CFP: A Cross-layer Recommender System with Fine-grained Preloading for Short Video Streaming at Network Edge

Dezhi Ran†, Yuanxing Zhang†, Ye Yuan†, Kaigui Bian†‡
†*School of Electronics Engineering and Computer Science, Peking University, Beijing, China*
‡*National Engineering Laboratory for Big Data Analysis and Applications, Beijing, China*
Email: {dezhiran, longo, yuanye_pku, bkg}@pku.edu.cn

***Abstract*—Nowadays, short video feed has attracted billions of mobile users all around the world to interact with content effortlessly, yielding an explosive growth of short video commerce. Typically, users watch full-screen short videos of a few seconds one-by-one in a watch-list generated by recommender systems, skipping those they are not interested in. However, the recommender system at the cloud makes a user-interest-specific decision mostly based on the users' behavior data collected within the application itself (e.g., users' view history), without examining the lower-layer network and communication statistics. When the playback choked due to the limited network bandwidth, the user will probably skip the video, leading to a waste of bandwidth and degradation of the user's quality of experience (QoE). Meanwhile, the excessive number of user requests to video contents raises a heavy computational load and communication cost for the recommender system at the cloud to determine which videos to be recommended and delivered to each user in a real-time manner. The advance of edge computing provides a promising avenue of deploying edge nodes with caches (e.g., household devices) beyond cloud and edge servers, such that the recommender system in the cloud can place popular video contents closer to client users, and meanwhile the contents are delivered to client users with good network condition. In this paper, we propose CFP, a cross-layer recommender system for short video streaming with fine-grained preloading technique at the network edge. CFP jointly optimizes the recommendation effect of the video application and the content preloading efficiency under various network conditions at the network edge. CFP takes a two-stage approach: the cloud server first seeks to perform edge-wise instead of user-interest-specific recommendation with neural collaborative filtering recommender, preloading a list of candidate videos to edge nodes, and each edge node, deploying the GRU with attention, then delivers the proper video contents to the client user device according to the user's preference. Trace-driven emulations demonstrate the efficiency of the proposed CFP scheme.**

***Keywords*-Short video, preloading, edge cache, recommendation, quality-of-experience, quality-of-service**

I. Introduction

Video-on-demand (VoD) service has become one of the most popular ways of entertainment over the Internet. According to [1], by 2022, 82% of all mobile data will be consumed by videos, among which short video mobile applications (e.g., TikTok) will take up a large proportion. Meanwhile, to provide high quality of service (QoS) for users, a significant issue for video service providers (VSPs) is to maintain the availability of services while reducing the cost on both servers [2] and bandwidth consumption [3]. The main challenge arises at the peak hours when massive requests towards the hot videos and newly-released episodes are received by the servers in a short period. Traditional video content delivery strategies suffer from breakdowns or crashes when the sudden burst of traffic imposes unbearable overload on the cloud server [4].

The advance of cache strategies in edge computing systems [5] and the widespread of internet data centers (IDCs) [6] enable VSPs to disseminate the video delivery task to the client-side overlays, where clients are capable of requesting specific resources from the edge caches or edge nodes instead of from the data centers or cloud servers. Within this kind of edge network, VSPs may place the video replicas to edge nodes that have caches near the edge of the network to reduce response times and avoid unnecessary data movement between clients and the cloud [7], [8].

A recent trend for video applications is short video feed where short videos are displayed one by one in full screen, which is more popular for attention capturing compared to the traditional video feed or news feed—to capture the user attention right away and consolidate the message within a few seconds in the video [9]. The short videos displayed to users are selected upon personalized recommendations by the cloud server [10]. Short video applications are built on the techniques of streaming long videos, which can integrate the state-of-the-art transmission techniques [11], video processing techniques [12] and panoramic techniques [13]. However, there exists a problem called the cascade of short video rebuffering: when the bandwidth is limited and a user quickly skips the first short video in a list, the second video may not have been loaded yet, and hence the user may suffer from rebuffering, which may lead the user to further skipping the next video. Figure 1 shows a typical scenario of the short video feed application. To take full advantage of the caches at edge nodes, video applications could preload some video resources with appropriate behavior pattern recognition [14], [15] to reduce the start-up delay of the playbacks, improve the potential video quality, and direct the requests to the edge nodes. The short video replicas, if they are identified to match the user preferences, will be

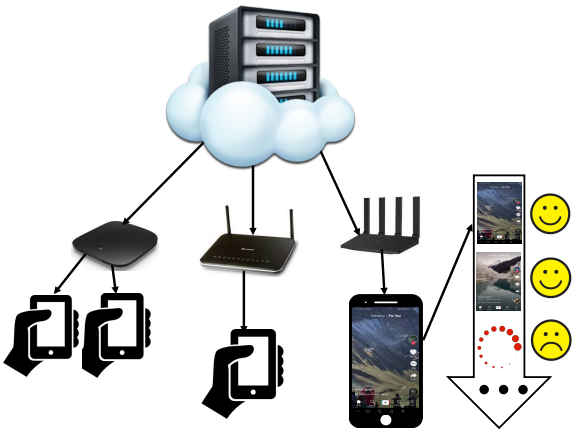


Figure 1: The scenario of short video streaming. The user downloads a pre-determined watch-list of short videos directly from the cloud server. The user views the videos one-by-one and can skip the videos she/he is not interested in. When the bandwidth is limited, or the user skips the first short videos quickly, the second video may not have been loaded yet, and hence the user may suffer from rebuffering, which may lead the user to further skipping the second video, and so on.

preloaded when the application launches and then ready to play in order. There are three main challenges to make the precaching scheme efficient.

