가 채워질 수 있다. 그러므로 본 연구에서는 제한된 용량의 프락시 캐시 시스템의 가용성을 향상시키기 위해서 멀티미디어 데이터의 특성과 사용자 접근 패턴의 특성을 고려하여 기존 웹 서버에서의 파일 단위 캐쉬 관리 정책을 고정크기 세그먼트에 적용하는 프락시 캐싱 기법을 제안한다. 이때 세그먼트의 크기는 클라이언트와의 대역폭 제어 관리를 고려하여 선정함으로써 캐쉬관리와 대역폭에 따른 전송의 효과를 얻고자한다.

본 논문의 구성은 다음과 같다. 2장에서는 멀티미디어 웹 캐싱에 관한 관련 연구를 분석하고 3장에서는 본 연구에서 제안하는 프락시 캐싱 정책을 제안한다. 4장에서는 제안 기법을 시뮬레이션을 통해 분석 제시하고, 마지막으로, 5장에서 결론과 향후 연구방향을 제시한다.

## 2. 관련연구

### 2.1 웹 캐시 시스템

웹에서는 전통적 메모리 시스템과는 달리 저장되는 자료가 파일단위이고, 웹 참조는 대부분이 읽기(read)이며, 또한 참조의 인기도에 따라 파일의 참조 수가 정해지는 특성을 가지므로 기존 메모리 시스템의 대표적인 LRU 제거 정책은 적절하지 않다고 분석되었다[6]. 따라서 웹 참조의 특성에 따른 참조 횟수, 저장 크기, 캐시에 저장된 시간 등을 활용하는 제거 방식들이 제시되었다. 즉, LRU이외에 LFU, SIZE등의 기법들이 연구되었다. 또한 비용함수(또는 이익함수)를 이용한 정책으로 SLRU(Self-adjusted LRU), 정적캐시(Static Cache) 및 WLFU (Weighted- LFU) 등의 제거 기법들이 제시되었다. SLRU 기법에서는 파일을 캐쉬에 저장하여 얻는 이익을 나타내는 다음 식을 사용하였다.

$$\text{이익} = \frac{1}{T - T'} * SIZE$$

T :현재시간, T': 최종 참조시간, SIZE:파일크기

위 식의 값이 가장 작은 파일을 캐시에서 제거하는 정책을 사용하였다.

### 2.2 멀티미디어 데이터와 프록시 캐싱

텍스트나 이미지에 대한 전통적인 웹 프락시 캐싱이 높은 성능향상을 제공했음에도 불구하고 멀티미디어 데이터의 캐싱에는 효율적으로 이용되지 못했다. 따라서 제한된 용량의 캐쉬 가용성을 높이기 위하여 인기도에 따라 캐싱의 결정, 캐싱 저장량등을 차별하는 방법들이 제안되었다. 인기도를 결정하는 요소에는 데이터의 총 참조수, 일정 시간 동안의 참조수, 재생량, 최근성등이 고려되었다. [3]는 프락시 캐싱의 양을 유동적으로 변화시켜서 멀티미디어 데이터의 인기도에 따른 디스크 사용공간을 차별화 하는 방법을 제안했다. [1]은 프록시 캐시에 멀티미디어 데이터 파일 전부를 저장하는 대신에 파일의 앞부분만을 캐시에 저장하는 prefix 방법을 제안하였다. 이 제안 방법은 멀티미디어를 디스플레이하기 위한 클라이언트의 초기 지연을 감소하고 프락시 서버 캐싱의 저장 공간을 줄일 수 있는 제안이다. [2]에서는 대부분의 멀티미디어 데이터들은 부분적으로 캐쉬된다는 분석에 따라서 가변적 세그먼트 기반의 접근 방법을 제안하였다. 이 제안 방법은 프락시에 의해서 수신된 멀티미디어 데이터의 블록들을 가변 크기의 세그먼트로 그룹화 시키고 각 세그먼트 단위로 이익함수를 계산하여 캐쉬를 관리하는 정책이며, 이익함수는 다음과 같다.

