Video Quality and Popularity-aware Video Caching in Content Delivery Networks

Yijun Sun, Zehua Guo*, Songshi Dou, Yuanqing Xia
Beijing Institute of Technology

***Abstract*—Content Delivery Network (CDN) is a popular service to accelerate object transmission by dynamically caching popular objects at cache points near users. Existing video caching schemes for CDN do not consider some important components of Quality of Experience (QoE). In this paper, we jointly consider video quality and popularity to design a new QoE metric called Video Hit Experience (VHE) and propose an efficient video caching algorithm named Hit ExpeRience-based videO caching (HERO) to improve VHE. Preliminary results show that HERO outperforms existing solutions.**

I. Introduction

Content Delivery Network (CDN) accelerates objects transmission by dynamically caching popular objects (e.g., videos, audios) at cache points near users. Thus, many users can quickly fetch popular objects, and precious backbone network bandwidth is saved. Video is a popular application for CDN. However, a limited number of cache points are deployed with limited storage space and cannot cache all videos. Therefore, efficiently caching videos is a key concern for CDN providers.

Many existing caching solutions (e.g., Least Recently Used (LRU)-K [1], TinyLFU [2], and Greedy Dual Size Frequency (GDSF) [3]) focus on improving caching performance based on hit ratio metrics (e.g., object hit ratio [4] and byte hit ratio). However, the essence of CDN is to improve the Quality of Experience (QoE) for users. Simply improving hit ratio cannot fully depict QoE. Comprehensively modeling QoE is a complicated task since QoE is related to many factors (e.g., video access delay [5] and video quality [6]).

In this paper, we introduce a new QoE metric called Video Hit Experience (VHE) that jointly considers the impact of video hit ratio and video quality on QoE. Based on the new metric, we propose Hit ExpeRience-based videO caching (HERO), which consists of a proactive module and a reactive module, to improve QoE. The proactive module periodically updates caching videos based on video quality and video popularity collected in a period of time. The reactive module uses eviction and admission policy to dynamically replace out-of-date videos with popular videos according to incoming user requests. Preliminary results show that HERO outperforms LRU and GDSF by improving VHE up to 14% and up to 36%, respectively.

II. Background and motivation

A. QoE modeling and hit ratio

Many factors play important roles in video QoE modeling and can be generally grouped into three categories [7]: human-related factors (e.g., emotion), network system-related factors (e.g., video accessing delay), and video-related factors (e.g, video quality). Popular caching metrics (e.g., object hit ratio [4] and byte hit ratio) only consider system-related factors and are not good enough to evaluate the QoE of videos.

*This work is supported by National Natural Science Foundation of China under Grant 62002019 and Beijing Institute of Technology Research Fund Program for Young Scholars. Corresponding author is Zehua Guo.

B. Opportunity

Some works on QoE measurement inspire us to consider new factors to model QoE. In [6], the video quality consists of video resolution and bitrate, and different qualities lead to different QoEs. The QoE of video v_i with video quality L_j is described as $q_{i,j} = m * b^n + o$, where m, n, and o are specific constants related to video resolution, and b denotes the bitrate of video. The relationship between QoE and video quality indicates that high quality video deserves higher chance to be cached.

C. Challenges

We face two major challenges in designing a new QoE metric and a caching scheme. To effectively evaluate the impact of different factors on QoE, a new QoE metric needs to combine video quality and hit ratio. For efficiently caching videos, a good solution needs to jointly consider the stochastic video popularity for a period of time and user request variation.

III. Design

In this section, we introduce our QoE metric named VHE and propose an efficient caching scheme to improve VHE.

A. VHE modeling

VHE is modeled by jointly considering video quality and hit ratio. We model the overall QoE by accumulating the QoE of videos under different qualities when these videos are hit by user requests. VHE takes hit ratio into account by dividing the overall QoE to the number of requests. Thus, VHE is modeled as $VHE = \frac{\sum_{h=1}^{H}\sum_{i=1}^{N}\sum_{j=1}^{M} q_{i,j}^{h}}{H}$, where i is the index of video v_i, j is the index of video quality L_j, $q_{i,j}^{h}$ denotes that h-th request is hit with QoE $q_{i,j}$ [6], M denotes the number of video qualities, N denotes the number of videos, and H denotes the total number of requests.

B. Overview of HERO

We propose HERO to improve QoE by caching popular videos with high VHE. Fig.1 shows the structure of HERO, which consists of a proactive module and a reactive module. In the proactive module, we periodically collect trace data from user request history and evaluate the popularity of videos based on the request frequency in the trace data. We can formulate

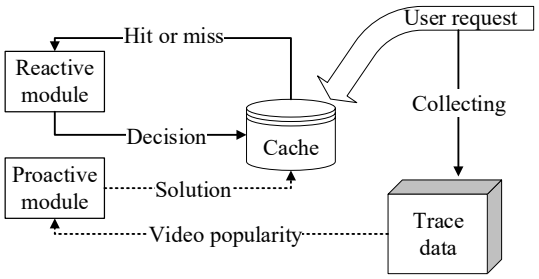

Fig. 1. Structure of HERO.

an optimization problem to periodically select caching videos. However, due to the high computational overhead for solving the problem, it is not easy to refresh cache points through proactive scheme frequently. Thus, we design the reactive module to make quick response to user requests. In the reactive module, we dynamically replace out-of-date videos with popular videos to accommodate the changing video popularity. By combining proactive module and reactive module together, HERO works periodically and dynamically to achieve high QoE.