- **Preloading cache $\neq$ Regular cache**. *Preloading cache* focuses on pushing video resources that will match user preferences in the future from the cloud. However, regular caches at edge nodes only passively receive and store video contents that have been previously viewed by users.
- **Limited cache storage of edge nodes**. As the edge node is usually a household device such as OTT (over the top) device, only a small part of its storage can be used as a preloading cache. In other words, the edge node can only prepare a limited number of video resources simultaneously.
- **Excessive preloading**. It is not appropriate nor feasible to preload all videos to the edge node in the overlay, which may cause severe network congestion and massive overload on both the edge node and the client. The edge node should determine which video resources to preload, given the trade-off of the resource availability in caches and the limited cache storage.

To address the above challenges, in this paper, we propose a novel fine-grained preloading technique for short video streaming upon the off-the-shelf edge node network, named CFP (cross-layer recommender with fine-grained preloading for short video streaming at the network edge). CFP employs a two-stage recommender system to determine the video contents to deliver (1) from cloud to preloading caches at the edge node, and (2) from the edge node to client devices. Before the peak hours, for each edge node, the cloud server of CFP prepares a set of video resources, called *preloading candidate set* (PC-set), which will be pushed to edge nodes and will only be available during a specific period of the peak hours. These videos in PC-set are expected to be popular among the group of users within the edge nodes' overlay.

- The first stage of CFP is called the push stage, where each edge node collects the historical data of user behaviors within the overlay and attempts to push several video resources from the PC-set prepared by the cloud to fulfill the preloading requirement of as many clients as possible to the overlay. The push stage embeds the users' behaviors into latent factors and applies the multi-layer perceptron (MLP) to estimate the click-through-rate (CTR) of each video resource in the PC-set.
- The second stage of CFP is called the pull stage, where each edge node recommends the most appropriate videos from the preloading cache at the edge node to the clients. We utilize a gated recurrent unit (GRU) network with attention over the embedding of short videos and feedbacks from specific users to capture their personal preferences and make accurate recommendations to the user.

We construct a dataset by the real-world records in the Avazu CTR prediction dataset and examine the performance of the proposed CFP framework with several state-of-the-art schemes on both accuracy and QoS. Results indicate that CFP provides accurate short videos to reach high CTR and can significantly improve QoS of the short video applications.

The rest of paper is organized as follows: We introduce the background knowledge and related work in Section II. We formulate the short video preloading problem in Section III Section IV presents the design of the proposed system. We provide the implementation scheme in Section V. Then, we conduct experiments and analyze the results in Section VI. Finally, we conclude our work in Section VII.

II. Related Work

Edge-assisted video streaming. The edge network helps reduce the distance between clients and data sources, which is a promising solution in the video streaming application. The core task of designing an efficient edge-assisted video streaming system is to find a replacement scheme of the cached content [16]. Apart from the legacy method such as least recently used (LRU), the prediction on the popularity of the content provides an evidence for increasing the hit ratio of the caches [17] or collaboratively reducing the bandwidth cost of the entire edge network [3], [18]. It is also feasible to control the content cached in the edge nodes by the central

server to optimize multiple QoS objectives based on the real-time collected statistics [2].

Content and resources preloading. Preloading content and resources contribute to accelerating the launching of applications, e.g., preloading resources to the memory of devices based on the identification of usage patterns can significantly reduce the initializing time [19], [20]. Regarding video applications, especially the short video applications, the playback of preloaded content enables the virtually unperceived interruptions or delays and thereby improves the QoE [21]. Attempts have been made on preloading videos under traditional client-server architecture [22]. To the best of our knowledge, CFP is the first work to execute preloading at edge nodes for improving QoE of video streaming services within the edge network.

Sequential recommendation in edge network. The accuracy of recommendation determines the performance of the efficiency of preloading [23], [24]. In many E-commerce applications and VoD applications, sequential modeling of user preferences has shown prominent performance with merely implicit feedbacks. For example, GRU4REC [25] utilizes GRU to capture the sequential user behavior for the recommendation, and NARM [26] introduced the attention mechanism to learn the user's preference better. In terms of the edge network, recommendation strategies have been deployed on edge caches to update the cached content for high hit ratio [27]. CFP combines both user-level recommendation and edge-level recommendation, and presents a two-stage recommendation scheme for the efficient preloading.

III. Problem Formulation

In this section, we formulate the preloading problem and introduce the QoE objectives.

