KEPC-Push: A Knowledge-Enhanced Proactive Content Push Strategy for Edge-Assisted Video Feed Streaming

Ziwen Ye, *Peng Cheng Laboratory and Tsinghua Shenzhen International Graduate School;* Qing Li, *Peng Cheng Laboratory;* Chunyu Qiao, *ByteDance;* Xiaoteng Ma, *Tsinghua Shenzhen International Graduate School;* Yong Jiang, *Peng Cheng Laboratory and Tsinghua Shenzhen International Graduate School;* Qian Ma and Shengbin Meng, *ByteDance;* Zhenhui Yuan, *University of Warwick;* Zili Meng, *HKUST*

https://www.usenix.org/conference/atc24/presentation/ye-ziwen

This paper is included in the Proceedings of the 2024 USENIX Annual Technical Conference.

July 10–12, 2024 • Santa Clara, CA, USA

978-1-939133-41-0

Open access to the Proceedings of the 2024 USENIX Annual Technical Conference is sponsored by

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

KEPC-Push: A Knowledge-Enhanced Proactive Content Push Strategy for Edge-Assisted Video Feed Streaming

Ziwen Ye
Peng Cheng Laboratory, Tsinghua Shenzhen International Graduate School

Qing Li
Peng Cheng Laboratory

Chunyu Qiao
ByteDance

Xiaoteng Ma
Tsinghua Shenzhen International Graduate School

Yong Jiang
Peng Cheng Laboratory, Tsinghua Shenzhen International Graduate School

Qian Ma
ByteDance

Shengbin Meng
ByteDance

Zhenhui Yuan
University of Warwick

Zili Meng
HKUST

Abstract

Video Feed Streaming (e.g., TikTok, Reels) is increasingly popular nowadays. Users will be scheduled to the distribution infrastructure, including content distribution network (CDN) and multi-access edge computing (MEC) nodes, to access the content. Our observation is that the existing proactive content push algorithms, which are primarily based on historical access information and designed for on-demand videos, no longer meet the demands of video feed streaming. The main reason is that video feed streaming applications always push recently generated videos to attract users' interests, thus lacking historical information when pushing. In this case, push mismatches and load imbalances will be observed, resulting in degraded bandwidth cost and user experience. To this end, we propose KEPC-Push, a Knowledge-Enhanced Proactive Content Push strategy with the *knowledge* of video content features. KEPC-Push employs knowledge graphs to determine the popularity correlation among similar videos (with similar authors, contents, length, etc.) and pushes content based on this guidance. Besides, KEPC-Push designs a hierarchical algorithm to optimize the resource allocation in edge nodes with heterogeneous capabilities and runs at the regional level to shorten the communication distance. Trace-driven simulations show that KEPC-Push saves the peak-period CDN bandwidth costs by 20% and improves the average download speeds by 7% against the state-of-the-art solutions.

1 Introduction

Video feed streaming service (e.g., TikTok, Reels, Kuaishou) is becoming increasingly popular [9]. In video feed streaming applications, users keep scrolling down and the applications will recommend users with videos of users' interests [16]. As shown in Figure 1, to improve user experience and reduce bandwidth cost, upon the traditional content delivery network (CDN), content providers nowadays adopt hierarchical CDN-Edge architectures [3, 15, 45]. Deploying cache servers at the edge of CDN can further bring the popular contents closer to users, saving the traffic cost and reducing the delay for users.

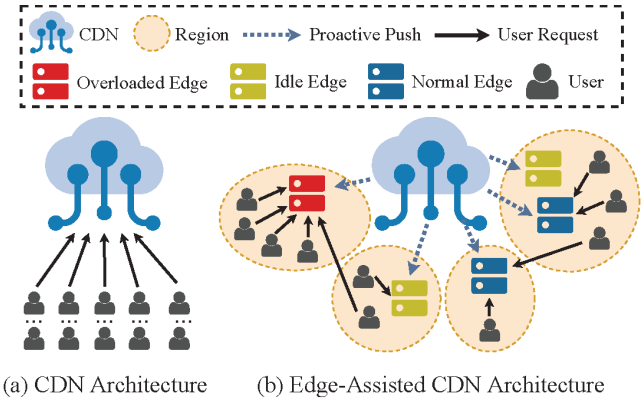


Figure 1: Architectures of CDN and CDN-Edge systems

However, video feed streaming poses new challenges to which videos to cache in the CDN-Edge architecture. To utilize the bandwidth during idle time, existing CDN-Edge architecture usually predicts the peak-period popularity of videos based on historical access data [36, 46], and pushes the popular ones to edge cache servers in advance [12, 14, 15, 26, 47]. However, for video feed streaming applications, content providers need to push the freshest videos to attract users' interests, where historical data is inaccessible. For example, in our measurements, as shown in Figure 2, we use web crawlers to retrieve preferentially recommended videos across different platforms, targeting hundreds of generic keywords, and analyze the publishing time of these videos. As we can see, the median time between video upload by the creator and distribution to the users is around 20 hours in the video feed streaming, which is not enough to even gather the historical information from one day before, while for Video-On-Demand (VoD), the time could be days to weeks. In this case, the ab-

Corresponding Author: Yong Jiang (jiangy@sz.tsinghua.edu.cn), Qing Li (liq@pcl.ac.cn)

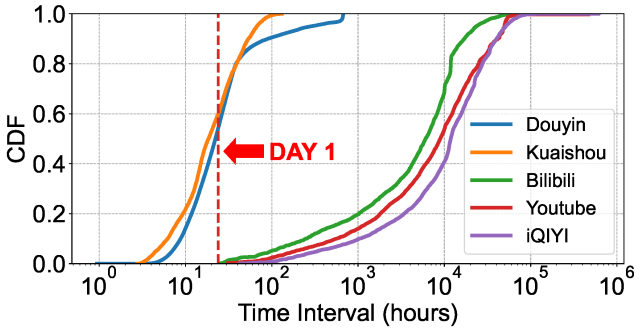


Figure 2: Time interval from video uploads to user viewing. Video feed streaming (Douyin and Kuaishou) has a shorter interval of orders of magnitudes than video-on-demand streaming (Bilibili, YouTube, and iQIYI).

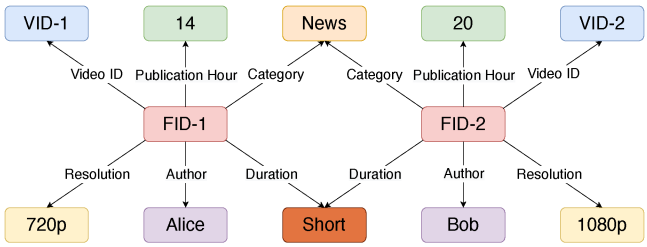


Figure 3: An illustration of video file knowledge graph

sence of historical data can lead to imprecise push, which in turn leads to observed load imbalances and mismatches, as shown in the edge-assisted CDN network of Figure 1. As a result, users' requests have to traverse longer to reach the correct node, which ultimately degrades both bandwidth cost and user experience.

To address the problems outlined above, our main insight is that *knowledge* derived from the content features of video files can help with the decision of video push. Figure 3 illustrates the knowledge graph constructed from the file content features. It includes features such as video category and duration, which provide additional information for determining video popularity trends during the cold-start phase. Importantly, these features are obtained at the time of video creation and do not rely on historical data. With the help of *knowledge*, edge load fluctuation can be effectively reduced by precisely pushing the video files to the most appropriate edge nodes. Therefore, we propose a Knowledge-Enhanced Proactive Content Push (KEPC-Push) strategy, aimed at improving the user's experience and bandwidth cost.

However, it is challenging to capture effective information from auxiliary knowledge to guide the load-balanced proactive push for heterogeneous edge nodes. The features of one video file are multi-dimensional, including creators, categories, publication times, and so on. KEPC-Push needs to select suitable features and extract valuable knowledge from them. Meanwhile, the processing capacity of edge nodes varies a lot, and some of them might only handle the parallel transmission of dozens of videos [10, 47]. Our measurements in production (§2.2 and §2.3) further confirm that the diverse popularity of video files and the heterogeneous capacity of edge nodes drastically affect the user experiences. KEPC-Push needs to carefully consider the heterogeneity of nodes to avoid overloads. Moreover, it is also challenging to design a fine-grained and lightweight scheduling scheme for large-scale fresh video feed streaming services. For example, applications such as Douyin [2] and Kuaishou [1] have hundreds of millions of monthly active users scattered across vast regions, requiring the strategy to be lightweight and scalable.

In response, we introduce two modules respectively to address the challenges. (1) a File Feature Embedding (FFE) module. As a first step, the FFE uses a knowledge graph embedding algorithm to identify the underlying popularity correlation among files. Additionally, a collaborative filtering embedding algorithm is introduced to supplement the representations of new files constantly created during the proactive push process. As a result, the continuously updated representation library will be used to guide the deployment of files to optimal edges in subsequent tasks. (2) a File Deployment Scheduling (FDS) module. The FDS first solves the allocation imprecision problem by implementing a region-level allocation mechanism, which dynamically configures the appropriate number of to-be-pushed replicas and the replaceable cache space for different regions, achieving precise file scheduling. Next, to alleviate the load imbalance problem, a clustering-based deployment mechanism is designed to make timely load-balanced push decisions with the representation library.

To demonstrate the performance of KEPC-Push, we compare it with several state-of-the-art proactive content push algorithms through comprehensive experiments using real video traces collected from Douyin, a leading video content provider in China. Experimental results show that: (i) KEPC-Push significantly optimizes the bandwidth cost and user experience, saves the peak-period CDN bandwidth by 20%, and improves request download speeds by 7%; (ii) The proposed load-balanced push mechanism effectively copes with load fluctuations caused by potential popularity bursts, decreases the ratio of overloaded edges by 19%, and improves the overall bandwidth utilization of the edge network by 22%; (iii) The proposed region-level allocation mechanism optimizes the request scheduling, reduces the cross-region request ratio by 10%, and reduces the repeated deployment of files.

The contributions of this paper are as follows:

- We propose a knowledge-enhanced proactive content push strategy for edge-based fresh video feed streaming systems, called KEPC-Push (§3);
- We design a file feature embedding (FFE) module that captures the underlying popularity correlation of files and provides guidance for subsequent deployment (§4);
- We propose a file deployment scheduling (FDS) module that allocates resources at the regional level, achieving

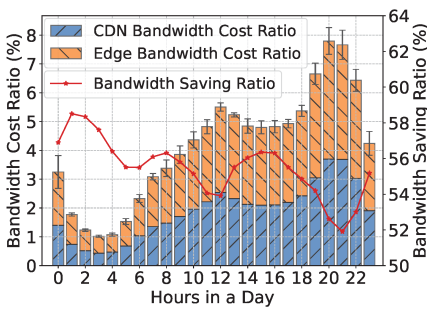


Figure 4: Change in bandwidth cost ratio and saving ratio within a day

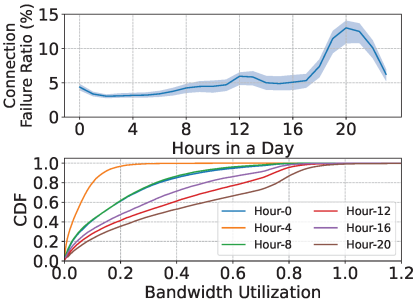


Figure 5: Change in core indicators of the edge network within a day

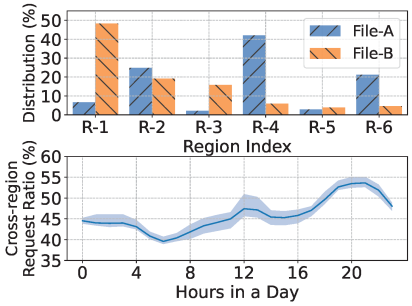


Figure 6: The heterogeneous distribution of user requests across regions

lightweight and load-balanced proactive push (§5);
- Comprehensive experiments through production traces demonstrate that both bandwidth cost and user experience are significantly optimized (§7).

