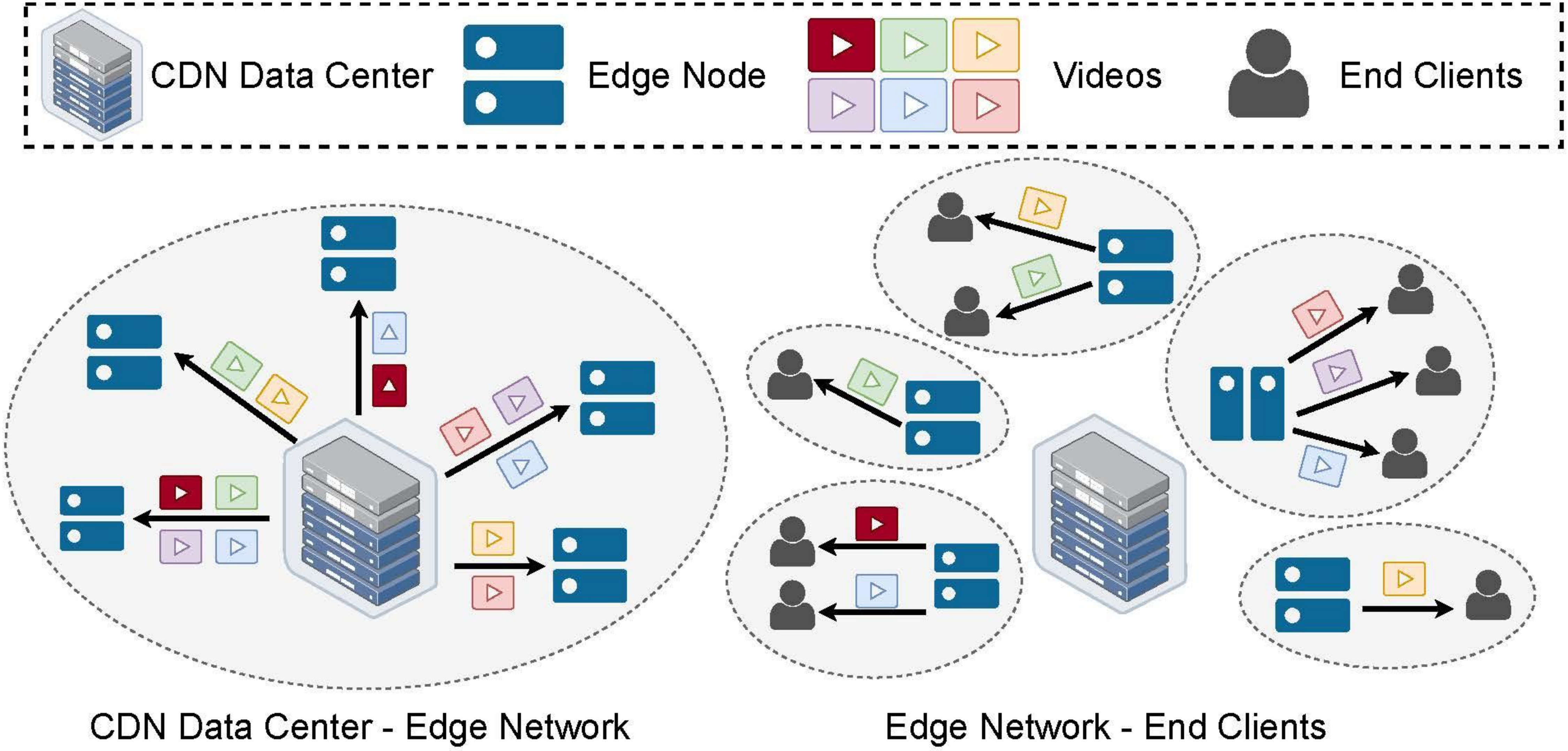
KEPC-Push: A Knowledge-Enhanced Proactive Content Push Strategy for Edge-Assisted Video Feed Streaming

