모바일 P2P 환경에서 객체 크기 기반의 협력적인 캐시 교체 정책

박교성, 송진우, 양성봉
연세대학교 컴퓨터과학과
e-mail:imkspark@cs.yonsei.ac.kr

Cooperative Cache Replacement Policy based on Object Size in Mobile P2P Environment

Kyo-Sung Park, Jin-Woo Song, Sung-Bong Yang
Dept. of Computer Science, Yonsei University

요 약

최근 모바일 환경에서의 P2P 협력적인 캐싱 기술에 대한 연구가 활발히 이루어지고 있다. 모바일 P2P 협력적인 캐싱이란 객체를 요청한 피어의 전파 범위 안에 있는 다른 피어들을 찾아 원하는 데이터가 있는지 살펴본 후 원하는 데이터가 주변 피어에 없을 때에만 서버에 요청을 보내는 방식으로 데이터 탐색 성능 개선과 제한된 피어의 저장 공간을 효율적으로 이용할 수 있는 장점이 있다. 본 논문에서는 모바일 P2P 네트워크 환경에서 이동성을 가진 피어의 저장 공간을 효과적으로 이용하기 위한 객체 크기 기반 P2P 협력적인 캐시 교체 정책을 제안하고자 한다. 가까운 위치 좌표를 가진 피어들을 하나의 그룹으로 묶은 그룹 기반의 모델링 환경에서 객체의 크기에 따른 캐시 교체 정책을 실제 웹 로그 트레이스에 적용하여 실험하였다. 실험 결과, 제안하는 교체 정책이 기존의 교체 정책들과 비교하여 더 우수한 성능을 보였다.

1. 서론

최근 모바일 환경에서의 P2P 협력적인 캐싱 기술에 대한 연구가 활발히 이루어지고 있다. 모바일 P2P 협력적인 캐싱이란 요청을 발생한 피어(모바일 디바이스)의 전파 범위 안에 있는 다른 피어들을 찾아 원하는 데이터가 있는지 살펴본 후, 원하는 데이터가 주변 피어에 없을 때에만 서버에 요청을 보내는 방식이다. 피어 성능의 꾸준한 개선으로 인하여 피어들끼리 정보를 공유하는 협력적인 캐싱이 많이 사용되고 있다[1].

모바일 P2P 협력적인 캐싱 시스템은 일반적으로 모바일 클라이언트 캐시, MSS(mobile support station) 캐시, MSS 디스크의 3계층으로 이루어져 있다[1]. 피어가 MSS에게 객체를 요청하면 MSS는 MSS 캐시에서 객체를 찾아보고 MSS 캐시에 원하는 객체가 없을 경우 MSS 디스크로부터 요청한 객체를 찾아 피어의 캐시에 전달하면서 동작한다.

P2P 네트워크 환경에서 이웃을 찾는 방법은 다음과 같은 세 가지가 있다. 첫째, 디렉토리 기반(directory-based)의 방식으로 중앙 서버에 각 객체를 가진 피어들을 등록해 놓고 객체를 요청한 피어에게 해당 객체를 소유한 이웃 피어의 목록을 알려주는 시스템이다[2]. 두 번째는 플러딩 기반(flooding-based)의 방식이다[3]. 이는 자신과 인접한 모든 이웃에게 객체 요청 메시지를 보내는 방식으로 과도한 메시지 전달로 인해 네트워크에 과도한 부하가 생기는 단점이 있다. 세 번째는 DHT 기반(distributed hash table-based)의 방식으로 플러딩 기반 방식에 비하여 적은 메시지 전달로 빠르게 원하는 객체를 찾을 수 있지만 요청도가 높은 객체를 소유한 피어는 많은 부하를 가지게 된다[4].

본 논문에서는 디렉토리 기반의 모바일 P2P 캐싱 시스템에서 위치 좌표와 이동성이 유사한 피어들을 하나의 그룹으로 묶고 객체의 크기에 따라 다른 교체 정책을 적용하는 캐시 교체 정책 (OSRP : object size-based replacement policy)을 제안한다.