2 Background and Motivation

In this section, we explain the background (§2.1), existing problems (§2.2), and root causes (§2.3) in the video feed streaming through large-scale measurement, and we summarize the challenges of addressing the problem (§2.4).

2.1 Edge-assisted Video Streaming System

To further optimize the user's experience, the CDN-Edge-hybrid content delivery architecture is replacing the CDN architecture. As shown in Figure 1, instead of users directly accessing the CDN node, users now will be scheduled to a series of edge nodes nearby. Edge nodes can be in the gateway of a campus or a building, which are much closer to the users. In this way, the access latency can be further improved by at least a half [6] to the level of 10-20 ms [28–30].

Unlike CDN, CDN-Edge architecture adopts proactive push strategies more often due to cost considerations. The bandwidth in an edge-assisted video streaming system is charged by the peak-time CDN bandwidth metering (e.g. 95th percentile bandwidth metering [33]), which can be reduced by predicting which files will gain the highest peak-period popularity, and proactively pushing them to the edge network in advance. For example, 20:00∼22:00 are usually the busy hours in the day, and pushing replicas at that time will incur additional CDN bandwidth expenditure. Therefore, on the server side, during the whole off-peak periods (0:00∼20:00 and 22:00∼24:00), the content provider periodically selects the files whose access frequency is higher than the preset access frequency threshold within the latest hour, and predicts their popularity during the next evening peak period (20:00∼22:00). Then a certain number of replicas, determined by the predicted popularity [12], will be proactively pushed to the edge network. On the client side, users constantly make requests and wait for the service from CDN or the edge network based on the cache status and workload of edge nodes.

However, the current edge-assisted video streaming system is inefficient at utilizing edge networks to save bandwidth [47]. We perform an in-depth measurement study over Douyin, one of the most popular video feed streaming applications. As illustrated in Figure 4, we record the bandwidth consumption data over a typical week, and investigate the distribution of hourly bandwidth consumed on CDN and edge networks to the total daily traffic, which is represented by CDN (edge) bandwidth cost ratio. Apparently, bandwidth costs are highest during the evening peak period, but bandwidth saving ratios (proportion of edge bandwidth cost to the total traffic of that hour) decrease significantly during the peak hours, which makes it difficult to reduce peak-period CDN bandwidth.

2.2 Observations

To understand the limitation of edge-accelerated streaming systems, especially for video feed streaming, we have the following two observations.

Load imbalance limits edge network's performance. We investigate the performance of edge nodes and show the results in Figure 5. As illustrated in the top sub-figure of Figure 5, the connection failure ratio of requests increases rapidly during the peak period. This is because many edge nodes are overloaded and therefore cannot serve more new requests, and this is even more severe for edge nodes since the capacity of each edge node is much less than a CDN node. However, in the bottom sub-figure of Figure 5, which shows the Cumulative Distribution Function (CDF) of edges' Bandwidth Utilization (BU), we can see that many idle nodes are not utilized effectively when other nodes are overloaded. Specifically, during the evening peak period, when the proportion of edge nodes with high load ($BU > 0.8$) exceeds 15%, 52% of edge nodes have low load ($BU < 0.4$). In summary, the highly-skewed workload across different edge nodes limits the utilization of the edge network.

Imprecise resource allocation degrades user's experience. In a large-scale online video streaming platform, massive users and edges are distributed over a wide area, and the geographical distribution of requests is unbalanced. As a

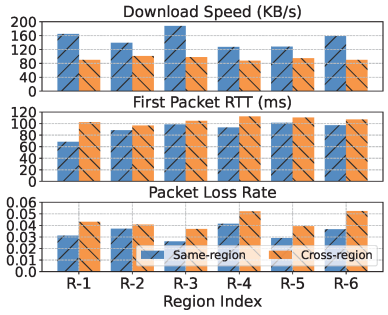


Figure 7: The QoS metrics for same-region and cross-region scheduling

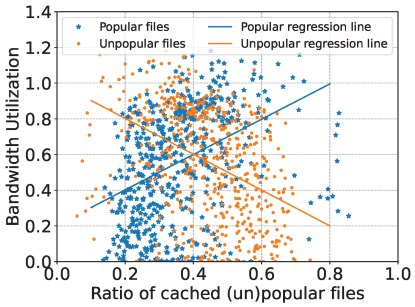


Figure 8: The impact of file popularity on bandwidth utilization

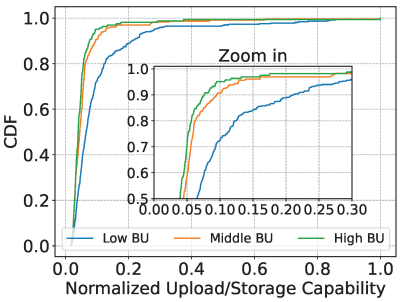


Figure 9: The impact of node capability on bandwidth utilization

practical example, we randomly select two video files from our production traces and present their heterogeneous user distribution in the top sub-figure of Figure 6. We can see that most users of File-A are from region-4 (about 42%), while dominant users are from region-1 (about 48%) for File-B. Furthermore, the bottom sub-figure of Figure 6 shows that current inaccurate resource allocation results in about half of all requests requiring cross-region scheduling. Compared with same-region scheduling, cross-region user requests will degrade the user's QoS (e.g. download speed, RTT and packet loss rate), which is shown in Figure 7. Whenever the download speed of a user request is insufficient, the content delivery task will be taken over by the CDN with a faster download speed to guarantee smooth playback of the video. Therefore, we should allocate resources appropriately across regions to reduce distances between users and sources, so that both CDN bandwidth and user experience can be improved.

The current proactive push strategy based on historical information will bring about the above-mentioned problems that need to be solved urgently, so it is not suitable for fresh video feed streaming services. Allocation imprecision problems are caused by ignorance of imbalanced request distribution, whereas the root causes of load imbalance problems are more complex and will be elaborated below.

2.3 Causes of Unbalanced Workload

To investigate what leads to the highly-skewed workload, we analyze the monitoring data of edge nodes and present the results in Figure 8 and Figure 9.

The first cause of the imbalance workload is the diverse popularity among video files. We count the bandwidth utilization of one thousand edge nodes and the popularity of cached files in these nodes during the evening peak period. As illustrated in Figure 8, we present how the bandwidth utilization varies with the ratio of cached popular and unpopular files by drawing scatter points and corresponding linear regression lines, in which (un)popular files are those with popularity falls within the top (bottom) 40%. We can see that as the proportion of cached popular files increases, edge nodes are more likely to be in a state of high load, while the trend is reversed for the unpopular files (their Pearson correlation coefficients are around ± 0.3). This is because storing a large number of popular files means the edge node might handle more requests in parallel, thus improving bandwidth utilization.

The second cause of the imbalance workload is heterogeneous edge capacity. As illustrated in Figure 9, we divide all edge nodes into three groups based on their bandwidth utilization, and calculate the CDF of their normalized upload/storage capability (i.e. upload bandwidth per unit of storage space). The results demonstrate that the edge nodes with lower capability are more likely to suffer heavy loads. Because upload bandwidth capability is decreased, the edge node's transmission bottleneck is reduced, and storage space is increased, more tasks can be handled simultaneously by the edge node.

Based on the above conclusions, we should not cache excessive files with high peak-period popularity on the same edge node, especially the edge with weak capability.

2.4 Challenges

In light of the data-driven insights above, we can develop an intuitive strategy that comprehensively considers the predicted popularity of cached files, the heterogeneous edge capabilities, and the geographical distribution of requests. By balancing edge workloads and allocating resources geographically, the peak-period CDN bandwidth and user experience can be improved. However, a practical proactive push mechanism still faces the following challenges.

Challenge-1: How to achieve load-balance proactive push for heterogeneous edge nodes without sufficient historical information. In video feed streaming services, it is difficult to accurately predict the peak-period popularity of to-be-pushed videos. On the one hand, fresh videos are always quickly deployed to the edge network, so there is only sparse historical information. On the other hand, the proactive push mechanism makes decisions periodically during the whole off-peak period, which means file popularity needs to be predicted over a long period of time. Since the accuracy of prediction declines significantly with the loss of historical information and the increase of forecast time span [35], the prediction results are not credible. Moreover, heterogeneous

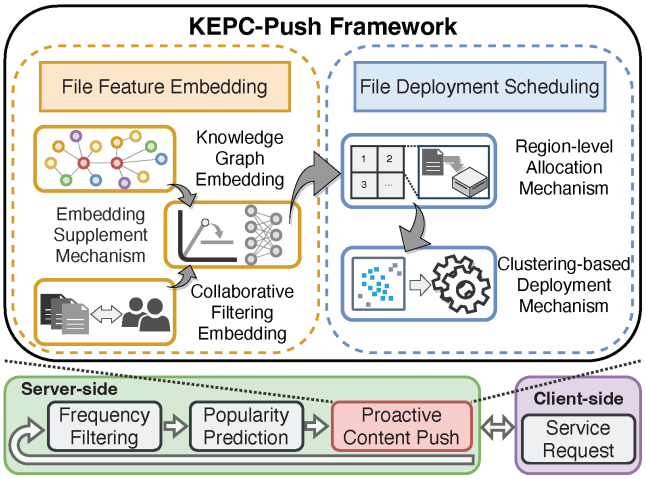


Figure 10: The system overview of KEPC-Push

edge capabilities further disrupt file deployment decisions.

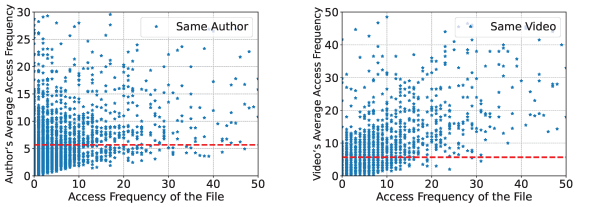
Challenge-2: How to ensure that the scheduling scheme can satisfy systematic requirements for large-scale video streaming services. First of all, in order to achieve precise proactive push, a flexible allocation mechanism that can dynamically adjust the number of replicas for files and the replaceable cache space for edge nodes should be designed at the regional level. In addition, at each proactive push decision, tens of thousands of files need to be pushed to the edge network, the delay of which will degrade the users' Quality of Experience (QoE) and increase bandwidth cost. Therefore, a lightweight scheduling scheme is necessary to ensure that the system can run in real-time.

3 The KEPC-Push Framework

To address the challenges outlined above, we propose the KEPC-Push, a knowledge-enhanced proactive push strategy. Figure 10 illustrates the system overview of KEPC-Push, which can mitigate the load imbalance and allocation imprecision problems by introducing auxiliary knowledge derived from the content features of video files. Specifically, two modules are designed to address the two challenges respectively.

