동적 인기도 콘텐츠를 활용한 이동성 인식 엣지 캐싱 알고리즘

이태윤 [1], 이수경 [1]
[1] 연세대학교 컴퓨터과학과
tylee814@yonsei.ac.kr, sklee@yonsei.ac.kr

Mobility-Aware Edge Caching Algorithm with Dynamic Content Popularity

Tae-Yoon Lee[1], SuKyoung Lee[1]
[1]Dept. of Computer Science, Yonsei University

요 약

이동성 기반의 기존 엣지 캐싱 연구에서는 인기도가 짧은 시간 급격하게 변화하는 SNM(Shot Noise Model) 콘텐츠를 반영하지 않았다. 동적 인기도 특성을 다루지 않는 경우, 잦은 캐시 미스가 발생하므로 SNM 콘텐츠를 고려하는 것은 중요하다. 이에 본 논문은 이동성을 고려한 기존 연구에 SNM 콘텐츠를 함께 고려하고, 시뮬레이션을 통해 기존 연구 대비 제안 알고리즘의 향상된 캐시 적중률을 확인한다.

Key Words: Dynamic popularity, Shot Noise Model(SNM), Mobility, Edge Caching

1. 서론

최근 스마트 디바이스 및 스트리밍 서비스의 수요 증가에 따른 네트워크 트래픽을 효과적으로 관리하기 위한 방법으로 Mobile Edge Computing(MEC) 기술이 주목받고 있다. MEC 는 네트워크 엣지에 캐시를 설치하여 사용자에게 보다 가까운 곳에서 서비스를 제공하므로 낮은 지연 시간을 제공하고, 네트워크 부하를 감소시킬 수 있다[1, 4].

이러한 MEC 환경에서 사용자가 이동하는 경우, 요청 콘텐츠가 캐싱되어 있지 않은 엣지 서버로부터 제공받아야 하는 상황이 발생해 서비스 응답시간이 증가한다. 따라서 지연시간 감소를 위한 이동성 기반의 캐싱 연구가 활발히 진행되고 있다[1, 2]. 하지만 기존의 이동성 기반 연구는 콘텐츠를 Independent Reference Model (IRM)만으로 고려하여 스포츠 뉴스와 같이 인기도(popularity)가 짧은 시간 급격하게 변화하는 Shot Noise Model(SNM)콘텐츠를 반영하지 않았다. 동적 인기도 특성을 다루지 않는 캐싱 알고리즘은 잦은 캐시 미스(cache miss)가 발생하므로 SNM 콘텐츠를 고려하는 것은 중요하다. 그러나 동적 인기도를 다룬 연구[3, 4]에서는 이동성을 고려하지 않아 캐시 성능이 낮아진다.

본 연구에서는 이동성을 고려한 기존 연구에 SNM 콘텐츠를 함께 고려함으로써 변화하는 인기도에 대응하는 알고리즘을 제안한다. 제안 알고리즘은 이동성을 기반으로 체류시간을 예측하고, 체류 시간이 긴 사용자들의 요청을 토대로 캐싱을 결정한다. 시뮬레이션에서는 기존 연구와 비교를 통해 캐시 적중률(hit ratio)의 향상된 성능을 증명한다.

2. 시스템 모델

본 연구에서는 여러 엣지 서버 e로 구성된 MEC 환경을 고려한다. 사용자는 거리가 가장 가까운 하나의 엣지 서버와 연결되어 있다고 가정하고, 연결된 엣지 서버에 IRM 또는 SNM 콘텐츠 f를 요청한다. 그리고 이동한 엣지 서버에 시간 T를 전달한다. 이동성은 Markov Renewal Process(MRP)로 모델링하고, 평균 체류시간은 uniform 분포의 확률밀도함수로 가정한다. 엣지 서버는 사용자가 전달한 이동 발생 시간 T와 콘텐츠의 인기도 p_f를 저장한다. 엣지 서버의 IRM/SNM 캐시는 각각 C_I와 C_S로 나타낸다. 요청은 IRM 또는 SNM 콘텐츠로 구분되고, 인기 분포는 각각 Zipf 분포와 Pareto 분포를 따른다고 가정한다[3, 4].