논문의 구성은 다음과 같다. 2장에서 관련 연구를 소개하며, 3장에서 본 연구에서 제안하는 모바일 캐싱 시스템의 교체 정책을 기술한다. 4장에서 실험을 통한 성능 평가 결과를 설명하고, 5장에서 결론을 맺는다.

2. 관련 연구

2.1 그룹 모빌리티 모델(Group Mobility Model)

무선 네트워크 환경에서 피어는 다양한 방향과 속도로 이동할 수 있다. 모빌리티 모델은 이러한 피어 각각에 대해서 속도와 방향을 표현할 뿐 아니라 국부적인 지역에 속한 모든 피어들이 가지는 공통의 움직임도 나타내는데 목적을 둔다. 모빌리티 모델 중에서 RPGM(reference point group mobility) 모델은 가까운 위치의 피어들은 비슷한 액세스 패턴(access pattern)과 이동성을 가질 확률이 높다는 사실을 바탕으로 지역적으로 유사한 피어를 하나의 그룹으로 묶은 모델이다[5].

RPGM 모델의 각 그룹은 그룹의 위치, 속도, 방향, 가속 등의 이동성을 나타내는 중점을 가지고 있다. 각 피어는 그룹의 범위 안에서 그룹의 중점 부근에 임의로 위치가 정해진다. 따라서 각 피어는 그룹의 이동성 안에서 독립적으로 개별적인 이동이 가능하다.

2.2 기존의 캐시 교체 정책

대표적인 캐시 교체 정책으로는 다음과 같은 것들이 있다. 첫째, 최신 기반(recency-based)의 방식이다. 이는 객체마다 참조된 시간을 기록하여 가장 오래된 객체를 캐시에서 내보내는 방법으로 대표적으로 LRU(least recently used)를 들 수 있다[6]. 둘째, 빈도 기반(frequency-based)의 방식으로 객체의 총 요청 횟수를 바탕으로 가장 적게 참조된 객체가 교체 대상이 되는 방법으로 LFU(least frequently used)가 대표적인 방법이다. 마지막으로, 함수 기반(function-based)의 방식이 있다. 이는 객체의 각 요소에 가중치를 고려한 수식을 이용하여 최대 또는 최소 값의 객체를 교체 대상으로 삼는 방법으로 대표적인 것으로는 GD-Size가 있다[7].

3. 객체 크기 기반의 교체 정책

3.1 기본 동작

모바일 P2P 시스템은 하나의 피어가 특정 객체를 요청하면 해당 피어와 같은 그룹에 속한 다른 이웃 피어들에게 요청된 객체가 있는지를 찾아본다. 만약 요청된 객체를 가진 이웃 피어가 있다면 그 이웃 피어로부터 객체를 서비스 받는다. 그렇지 못한 경우에는 피어는 MSS에게 객체를 요청하고 MSS로부터 요청된 객체를 받게 된다.

그러나 모바일 환경에서는 각 피어의 저장 공간이 기존 인터넷 P2P 환경의 피어들에 비하여 크게 제한되어 있다. 따라서 저장 공간이 제한적인 피어의 저장 효율성을 높이면서 캐싱 시스템의 성능을 높이기 위한 교체 정책이 요구된다. 본 논문에서는 제안하는 교체 정책은 객체 크기에 따라 서로 다른 교체 정책을 사용한다. 즉, 작은 크기의 객체는 객체를 요청한 피어마다 로컬 캐시에 저장하고, 크기가 큰 객체는 같은 그룹안의 이웃 피어에 저장하는 정책을 적용한다.

요청한 객체를 피어의 로컬 캐시에 저장할 때, 로컬 캐시가 이미 가득 찼다면 객체의 크기를 확인한다. 객체 크기가 임계값보다 작다면 로컬 캐시에 저장된 객체들 중에서 교체 대상을 찾는다. 객체의 크기가 크거나, 오래 전에 참조된 것이거나, 요청 횟수가 적은 것일수록 교체 대상이 될 확률이 높아진다.

객체의 크기가 임계값보다 큰 경우에는 같은 그룹에 있는 이웃 피어 중에서 가장 활동성이 낮은 (최근 요청 횟수가 적은) 피어를 골라 그 피어를 저장 매체로 이용한다. 가장 활동성이 낮은 피어의 로컬 캐시도 가득 찬 경우에는 가장 오래 전에 참조되고 요청 횟수가 가장 적은 객체를 교체 하여 저장한다.