First, to cope with the problem of insufficient historical information, we propose the **file feature embedding (FFE) module** (§4). The FFE module extracts the underlying popularity correlation of files from the content features and collaborative behaviors by utilizing Knowledge Graph (KG) and Collaborative Filtering (CF) algorithms, respectively. The auxiliary information enhances the ability of KEPC-Push to learn the diversity of video files. And the acquired representation library will be used to guide the deployment of files to optimal edge nodes in subsequent tasks.

Second, to design a fine-grained and lightweight scheduling scheme for large-scale video feed streaming services, we propose the **file deployment scheduling (FDS) module** (§5). The FDS module allocates resources at the regional level and



(a) Created by the same author (b) Belonging to the same video

Figure 11: The radiation effect of the file popularity

reduces the complexity of the proactive push problem. In detail, it dynamically configures the number of replicas and the size of replaceable cache space for different regions, and designs a clustering-based deployment mechanism to assist the deployment of files based on the learned representation vectors, achieving a load-balanced proactive push.

Furthermore, we also define the proactive push problem in Appendix §A, where the load imbalance and allocation imprecision issues that we need to address are formulated in detail. However, since the problem is NP-hard, it is necessary to propose our heuristic-based solution, namely, KEPC-Push.

4 File Feature Embedding

In this section, we introduce the file feature embedding module, which solves three sub-problems in capturing the underlying patterns of files from the knowledge data, namely, which content features are related to popularity trends, how to use these features as a guide for the file deployment, and how to handle constantly published new files during the push process.

4.1 Content Feature Selection

In the video streaming system, contents are delivered in the format of files rather than videos. This is because, due to differences in resolution, codec type and other content features, one video may be encoded into multiple files in advance to quickly respond to user requests. Our data analysis uncovers some useful phenomena and identifies the content features that determine the popularity correlation between files.

The popularity radiation effect. Files created by the same author or belonging to the same video are more likely to have similar popularity trends. As illustrated in Figure 11, we randomly select five thousand files from all the files requested during the evening peak period, and present the access frequency of each selected file and the average access frequency of other files created by the same author (or belonging to the same video). With the increasing access frequency, it is statistically more likely that the average access frequency of the other files will be greater than that of all selected files (see red dashed lines). In other words, when an author (or a video) becomes popular, other files created by that author (or belonging to that video) will also be requested more frequently.

The popularity skewness effect. The popularity of files can skew significantly in some characteristics, such as cate-

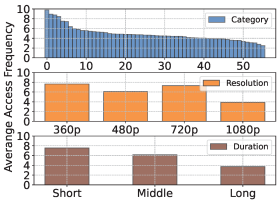


Figure 12: The skewed popularity in some characteristics

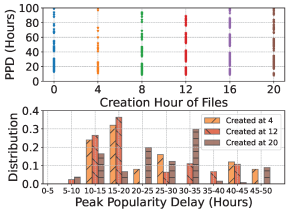


Figure 13: The distribution of peak popularity delay

gory, resolution and duration. For demonstration, we calculate the average access frequency of files with different characteristics during the evening peak period. As illustrated in the top sub-figure of Figure 12, we can see a three-fold difference in the average access frequency between the most popular and least popular categories among the 56 video categories (e.g., "News", "Movie" and "Game"). Similarly, file popularity is skewed in terms of resolution and duration.

The peak popularity delay effect. The timing of popularity is as important as which files will become popular. Therefore, we collect files created at different hours, and compute the time interval between when a file is created and when it reaches the popularity peak, which is called the Popularity Peak Delay (PPD). As illustrated in the top sub-figure of Figure 13, the distribution of the PPD is phased rather than continuous, because the popularity peaks of files mainly occur around the noon and evening peak periods. Besides, the distribution of the PPD varies significantly depending on the creation hour of files. In detail, we select the files published at 4, 12 and 20 o'clock and present the distribution of PPD in the bottom sub-figure of Figure 13, showing that they reach their popularity peaks at different times. For instance, we can see that most of the files published at 4 o'clock will reach peak popularity in the next 10∼20 hours, which is the afternoon and evening of that day, while for files published at 12 o'clock, it will be the evening of that day and morning of the next day. Meanwhile, as for the files published at 20 o'clock, they will dispersedly reach peak popularity over a long period in the future. This indicates that the files published at the same time tend to reach peak popularity at similar times, so they should be stored separately in the edge network.

The above findings indicate that there is a strong correlation between the file popularity and some key content features, and the files with the same content features show similar popularity trends, i.e., they reach popularity peaks at similar times and are more consistent in their popularity peaks. Therefore, files with different content features should be mixed and stored on the same edge node to avoid excessively high or low workloads in a short period of time.

4.2 Knowledge Graph Embedding

One consequent challenge is how to assess the similarity of files on the aforementioned content features. Considering the success of knowledge graphs in understanding mutual relations, we utilize the Knowledge Graph Embedding (KGE) method to learn representation vectors for different items.

Knowledge graphs can provide rich side information for items in content-enriched model [11]. A KG is a heterogeneous graph, where nodes represent entities, and edges represent relations between entities. By mapping items and their attributes into the KG, a general and compact context can be provided. As such, the KGE approaches can extract content features from the KG and accurately capture potential relations between items. We choose six content features to construct the knowledge graph, namely Video ID, Author, Category, Resolution, Duration and Publication Hour. An example of the KG is shown in Figure 3. In the KG, each edge is represented in the form of a triple (head entity, relation, tail entity), implying the specific relationship between the head entity and tail entity. For example, (FID-1, Author, Alice) indicates that Alice is the author of FID-1. Finally, the KGE can map entities and relations into low-dimensional representation vectors, in which the graph structure and semantic information are encoded.

To accurately learn the representation vectors, we choose TransR [24] method, which models entities and relations in separate spaces and trains vectors by projecting entities from entity space to relation space with projection matrices. We extract all the requested files and their content features within the last week, and input them into the TransR model.

In TransR, for each relation r, there is a corresponding projection matrix $\mathbf{M}_r$ that projects entities from entity space to relation space. With the mapping matrix, the projected vectors of the head entity and tail entity in triple (h,r,t) are defined as $\mathbf{h}_r = \mathbf{h}\mathbf{M}_r$ and $\mathbf{t}_r = \mathbf{t}\mathbf{M}_r$, respectively. The TransR model optimizes translation principle $\mathbf{h}_r + \mathbf{t} \approx \mathbf{t}_r$ to learn embeddings. As such, the score function of TransR is formulated as follows:

$$f_r(h,t) = \|\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r\|_2^2 = \|\mathbf{h}\mathbf{M}_r + \mathbf{r} - \mathbf{t}\mathbf{M}_r\|_2^2. \quad (1)$$

The training of TransR takes both correct triples and incorrect triples into consideration to encourage their discrimination, so the object for training is defined as the following margin-based score function

$$L = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} \max\left(0, f_r(h,t) + \gamma - f_r\left(h',t'\right)\right), \quad (2)$$

where γ is the margin, S is the set of correct triples and S' is the set of incorrect triples.

After the training process, we can get the representation vectors of all files, which are stored in a KG-based representation library *KG_library*. The distance between two vectors is inversely proportional to the popularity correlation of two files, and a stronger correlation means a more similar popularity trend, that is, the two files should be stored separately.

4.3 CF-based Embedding Supplement

Although the KG-based representation library has been acquired, new files are constantly being created during the proactive push process, and these files do not have corresponding vectors in the *KG_library*. Therefore, the *KG_library* needs to be updated in real-time to meet the requirements of file deployment scheduling. However, it is not acceptable to retrain the entire knowledge graph due to excessive training overhead, and the training time for millions of files may reach the day level. To reduce the update cost, we propose an Embedding Supplement Mechanism (ESM) to supplement the missing representation vectors of new files by introducing Collaborative Filtering Embedding (CFE) technology [34], which can capture the implicit relationship between new and old files from users' collaborative request behaviors.

Algorithm 1 shows the process of ESM, in which a library hit ratio threshold ξ is proposed to determine whether the entire knowledge graph needs to be retrained. There are two situations in which retraining may occur. First, recall that files are only proactively pushed to the edge network during the off-peak period, so there is ample time to update the input data and construct a new knowledge graph for retraining during the peak period. Second, if the library hit ratio of the scarce file list $\mathcal{L}$, which contains all files that are frequently accessed in the latest hour and need to be pushed, is smaller than the threshold ξ, the retraining will happen due to excessive omissions.

For other cases, we will use the CFE to obtain a CF-based representation library *CF_library* in three steps, which can be used to supplement representations. Specifically, the files requested by different users within the last four hours are first separated into independent sets, and the files accessed by the same user are considered to have a closer relationship. Next, we filter out the files whose access frequency is lower than the preset threshold. Finally, the item2vector method [5] and skip-gram model [13] are used to train the vector representations from the extracted sets. After acquiring *CF_library*, for any file f that cannot be retrieved in *KG_library*, we first find out the files that exist in both *KG_library* and *CF_library* from $\mathcal{L}$. Then we choose the file f' that has the closest vector distance to file f in *CF_library* from these files, and finally use the representation vector of f' in *KG_library* to temporarily supplement the representation vector of f.

There are two reasons for using the KGE to learn representation vectors and using the CFE to supplement them. On the one hand, the meta-data from the video streaming system is usually sparse and noisy, which means the vast majority of files are unpopular and lack user interaction information. In addition, the content features of files are available as soon as the videos are created. Therefore, we choose to construct the knowledge graph. On the other hand, the files that appear in the scarce file list are frequently accessed by users in the recent period. As such, sufficient collaborative behaviors can be used to identify which files are less likely to be requested consecutively and guide the embedding supplement, so the distance relationship among representation vectors in the *KG_library* can be maintained.

Algorithm 1: Embedding Supplement Mechanism (ESM)

Input: Scarce file list $f \in \mathcal{L}$, Library hit ratio threshold ξ, KG-based representation library *KG_library*, CF-based representation library *CF_library*.
Output: Updated KG-based representation library *updated_KG_library*.

```
Function EmbeddingSupplement():
    if |L ∩ KG_library.keys()| / |L| < ξ then
        updated_KG_library ← KG_training()
    else
        CF_library ← CF_training()
        updated_KG_library = KG_library
        IntersectionSet = KG_library ∩ CF_library
        for every file f in L do
            if f ∈ KG_library.keys() then
                Continue
            else
                Initial MinDistance ← +∞
                for f' ∈ IntersectionSet.keys() do
                    Distance = GetDistance
                      (CF_library[f], CF_library[f'])
                    if Distance < MinDistance then
                        MinDistance = Distance
                        updated_KG_library[f] =
                          updated_KG_library[f']
    return updated_KG_library
```

5 File Deployment Scheduling

In this section, we will introduce the file deployment scheduling module, which solves two sub-problems step by step, and finally completes the load-balanced proactive push decision. First of all, we need to determine how many replicas to push for each video and how much replaceable cache size each edge node should provide before each round of push. Then, how to match the to-be-pushed replica with the edge node based on the obtained representation library in a lightweight way is another challenge. To this end, we propose a Regional-level Allocation Mechanism (RAM) and a Clustering-based Deployment Mechanism (CDM).

5.1 Region-level Resource Allocation

Recall the observations mentioned in §2.2, many files have skewed geographical distributions of requests. Therefore, to reduce the ratio of cross-region request scheduling, we should determine the number of replicas at the regional level. Moreover, since edge nodes have heterogeneous capabilities and

fluctuating loads, we also should dynamically configure the replaceable cache size for different edges. Based on these facts, we design the region-level allocation mechanism, which makes independent allocation decisions for each region. And algorithm 2 summarizes the allocation process of the RAM.