A. List-wise Recommendation

In the scenario of short video recommendation, the cloud server usually yields a list of videos for users by recommender systems to avoid the rebuffering caused by consecutively skipping short videos, and to relieve the overload on server engendered by frequent requests [28]. Let $\mathbf{v} = (v_1, v_2, ..., v_N)$ be the watch list of videos, where each short video v_i is of size s_{v_i} and N is the length of the watch list. Denote I_i the indicator function of whether the user is fond of v_i:

$$I_i = \begin{cases} 1 & \text{user likes } v_i, \\ 0 & \text{user dislikes } v_i. \end{cases} \tag{1}$$

The user behavior to the short videos would be influenced by the bitrate, content of the video as well as the rebuffering time. Intuitively, the recommender systems should deliver lists of videos users are interested in.

B. Preloading of Short Videos

The accuracy of recommendation is not the only factor affecting the QoE of users and the QoS of the system. Distinct from traditional long-form video streaming, where a long video is divided into chunks and can be loaded one chunk after another [29], short videos, each lasts within tens of seconds, require fully downloaded before displayed; otherwise the user has to wait for the downloading process. Video applications could preload several short videos at appropriate time if the application is running foreground or background, leading to a minuscule start-up delay when the user begins to watch the short video feed. To take full advantage of the edge network where clients locate in overlays served by specific edge nodes, we exploit a certain part of the storage of edge nodes to be the *preloading cache* aside from the regular cache. Preloading cache only caches the short videos, which may be videos ads or videos to be released shortly (allowed for playback only after a specific time), to be preloaded to clients. Considering the limited storage of the edge nodes, let M be the maximal number of videos edge nodes can cache, and let $\mathbb{C} = \{c_1, c_2, ..., c_M\}$ represent the videos cached on the preloading cache. We call the set of the candidate of videos to be preloaded to clients as *preloading candidate set* (PC-set), denoted by $\mathbf{v}^{'}$. The edge node can select m ($m < M$) videos from $\mathbf{v}^{'}$ and preload them to clients. We assume that the available bandwidth varies with time and its value at timestamp t is $b(t)$. Given the short video list $\mathbf{v}$, for v_j started at timestamp t_j, the downloading time, denoted as D_j, can be computed by:

$$D_j = \begin{cases} 0 & v_i \text{ is preloaded}, \\ \Delta t_j + \delta_j & v_i \text{ is not preloaded}, \end{cases} \tag{2}$$

where Δt_j satisfies $v_j = \int_{t_j}^{t_j + \Delta t_j} b(t)dt$, and δ_j represents CODEC time for the video.

C. QoE Objective for Users

QoE is a measure of user experiences towards the applications, which is an important part of the optimization objective of CFP.

Accuracy of recommendation at server-side. To improve the efficiency of preloading, CFP is expected to construct the PC-set that can satisfy the preference of as many users in the overlay as possible. The server-side recommender system has been a sophisticated technique, and we are not expected to degrade the performance of the accuracy due to the existence of preloading cache. Let $\bar{\mathbf{v}}$ denote the videos preloaded to a specific client. We use Normalized Discounted Cumulative Gain (NDCG), which is common in evaluating recommender systems, to evaluate the accuracy of the preloaded videos, i.e.,

$$\mathbf{R} = \max_{\bar{\mathbf{v}}} \mathbb{E}_{\bar{\mathbf{v}} \sim \mathbf{v}'}[\text{NDCG}(\bar{\mathbf{v}})], \tag{3}$$

where $\mathbf{R}$ represents the accuracy of the PC-set, and the relevance value in calculating NDCG follows the results from the server-side recommender system.

Rebuffering time at user-side. The time user suffers from delay for each video j calculated by Eqn. (2) is D_j. We seek to minimize the average rebuffering time D :

$$\mathbf{D} = \min_{\mathbf{v}=(v_1,v_2,..,v_N)} \frac{\sum_{j=1}^{N} D_j}{N}. \tag{4}$$

Then, the QoE objective is formulated by:

Problem 1.

$$\arg\max_{\mathbf{v},\mathbf{v}'} \{\gamma_1 \mathbf{R} - \gamma_2 \mathbf{D}\}, \tag{5}$$

where (γ_1, γ_2),set to $(1.2, 0.1)$ in this paper, are non-negative parameters.

D. QoS Objective at Network Edge

QoS measures resource consumption and service quality, which is also an important part of the optimization of CFP. The short video streaming and preloading both consume the bandwidth resource. On the one hand, preloading can alleviate bandwidth consumption, especially during peak hours and reduce network congestion. On the other hand, excessive preloading is a waste of bandwidth and should be avoided.

Cache hit ratio. CFP is designed to maximize the hit ratio of content in the preloading cache, defined as:

$$\mathbf{H} = \mathbb{E}_{\mathbb{C}}[\frac{\sum_{v \in \mathbf{v}} \iota(v)}{N}], \tag{6}$$

where $\iota(v) \in \{0, 1\}$ indicates whether the user clicks (for video ads) or fully watched (for any short videos) video v. This metric ensures that the videos on the preloading cache own a high probability of conforming to user preference.