3.2 로컬 캐시에서의 교체 정책

객체를 요청한 피어의 로컬 캐시에는 임계값보다 작은 크기의 객체를 저장한다. 최대한 많은 수의 유용한 객체를 저장하기 위해 다음과 같은 수치로 객체를 비교한다.

$$Cost_{small} = \frac{Freq_{object} \times Recent_{object}}{Size_{object}} \quad (1)$$

식 (1)에서 $Size_{object}$는 객체 크기를, $Freq_{object}$는 객체의 요청 횟수를 표현하며 $Recent_{object}$는 얼마나 최근에 객체를 참조했는지를 나타낸다. 크기가 작거나, 요청 횟수가 많거나, 최근에 요청된 객체일수록 $Cost_{small}$값을 증가시키고 $Cost_{small}$값이 큰 객체가 캐시에 저장된다.

로컬 캐시에 저장되기를 원하는 객체와 로컬 캐시 안에 있는 객체의 $Cost_{small}$값을 비교하여 만약 로

컬 캐시 안에 있는 객체의 $Cost_{small}$값이 크다면 저장되기를 원하는 객체는 로컬 캐시 안에 저장하는 것을 포기한다. 반면, 로컬 캐시에 저장되기를 원하는 객체의 $Cost_{small}$값이 로컬 캐시 안에 있는 객체의 $Cost_{small}$값보다 크다면 로컬 캐시 안에 있는 객체를 삭제하고 저장되기를 원하는 객체를 로컬 캐시에 저장한다. 이와 같이 $Cost_{small}$값이 큰 객체를 로컬 캐시에 저장하여 교체 정책을 진행한다.

3.3 이웃 피어에서의 교체 정책

저장하려는 객체의 크기가 임계값보다 큰 경우 저장 공간의 효율성을 위해 같은 그룹에 속한 이웃 피어 중 가장 요청 횟수가 적은, 즉 활동성이 낮은 피어를 선택하여 그 피어에 객체를 저장한다. 이 때, 가장 활동성이 낮은 피어를 선택하기 위해서 아래와 같은 식을 이용한다.

$$Selection_{peer} = Freq_{peer} \times (RRT_{peer} - ST) \quad (2)$$

식 (2)에서 $Freq_{peer}$는 특정 피어가 객체를 요청한 총 횟수를 나타내고 RRT_{peer}는 가장 최근에 특정 피어가 요청한 시간(recent request time)을 나타내며 ST는 웹 로그 트레이스가 시작된 시간(start time)을 의미한다. 따라서 $RRT_{peer} - ST$ 값이 클수록 최근에 요청이 발생했음을 나타내어 피어의 활동성과 밀접한 관계를 형성한다. 한 그룹에 속한 모든 피어의 $Selection_{peer}$값을 계산하여 가장 작은 값의 피어를 가장 활동성이 낮은 피어로 결정한다.

저장 공간으로 쓰일 가장 활동성이 낮은 피어가 결정되면 선택된 피어의 로컬 캐시 안에서도 객체의 교체가 일어나게 된다. 여기서의 교체 정책은 객체의 크기를 고려하지 않고 객체의 요청 빈도와 최근 객체를 접근한 값을 가지고 결정하며 식 (3)에서 얻은 값을 사용한다.

$$Cost_{large} = Freq_{object} \times Recent_{object} \quad (3)$$

식 (3)에서 $Cost_{large}$값은 $Cost_{small}$값에 객체의 크기를 곱한 것과 같은 값이다. 이웃 피어의 캐시에서는 $Cost_{large}$값을 비교 요소로 두어 $Cost_{large}$이 작은 객체를 캐시에서 삭제될 객체로 선별하게 된다.