The number of replicas is calculated in two steps. First, the popularity prediction model based on XGBoost [7] will forecast the peak-period popularity of file f in region g, which is defined as $v_{f,g}$. The reason for predicting at a regional level is that different regions have different user bases and thus be interested in different content. To train the XGBoost for each region, which models the non-linear relationship between file features and regional peak-period access frequency, we use dynamic contextual features that will vary within a short time (e.g., the historical number and regional distribution of user requests) and constant semantic features that can be encoded into one-hot vectors (e.g., codec type, resolution, bitrate, author, number of followers, category, duration and publication time). Second, to ensure that each request can be responded by at least η edge nodes in parallel to guarantee the QoE, the required number of to-be-pushed replicas, defined as $n_{f,g}$, can be calculated based on the network cache information $x_{f,e}$, which represents whether file f is stored on the node e.

After calculating the total size of the scarce file list, we need to assign different replaceable cache sizes to heterogeneous edge nodes. If an edge's bandwidth utilization is higher than the preset threshold μ in the previous proactive push iteration, it will not be assigned new deployment tasks. Otherwise, for those edge nodes with low load, the ones with larger cache space or lower bandwidth utilization will be allocated larger replaceable cache sizes, which helps to evenly share the workload on different edge nodes.

The proposed region-level allocation mechanism solves the allocation imprecision problem and offers two benefits. On the one hand, a precise replica allocation improves the download speed by increasing the number of same-region scheduling, while a reasonable replaceable space allocation helps to dynamically adjust the workload. On the other hand, accurate scheduling and balanced load can improve the overall network utilization, reducing the incorrect cache eviction of valuable files. Therefore, the number of replicas deployed repeatedly can be reduced. As a result, the two benefits feed off one another in a virtuous cycle.

Algorithm 2: Region-level Allocation Mechanism (RAM)

Input: Geographical regions $g \in \mathcal{G}$ and corresponding scarce file list $f \in \mathcal{L}_g$, edge node set $e \in \mathcal{E}_g$; File size s_f, edge bandwidth utilization U_e, edge cache space C_e and cache state $x_{f,e}$; Minimum number of connections η; Bandwidth utilization threshold μ;

Output: Replaceable cache size $RC_e \in \mathcal{RC}$ for e; Number of replicas $n_{f,g} \in \mathcal{N}$ for f in g.

Function *RegionAllocation*():
  for *every geographical region* $g \in \mathcal{G}$ **do**
    `// Compute the number of replicas`
    for *every scarce file* $f \in \mathcal{L}_g$ **do**
      $v_{f,g} \leftarrow XGBoostPopularityPredict()$
      $n_{f,g} = max\{\eta \cdot v_{f,g} - \sum_{e \in \mathcal{E}_g} x_{f,e}, 0\}$
    `// Compute the size of all replicas`
    $S_g = \sum_{f \in \mathcal{L}_g} (s_f \cdot n_{f,g})$
    `// Compute replaceable cache size`
    for *every cache node* $e \in \mathcal{E}_g$ **do**
      if $U_e \geq \mu$ **then**
        $RC_e = 0$
      else
        $RC_e = S_g \cdot \frac{C_e \cdot (\mu - U_e)}{\sum_{e \in \mathcal{E}_g} C_e \cdot (\mu - U_e)^+}$
  $\mathcal{RC} = \{RC_e \mid e \in \mathcal{E}\}$, $\mathcal{N} = \{n_{f,g} \mid g \in \mathcal{G}, f \in \mathcal{L}_g\}$
  return $\mathcal{RC}, \mathcal{N}$

5.2 Load-Balanced Proactive Push Decision

After determining the number of to-be-pushed replicas for scarce files and the replaceable cache size for edge nodes, another challenge is to achieve a lightweight and load-balanced push by completing appropriate push pairing between replicas and edges. To tackle this problem, we propose a clustering-based deployment mechanism, which is shown in algorithm 3.

At the beginning of each proactive push decision, we first supplement *KG_library* through algorithm 1 and use algorithm 2 to allocate resources. Then, we will execute independent file deployment decisions for each region. According to the acquisition method of representation vectors, we know that files with close vector distances have similar popularity trends, which means they need to be stored separately to balance the load. Theoretically, we should maximize the vector distance of all cached files in each edge node during the push process, but it is impractical due to the extremely high computational complexity. Therefore, we propose a heuristic-based solution to make it run in a real-time way.

In our solution, with the help of the representation vectors stored in *KG_library*, files in each region's scarce file list $\mathcal{L}_g$ are first clustered into K different clusters by K-means clustering strategy. Then we initialize a $K \times |\mathcal{E}_g|$ matrix called *DistanceMatrix*, in which *DistanceMatrix*$[k][e]$ stores the distance between the center vector of cluster k and the mean vector of all cached files in node e. To prioritize the deployment of popular files, we sort files in $\mathcal{L}_g$ according to their predicted popularity and push them in order. For file f, we first use the function *GetMaxDistanceCacheNodes*() to find the number of $\varepsilon \cdot n_{f,g}$ candidate nodes in *DistanceMatrix*. These candidate nodes have the *top*-$\varepsilon \cdot n_{f,g}$ farthest distance from the center vector of the cluster where f is located and have enough replaceable cache size to store f, which are defined as $\mathcal{E}_{f,g}$. Finally, we sort the edge nodes in $\mathcal{E}_{f,g}$ by their upload/storage capabilities, and the *top*-$n_{f,g}$ nodes with the strongest capability I_e are selected to store f. In this way, we

Algorithm 3: Clustering-based Deployment Mechanism (CDM)

Input: Geographical regions $g \in \mathcal{G}$ and corresponding scarce file list $f \in \mathcal{L}_g$, edge node set $e \in \mathcal{E}_g$; Edge cache space C_e, edge upload bandwidth bottleneck B_e, edge cache file set $\mathcal{D}_e$, predicted file popularity $v_{f,g}$ and number of replicas $n_{f,g}$; Number of clusters K; Search amplification factor of candidate nodes ε.

Output: Proactive push decision $\mathcal{PD}$.

Function *ProactivePush*():
- $KG\_library \leftarrow EmbeddingSupplement()$
- $\mathcal{RC}, \mathcal{N} \leftarrow RegionAllocation()$
- **for** *every geographical region* $g \in \mathcal{G}$ **do**
  - `// Perform K-means clustering`
  - $Clusters \leftarrow KMC(\mathcal{L}_g, K, KG\_library)$
  - Initial $DistanceMatrix \leftarrow [\,][\,]$
  - **for** $k \in K$ **do**
    - **for** $e \in \mathcal{E}_g$ **do**
      - $DistanceMatrix[k][e] \leftarrow GetDistance(Clusters[k].center, KG\_library[\mathcal{D}_e].mean)$
  - `// Select edge nodes to cache files`
  - Sort files f in $\mathcal{L}_g$ by their popularity $v_{f,g}$
  - **for** *every file* f *in* $\mathcal{L}_g$ *in order* **do**
    - $\mathcal{E}_{f,g} \leftarrow GetMaxDistanceCacheNodes(DistanceMatrix[f.cluster], \varepsilon \cdot n_{f,g}, \mathcal{RC})$
    - Sort nodes e in $\mathcal{E}_{f,g}$ by $I_e = B_e / C_e$
    - Extract the $top\text{-}n_{f,g}$ nodes to cache f and record the result in $\mathcal{PD}$
    - Update the replaceable cache size in $\mathcal{RC}$
- **return** $\mathcal{PD}$

achieve both lightweight and load-balanced proactive push.

In the CDM, to accelerate file deployment decisions, we simplify the calculation of *DistanceMatrix* to reduce the computation costs. And the computational complexity of each *DistanceMatrix* is reduced from $O(|\mathcal{L}_g| \cdot |\mathcal{E}_g| \cdot |\mathcal{D}_e| \cdot d^2)$ to $O(K \cdot |\mathcal{E}_g| \cdot d^2)$, where d is the embedding size. Actually, the average vector distance between each scarce file and all cached files in each node should be recorded in *DistanceMatrix*. However, based on the principle of triangle inequality [8], the simplified result can still represent the lower bound of the actual distance, which can also provide a guide for file deployment.

6 Experiment Setup

6.1 Dataset

The dataset is collected with the help of Douyin, a leading online video-sharing platform, and it consists of three components: client requests, file features and device information.

The client request data contains 60 million client traces spanning three weeks in September 2022, in which each trace represents a responded user request consisting of four key fields (e.g., "Client ID", "File ID", "Device ID" and "Timestamp"). A total of 13 million files are requested by 80 thousand unique users from six different regions, and these requests are served by 500 edge devices. The file feature data records files' content features that are used to construct the knowledge graph and predict the peak-period popularity (e.g., "Resolution", "Duration", "Category", "Publish Hour", "Author ID", "Video ID", "Bitrate" and "Codec Type"). And the device information data records the inherent capabilities and service information of all edge nodes (e.g., "Upload Bandwidth Bottleneck" (mean: 25.65 Mbps; std: 29.62 Mbps), "Storage Space" (mean: 131.61 GB; std: 113.80 GB), "Used Bandwidth", "Used Cache Size", and "Cache Information"), which can be collected from the service monitors.

6.2 Trace-driven Simulation

Implementation. To verify the performance of our proposed method, we build a trace-driven experiment platform, which can simulate parallel user requests based on the prepared traces. The hardware and software setup of the experimental platform is described in Appendix §B. In order to keep KEPC-Push and other baselines consistent with the online deployment environment, we set 5 minutes as a decision-making cycle, and replicas are pushed to the edge network only between 0:00 and 20:00. In the experiment, a central monitor is used to record the cache file list and access request list of all edge nodes in each cycle, which are updated in real-time and used for performance evaluation. The server-side and client-side workflows are the same as mentioned in §2.1, in which the preset access frequency threshold is five requests per hour. Additionally, the edge devices adopt the Least Recently Used (LRU) algorithm [22] to implement cache replacement. For those parameters used in KEPC-Push, sensitivity analyses are conducted to determine the optimal settings, and we discuss this further in Appendix §C. Ultimately, we empirically set $[\xi, \eta, \mu, \varepsilon, K] = [0.9, 5, 0.9, 3, 16]$.

Training. In the KGE training, we extract all the files requested within the last week for every training, and construct the knowledge graph with their content features. During the training, the dimension of entity embedding and relation embedding are both set to 128. As for the CFE training, we extract the user request sets from the client traces within the last four hours according to the strategies mentioned in §4.3, then input them into the skip-gram model for training, in which the size of the hidden layer is set to 128. Besides, we set one hour as the time period for popularity prediction, and we predict the peak-period popularity based on the data from the past 12 time periods (fill missing values with 0). Since we need the data of two-week length to train XGBoost models for each region, we use the last one-week data for evaluation.