System throughput. The system throughput determines the cost of the video platform, which can be divided into two parts: (1) Pushing the video resources to the preloading cache at edge node; (2) The cache-missed video streaming from the server. Therefore, the overall server-side throughput would be calculated by:

$$\hat{\mathbf{B}} = \sum_{i=1}^{M} s_{c_i} + \sum_{\mathbf{v}} \sum_{i=1}^{N} (1 - \mathbb{I}(v \in \mathbb{C})) s_{v_i} \tag{7}$$

where $\mathbb{I}$ is the indicator function.

Combining the above two, the QoS objective of CFP is formulated as:

Problem 2.

$$\arg\max_{\mathbf{v},\mathbb{C}} \gamma_3 \mathbf{H} - \gamma_4 \hat{\mathbf{B}}, \tag{8}$$

where (γ_3, γ_4), set to $(1.8, 2 \times 10^{-5})$ in this paper, are non-negative parameters.

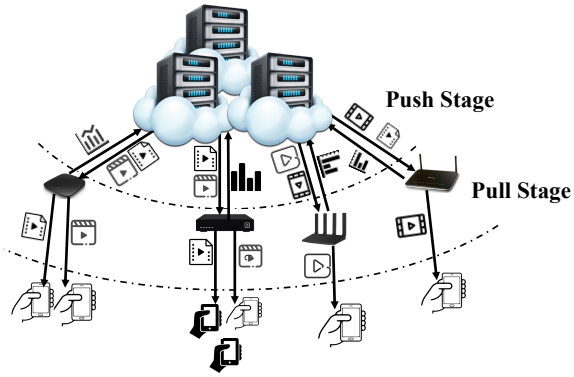


Figure 2: The architecture of two-stage short video recommendation with edge nodes that have preloading caches. The edge node collects and sends the historical playback statistics of the clients in the overlay to the cloud. The cloud server responds with short video lists back to the edge node based on the push-stage algorithm to construct the PC-set. When a client within the overlay launches the application or runs in the background with available resources, the client would preload several videos from PC-set based on the pull-stage algorithm.

IV. Two-stage Recommender Scheme

The objective of the preloading of CFP is to balance QoE and QoS of the edge network defined by Problems 1 and 2 respectively, and achieve the overall-best performance. In this section, we introduce the two-stage design of the cross-layer recommender system in detail. Fig. 2 illustrates the overall architecture of CFP.

A. Push Stage: Edge-aware Cloud Recommendation

To make full use of the preloading cache for improving the efficiency of preloading, the first stage of CFP should determine which videos should be cached on the preloading cache. CFP employs a push stage to select several videos as the PC-set to the edge node. Specifically, the server prepares a list of videos that deserve to be preloaded to clients. These videos can be manually labeled or predicted by the popularity of the generator of the videos. The server collects the statistics on playback and feedbacks reported from the edge node, and meanwhile some advertisers need their advertisement to display to users based on location information, i.e., they require their ads to be delivered to people reside in a specific area. Accordingly, the server selects M videos to be the PC-set. The server needs to balance between accuracy and efficiency of recommendation in the push stage. In this stage, we utilize the multi-layer perceptron (MLP) [30] to analyze the relevance of each video and generate the PC-set. Suppose that the videos are mapped into embedding according to the textual description and video content analysis. Let $x_v \in \mathbb{R}^{\delta}$ denote the embedding of short video v in δ-dimension. Suppose that the

historical playback information retrieved and reported by the edge node is represented by a vector, denoted as e (will be introduced in Sec. IV-B). Then, we decide whether a video is appropriate to be preloaded to clients in the corresponding overlay by:

$$s_v = \text{MLP}([x_v; e]; \theta_\text{C}), \tag{9}$$

where s_v represents the score of video v, "[;]" indicates the concatenation and θ_C represents the parameters of the MLP network. Finally, the videos with the top-M scores will be sent back to the edge node.

To obtain the optimal parameters of the model, we train the model by optimizing the loss:

$$\mathcal{L}_\text{C}(\theta_\text{C}) = \sum_v \zeta_v ||s_v - \mathbb{I}(\zeta_v \geq 1)||^2 + \lambda \sum_v ||x_v||^2, \tag{10}$$

where ζ_v represents the number of positive feedbacks towards the video in the dataset. Note that as most of the videos receive no feedback in the dataset, we make negative sampling by the proportion of 1:1 during training.

In particular, for the overlay with little history information, i.e., e is unreliable or missing, the PC-set will be generated merely based on the predicted popularity, i.e., assuming that the users in the overlay own the global preferences. Here, the popularity of a video may be roughly predicted according to the popularity of the previous videos by the same creator.

B. Pull Stage: Video Recommender at Edge

With the PC-set on the edge node, several videos will be preloaded to the client based on the personal statistics. CFP applies a pull-stage recommender system to analyze the relevance between the videos in the PC-set and the user preferences for consideration of QoS and QoE.

The edge node receives the PC-set from the cloud server during the push stage. Apparently, due to the cache capacity, videos in PC-set may not fulfill the requirements of all clients within the overlay. When a client launches the video application or runs in the background with available resources, the client would register to the edge node and fetch the information of the PC-set. A GRU-based recommender system is embedded in the client to preload several videos from the preloading cache. For the consideration of storage and bandwidth consumption of both client and edge node, it is not appropriate to preload all videos in PC-set to clients. Intuitively, preloading attempts to reach the highest CTR throughout the overlay from the perspective of the edge node.

