모바일 환경에서 멀티미디어 컨텐츠의 요청패턴 분석에 기반한 프록시 캐싱 알고리즘

이상민O 김법중 낭종호
서강대학교 컴퓨터학과
{lesminO, neties}@mlneptune.sogang.ac.kr, jhnang@ccs.sogang.ac.kr

A Proxy Caching Algorithm based on the Analysis of Multimedia Contents Request Patterns in Mobile Environments

Sangmin LeeO Bubjung Kim Jongho Nang
Dept. of Computer Science Sogang University

요 약

본 논문에서는 실제 모바일 환경에서의 멀티미디어 컨텐츠의 요청 패턴을 분석하고 그 결과를 반영한 캐싱 알고리즘을 제안한다. 로그 데이터를 분석한 결과 컨텐츠의 연속적인 요청이 매우 짧은 시간 동안 이루어지는 시간적인 특성을 발견했으며, 컨텐츠의 다른 버전 사이의 요청 시간이 짧음을 통해서 버전별 컨텐츠의 요청이 응집되어 나타남을 확인했다. 제안된 알고리즘에서는 시간적인 특성과 컨텐츠의 인기도 특성을 반영하기 위해 측정 시간 윈도우 내에서 컨텐츠의 요청 횟수를 측정하고, 요청 횟수의 임계값 이상의 컨텐츠에 대해서 유지 시간 단위만큼의 컨텐츠 만료 시간을 정한다. 또한 공간적인 특성을 이용하여 컨텐츠가 캐시에서 제거될 때, 만료시간이 지난 컨텐츠의 다른 버전을 동시에 캐시에서 제거하는 방법을 사용한다. 실험 결과, 캐시의 크기가 작은 경우 제안된 알고리즘이 기존 알고리즘에 비해 1~5% 정도 성능향상을 보였다.

1. 서 론

인터넷 사용 인구가 늘어가고 네트워크 대역폭이 늘어남에 따라 대용량 멀티미디어 데이터의 전송도 빈번해졌다. 이로 인한 네트워크의 트래픽을 줄이고 클라이언트의 요청 지연시간을 줄이기 위해 프록시 캐싱 방법이 도입되었다. 프록시 캐싱 방법은 고전적인 LRU, LFU 방법을 기반으로 하고 있으며 웹 환경에서는 요청된 객체의 참조 횟수, 최근 참조된 시간 정보, 참조되는 객체의 크기, 객체의 평균 검색시간, 객체의 만료 시간이나 객체의 수정시간 등을 파라미터로 사용한다[1]. 본 논문에서는 실제 로그 데이터를 이용한 모바일 환경에서 멀티미디어 컨텐츠의 요청 패턴을 분석하고, 그 결과를 기반으로 보다 효율적인 캐싱 알고리즘을 제안하며 기존 알고리즘과의 비교분석을 통해 제안한 알고리즘의 성능을 살펴본다.

2. 모바일 환경에서의 멀티미디어 컨텐츠 요청 패턴 분석

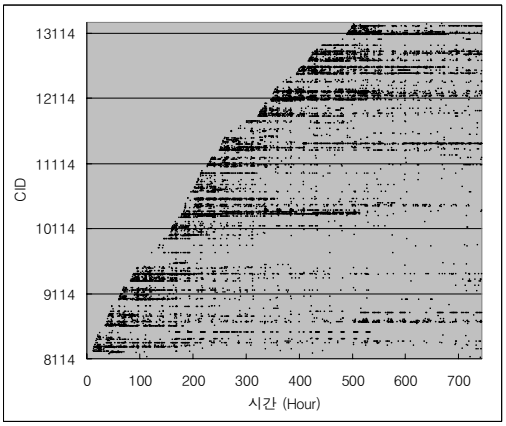
로그 데이터는 A사의 모바일 동영상 서비스를 기반으로 2004년 5월 1일부터 2004년 5월 31일까지의 컨텐츠 서버의 로그를 이용하였다. 컨텐츠들은 해상도와 코덱 버전에 따라 3가지로 나뉜다. 112x96 해상도를 가지는 V_SMALL버전, 128x112의 해상도를 가진 V_MEDIUM 버전과 176x144의 해상도를 가진 V_ LARGE 버전이 있다. 각각의 컨텐츠를 버전 별로 살펴보면, 컨텐츠의 수는 V_SMALL 버전이 10,000개 정도로 가장 많지만 실제 요청이 오는 횟수는 V_LARGE 버전이 270,000회 정도로, 한달 동안의 요청 횟수의 절반을 차지했다. 이를 통해서 많은 컨텐츠를 가지는 버전이 많은 요청이 오는 것은 아니라는 것을 알 수 있다.