Baseline Algorithms. To evaluate the performance of our KEPC-Push, we choose some existing algorithms modified to proactive push strategies. **(1) Origin-Push**: This method

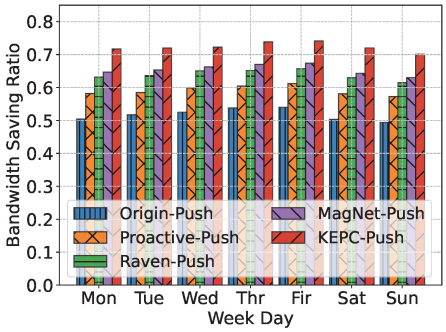
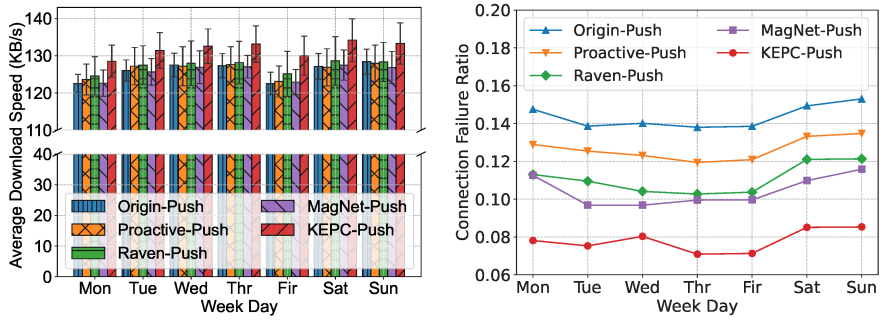


Figure 14: Evaluation of the evening peak-period CDN bandwidth saving



(a) Average Download Speed

(b) Connection Failure Ratio

Figure 15: Evaluation of the user experience improvement

selects a certain number of edge devices with low bandwidth utilization to push replicas that are predicted to be scarce, which is consistent with the online deployment environment. **(2) Proactive-Push [47]**: This method pushes scarce files with higher predicted peak-period popularity to edge devices with stronger upload capability, taking into account the heterogeneity of edge device capacities. **(3) Raven-Push [17]**: This method replaces the original LRU cache policy of Proactive-Push with a Belady-guided approach, which learns and evicts the cached content with the farthest arrival time, optimizing the performance of reactive caching. **(4) MagNet-Push [31]**: This method utilizes collaborative filtering to capture the underlying patterns of historical traces that guide pushes and requests, where user collaborative behaviors serve as key information to enhance the push strategy. The more detailed algorithm differences are shown in Table 1 in Appendix §D.

Performance Metrics. We use several metrics to validate performance improvement and challenge resolution.

- *Bandwidth Cost (Goal-1)*: **Bandwidth Saving Ratio** records the ratio of bandwidth saved by edge networks;
- *User Experience (Goal-2)*: The **Average Download Speed**, **Packet Loss Rate**, **RTT** and the **Connection Failure Ratio** (the percentage of requests not served due to overloaded edge nodes) of user requests are measured;
- *Load Imbalance Problem (Challenge-1)*: The **Ratio of Overloaded Nodes**, **Overall Bandwidth Utilization** of all edge nodes, and average **Number of Concurrent Requests** per node are recorded to reflect load balancing;
- *Allocation Imprecision Problem (Challenge-2)*: The **Cross-region Request Ratio** (the ratio of requests scheduled across regions) and **Number of Pushed Replicas** are used to reflect the accuracy of resource allocation.

7 Performance Evaluation

7.1 Saving in CDN Bandwidth Consumption

We first evaluate the CDN bandwidth saving ratio during the evening peak period to observe the performance of different algorithms in reducing peak-time bandwidth consumption.

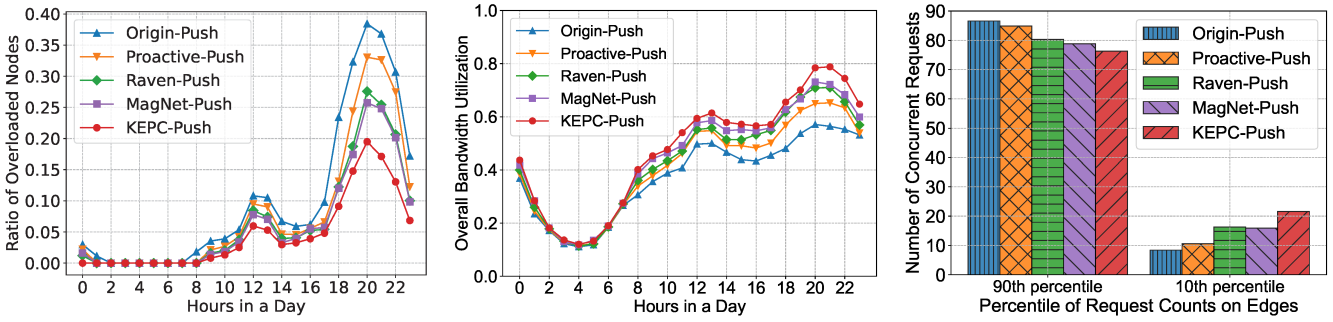
As we can see in Figure 14, on the basis of Origin-Push, the other four algorithms all further reduce the CDN bandwidth consumption, which is expected, because they deploy replicas in the appropriate nodes by optimizing the proactive or reactive caching. In this way, load imbalance and allocation imprecision problems are mitigated to varying degrees, so that more peak-period traffic can be offloaded by the edge network. Furthermore, KEPC-Push saves the most CDN bandwidth among them. Specifically, KEPC-Push improves the bandwidth saving ratio by 6.9%, 8.4%, 13.2%, and 20.5% compared with MagNet-Push, Raven-Push, Proactive-Push and Origin-Push, respectively. This indicates that KEPC-Push achieves excellent bandwidth saving performance through the knowledge-enhanced proactive content push strategy.

7.2 Improvements in User Experience

To verify the user experience improvements of KEPC-Push, we measure the average download speed and connection failure ratio of user requests, which are shown in Figure 15.

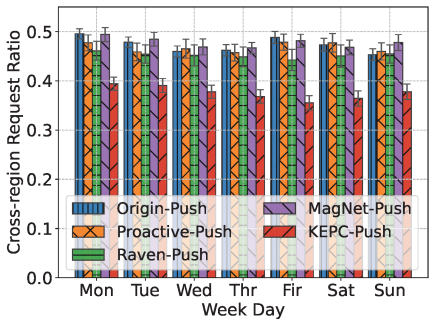
First of all, KEPC-Push significantly improves the average download speed of user requests. Figure 15(a) shows that the introduction of KEPC-Push leads to an increase in the average download speed (by about 7%). This is because the regional resource allocation effectively reduces the communication distance, which also benefits the RTT and packet loss rate (discussed in Appendix §E). The improvement of average download speed is extremely important, not only to ensure the smoothness of user viewing, but also to avoid requests being redirected to CDN due to insufficient transmission capability.

Besides, KEPC-Push enables users to suffer less connection interruption. As shown in Figure 15(b), we can observe that KEPC-Push reduces connection failure ratio by 2.6%, 3.2%, 4.8% and 6.5% compared with MagNet-Push, Raven-Push, Proactive-Push and Origin-Push, respectively. Since KEPC-Push uses knowledge graphs and collaborative filtering technologies to extract the popularity correlation, it stores files with similar popularity trends separately, avoiding excessive requests guided to the same node. In contrast, other algorithms perform unsatisfactorily. MagNet-Push ignores

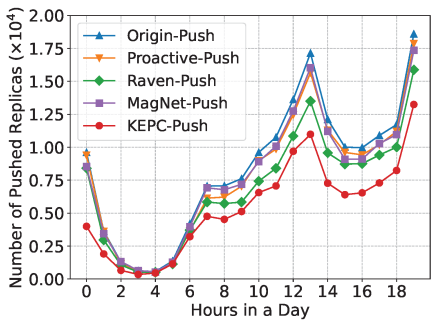


(a) Ratio of Overloaded Nodes (b) Overall Bandwidth Utilization (c) Number of Concurrent Requests

Figure 16: Evaluation of the load balancing performance in the edge network



(a) Cross-Region Request Ratio



(b) Number of Pushed Replicas

Figure 17: Evaluation of the resource allocation performance in the edge network

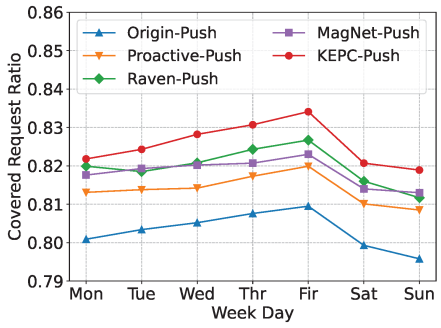


Figure 18: Evaluation of the evening peak-period cache hit performance

the utilization of auxiliary knowledge about content features, which is especially important in the cold-start phase with only sparse user access behaviors. Besides, the performance of Raven-Push and Proactive-Push relies heavily on the prediction results of file popularity and arrival patterns, which are often not accurate enough during the off-peak period.

7.3 Load Balancing Performance Evaluation

Figure 16 summarizes the optimization performance of all methods on the load imbalance problem in the edge network.

First, we compute the proportion of edge nodes that have reached an overloaded state at any hour within a day. As shown in Figure 16(a), KEPC-Push significantly reduces the ratio of overloaded nodes, especially during the evening peak period. Specifically, KEPC-Push decreases the peak-period overloaded node ratio by 6.2%, 8.1%, 13.6% and 18.9% compared with MagNet-Push, Raven-Push, Proactive-Push and Origin-Push, respectively. In this case, the edge network's transmission pressure can be distributed more evenly.

Then, to evaluate the impact of load balancing optimization on the entire edge network, we compute the overall bandwidth utilization of all edge nodes. As illustrated in Figure 16(b), the overall bandwidth utilization of KEPC-Push is higher than other algorithms during the whole day. This is because KEPC-Push pushes files with similar popularity trends to different edge nodes, keeping the access traffic of each node within an acceptable range. Therefore, edge nodes are not only less likely to become overloaded, but also more likely to effectively utilize their bandwidth.

Additionally, Figure 16(c) depicts the concurrent request counts at the 90th percentile and 10th percentile among all edges during the evening peak period. KEPC-Push exhibits the least number of requests at the 90th percentile, while having the highest number of requests at the 10th percentile. This also demonstrates that KEPC-Push achieves the transfer of access traffic from high-load nodes to low-load nodes through appropriate cache configuration.

7.4 File Allocation Performance Evaluation

As shown in Figure 17, we analyze the optimization performance of all methods on the allocation imprecision problem.

Figure 17(a) shows the cross-region request ratio achieved by different algorithms in one week, from which we can obtain the following conclusions. By comparing KPEC-Push to other baselines, it can reduce the cross-region request ratio by about 8%, which means that more user requests can be served by edge nodes in the same region, shortening the communication distance. And it is an important reason for the increase in average download speed. Besides, KPEC-Push achieves precise geographical resource allocation through the region-level allocation mechanism, so that more requests are scheduled within the same region.