Without loss of generality, we assume that the watching the videos completely indicates the preferences of the users, while skipping a video indicates that the short video is against user interests. Denote I_i the implicit feedback of whether the user likes v_i, i.e., click or fully watch. The client-side recommender system predicts the implicit user feedback I'_i $(1 \leq i \leq n)$ based on the previous watch list and the corresponding feedback. CFP utilizes the GRU [31] to encode the sequential watching record of a user:

$$g_i = \text{GRU}([x_i; l_{i-1}], g_{i-1}; \theta_\text{FG}), \tag{11}$$

where g_i is the hidden state of the GRU at the i-th position, encoding the subsequence to a single vector with parameter θ_FG.

To capture the latest information from the playback list of a client, we employ the attention mechanism [32], which allows the network to dynamically select and combine different parts of the input sequence, to generate the attentive representation at every position of the sequence:

$$o_i = \sum_{j=1}^{i-1} \alpha_{ij} g_j, \tag{12}$$

where α_{ij} is the normalized weighted factor that determines which part of the activation sequence should be emphasized or ignored in the prediction, depicted by

$$\alpha_{ij} = \text{softmax}(v_a^T \tanh(W_1 g_j + W_2 g_i)), \tag{13}$$

where v_a^T, W_1, W_2 are the parameters of the attention mechanism.

Then, we use another MLP to decode the user behavior for estimating the feedbacks:

$$l'_i = \sigma(\text{MLP}(o_i; \theta_\text{FM})), \tag{14}$$

where $\sigma(\cdot)$ indicates the sigmoid function and θ_FM represents the parameters to be learnt. We take cross entropy loss between the ground-truth feedback and the predicted feedback to update the parameters, i.e., the loss function is set to:

$$\mathcal{L}_\text{F}(\theta_\text{FG}, \theta_\text{FM}, W_1, W_2, v_a) = -\sum_{i=1}^{N} l_i \log l'_i. \tag{15}$$

Beyond those, we take the mean of user preference from all clients served by the edge node to be the summary vector of the overlay, i.e., $e = \mathbb{E}[o_N]$, which will be reported to the central server for requesting the new PC-set.

C. Joint Training for Two stages

To incorporate the push- and pull-stage recommendation, we alternate the training of the two stages by fixing one stage and updating the other. The updating of the parameters of the two stages in an online manner follows the same alternative training mechanism. The training process would provide several sets of parameters of the model on the offline training dataset (e.g., we reserve 10 in this paper), and evaluate the performance on the validation set. The one reaching the best performance on the summation of Eqn. (5) and (8) will be deployed to the system as a solution to both Problems 1 and 2.

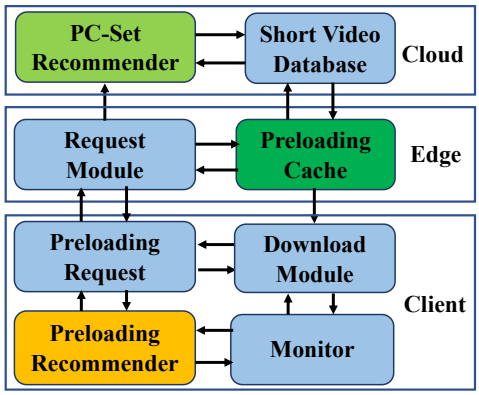


Figure 3: The architecture of system implementation.

V. Implementation

The architecture of the system implementation is illustrated in Figure 3, including the cloud server modules, the edge node modules and client modules.

Cloud server. The cloud server is responsible for handling requests from edge nodes and clients. The *short video database* is an offline module. Some videos in the short video database are collected to be the preloading candidate, from which the *PC-set recommender* follows the design of CFP to generate the PC-set. It exposes Restful API to all edge nodes and processes preloading requests from edge nodes.

Edge node. The edge node modules can be implemented on a variety of edge devices. The *request module* takes charge of video preloading requests from clients, sending the cache information to and receiving the PC-set from the cloud server. The short-videos in the PC-set will be cached in the *preloading cache*, as well as the corresponding embedding. To monitor the local user preferences within the overlay, the edge node should collect information from clients. The request from clients will also be responded by the *request module* to preload the corresponding videos to clients.

Client. The playback statistics are collected by the *monitor* on the client. Based on the statistics, when the application launches or runs in the background with available resources, the *preloading recommender* decides videos to be preloaded in the client. The *preloading request* module would send requests to the edge node for preloading. Besides, the client should periodically report the representation of the user preference to the edge node to construct the summary vector of the overlay.

Emulation platform. The emulation system helps evaluate the performance of the edge network with preloading based on the offline dataset and could monitor the playback details from server, edge, and client perspectives. In this case, we develop an emulation platform by Python based on the architecture introduced in this section. Given the offline playback statistics and network measurements, the system can simulate the whole process of the playback. When the offline emulation is finished, the system can be deployed at the edge network with efficient and accurate video preloading.

VI. Performance Evaluation

In this section, we evaluate the performance of CFP over real-world dataset.