<표1> 버전 별 통계 (2004년 5월 한달)

버전	컨텐츠 개수	요청 횟수	전체 크기 (MB)	평균 요청 횟수
V_SMALL	10,307	184,881	3,254	17.9
V_MEDIUM	5,557	74,284	1,943	13.4
V_LARGE	9,661	273,302	3,117	28.3
전체	25,525	532,467	8,314	20.9

2.1. 시간적인 특성

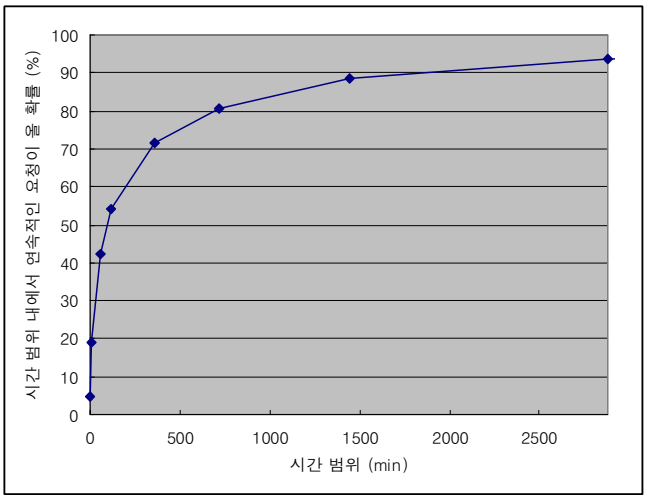
실제로 어떤 시간적인 특성이 있는지 찾기 위해서 <그림1>과 같이 실제 로그 데이터의 시간적인 특성의 분포를 살펴보았다.



<그림1> 컨텐츠의 시간에 따른 요청 분포

· 본 연구는 한국과학재단 목적기초연구(R01-2002-000-00141)지원으로 수행되었음

CID는 컨텐츠의 ID를 말하며 각 CID는 컨텐츠의 처음 요청이 온 순서대로 번호를 매겨서 8000번 대부터 5000개를 발췌하였다. 가로로 나타나는 점이 직선으로 나타나는 컨텐츠는 오랜 시간 동안 지속적인 요청이 오는 컨텐츠를 나타낸다. 요청의 지속시간은 컨텐츠 별로 많이 다르며 컨텐츠가 긴 시간 동안 요청이 지속되는 경우는 매우 적음을 알 수 있다. 이는 컨텐츠에 대한 사용자의 선호도와 관련이 있지만, 사용자의 선호도를 시간적으로 볼 때 지속시간의 범위는 매우 다양하며 대부분 짧은 시간 동안 지속되는 것을 알 수 있다. 따라서 캐시에 컨텐츠를 유지할 때 컨텐츠 별로 다른 시간 동안 유지되어야 함을 알 수 있고, 컨텐츠의 요청이 연속적으로 올 때 그 시간 범위를 조절함으로써 캐시의 성능을 높일 수 있음을 알 수 있다.