$$Zipf(f) = f^{-\delta} / \sum_{j=1}^{f} j^{-\delta}$$
$$Pareto(V) = \beta V_{min}^{\beta} \; v^{-\beta} \quad (1)$$

여기서 Zipf 분포의 δ는 분포 특성을 나타내며, Pareto 분포는 분포 특성을 나타내는 β와 요청 횟수 v, 최소 요청 횟수인 V_{min}로 표현된다.

3. 캐싱 알고리즘

3.1 체류시간 예측

엣지 서버는 MRP 를 활용하여 체류시간을 예측한다[1]. MRP 로 모델링한 이동성$\{(e_i, T_i): i \geq 0\}$은 i번째로 이동한 엣지 서버 e_i와 이동이 발생한 시간 $t - 1 \leq T_i \leq t$로 정의된다. t는 엣지 서버에서 예측이 수행되는 현재 시간을 의미한다.

체류시간은 이동 발생 시간 T와 이동 확률 Pr을 토대로 계산된다. $T_i \geq t$인 경우, 엣지 서버 e_i로 이동할 확률은 $\Pr(e_i) = 1$ 이다. i 번째 이동발생 시간이 $T_i < t$ 인 경우, 엣지 서버 e_j에서 e_i로 이동할 확률이 이면 엣지 서버 e_i 로 이동할 확률은 $\Pr(e_i) = \prod_{k=0}^{i-1} Pr_{k,k+1}$이다. 따라서, 엣지 서버 e_i에 머무르는 체류시간은 식(2)와 같이 계산할 수 있다[1, 2].

$$r_i = [t - T_i]\Pr(e_i) \quad (2)$$

이때, [a]에서 $a < 0$이면 0 이고, $a \geq 0$이면 a 다. 예측된 체류시간 r_i 는 콘텐츠 캐싱에 이용된다.

3.2 IRM 콘텐츠 캐싱

엣지 서버는 체류시간을 예측한 다음으로 IRM 콘텐츠 캐싱을 수행한다. 이동성 유사도를 계산하기 위해 예측한 체류시간을 활용한다. r_{th}미리 정의된 체류시간 임계 값일 때, L_u는 과거 t_o시간 동안 $r_i \geq r_{th}$ 를 만족하는 사용자 u의 엣지 서버 집합으로 정의한다. L_u를 토대로 Jaccard similarity 를 활용한 사용자 u, v의 이동성 유사도는 식(3)과 같다[5].

$$sim_{u,v} = \frac{|L_u \cap L_v|}{\min\{|L_u|, |L_v|\}} \quad (3)$$

식(3)의 $L_u \cap L_v$는 사용자 u, v가 동시에 방문한 엣지 서버 집합을 의미한다. 엣지 서버는 계산한 유사도를 이용하여 사용자를 체류시간이 유사한 클러스터로 나누기 위해 K-Medoids 클러스터링을 활용한다. K-Medoids 클러스터링은 이동성 유사도 $sim_{u,v}$를 입력으로 K개의 클러스터를 결정한다[5].

그 다음, 클러스터에서 인기도가 높은 콘텐츠를 예측하기 위해 MLP(Multi-layer Perceptron)를 활용한다[3]. 과거 t_o동안 콘텐츠 f의 인기도가 $\left[p_{k,f}^{t-t_o}, \ldots, p_{k,f}^{t-1}\right]$일 때, MLP 는 엣지 서버에 저장된 인기도를 토대로 시간 t의 인기도 $\hat{p}_{k,f}$를 출력한다. 각 클러스터에서 예측된 콘텐츠 f의 인기도는 식(4)와 같이 병합된다.