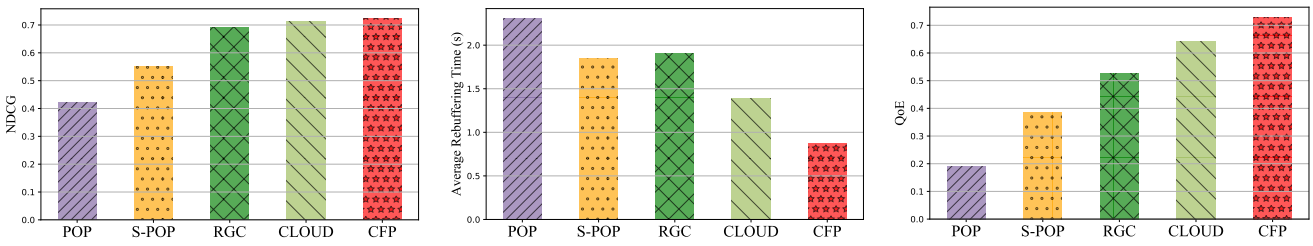
A. Settings

Dataset construction. As there are few datasets that cover server-side, client-side, and edge-side statistics at the same time, we construct the dataset for emulation by combining several public datasets collected in real-world applications.

- **Videos**. We collect one hundred short videos from TikTok[1] and encode these short videos by FFMPEG with X.264 under the average bitrate (ABR) mode, where each video is assigned by 8Mbps (720p). We might assume that all the preload videos are of the same bitrate to guarantee the quality of preloading video delivery.
- **Feedback sequence**. We select 5000 feedback sequences of length 20 from the real-world Avazu CTR prediction dataset [2]. Some labels are attached to the videos, which can be used to learn the video embedding. The users are divided into 250 groups based on their position, and each group is served by an edge node. We pair the IDs of ads in the CTR dataset with the randomly generated short videos, and thus each short video is assigned with a unique ID. "0" in the dataset represents that the user is not interested in the video. "1" in the dataset represents that the user is interested in the video (i.e., click or fully watch). For the evaluation, 30% of feedback sequences are selected as the test set, which are guaranteed not to appear in the training procedure. For each feedback sequence in the test set, the first half of the sequence are exposed to the model as the historical information and the second half in the sequence are used for prediction.Suppose that the requests in the test set are sent serially.
- **Bandwidth traces**. We choose and scale 200 bandwidth traces with various bandwidth patterns from public datasets [33], [34] to simulate different network bandwidth conditions. The average bandwidth ranges from 4Mbps to 20Mbps, and 70% of bandwidth falls between 10Mbps and 16 Mbps. The bandwidth traces are paired to the users with a sequence of feedbacks as the bandwidth between the client and the server. We assume that the bandwidth between clients and the edge node is constant as 20Mbps as they locate in the same overlay.

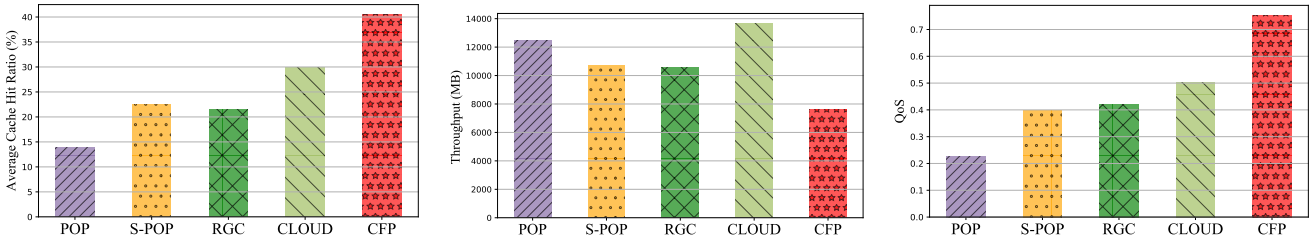
[1]https://www.tiktokapp.cc/

[2]https://www.kaggle.com/c/avazu-ctr-prediction/data



(a) Quality of Recommendation (b) Average Rebuffering Time (c) Quality of Experience

Figure 4: The performance of QoE metrics by different systems.



(a) Cache Hit Ratio (b) Throughput Consumption (c) Quality of Service

Figure 5: The performance of QoS metrics by different systems.

Experiment setup. Each user in the dataset is attached with a preloading duration ranging from zero to five seconds under uniform distribution and will preload at most five short videos. The size of the PC-set, i.e., M, is set to 20 for consideration of the storage usage on the edge node for the preloading cache. To ensure the efficiency of the two-stage models, the dimension of the embedding is set to 96, and thereby the pull-stage model can run in real-time on off-the-shelf mobile devices. A GRU4REC model is deployed on the server to make the regular recommendation other than preloading. Note that we also prepare a regular cache of same size following LRU as the preloading cache to assist the short video streaming in the overlay. We emulate the streaming of the short videos over the users and collect NDCG (w.r.t. the relevance value from server-side GRU4REC), rebuffering time, cache hit ratio, and throughput consumption of server.

Baseline. We compare the performance of CFP with several common or state-of-the-art frameworks from both QoS and QoE aspects.