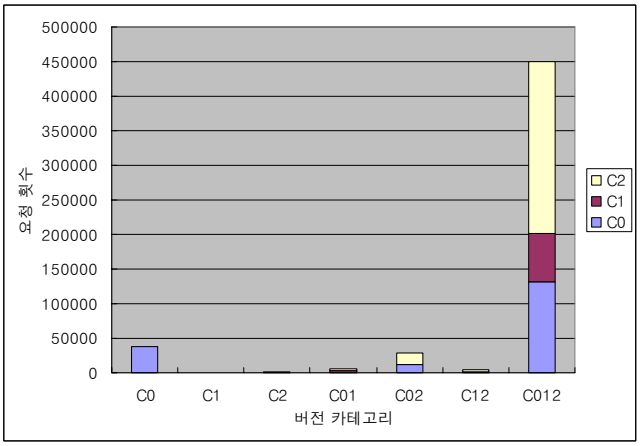


<그림2> 시간에 따른 연속적인 요청이 올 확률

같은 컨텐츠가 어느 정도의 시간 범위 내에서 연속적인 요청이 오는지를 알아보기 위해서, 시간 범위 내에서 연속적인 요청이 올 확률을 측정했다. <그림2>는 시간 범위를 변화시키면서 연속적인 요청이 올 확률을 구한 것이다. 1분 에서 360분 범위 내에서는 연속적인 요청이 올 확률은 급격하게 증가하며 시간 범위가 360분일 때 71%의 확률을 보여준다. 360분 이상의 시간 범위에서는 확률의 완만한 증가를 볼 수 있다. 시간이 1440분 (1일) 일 경우에는 연속적인 요청이 오는 확률은 90%에 육박하며, 그 이상에는 거의 변화가 없음을 볼 수 있다. 같은 컨텐츠에 대해서 컨텐츠의 평균적인 요청시간 차이가 367분(6시간)정도 임을 감안할 때, 이 범위 내에서 연속적인 요청이 오는 컨텐츠는 약 70%이며 이는 연속적인 요청이 오는 확률의 상승세가 둔화되는 지점이다. 기존의 캐싱 알고리즘과 연관 지어 생각해보면, LRU는 최근의 요청이 오지 않은 컨텐츠를 캐시에서 제거하게 되는데 요청이 오는 주기에 따라 캐시의 유지되는 시간범위가 달라진다. 하지만, 캐시를 유지할 때 연속적인 요청이 오는 시간 범위를 알 수 있고 그 시간 범위 내에서 캐시를 유지하게 한다면 더 높은 캐시 효율을 얻을 수 있을 것이다.

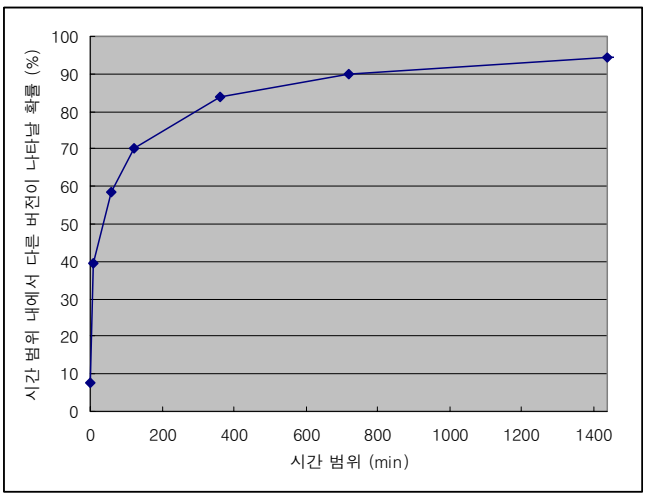
2.2. 공간적인 특성

공간적인 특성은 특정 컨텐츠들 사이에서 일어나는 특성으로, 컨텐츠들이 짧은 시간 내에 동시에 요청이 올 경우의 관계를 말한다. 앞에서 살펴본 바와 같이 컨텐츠들은 최대 3가지 버전을 가질 수 있는데, <그림3>은 V_SMALL 버전을 C0, V_MEDIUM 버전을 C1, V_LARGE 버전을 C2라고 표시한 것으로 두 가지 버전 이상을 가지는 경우에서는, 예를 들어 V_SMALL 버전과 V_MEDIUM 버전을 가지는 경우, C01와 같이 표현하였다. 버전 카테고리 별 요청 횟수 분석 결과, 버전을 1개나 2개 가지고 있는 경우보다 3가지 모든 버전을 가지고 있는 경우가 압도적인 비율로 많음을 알 수 있다.