$$\frac{1}{T - T'} * \frac{1}{SIZE}$$

T:현재 시간, T': 객체에 기록된 마지막 참조시간, SIZE: 세그먼트의 크기

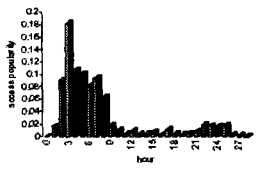
## 3.프락시 서버의 캐싱 기법

### 3.1 멀티미디어 데이터의 세그먼트관리

멀티미디어 데이터는 블록크기의 단위로 나누어져 서버에 저장된다. 클라이언트와 서버사이의 네트웍에 존재하는 프락시 서버는 캐쉬의 관리를 간편히 하기 위해서 여러 개의 블록을 하나의 세그먼트 단위로 그룹화 할 수 있으며, 이를 파일 단위로 관리하면 캐쉬 관리의 연산이 간단해진다. 세그먼트 단위로 캐쉬를 관리할 경우 세그먼트의 크기는 클라이언트와 프락시 사이의 데이터 전송 및 캐쉬 관리의 효율성에 따라 고정적 또는 가변적 크기로 정할 수 있다. 본 연구에서는 웹 서버 캐싱에서 사용한 파일 단위 캐싱기법에 멀티미디어 데이터의 특성을 고려하여 파일 단위로 상태 저장과 연산을 수행하되, 고정 크기 세그먼트의 캐쉬 관리 기법을 사용하였다. 세그먼트 단위로 저장하는 장점중의 하나는 각 멀티미디어 파일에 대해 캐싱되는 용량을 달리할 수 있어 캐쉬의 활용도를 높일 수 있는 것이다. 캐쉬에 저장되는 멀티미디어 파일의 크기는 인기도에 따라서 차별화 되어 저장된다. 인기도가 높은 상위 몇 개의 멀티미디어 파일은 전부 저장되어지며, 인기도가 낮은 멀티미디어 파일은 일부분 저장되어지며, 인기도가 낮더라도 초기지연을 감소시키기 위해서 항상 멀티미디어 데이터의 처음 일정 부분을 캐쉬에 저장되어 지도록 한다.

### 3.3 캐쉬 저장과 제거

[그림1]은 멀티미디어 데이터가 생성된 후 시간의 변화에 따라 요청되어지는 횟수를 나타낸 것이며, 생성 시간초기에는 높은 참조횟수를 보이다가 시간이 경과 할 수록 급격히 참조횟수가 감소하는 특성을 나타낸다[3]. 본 연구에서는 이러한 멀티미디어 데이터의 생성 시기와 참조횟수, 크기의 특성을 고려하여 프락시 서버의 캐싱 기법을 제안한다.




[그림1]멀티미디어의 생명주기

사용자의 초기 지연을 감소시키기 위해 [1][2]에서와 같이 멀티미디어 파일의 앞 부분만을 저장 관리하는 캐쉬와 나중 부분을 관리하는 캐쉬를 사용하였다. 앞 부분의 세그먼트는 모든 멀티미디어 파일에 대하여 고정된 크기로 제공하며, 많은 멀티미디어 파일을 저장하게 한다. 나중 세그먼트에 대해서는 가변크기 세그먼트에서의 연산 과부하를 줄이고, 멀티미디어 데이터의 특성인 참조횟수가 많은 데이터일수록 요청될 확률이 높다는 인기도 특성과 최근에 생성된 것일수록 요청확률이 높다는 최근성 분석을 이용하여 캐싱 값을 결정하는 이익함수를 사용하였다.

$$\frac{n^2}{T-S} * \frac{1}{SIZE}$$

n: 파일의 참조횟수, T: 현재시간, SIZE: 캐쉬에 저장된 파일의 크기, S: 최초로 파일이 요청된 시간

[그림2]은 본 연구에서 사용한 캐싱 알고리즘이다.

```
if(i < Kmin){ //초기 세그먼트를 저장하기 위한 캐쉬
  if(세그먼트 i를 저장 할 수 있는가?){
    제거할 파일의 초기 세그먼트를 선택, 제거;
    새로운 파일의 초기 세그먼트 i를 저장;
  }
}else{ //나중 세그먼트를 위한 캐쉬
  if(파일이 처음 참조 되었다면?)
    exit;
  while((세그먼트 i를 저장할 공간이 없고)and
        (제거할 세그먼트가 계속 발견된다면)){
    세그먼트 i가 포함된 파일 인기도를 갖는 객체 P;
    가장 낮은 인기도를 갖는 파일 객체 J;
    if(((n+n)/(T-S)) * P > ((n+n)/(T-S)) * J){
      객체 J의 마지막 세그먼트 재배치;
      캐쉬의 여유공간을 증가;
    }
  }
  if(파일P의 세그먼트i를 저장할 공간이 있다면)
    세그먼트 i를 캐싱;
}
```