1) **POP**. This baseline differs from CFP on the push stage. It collects the summary vector of all edge nodes and calculates the mean of them, based on which the server computes the PC-set by Eqn. (9) and send it to all edge nodes.
2) **S-POP**. This baseline differs from CFP on the pull stage. When the edge node receives the PC-set, only the top of them will be preloaded indistinguishably to the clients in the overlay.
3) **RGC**. Recommender with general cache (RGC) replaces the preloading cache by another regular cache of the same size. No preloading strategies are included in this algorithm.
4) **CLOUD**. This baseline excludes the preloading cache and thereby clients have to send their representation of preference directly to the server and receive the videos to be preloaded.

B. Results of QoE Metrics

We repeat the emulation by 20 times and reported the average statistics. Fig. 4 illustrates the QoE metrics achieved by different schemes. We observe that POP and S-POP do not work well on the QoE metrics, where we can infer that it is not sufficient to make recommendations and preloading merely based on popularity. Beyond that, on the NDCG metric as shown in Fig. 4(a), we observe that CFP, RGC and CLOUD perform nearly the same, indicating that the two-stage design does not degrade the accuracy of the recommendation. Meanwhile, CFP and CLOUD reaches the lowest rebuffering time due to the utilization of appropriate preloading, as shown in Fig. 4(b). CFP outperforms the other baselines on the overall QoE objective (as shown in Fig 4(c)), which reveals the significance of CFP in improving user experience.

C. Results of QoS Metrics

Fig. 5 illustrates the results on QoS metrics. Fig. 5(a) depicts the cache hit ratio of different systems, where we find that POP, S-POP and RGC do not work well owing to the lack of prediction on user preference. Note that RGC measures the hit ratio on the regular cache as RGC

does not involve preloading strategies. Fig. 5(b) shows the throughput consumption of the overall edge network. CLOUD consumes most bandwidth as there is no preloading cache to reuse the resources during the preloading phase. CFP takes advantage of the two-stage recommendation and avoids invalid preloading cache and consequently reaches the best performance on QoS (as shown in Fig. 5(c)). Besides, it can be inferred that the regular cache and preloading cache cooperate well in the design of CFP.

D. Advantage of Two-Stage Design

The results demonstrate the effectiveness of two-stage video preloading and recommendation at the edge. Specifically, the alternative training guarantees that the recommendation accuracy would not be degraded due to the two-stage framework (as shown in Fig. 4(a)). The push stage of CFP reduces the number of short video candidates (i.e., PC-set), which significantly improves the reusing of resources, indicated by the increment on cache hit ratio (as shown in Fig. 5(a)) and the decrement on throughput consumption (as shown in Fig. 5(b)). Preloading video resources enable the client to prepare several short videos before the playback begins, and thereby the client would suffer less rebuffering under various network patterns (as shown in Fig. 4(b)). The recommender systems on both the server and edge nodes are designed to balance the accuracy and efficiency, indicating the robustness and practicability of CFP to be deployed in real-world video applications. These results confirm the state-of-the-art performance of designed CFP.

VII. Conclusion

The emerging short video applications incur challenges on the network QoS and the user QoE. In this paper, we propose CFP, a cross-layer recommender system with fine-grained preloading technique at the off-the-shelf video edge network. Specifically, CFP yields an appropriate list of videos based on the statistics of the edge overlays, pushs and caches proper preloading videos from PC-set to the edge nodes. Then clients could request several videos to be preloaded when the application launches or runs in the background with available resources, which helps reduce the server throughput while improving QoE of users. The design of CFP ensures that it can be deployed in real-world applications and updated periodically based on the latest statistics in an online manner. We construct a dataset from real-world network log and user behaviors towards videos. The emulation results illustrate the state-of-the-art performance of CFP.

Acknowledgment

This work is partially supported by National Key Research and Development Program No. 2017YFB0803302, Beijing Academy of Artificial Intelligence (BAAI), and NSFC 61632017.

References

[1] V. Cisco, "Cisco visual networking index: Forecast and trends, 2017–2022 white paper," *Porto Salvo, Lisboa. Disponível em:¡ https://www. cisco. com/c/pt_pt/about/press/news-archive-2018/20181127. html¿, Acesso em*, vol. 17, 2019.

[2] J. Zou, C. Li, C. Zhai, H. Xiong, and E. Steinbach, "Joint pricing and cache placement for video caching: A game theoretic approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 7, pp. 1566–1583, 2019.

[3] Y. Zhang, C. Gao, Y. Guo, K. Bian, X. Jin, Z. Yang, L. Song, J. Cheng, H. Tuo, and X. Li, "Proactive video push for optimizing bandwidth consumption in hybrid cdn-p2p vod systems," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 2555–2563.

[4] T. Yang, J. Jiang, P. Liu, Q. Huang, J. Gong, Y. Zhou, R. Miao, X. Li, and S. Uhlig, "Elastic sketch: Adaptive and fast network-wide measurements," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 561–575.

[5] K. Bian, C. Gao, Y. Tao, Y. Zhang, L. Song, S. Dong, and X. Li, "Learning at the edge: Smart content delivery in real world mobile social networks," *IEEE Network*, vol. 33, no. 4, pp. 208–215, 2019.