$$\hat{p}_f = \sum_{k=1}^{K} \alpha_k \cdot \hat{p}_{k,f} \;, \quad \alpha_k \in [0, 1] \quad (4)$$

$\sum_{k=1}^{K} \alpha_k = 1$인 식(4)의 α_k는 체류시간이 긴 사용자 클러스터에서 요청할 콘텐츠에 큰 가중치를 부여함으로써 캐시 우선순위를 높인다. 예측한 인기도 리스트 $\hat{p} = [\hat{p}_1, \ldots, \hat{p}_f]$는 내림차순 정렬하여 IRM 캐시 사이즈 C_I가 full 상태일 때까지 인기도가 높은 콘텐츠부터 순서대로 캐싱한다.

3.3 SNM 콘텐츠 캐싱

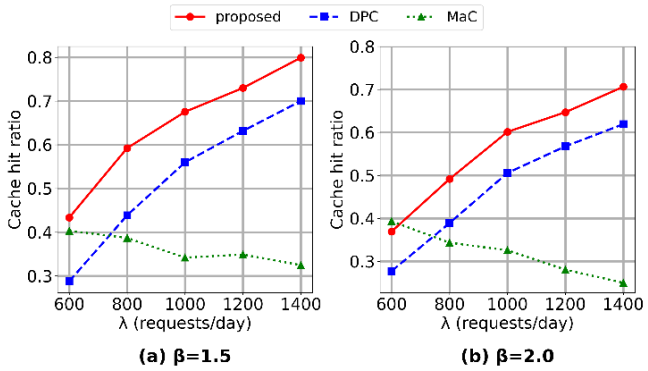
SNM 콘텐츠 캐싱은 기존 LRU 기법을 활용하고, 미리 정의된 체류시간 임계 값 r_{th}을 도입한다. SNM 콘텐츠 요청이 발생한 경우, 엣지 서버는 SNM 캐시 C_S에 요청 콘텐츠가 존재하는 지 확인한다. 캐싱되어 있는 경우 사용자에게 콘텐츠를 제공하고, 그렇지 않은 경우 original 서버로부터 콘텐츠를 제공한다. 만약 C_S가 full 상태일 경우, 가장 오래전 요청된 콘텐츠를 제거하고, 엣지 서버에서 체류시간이 r_{th}이상인 사용자가 요청한 콘텐츠를 캐싱한다. 체류시간은 최근 예측된 정보를 사용한다.

4. 시뮬레이션 결과

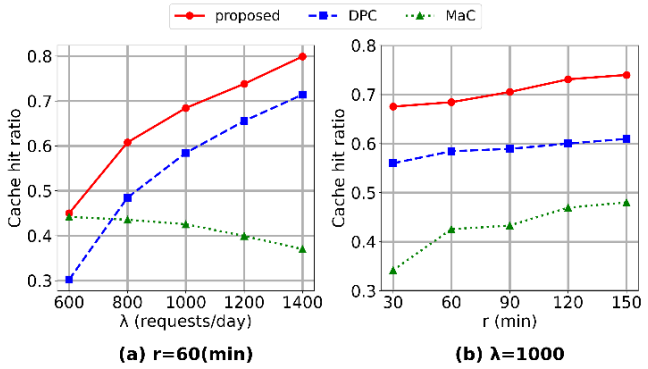
제안 알고리즘의 성능을 평가하기 위해 Python 기반의 시뮬레이터를 구현하고, 서비스 범위 내 100 명의 사용자가 4 개의 엣지 서버에 골고루 분포된 상황을 가정하였다. 이동성은 체류시간 γ을 평균 [30, 150] 분으로 가정하여 Random Waypoint Model 을 토대로 발생하였다[1, 2]. IRM 콘텐츠 요청은 MovieLens 1M[1]에서 하루 평균 1000 회 요청이 발생하도록 데이터 셋을 추출하였고, IRM 인기도 예측을 위한 MLP 의 파라미터는 Grid Search 를 활용해 결정하였다. SNM 콘텐츠 요청은 Pareto 분포의 β와 V_{min}을 각각 1.5 와 3 으로 설정하고, 하루[600, 1400]회 요청(λ)을 발생하였다[3, 4]. 콘텐츠와 캐시 크기는 각각 1MB, 600MB 로 고정하였다. 제안 알고리즘의 성능은 IRM/SNM 콘텐츠를 고려한 동적 인기도 기반 캐싱(DPC; Dynamic Popularity based Caching)[3, 4]과 이동성을 고려한 Mobility-aware Caching(MaC) [1, 2]의 캐시 적중률을 비교하였다.