In addition, Figure 17(b) shows the number of video replicas proactively pushed to the edge network during the off-

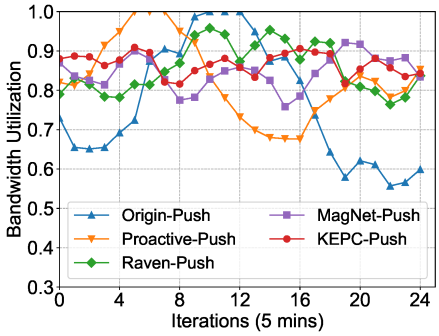


Figure 19: Temporal variation of evening peak-period bandwidth utilization

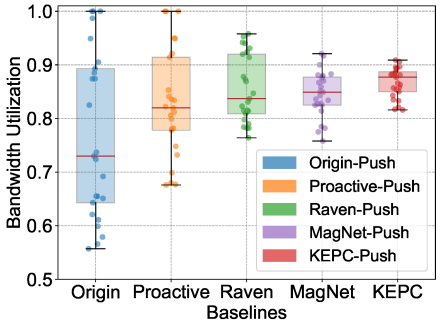


Figure 20: Boxplot graph of evening peak-period bandwidth utilization

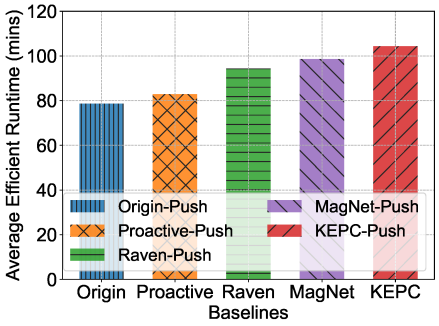


Figure 21: The average efficient runtime of edge nodes during the evening peak period

peak period within one day. Obviously, KEPC-Push can significantly reduce the number of replicas pushed, because precise resource allocation increases the cache lifetime of valuable files, reducing the number of replicas deployed repeatedly. Besides, Raven-Push also reduces deployment overhead by using an advanced Belady-guided cache replacement algorithm.

7.5 Cache Hit Performance Evaluation

As illustrated in Figure 18, KEPC-Push improves the covered request ratio (the percentage of the requested files for which there are enough replicas in the edge network) by about 2%, which means KEPC-Push's cache hit performance consistently outperforms other algorithms in different situations.

Actually, there are two main reasons for the performance improvement in the covered request ratio. On the one hand, KEPC-Push achieves a more precise allocation of scarce resources by introducing geographical information in the popularity prediction. Therefore, users can find more available replicas for their requests within the same region. On the other hand, due to the aforementioned reason, KEPC-Push reduces incorrect cache eviction of valuable files, thereby improving the coverage request ratio and reducing CDN bandwidth cost.

7.6 Ability to handle potential popularity burst

To intuitively demonstrate how KEPC-Push handles potential popularity bursts and improves overall bandwidth utilization, we select a typical edge node and count its bandwidth utilization every five minutes during the evening peak period.

Figure 19 shows the temporal variation of bandwidth utilization for the same edge node during the two-hour evening peak period (20:00∼22:00) when deploying different proactive push strategies. It can be seen that among all strategies, KEPC-Push does not maintain optimal bandwidth utilization at all times, but its bandwidth utilization fluctuates less and its performance is more stable, which is what we expect. Origin-Push and Proactive-Push do not consider the popularity correlation between cached resources, which will cause edge nodes to be overloaded or only have low bandwidth utilization in some periods. Raven-Push and MagNet-Push utilize patterns of historical traces to capture the potential correlation between user requests, thereby optimizing content push and request guidance, thus mitigating fluctuations in bandwidth utilization. However, the enriched information contained in the content features of files is ignored in Raven-Push and MagNet-Push, while KEPC-Push effectively uses them to construct the knowledge graph and complete further performance optimization. Figure 20 shows the boxplot graph that can reveal the distribution of bandwidth utilization. Compared to other solutions, KEPC-Push can handle the potential popularity bursts and maintain a stable and higher peak-period bandwidth utilization, thus achieving better performance.

Moreover, Figure 21 illustrates the average effective runtime ($0.7 \leq BU \leq 0.95$) of all edge nodes during the evening peak period for different schemes. Evidently, the more stable performance prevents edge nodes from experiencing overload or idle states, bringing more efficient runtime.

7.7 Ablation Experimental Evaluation

In the KEPC-Push strategy, we propose two modules (namely FFE and FDS) to respectively address the load imbalance and allocation imprecision problems. To validate the role of each module, we design the ablation study by replacing the FFE and FDS modules with other alternative solutions and assessing the performance. In Figure 22, we conduct seven experiments using one-week data and record the differences (mean/maximum/minimum) in overall bandwidth utilization and cross-region request ratio under different scenarios.

In the FFE module, KEPC-Push adopts a combination of knowledge graph and collaborative filtering to train and supplement the representation library, and measures the similarity of files based on this. Correspondingly, we propose two alternative mechanisms, including (1) performing label encoding on six content features separately and calculating similarity based on label differences, and (2) performing one-hot encoding on all classifications of different features and calculating similarity using Euclidean distance. As for the FDS module that completes resource allocation at the regional level, we use the global resource allocation mechanism as an alternative to validate the necessity of region-level resource allocation.

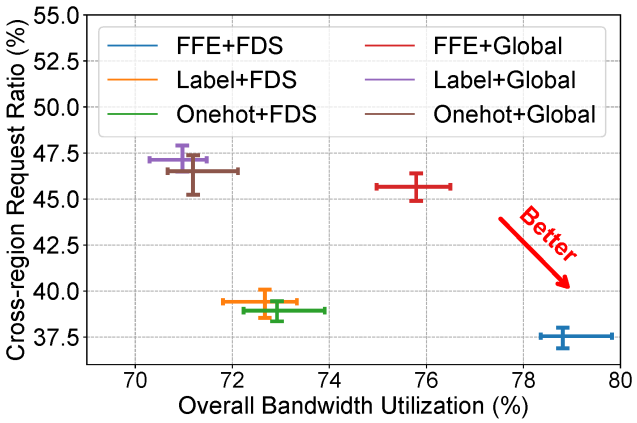


Figure 22: Ablation experiment of modules in KEPC-Push

As illustrated in Figure 22, among all the combination schemes, KEPC-Push (FFE+FDS) exhibits the best performance, with the highest overall bandwidth utilization and the lowest cross-region request ratio, demonstrating the important roles of the FFE and FDS modules. The reasons why the KEPC-Push strategy achieves the best performance are multifaceted. Firstly, in the file feature embedding dimension, although the representations of files can be easily obtained using label encoding and one-hot encoding, and there is no need to supplement the missing representation, their performance in file similarity measurement is unsatisfactory compared with the FFE module. On the one hand, it is challenging to precisely define the weight relationships between different feature dimensions. On the other hand, accurately quantifying the differences within the same feature dimension is inseparable from graph networks that contain implicit information such as feature quantity and interconnected relationships. However, the FFE module that adopts KGE and CFE can more effectively capture mutual relationships and improve the overall bandwidth utilization. Secondly, in the file deployment scheduling dimension, compared to global resource deployment, the FDS module significantly reduces the cross-region request ratio, which can not only help shorten the user request span, but also improve various QoS metrics.

8 Related Work

Edge-assisted Content Delivery. To alleviate the CDN bandwidth consumption, reactive and proactive edge caching mechanisms have been widely deployed in video content delivery [18, 41]. Reactive caching policies [10, 20, 43] determine whether to cache the requested content by analyzing historical traces and video features. However, the caching decision occurs after the user has already requested the content, which is not timely to alleviate the traffic pressure. Proactive caching policies [4, 46, 47] predict the files with high peak-period popularity and proactively push them to the edge network in advance. In this way, the bandwidth consumption can be significantly reduced during the peak periods, but we have extensively analyzed the existing problems in §2.2.

Load Balancing Strategy. To balance the workload in a multi-cache network, many cooperative edge caching mechanisms have been proposed [19, 37]. In centralized caching solutions [21, 32, 38, 44], one or more central nodes are utilized to synchronize information and schedule requests, which brings the central nodes excessive pressure. In decentralized caching solutions [27, 31, 39, 40], each node interacts local cache status with others and formulates its own caching policy in a distributed way, however, due to high computational complexity or excessive request hops, they cannot be practically utilized in weak edges. Furthermore, none of them introduce auxiliary knowledge data to guide global file deployment, instead optimizing decisions in existing caching scenarios.

Video Recommendation. With the rapid development of the internet, the volume of videos has grown exponentially [11]. To help users pick out what interests them among massive choices, recommender systems have been applied. Recommendation models can be divided into collaborative filtering models and content enriched models [42], both of which reveal potential relationships between videos, and the relations can also be used to guide proactive content cache [23, 25].

9 Conclusion

In this paper, we analyze the challenges of proactive push solutions for edge-assisted video feed streaming, and propose a knowledge-enhanced proactive content push strategy named KEPC-Push. The KEPC-Push has two innovative modules. The first part is the file feature embedding module, which utilizes the knowledge graphs and collaborative filtering technologies to capture the underlying popularity relations of files from the content features and collaborative behaviors, and the obtained relations can be used to guide the deployment of files to optimal edges. The second part is the file deployment scheduling module, which achieves precise resource allocation and load-balanced proactive push by using a region-level allocation mechanism and a lightweight clustering-based deployment mechanism, respectively. The trace-driven simulation has proven that KEPC-Push outperforms existing algorithms through extensive experiments, as it can further save the peak-period CDN bandwidth costs by 20%, increase the overall bandwidth utilization of the edge network by 21%, and improve the download speeds by 7%.

10 Acknowledgement

This work is supported by the Major Key Project of PCL under grant No. PCL2023A06-4, the National Key Research and Development Program of China under grant No. 2022YFB3105000, and the Shenzhen Key Lab of Software Defined Networking under grant No. ZDSYS20140509172959989.

References

[1] Number of daily and monthly active users of kuaishou from 2017 to 2022 | statista. https://www.statista.com/statistics/1239708/kuaishou-daily-and-monthly-active-user-numbers/.

[2] Number of monthly active users of douyin in china from november 2021 to december 2022 | statista. https://www.statista.com/statistics/1361354/china-monthly-active-users-of-douyin-chinese-tiktok/.

[3] ADHIKARI, V. K., GUO, Y., HAO, F., VARVELLO, M., HILT, V., STEINER, M., AND ZHANG, Z.-L. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In *2012 Proceedings IEEE INFOCOM* (Orlando, FL, May 2012).

[4] BACCOUR, E., ERBAD, A., BILAL, K., MOHAMED, A., AND GUIZANI, M. Pccp: Proactive video chunks caching and processing in edge networks. *Future Generation Computer Systems 105* (2020), 44–60.

[5] BARKAN, O., AND KOENIGSTEIN, N. Item2vec: Neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)* (Vietri sul Mare, Italy, September 2016).

[6] CHARYYEV, B., ARSLAN, E., AND GUNES, M. H. Latency comparison of cloud datacenters and edge servers. In *GLOBECOM 2020-2020 IEEE Global Communications Conference* (Taipei, Taiwan, December 2020).

[7] CHEN, T., AND GUESTRIN, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, August 2016).

[8] ELKAN, C. Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th international conference on Machine Learning (ICML-03)* (Palo Alto, California, March 2003).

[9] GRECE, C. Trends in the vod market in eu28. *Strasbourg: European Audiovisual Observatory. https://rm. coe. int/trends-in-the-vod-market-in-eu28-final-version/1680a1511a* (2021).

[10] GUAN, Y., ZHANG, X., AND GUO, Z. Caca: Learning-based content-aware cache admission for video content in edge caching. In *Proceedings of the 27th ACM International Conference on Multimedia* (New York, NY, USA, October 2019).

[11] GUO, Q., ZHUANG, F., QIN, C., ZHU, H., XIE, X., XIONG, H., AND HE, Q. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering 34*, 8 (2020), 3549–3568.