[그림2]파일P의 세그먼트 i를 위한 캐싱 알고리즘

## 4.성능분석

### 4.1 성능 분석 환경

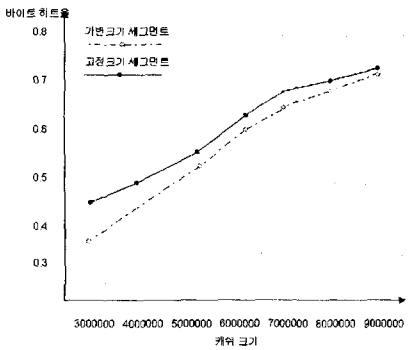
본 연구에서 제안한 프락시 캐쉬 서버 모델의 평가를 위해 [2]와 유사한 환경 [그림3]에서 이벤트 처리 시뮬레이션을 통해 성능을 분석하였다.

| 기호 | 정의(기본값) |
|---|---|
| B | 각 파일의 평균 블럭수(2,000블럭) |
| t | 평균 요청간격(60초) |
| C | 전체 캐쉬 용량(400,000블럭) |
| Cinit | 초기세그먼트를 저장하기 위한 부분(10%) |
| Kmin | 초기 세그먼트(4개의 세그먼트,32블럭) |
| M | 파일의 수 (2,000) |
| Zipf (x,M) | 파일의 Zipf법칙 (Zipf(0.2,2000)) |
| k | 인기 파일의 최대 이동간격(5) |
| R | 인기 파일의 이동 사이의 요청 숫자(1560) |
| n | 파일의 요청횟수(참조수) |

[그림3] 성능 분석 환경

### 4.2 시뮬레이션 결과

[2]에서는 가변크기 세그먼트기법과 prefix/suffix, full 캐쉬 기법을 비교 평가하였으며, 가변크기 세그먼트기법의 히트율이 가장 우수한 성능을 보였다. 따라서, 본 연구에서는 가변길이 세그먼트 기법과 본 연구에서 제안한 고정길이 세그먼트 기법의 캐쉬 크기에 따른 바이트 히트율을 비교하였다.




[그림4]캐쉬 크기에 따른 히트율 비교

[그림4]에서 보이듯이 본 연구에서 제안한 고정크기 세그먼트 기법이 가변크기 세그먼트 기법에 비해 향상된 히트율을 보여준다.

## 5. 결론 및 향후 연구

멀티미디어 특히 비디오 데이터는 대용량 파일 단위의 저장 특성을 가지며 사용자 접근 패턴이 기존의 웹 객체인 이미지 또는 텍스트와 다르기 때문에 멀티미디어 프락시 서버를 위한 새로운 관리 기법이 요구되어 왔다. 본 연구에서 제시한 고정크기 세그먼트와 캐쉬 제거 기법은 가변크기 세그먼트 기법보다 프록시 서버의 연산과 상태저장 과부하를 줄일

수 있으며, 또한 높은 캐쉬 히트율을 보였다. 제안 기법과 연계하여, 클라이언트와 프락시 서버사이의 대역폭을 고려하여 적절하게 세그먼트를 전송하여 멀티미디어 재생의 QoS를 높이기 위한 세그먼트 크기 선정에 대한 연구와 대역폭에 따른 전송제어 방법의 연구가 수행 중이다. 또한, 멀티미디어 파일의 히트율에 영향을 미치는 파일의 인기도에 대한 분석과 사용자의 접근 패턴에 따른 초기 세그먼트의 크기 선정 방법도 함께 연구되고 있다.

## 6.참고문헌

[1]Subhabrata Sen, Jennifer Rexford and Don Towsely, "Proxy Prefix Caching For Multimedia Streams," IEEE INFOCOM, April, 1999, www.ieee-infocom.org/1999/papers/09d_04.pdf

[2]Kun-Lung Wu, Philip S.Yu, and Joel L.Wolf, "Segment-Based Proxy Caching of Multimedia Streams," www10.org/cdrom/papers/183/index.html

[3]박성호, 임은지, 최태욱, 정기동, "인터넷상에서 NOD 서비스를 위한 연속 미디어 전송 및 푸쉬-캐싱 기법," 한국정보처리학회 논문지, 제7권, 제6호, pp. 1766~1777.

[4]M. A. Goulde, "Network caching guide optimizing web content delivery," March, 1999, http://inktomi.com/products/network/traffic/tech/CachingGuide.pdf

[5]J. Jung, "Enhancing Web Caching Architecture with the Replication of Large Objects," Master thesis, Korea Advanced Institute of Science and Technology, Korea, February, 1998.

[6]안효범, 조경산, "웹 서버의 참조 특성 분석과 성능 개선," 한국정보처리학회 논문지, 제8-A권, 제3호 pp. 0201~0208.