그림 1(a)는 $\beta = 1.5$, $K = 2$, $r = 30$(min), r_{th}=30(min)인 경우, SNM 콘텐츠 요청 횟수에 따른 캐시 적중률을 나타낸다. 제안 알고리즘은 SNM 요청이 증가할수록 높은 성능을 보이며, DPC 와 MaC 보다 평균 27%, 85% 높은 적중률을 보였다. 그림 1(b)는 그림 1(a)의 시뮬레이션 환경에서 β를 2.0 으로 변경했을 때, SNM 요청 횟수에 따른 캐시 적중률이다. β가 2.0 인 경우, β가 1.5 일 때보다 SNM 의 인기도가 낮아 적중률이 전체적으로 감소한다. 하지만 제안 알고리즘은 DPC 와 MaC 보다 각각 평균 17%, 66%씩 높은 캐시 성능을 보였다.

그림 2(a)에서는 그림 1(a)의 시뮬레이션 환경에서 체류시간을 60(min)으로 변경하였고, 캐시 적중률은 DPC 와 MaC 보다 평균 22%, 60% 개선되었다. 그림 2(b)는 λ가 1000 일 때, 평균 체류시간에 따른 캐시 적중률이다. 체류 시간이 짧은 r=30(min)일 경우에 DPC



(그림 1) SNM 요청 λ 에 따른 캐시 적중률



(그림 2) 체류 시간 r 에 따른 캐시 적중률

와 MaC 보다 최대 23%, 86% 향상되었다. 그림 2 를 토대로 제안 알고리즘은 기존 알고리즘에 비해 평균 체류시간이 짧을수록 캐시 성능이 향상함을 확인하였다.

5. 결 론

본 연구에서는 캐시 적중률을 향상시키기 위해 동적 인기도 콘텐츠를 활용한 이동성 기반의 캐싱을 제안하였다. 시뮬레이션을 통해 제안 알고리즘이 기존 알고리즘에 비해 캐시 적중률이 향상된 것을 증명하였고, 동적 인기도와 체류시간에 따른 성능을 확인하였다. 이 짧을수록 캐시 성능이 향상함을 확인하였다.

Acknowledgement

이 연구논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구결과임(No. 2022R1A2B5B01001683).

참고문헌

[1] M. K. Somesula et al., “Contact Duration-aware Cooperative Cache Placement using Genetic Algorithm for Mobile Edge Networks,” in Comput. Netw., vol. 193, article 108062, July 2021.

[2] Y. Ye et al., “Mobility-Aware Content Preference Learning in Decentralized Caching Networks,” in IEEE Trans. Cogn. Commun. Netw., vol. 6, no. 1, pp. 62-73, March 2020.

[3] K. Qi et al., “Learning a Hybrid Proactive and Reactive Caching Policy in Wireless Edge Under Dynamic Popularity,” in IEEE Access, vol. 7, pp. 120788-120801, Aug. 2019.

[4] Y. Hao et al., “Human-Like Hybrid Caching in Software-Defined Edge Cloud,” in IEEE Internet Things J., vol. 7, no. 7, pp. 5806-5815, July 2020.

[5] T. Li et al., “Lifecycle-Aware Online Video Caching,” in IEEE Trans. Mob. Comput., vol. 20, no. 8, pp. 2624-2636, Aug. 2021.