<그림3> 버전 카테고리 별 요청 횟수

이러한 컨텐츠의 여러 버전 사이의 응집도를 구하기 위해 컨텐츠 하나의 버전이 요청되었을 때, 다른 버전이 요청이 되는 확률을 구해보았다. 그 결과 <그림4>와 같이 1분에서 120분 사이에서 다른 버전이 나타날 확률이 급격하게 증가하며, 120분의 시간 범위 내에서 70%의 확률을 보여준다. 120분이 지나면 점점 완만한 곡선을 나타나게 되며 720분(12시간)에는 확률이 90%에 이르게 된다. 따라서 각 컨텐츠가 가지는 다양한 버전들을 캐시에 적용시킬 때 하나로 묶어서 관리하는 방법을 생각해 볼 수 있다.



<그림4> 시간 범위 내에서 컨텐츠의 다른 버전이 요청될 확률

3. 모바일 환경에서 멀티미디어 컨텐츠를 위한 프록시 캐싱 알고리즘

2장에서 살펴본 시간적인 특성을 반영하기 위해, 지속적인 요청을 받는 컨텐츠에 대해 캐시에서 제거되지 않는 우선권을 가지는 만료 시간($T_i^{expired}$)을 컨텐츠에 대한 요청이 올 때마다 <그림5>와 같은 방식으로 설정한다.

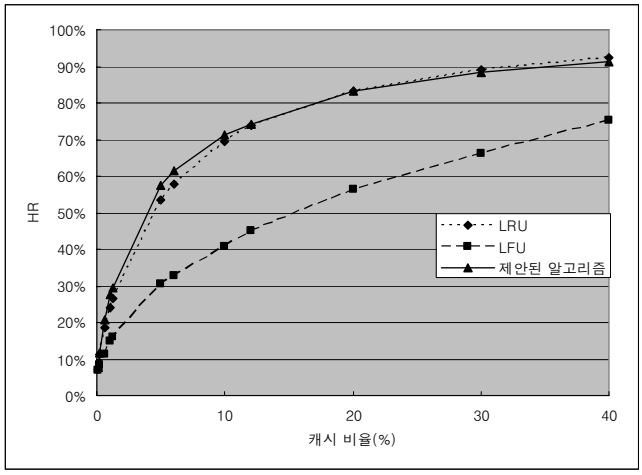
```
Procedure Set_T_expired(C_i, T)
{
    LAT_i = Current Time T
    N = F_i

    for(j=1;j<N;i++){
      if (PT_i^j < T_i - T^dur){
        Pop(PT_i^j)
        F_i --
        continue
      }
      Else{
        Push(PT_i^j, T)
        F_i ++
        break
      }
    }
    If( F_i >= F_thres ){
        T_i^expired = T + T^dur
    }
}
```