C. Proactive module

In the proactive module, we formulate an integer programming problem named *Video Caching and Quality Assignment (VCQA)* problem to determine caching videos and their video qualities (i.e., video resolution and video bitrate). In this problem, binary variables are used to control video caching decision and video quality assignments, and cache node space limitation is the constraint. The objective is overall QoE, which is the sum of the multiplied QoE value of video quality and video's popularity divided by the size of video. The popularity can be obtained from the request frequency and recency in the trace data through a decreasing exponential function [8]. By maximizing the objective function, we can cache popular videos with appropriate quality in limited space. Therefore, we can maximize VHE.

Generally speaking, an integer programming problem is a NP-hard problem with high computation complexity when the problem has a large solution space. To efficiently solve the VCQA problem within an acceptable time when the problem scale is large, we design a heuristic algorithm. The general idea of the heuristic algorithm is to relax the original problem into a linear programming problem and then finely round the solution of the linear programming problem to get the final result of the original problem. The heuristic solution helps us periodically select videos and cache them with suitable quality to improve users' experience.

D. Reactive module

In the reactive module, we consider several requests information (e.g., recency, frequency, size, and QoE) together to decide how to process each incoming request. We use *caching profit* to evaluate whether a video is worthy to be cached. The caching profit of a video is defined as the number of requests for the video multiplied with the QoE of the video and divided

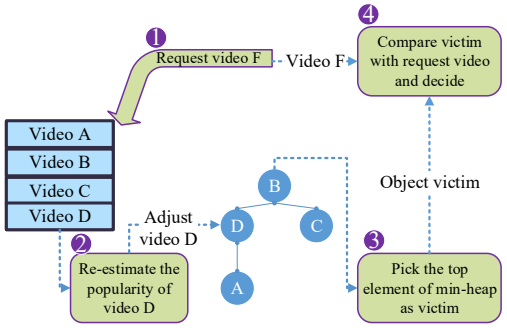

Fig. 2. An example that shows how the reactive module works when the request misses a cached video. Each circle with a number shows the number of the procedure order.

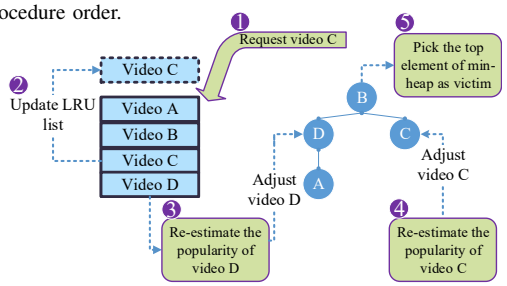

Fig. 3. An example that shows how the reactive module works when the request hits the cached videos.

by the size of the video and then subtracting a penalty item used to prevent the least recent accessed video from becoming cache pollution [9]. The penalty of a video is proportionally to the number of the times that the video becomes the least recent accessed video from the time it is hit until now. This model aims to reactively cache popular videos with small size and high quality to maximize VHE.

Following the above idea, we design an eviction policy and an admission policy to make quick response to user requests. The eviction or admission control takes place when a request comes but misses in the caching node. When the request misses, we calculate the caching profit of the new video and compare it with the cached video with the least caching profit. If the caching profit of the new video is larger than the cached video, we evict the cached video immediately. If the caching node has enough space to cache the new video, we cache it; otherwise, we do not take any further action. If an incoming request hits, we calculate the penalty item of the requested video and the least recently requested video. Then, we can get the caching profit of these videos and use it when the next time eviction and admission control take place.

We leverage the data structures of list, min-heap, and map in our implementation. We use a LRU list and a min-heap to store recency information and caching profit. In the LRU list, we can easily get the least recently requested video and use the number of times that the video stays at the bottom of LRU list to calculate the penalty. We use caching profit as the key value of min-heap to get the video with least caching profit. By maintaining the min-heap, we can get victims in O(log(n)). We use a map to construct a one-to-one mapping between LRU and min-heap.

We use two examples in Fig.2 and Fig.3 to show how the reactive module works. In these examples, LRU list and the heap have four videos. Video D is at the bottom of the list, and video B is the top of the heap. Fig.2 shows an example that how the reactive module works when the request misses a cached video. In step 1, a request of video F arrives. In step 2, we re-calculate the penalty item of video D and adjust the min-heap based on the new caching profit. In step 3, Video B is still at the top of the heap and we take it as victim video. In step 4, we use the admission and eviction policy to decide whether to replace video B with the request video F. Fig.3 shows an example that how the reactive module works when the request hits the cached videos. In step 1, a request of object C arrives. In step 2, we first update the LRU list by placing object C at the top of the list. In step 3, Because of the changing popularity, we recalculate the caching profits of videos D and C using the new penalty items. In step 4, we adjust video D and video C in the min-heap based on their caching profits. In step 5, video B is still at the top of the heap and we take video B as victim video.