[12] GUO, Y., ZHANG, Y., YANG, Z., BIAN, K., TUO, H., AND DAI, Y. Atdps: An adaptive time-dependent push strategy in hybrid cdn-p2p vod system. In *2018 IEEE International Conference on Communications (ICC)* (Kansas City, MO, USA, May 2018).

[13] GUTHRIE, D., ALLISON, B., LIU, W., GUTHRIE, L., AND WILKS, Y. A closer look at skip-gram modelling. In *Proceedings of the fifth international conference on language resources and evaluation (LREC'06)* (Genoa, Italy, May 2006).

[14] HAN, S., SU, H., YANG, C., AND MOLISCH, A. F. Proactive edge caching for video on demand with quality adaptation. *IEEE Transactions on Wireless Communications 19*, 1 (2019), 218–234.

[15] HAN, S., TAN, X., QI, K., YANG, C., MOLISCH, A. F., LU, Y., ZHENG, J., AND LI, Y. Rethinking the gain of multicasting and proactive caching for vod service. *IEEE Wireless Communications 27*, 5 (2020), 133–139.

[16] HE, J., HU, M., ZHOU, Y., AND WU, D. Liveclip: Towards intelligent mobile short-form video streaming with deep reinforcement learning. In *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video* (New York, NY, USA, June 2020).

[17] HU, X., RAMADAN, E., YE, W., TIAN, F., AND ZHANG, Z.-L. Raven: Belady-guided, predictive (deep) learning for in-memory and content caching. In *Proceedings of the 18th International Conference on Emerging Networking EXperiments and Technologies* (New York, NY, USA, November 2022).

[18] JEDARI, B., PREMSANKAR, G., ILLAHI, G., FRANCESCO, M. D., MEHRABI, A., AND YLÄ-JÄÄSKI, A. Video caching, analytics, and delivery at the wireless edge: A survey and future directions. *IEEE Communications Surveys & Tutorials 23*, 1 (2021), 431–471.

[19] KHAN, M. A., BACCOUR, E., CHKIRBENE, Z., ERBAD, A., HAMILA, R., HAMDI, M., AND GABBOUJ, M. A survey on mobile edge computing for video streaming: Opportunities and challenges. *IEEE Access 10* (2022), 120514–120550.

[20] KIRILIN, V., SUNDARRAJAN, A., GORINSKY, S., AND SITARAMAN, R. K. Rl-cache: Learning-based cache admission for content delivery. In *Proceedings of the 2019 Workshop on Network Meets AI & ML* (New York, NY, USA, August 2019).

[21] LARBI, A., BOUALLOUCHE-MEDJKOUNE, L., AND AISSANI, D. Improving cache effectiveness based on cooperative cache management in manets. *Wireless Personal Communications 98*, 3 (2018), 2497–2519.

[22] LEE, D., CHOI, J., KIM, J.-H., NOH, S. H., MIN, S. L., CHO, Y., AND KIM, C. S. Lrfu: A spectrum of policies that subsumes the least recently used and least frequently used policies. *IEEE transactions on Computers 50*, 12 (2001), 1352–1361.

[23] LI, Z., GAO, X., LI, Q., GUO, J., AND YANG, B. Edge caching enhancement for industrial internet: A recommendation-aided approach. *IEEE Internet of Things Journal 9*, 18 (2022), 16941–16952.

[24] LIN, Y., LIU, Z., SUN, M., LIU, Y., AND ZHU, X. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence* (Austin, Texas, USA, February 2015).

[25] LIU, Z., SONG, H., AND PAN, D. Distributed video content caching policy with deep learning approaches for d2d communication. *IEEE Transactions on Vehicular Technology 69*, 12 (2020), 15644–15655.

[26] MA, M., WANG, Z., SU, K., AND SUN, L. Understanding the smartrouter-based peer cdn for video streaming. *international conference on computer communications and networks* (2016).

[27] MA, M., WANG, Z., YI, K., LIU, J., AND SUN, L. Joint request balancing and content aggregation in crowdsourced cdn. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (Atlanta, GA, USA, June 2017).

[28] MENG, Z., KONG, X., CHEN, J., WANG, B., XU, M., HAN, R., LIU, H., ARUN, V., HU, H., AND WEI, X. Hairpin: Rethinking packet loss recovery in edge-based interactive video streaming. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)* (Santa Clara, CA, USA, April 2024).

[29] MENG, Z., WANG, T., SHEN, Y., WANG, B., XU, M., HAN, R., LIU, H., ARUN, V., HU, H., AND WEI, X. Enabling high quality {Real-Time} communications with adaptive {Frame-Rate}. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)* (BOSTON, MA, USA, April 2023).

[30] MOHAN, N., CORNEO, L., ZAVODOVSKI, A., BAYHAN, S., WONG, W., AND KANGASHARJU, J. Pruning edge research with latency shears. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks* (Virtual Event USA, November 2020).

[31] PENG, J., LI, Q., MA, X., JIANG, Y., DONG, Y., HU, C., AND CHEN, M. Magnet: Cooperative edge caching by automatic content congregating. In *Proceedings of the ACM Web Conference 2022* (New York, NY, USA, April 2022).

[32] QIAO, G., LENG, S., MAHARJAN, S., ZHANG, Y., AND ANSARI, N. Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks. *IEEE Internet of Things Journal 7*, 1 (2020), 247–257.

[33] STANOJEVIC, R., LAOUTARIS, N., AND RODRIGUEZ, P. On economic heavy hitters: shapley value analysis of 95th-percentile pricing. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement* (New York, NY, USA, November 2010).

[34] SU, X., AND KHOSHGOFTAAR, T. M. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence 2009* (2009).

[35] TAN, J., LIU, W., WANG, T., ZHAO, M., LIU, A., AND ZHANG, S. A high-accurate content popularity prediction computational modeling for mobile edge computing using matrix completion technology. *Transactions on Emerging Telecommunications Technologies 32*, 6 (2021), e3871.

[36] TANG, S., LI, Q., MA, X., GAO, C., WANG, D., JIANG, Y., MA, Q., ZHANG, A., AND CHEN, H. Knowledge-based temporal fusion network for interpretable online video popularity prediction. In *Proceedings of the ACM Web Conference 2022* (New York, NY, USA, April 2022).

[37] TIAN, A., FENG, B., ZHOU, H., HUANG, Y., SOOD, K., YU, S., AND ZHANG, H. Efficient federated drl-based cooperative caching for mobile edge networks. *IEEE Transactions on Network and Service Management* (2022), 1–1.

[38] UGWUANYI, E. E., GHOSH, S., IQBAL, M., DAGIUKLAS, T., MUMTAZ, S., AND AL-DULAIMI, A. Co-operative and hybrid replacement caching for multi-access mobile edge computing. In *2019 European Conference on Networks and Communications (EuCNC)* (Valencia, Spain, June 2019).

[39] WANG, F., WANG, F., LIU, J., SHEA, R., AND SUN, L. Intelligent video caching at network edge: A multi-agent deep reinforcement learning approach. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications* (Toronto, ON, Canada, July 2020).

[40] WANG, X., LI, R., WANG, C., LI, X., TALEB, T., AND LEUNG, V. C. M. Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching. *IEEE Journal on Selected Areas in Communications 39*, 1 (2021), 154–169.

[41] WU, H., FAN, Y., WANG, Y., MA, H., AND XING, L. A comprehensive review on edge caching from the perspective of total process: Placement, policy and delivery. *Sensors 21*, 15 (2021), 5033.

[42] WU, L., HE, X., WANG, X., ZHANG, K., AND WANG, M. A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2022), 1–1.

[43] YAN, G., AND LI, J. Rl-bélády: A unified learning framework for content caching. In *Proceedings of the 28th ACM International Conference on Multimedia* (New York, NY, USA, October 2020).

[44] ZHANG, F., HAN, G., LIU, L., MARTINEZ-GARCIA, M., AND PENG, Y. Joint optimization of cooperative edge caching and radio resource allocation in 5g-enabled massive iot networks. *IEEE Internet of Things Journal 8*, 18 (2021), 14156–14170.

[45] ZHANG, R.-X., YANG, C., WANG, X., HUANG, T., WU, C., LIU, J., WU, C.-G., AND AGGCAST, L. S. . Aggcast: Practical cost-effective scheduling for large-scale cloud-edge crowdsourced live streaming. In *Proceedings of the 30th ACM International Conference on Multimedia* (New York, NY, USA, October 2022).

[46] ZHANG, Y., BIAN, K., TUO, H., CUI, B., SONG, L., AND LI, X. 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)* (Chicago, IL, USA, August 2018).

[47] ZHANG, Y., GAO, C., GUO, Y., BIAN, K., JIN, X., YANG, Z., SONG, L., CHENG, J., TUO, H., AND LI, X. Proactive video push for optimizing bandwidth consumption in hybrid cdn-p2p vod systems. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications* (Honolulu, HI, USA, April 2018).

A Proactive Push Problem Definition

In the network model of the online video streaming services, given a proactive push scheme $\mathcal{A}$, a file set $\mathcal{F} = \{1,\cdots,f,\cdots,F\}$, a cache set $\mathcal{E} = \{1,\cdots,e,\cdots,E\}$, a client set $\mathcal{U} = \{1,\cdots,u,\cdots,U\}$ and a geographical region set $\mathcal{G} = \{1,\cdots,g,\cdots,G\}$, we define s_f as the size of file f, define R_f^t as the number of requests for file f at the time slot t, and use $r_{f,i}^t$ to represent the *i-th* user request for file f at the time slot t.

The optimization objective of the proactive push problem is to find an appropriate push scheme $\mathcal{A}$ to minimize the peak-time traffic from CDN servers, which is defined as PTT:

$$PTT = \max_t \left\{ \sum_{f\in\mathcal{F}} \sum_{e\in\mathcal{E}} p_{f,e}^t \cdot s_f + \sum_{f\in\mathcal{F}} \sum_{1\le i\le R_f^t} q_{f,i}^t \cdot s_f \right\}, \quad (3)$$

where $p_{f,e}^t$ indicates whether file f is proactively pushed to edge node e at the time slot t, $q_{f,i}^t$ indicates whether the user request $r_{f,i}^t$ is redirected to CDN servers, and both of them are influenced by the proactive push scheme $\mathcal{A}$.

The first part of PTT is the amount of the CDN bandwidth consumed by proactively pushing files, and the second part represents the CDN bandwidth generated by redirecting requests from edge to CDN due to cache miss, overloaded node and insufficient download speed. In detail, $q_{f,i}^t$ is defined as

$$q_{f,i}^t = 1 - x_f^t \cdot y_{f,i}^t \cdot z_{f,i}^t, \quad (4)$$

where x_f^t represents whether there are enough candidate nodes that cache file f in the edge network, $y_{f,i}^t$ represents whether there are enough non-overloaded nodes among the candidate nodes to serve the request $r_{f,i}^t$, $z_{f,i}^t$ represents whether the selected nodes can provide a sufficient download speed to meet the requirement of the user, and they are all 0/1 variables.