4. 실험 결과 및 분석

4.1 실험 환경

실험을 수행하기 위해 사용한 웹 로그는 1998년 월드컵 데이터 셋[8]이다. 이 데이터 셋은 타임스탬프, 클라이언트 ID, 객체 ID 및 객체 크기로 구성되어 있다. 이것은 모두 2,770,108명의 클라이언트가 총 1,352,804,107개의 요청을 발생하는 데이터 셋으로 실험을 위해 로그 파일 중 임의로 10일의 로그를 추출하였고 각각에 대해서 임의로 100명의 클라이언트를 선정한 후 이들의 객체 요청을 임의로 1시간씩 선택하였다. 결과 값은 각 경우의 평균값을 계산하였다.

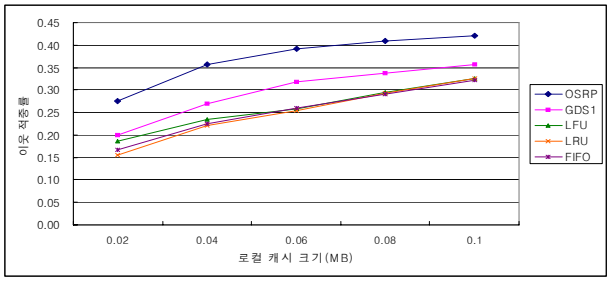
모바일 네트워크는 RPGM 모델을 사용하였고 225m의 MSS 서비스 영역과 각 피어 당 75m의 전파 범위를 가진다. 객체의 크고 작음을 분류하는 임계값은 250부터 10000까지 다양하게 실험을 하였고 결과 그래프 경우에는 임계값을 500으로 설정하여 실험하였다. 식 (1)에서 $Recent_{object}$의 초기값을 10으로 설정되었고 해당 피어에 요청이 발생할 때마다 1씩 감소하다가 이 객체에 다시 요청이 들어온 경우에 10의 값을 주어 가장 최근에 객체가 참조되었음을 나타낸다. 또 객체의 크기가 변경된 경우에 객체는 업데이트된 것으로 간주하여 객체의 일관성을 유지하도록 하였다.

본 논문에서는 제안하는 교체 정책인 OSRP를 실제 웹 로그 트레이스로 실험을 하여 LRU, LFU, GDS1, FIFO와 비교한다.

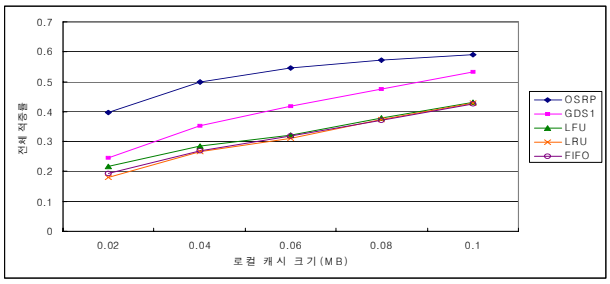
4.2 성능 평가 기준

캐시 교체 정책의 성능을 평가하기 위해 사용한 기준은 적중률과 바이트 적중률이 있다. 적중률은 전체 요청 횟수 중에서 로컬 또는 이웃 캐시로부터 요청에 대한 응답이 이루어지는 횟수에 대한 비율을 나타낸다. 바이트 적중률은 모든 요청한 객체의 크기에서 로컬 또는 이웃 캐시로부터 응답이 이루어지는 객체의 크기가 차지하는 비율을 의미한다. 본 논문에서는 제안하는 교체 정책과 기존의 교체 정책들의 적중률과 바이트 적중률을 비교하여 실험하였다.