IV. Preliminary Results

A. Simulation setup

In our simulation, we use Poisson distribution to simulate the variation of user requests over a period of time, and the popular distribution of the video conforms to Zipf distribution [4]. λ of Poisson distribution is 1, and α of Zipf distribution is 0.75. Everyday 10% videos become out of fashion with low accessing frequency, and 10% new videos appear in the request sequence. We use three typical video qualities: (128kb, 224p), (256kb,360p), (512kb,360p) [6], and the access possibilities for three qualities are equal. All videos are cached in form of chunks with a length of 10 seconds. Existing study [9] shows Adaptsize performs poorly when the size of objects are similar, and thus we do not compare HERO with Adaptsize.

B. Comparison algorithms

Existing CDN caching algorithms mainly consider three factors: request frequency, request recency, and video size. Our evaluation selects three popular, representative algorithms for comparison:

LRU: When a new request comes, it replaces the least recently used video.

LFU: When a new request comes, it replaces the least frequently used video.

GDSF [3]: GDSF associates $H = \frac{frequency}{size}$ with all cached videos. When a video replacement occurs, the video with the lowest H is replaced, and the lowest H is subtracted from all videos' H. If a video is accessed again, its original value H is restored. In this way, the recently accessed video maintains a large H and is always cached.

Proactive-only: it denotes the result of deploying the proactive module only.

HERO: In the initial stage of each period, HERO caches videos based on the solution of proactive module. For the rest of the time, with the help of reactive module, HERO dynamically adjust caching content according to the changing popularity.

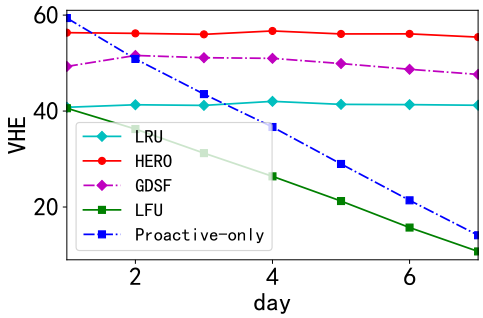

Fig. 4. Simulation result.

C. Result

We use VHE to evaluate the performance of different algorithms. Fig.4 shows the VHE of different algorithms in seven days. In Fig.4, the performance of LRU, GDSF, and HERO do not change over time which is because LRU, GDSF, HERO have no obvious preference for old videos. HERO always outperforms LRU and GDSF. This is because HERO considers the difference of video QoE in three qualities when taking admission and eviction control and cache popular video in high quality. Performance of proactive-only and LFU schemes decreases as linear function with negative slope. Proactive-only scheme suffers from cache pollution because proactive-only scheme caches video at the initial period and does not adjust the caching content based on user request. LFU prefers old videos to new videos because old videos usually have higher frequency, so the cache pollution phenomenon exists too.

V. Conclusion and Future Work

In this paper, we introduce VHE to measure video QoE by jointly consider video quality and hit ration and propose HERO to improve VHE for CDN by periodically and dynamically adjusting caching videos. In the future, we will further improve proactive and reactive modules and conduct comprehensive simulations to evaluate VHE's effectiveness and HERO's performance.

References

[1] E. J. O'neil and et al., "The lru-k page replacement algorithm for database disk buffering," *Acm Sigmod Record*, 1993.

[2] G. Einziger and et al., "Tinylfu: A highly efficient cache admission policy," *ACM ToS*, 2017.

[3] L. Cherkasova, *Improving WWW proxies performance with greedy-dual-size-frequency caching policy*. Hewlett-Packard Laboratories, 1998.

[4] D. S. Berger and et al., "Adaptsize: Orchestrating the hot object memory cache in a content delivery network," in *NSDI*, 2017.

[5] H. Zhao and et al., "Popularity-based and version-aware caching scheme at edge servers for multi-version vod systems," *IEEE TCSVT*, 2020.

[6] L. Toni and et al., "Optimal set of video representations in adaptive streaming," in *Proceedings of the 5th ACM MMSys*, 2014.

[7] K. Bouraqia and et al., "Quality of experience for streaming services: Measurements, challenges and insights," *IEEE Access*, 2020.

[8] D. Lee and et al., "Lrfu: A spectrum of policies that subsumes the least recently used and least frequently used policies." *IEEE Trans. Computers*, vol. 50, pp. 1352–1361, 12 2001.

[9] Z. Akhtar and et al., "Avic: a cache for adaptive bitrate video," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, 2019, pp. 305–317.