We define $\mathcal{E}(r_{f,i}^t)$ as the set of edge nodes that cache file f when processing the request $r_{f,i}^t$, and use $\mathcal{E}^O(r_{f,i}^t)$ and $\mathcal{E}^N(r_{f,i}^t)$ to indicate the subsets of overloaded and non-overloaded nodes in $\mathcal{E}(r_{f,i}^t)$, respectively. They are given by

$$\mathcal{E}\left(r_{f,i}^t\right) = \left\{ e \mid x_{f,e}^t = 1,\ e \in \mathcal{E} \right\}, \quad (5)$$

$$\mathcal{E}^O\left(r_{f,i}^t\right) = \left\{ e \mid U_e \ge \mu,\ e \in \mathcal{E}\left(r_{f,i}^t\right) \right\}, \quad (6)$$

$$\mathcal{E}^N\left(r_{f,i}^t\right) = \left\{ e \mid U_e < \mu,\ e \in \mathcal{E}\left(r_{f,i}^t\right) \right\}, \quad (7)$$

where $x_{f,e}^t$ represents whether file f is stored on the node e at the time slot t, U_e represents the current bandwidth utilization of node e, and we define μ as the threshold to distinguish overloaded nodes from non-overloaded nodes.

Therefore, x_f^t and $y_{f,i}^t$ are given by

$$x_f^t = \mathbb{I}\left(\left|\mathcal{E}\left(r_{f,i}^t\right)\right| \ge \eta\right), \quad (8)$$

$$y_{f,i}^t = \mathbb{I}\left(\left|\mathcal{E}^{\mathcal{N}}\left(r_{f,i}^t\right)\right| \ge \eta\right), \quad (9)$$

where η represents the least number of edge nodes that a client must connect to in parallel to guarantee the QoE.

In addition, we use $ds(r_{f,i}^t)$ to indicate the download speed at which the user downloads file f from the edge network when processing the request $r_{f,i}^t$. In the edge-assisted CDN, the request $r_{f,i}^t$ is split into η subrequests which will be sent to different edge nodes, and the download speed of each subrequest is mainly determined by the distance between the user and the edge node. Therefore, we use the triplet $(request, client, node)$ to represent it, such as the triple $(r_{f,i,j}^t, u(r_{f,i}^t), e(r_{f,i,j}^t))$ for the subrequest $r_{f,i,j}^t$. And $ds(r_{f,i}^t)$ can be given by

$$ds(r_{f,i}^t) = \sum_{1\le j\le\eta} ds\left(r_{f,i,j}^t, u\left(r_{f,i}^t\right), e\left(r_{f,i,j}^t\right)\right). \quad (10)$$

As such, $z_{f,i}^t$ can be written as

$$z_{f,i}^t = \mathbb{I}\left(ds(r_{f,i}^t) \ge ds_f\right), \quad (11)$$

where ds_f is the minimum download speed required to ensure that video file f can be played smoothly.

In summary, we can formally formulate the proactive push problem that aims to optimize the peak-time traffic as follows:

$$\begin{aligned} &\arg\min_{\mathcal{A}}: \ PTT \\ &\text{subject to}: \ (4),(5),(6),(7),(8),(9),(10) \textit{ and } (11) \end{aligned} \quad (12)$$

Problem (12) is an Integer Linear Programming (ILP) problem. As the problem is NP-hard, it is extremely challenging to solve the problem optimally within polynomial time. Therefore, we propose a heuristic-based solution for load balancing and resource allocation, named Knowledge-Enhanced Proactive Content Push (KEPC-Push).

B Hardware and Software Platforms

Our simulations and model training run on a server with two Intel(R) Xeon(R) E5-2620 v4 @ 2.10GHz CPUs, four NVIDIA GeForce RTX 2080 Ti GPUs, and 64G memory. The motherboard of the hardware platform is Huawei BC11HGSA0 (Version: V100R003). As for the software, the OS is Ubuntu 18.04.5 LTS, and the kernel is Linux 4.15.0-200-generic.

C Searching for the Appropriate Parameters

In this experiment, we find that the settings of two parameters in the clustering-based deployment mechanism have a significant impact on the performance of KEPC-Push, that is, the number of clusters K and the search amplification factor of

Table 1: Baseline algorithm comparison

Scheme	Origin-Push	Proactive-Push	Raven-Push	MagNet-Push	KEPC-Push
Predicting Peak-period Popularity	✔	✔	✔	✔	✔
Considering Edge Heterogeneity	✘	✔	✔	✔	✔
Belady-guided Cache Policy	✘	✘	✔	✘	✘
User Behavior Collaborative Filtering	✘	✘	✘	✔	✔
Utilizing Content Feature Knowledge	✘	✘	✘	✘	✔
Allocating Resources Geographically	✘	✘	✘	✘	✔

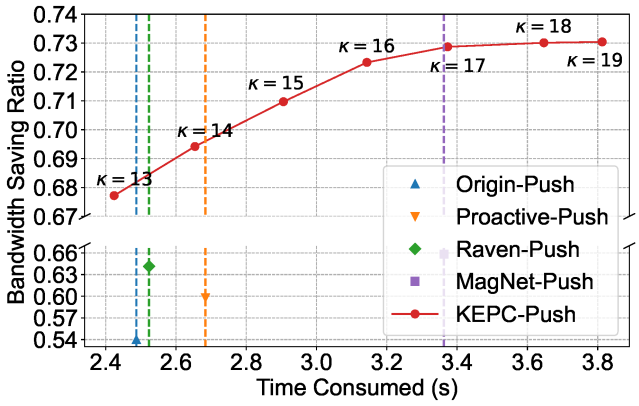


Figure 23: Effect of setting different parameters K (number of clusters)

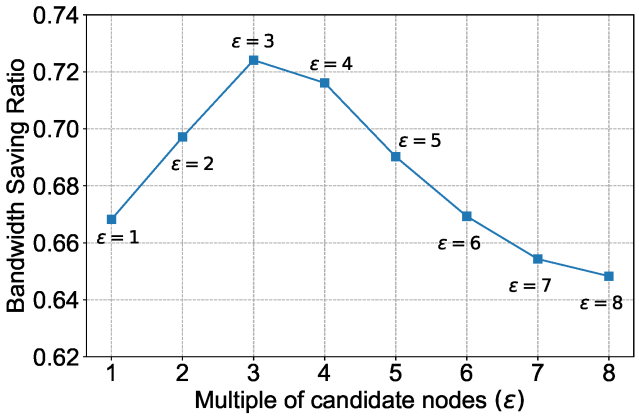


Figure 24: Effect of setting different parameters ε (search amplification factor)

candidate nodes ε. To search for appropriate parameters, we test the bandwidth saving ratio and running time with different parameter settings, and the results are shown in Figure 23 and Figure 24.

As illustrated in Figure 23, the bandwidth saving ratio increases with the increase of K, but the time consumption increases as well. This indicates that although the increase of K improves the accuracy of vector distance calculation, it also brings additional computing burdens. For comparison, we present the time consumption and bandwidth saving ratios of the proactive push decision processes achieved by other baselines. To obtain an appropriate parameter K with the competitive bandwidth saving ratio and time consumption, we finally set $K = 16$.

We demonstrate the impact of the search amplification factor on the bandwidth saving ratio in Figure 24, varying ε from 1 to 8. It is clear that the bandwidth saving ratio reaches the maximum value when $\varepsilon = 3$, and gradually decreases when ε is greater or smaller. This is because ε determines the trade-off between finding the edge node with the furthest vector distance and the node with the strongest individual capability. If ε is too small, it is difficult to choose nodes with strong capability from the candidate nodes to cache those scarce replicas with higher priority. On the contrary, if ε is too large, the results of clustering-based distance calculation cannot be used effectively, and the decision-making process will be similar to Proactive-Push.

D Algorithm Design Differences

Table 1 shows five different proactive push strategies used in the experiments and their design differences, which are mainly divided into the following six parts:

- **Predicting Peak-period Popularity:** All five strategies use the XGBoost model for peak-period popularity prediction before the proactive push. The training and testing data include the dynamic contextual features (e.g., the historical number and regional distribution of user requests) and constant semantic features (e.g., codec type, resolution, bitrate, author, number of followers, category, duration and publication time).
- **Considering Edge Heterogeneity:** This metric indicates whether the proactive push strategy takes into account the heterogeneity of edge nodes' capabilities when making push decisions, which is crucial for effectively improving the utilization of different edges.
- **Belady-guided Cache Policy:** This metric indicates whether the proactive push strategy employs the Belady-guided cache policy as the cache replacement algorithm within the edge nodes. By default, the LRU cache replacement algorithm is adopted.
- **User Behavior Collaborative Filtering:** This metric indicates whether the proactive push strategy utilizes collaborative filtering algorithms to analyze users' collaborative access behavior, assisting in content deployment

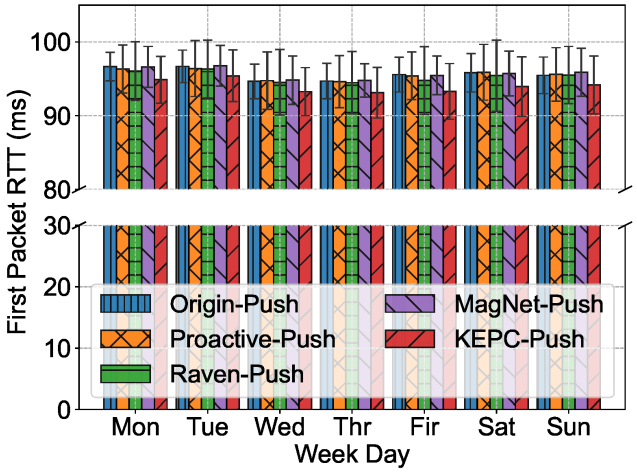


Figure 25: Evaluation of the average first packet RTT

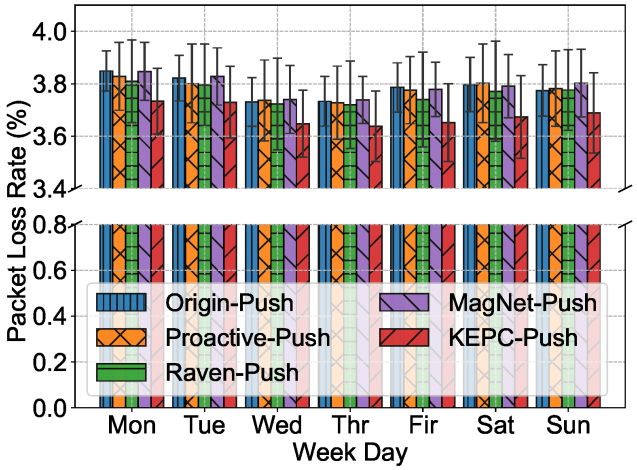


Figure 26: Evaluation of the average packet loss rate

decisions.

- **Utilizing Content Feature Knowledge:** This metric indicates whether the proactive push strategy introduces *knowledge* by leveraging file content features to construct a knowledge graph, enabling targeted handling of the video feed streaming service's cold-start phase.
- **Allocating Resources Geographically:** This metric indicates whether the proactive push strategy combines geographical information to achieve regional resource allocation. By default, resource allocation is global.

E Improvements in RTT and Packet Loss Rate

Figure 25 and Figure 26 respectively illustrate the first packet RTT and packet loss rate of user requests under different proactive push strategies. It can be observed that, similar to the video download speed discussed in §7.2, KEPC-Push also optimizes the RTT and packet loss rate to a certain extent due to the use of regional resource allocation. This shows that KEPC-Push can change some user requests originally scheduled for cross-region scheduling into same-region scheduling by optimizing push matching, thus optimizing QoS metrics.