<그림5> 컨텐츠의 만료시간 결정 알고리즘

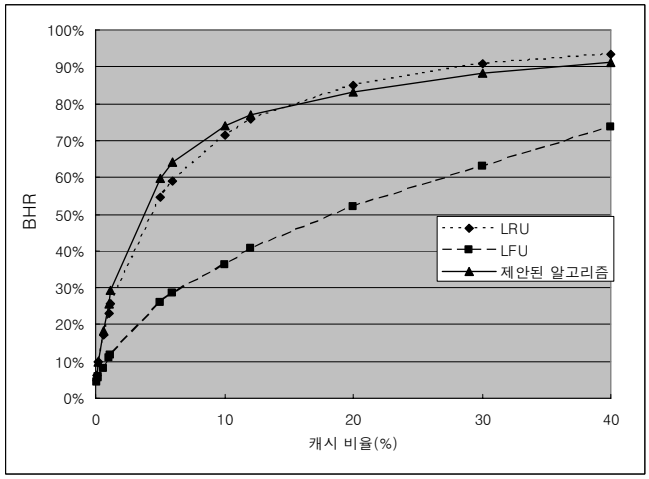
만료시간을 결정하기 위해서 컨텐츠의 인기도를 반영하기 위한 컨텐츠의 요청횟수(F_i)를 일정시간마다 측정하며, 요청된 시간으로부터 캐시에 얼마나 오래 저장할 것인지의 유지시간(T^{dur})과 일정 측정시간($T^{measure}$)동안의 요청횟수를 통해 요청횟수 임계값 (F^{thres})을 구해서 사용한다. PT_i^j는 컨텐츠 i의 지난 시간 동안의 요청시간을 저장한 큐 (j는 큐 안에서의 인덱스)를 의미한다. 또한, 앞서 살펴본 공간적인 특성으로, 한 컨텐츠의 여러 버전은 요청이 따로 떨어져서 오는 것이 아니라 짧은 시간 내에서 응집되어 나타남을 알 수 있었다. 이를 반영하여 컨텐츠의 한 버전이 캐시에서 제거될 때 다른 버전도 동시에 캐시에서 제거하는 방법을 사용한다. 단, 만료시간이 설정되어 있는 컨텐츠에 대해서는 컨텐츠가 제거되는 시점에 해당 컨텐츠의 버전이 만료되었을 경우에만 제거함으로써, 시간적인 특성과 공간적인 특성을 모두 반영하도록 한다.

4. 실험 결과 및 분석

3장에서 제안한 알고리즘을 기반으로, 2004년 5월 한 달간의 로그 데이터를 이용하여 시뮬레이션을 수행했다. 로그 분석 결과, $T^{measure}$ = 120분, T^{dur} = 24시간, F^{thres} = 2로 설정하였으며, <그림6>,<그림7>은 캐시 비율을 변경해가면서 HR(Hit Ratio)과 BHR(Byte Hit Ratio)을 통하여 기존의 알고리즘과 성능을 비교 측정한 결과이다. 기존 알고리즘 중에 LFU는 LRU에 비해서 현저히 떨어지는 성능을 보여주며, 캐시의 비율이 늘어날수록 그 격차는 좁아지고 있다. 제안한 알고리즘과 LRU는 거의 비슷한 성능을 보여주고 있는데, 캐시 비율이 10% 정도까지는 HR의 경우에는 제안한 알고리즘이 1~5% 정도의 성능 향상을 보여주고 있으나, 캐시 비율이 10% 이상일 경우에는 1~2% 정도의 성능 저하를 보여주고 있다. 이는 캐시의 크기가 커질수록 캐시에 저장할 수 있는 컨텐츠는 많아지는데 기존 알고리즘은 오랫동안 컨텐츠를 유지할 수 있지만, 제안한 알고리즘은 고정된 만료시간을 가지고 있어서 캐시의 교체 시기에 만료시간이 지난 컨텐츠가 우선 제거되기 때문에 이러한 성능 저하가 일어난다.



<그림6> 기존 알고리즘과의 HR 비교 그래프



<그림7> 기존 알고리즘과의 BHR 비교 그래프

5. 결론

본 논문에서는 2004년 5월 한 달 동안 서비스된 컨텐츠의 실제 로그 데이터를 이용하여 컨텐츠 요청의 패턴을 분석하여 시간적인 특성과 공간적인 특성을 구하고, 이러한 특성을 반영한 캐싱 알고리즘을 제안하였다. 제안한 캐싱 알고리즘을 실제 서비스에 적용한다면 적은 캐시 비율로도 성능 향상을 꾀할 수 있을 것이다.

6. 참고문헌

[1] Abdullah Balamash and Marwan Krunz, “An Overview of Web Caching Replacement Algorithms,” *IEEE Communications Survey & tutorials, Volume 6, No 2*, Second Quarter , 2004, pp. 44-56.