[6] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: optimization of distributed internet data centers in a multi-electricity-market environment," in *2010 Proceedings IEEE INFOCOM*. IEEE, 2010, pp. 1–9.

[7] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, "Qoe-driven mobile edge caching placement for adaptive video streaming," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 965–984, 2017.

[8] Y. Zhang, K. Bian, H. Tuo, B. Cui, L. Song, and X. Li, "Geo-edge: Geographical resource allocation on edge caches for video-on-demand streaming," in *2018 4th International Conference on Big Data Computing and Communications (BIGCOM)*. IEEE, 2018, pp. 189–194.

[9] J. Chen, X. Song, L. Nie, X. Wang, H. Zhang, and T.-S. Chua, "Micro tells macro: predicting the popularity of micro-videos via a transductive model," in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 898–907.

[10] Y. Zhang, P. Zhao, Y. Guan, L. Chen, K. Bian, L. Song, B. Cui, and X. Li, "Preference-aware mask for session-based recommendation with bidirectional transformer," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3412–3416.

[11] Y. Guan, Y. Zhang, B. Wang, K. Bian, X. Xiong, and L. Song, "Perm: Neural adaptive video streaming with multi-path transmission," in *IEEE INFOCOM 2020*, 2020.

[12] Y. Zhang, Y. Zhang, Y. Wu, Y. Tao, K. Bian, P. Zhou, L. Song, and H. Tuo, "Improving quality of experience by adaptive video streaming with super-resolution," in *IEEE INFOCOM 2020*, 2020.

[13] Y. Zhang, P. Zhao, K. Bian, Y. Liu, L. Song, and X. Li, "Drl360: 360-degree video streaming with deep reinforcement learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1252–1260.

[14] V. Srinivasan, S. Moghaddam, A. Mukherji, K. K. Rachuri, C. Xu, and E. M. Tapia, "Mobileminer: Mining your frequent patterns on your phone," in *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing*, 2014, pp. 389–400.

[15] F. Zhang, C. Xu, Y. Zhang, K. Ramakrishnan, S. Mukherjee, R. Yates, and T. Nguyen, "Edgebuffer: Caching and prefetching content at the edge in the mobilityfirst future internet architecture," in *2015 IEEE 16th International Symposium on a World of wireless, mobile and multimedia networks (WoWMoM)*. IEEE, 2015, pp. 1–9.

[16] S. Dernbach, N. Taft, J. Kurose, U. Weinsberg, C. Diot, and A. Ashkan, "Cache content-selection policies for streaming video services," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[17] S. Li, J. Xu, M. van der Schaar, and W. Li, "Trend-aware video caching through online learning," *IEEE Transactions on Multimedia*, vol. 18, no. 12, pp. 2503–2516, 2016.

[18] Y. Cui, N. Dai, Z. Lai, M. Li, Z. Li, Y. Hu, K. Ren, and Y. Chen, "Tailcutter: Wisely cutting tail latency in cloud cdns under cost constraints," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1612–1628, 2019.

[19] X. Liu, Y. Ma, X. Wang, Y. Liu, T. Xie, and G. Huang, "Swarovsky: Optimizing resource loading for mobile web browsing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 10, pp. 2941–2954, 2016.

[20] Z. Shen, K. Yang, W. Du, X. Zhao, and J. Zou, "Deepapp: a deep reinforcement learning framework for mobile application usage prediction," in *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, 2019, pp. 153–165.

[21] A. Devlic, P. Lungaro, P. Kamaraju, Z. Segall, and K. Tollmar, "Energy consumption reduction via context-aware mobile video pre-fetching," in *2012 IEEE International Symposium on Multimedia*. IEEE, 2012, pp. 261–265.

[22] D. Karamshuk, N. Sastry, M. Al-Bassam, A. Secker, and J. Chandaria, "Take-away tv: Recharging work commutes with predictive preloading of catch-up tv content," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2091–2101, 2016.

[23] T. Giannakas, P. Sermpezis, and T. Spyropoulos, "Show me the cache: Optimizing cache-friendly recommendations for sequential content access," in *2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2018, pp. 14–22.

[24] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. S. Shen, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 915–929, 2018.

[25] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2018, pp. 843–852.

[26] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 1419–1428.

[27] P. Sermpezis, T. Giannakas, T. Spyropoulos, and L. Vigneri, "Soft cache hits: Improving performance through recommendation and delivery of related content," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1300–1313, 2018.

[28] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang, "Deep reinforcement learning for page-wise recommendations," in *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018, pp. 95–103.

[29] T. Stockhammer, "Dynamic adaptive streaming over http–standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*, 2011, pp. 133–144.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[31] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[32] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[33] B. Meixner, J. W. Kleinrouweler, and P. Cesar, "4g/lte channel quality reference signal trace data set," in *Proceedings of the 9th ACM Multimedia Systems Conference*. ACM, 2018, pp. 387–392.

[34] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks," *IEEE Communications Letters*, vol. 20, no. 11, pp. 2177–2180, 2016.