4.3 실험 결과 및 분석

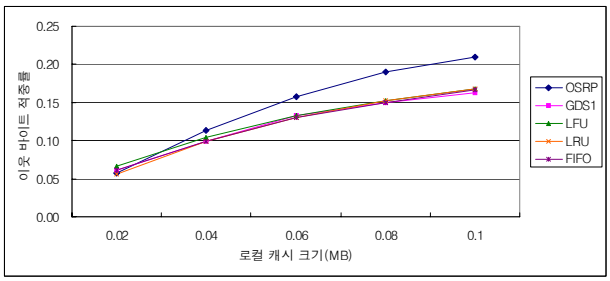


(그림 1) 캐시 크기에 따른 이웃 적중률

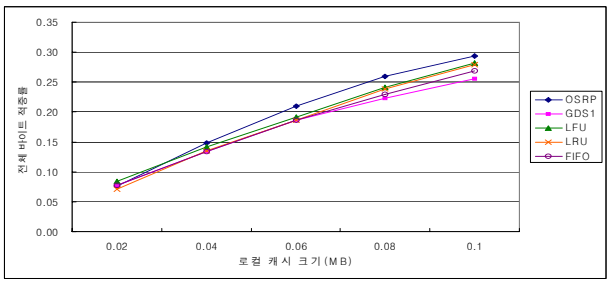


(그림 2) 캐시 크기에 따른 전체 적중률

그림 1과 그림 2는 로컬 캐시 크기를 변경해보면서 이웃 피어에 대한 적중률과 전체 적중률을 나타내었다. 두 그래프를 보면 제안한 OSRP가 다른 교체 정책에 비해 뚜렷하게 우수한 성능을 보이고 있다. 전체 적중률에서 다른 교체 정책들과 비교하면 최고 60%에서 최저 10%정도의 성능 향상을 보였고 제한된 저장 공간을 갖는 모바일 디바이스에 유용하도록 제안되었기 때문에 로컬 캐시 크기가 작을수록 적중률에 유리하는 것을 알 수 있다.



(그림 3) 캐시 크기에 따른 이웃 바이트 적중률



(그림 4) 캐시 크기에 따른 전체 바이트 적중률

그림 3과 그림 4는 캐시 크기를 변경시켜가면서 이웃 피어에 대한 바이트 적중률과 전체 적중률을 나타내었다. 이웃 피어에 대한 바이트 적중률은 캐시가 커질수록 다른 교체 정책과 큰 차이를 보이며 OSRP의 우수함을 보여준다. 전체 바이트 적중률은 다른 교체 정책보다 조금 높은 성능을 보이고 있는데 그 이유는 로컬 캐시는 임계값보다 작은 객체만 응답하여 로컬 바이트 적중률이 낮기 때문이다.

5. 결론

본 논문에서는 RPGM 모델링을 적용한 모바일 P2P 환경에서 객체 크기를 고려한 교체 정책을 제안하였다. 로컬 캐시에 저장되는 크기가 작은 객체는 많은 수의 객체가 캐시에 저장되기 때문에 적중률을 높였고 크기가 큰 객체는 이웃 피어에 저장되어 바이트 적중률을 증가시켰다. 결과적으로 제안한 OSRP 정책은 다른 교체 정책들과 비교하여 모바일 디바이스에 적합하며 우수한 성능을 나타냈다.

참고문헌

[1] Chi-Yin Chow, Hong Va Leong and Alvin Chan, "Peer-to-Peer Cooperative Caching in Mobile Environments," *Proc. International Conference on Distributed Computing Systems Workshops*, pp.528-533, 2004.

[2] K. Kong and D. Ghosal, "Mitigating server-side congestion in the Internet through pseudoserving," *IEEE/ACM Transactions on Networking*, Vol.7, No.4, pp.530-544, 1999.

[3] A. Stavrou, D. Rubenstein, and S. Sahu, "A lightweight, robust P2P system to handle flash crowds," *IEEE Journal on Selected Areas In Communications*, Vol.22, No.1, pp.6-16, 2004.

[4] S. Iyer, A. Rowstron, and P. Druschel, "Squirrel : A decentralized peer-to-peer Web cache," *Proc. Symposium on Principles of Distributed Computing*, pp.213-222, 2002.

[5] Xiaoyan Hong, Mario Gerla and Guangyu Pei, Ching-Chuan Chiang, "A Group Mobility Model for Ad Hoc Wireless Networks," Proc. *ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pp.53-60, 1999.

[6] Kai Cheng and Yahiko Kanbayashi, "LRU-SP : A Size-Adjusted and Popularity-Aware LRU Replacement Algorithm for Web Caching," *Proc. of the 24th IEEE Computer Society International Computer Software and Applications Conference*, pp.48-53, 2000.

[7] Shudong Jin and Azer Bestavros, "Popularity-Aware GreedyDual-Size Web Proxy Caching Algorithm," Proc. *of IEEE International Conference on Distributed Computer Systems*, pp.254-261, 2000.

[8] http://ita.ee.lbl.gov.