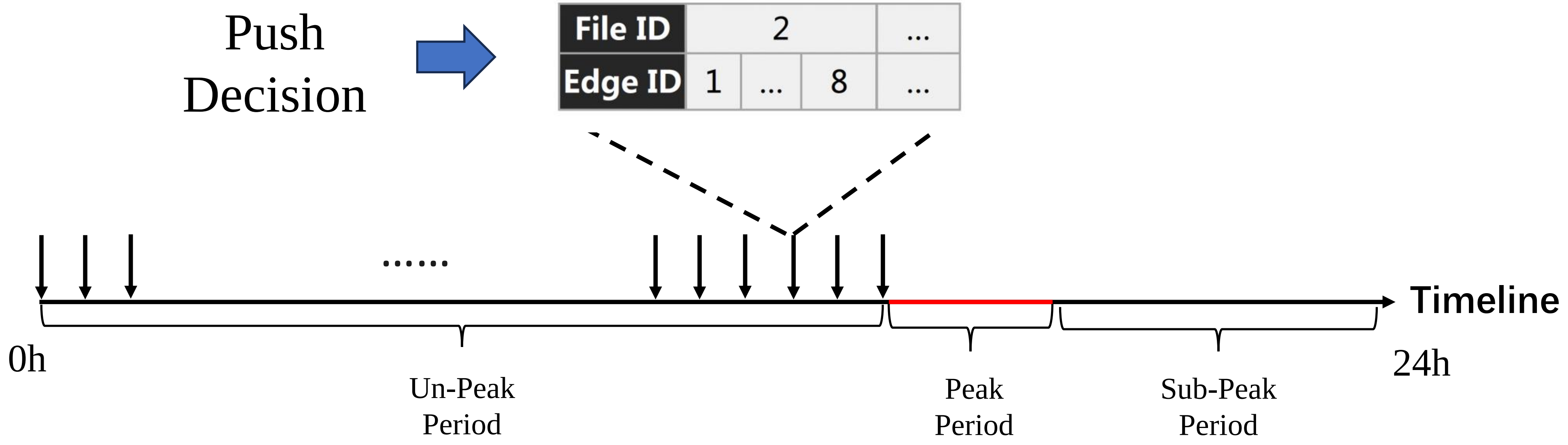
Ziwen Ye, Qing Li, Chunyu Qiao, Xiaoteng Ma, Yong Jiang, Qian Ma, Shengbin Meng, Zhenhui Yuan, Zili Meng

Background: Edge-assisted Video Delivery



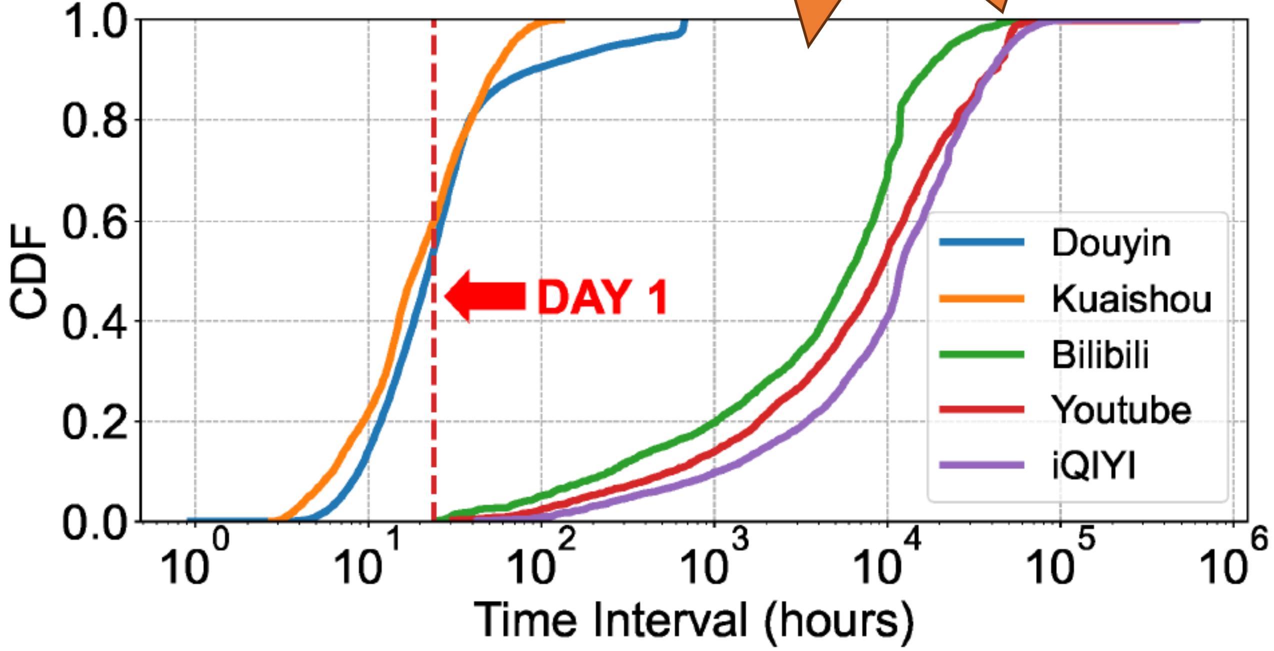
Background: Proactive Push Mechanism

- The bandwidth in an edge-assisted video streaming system is charged by the peak-time CDN bandwidth metering (e.g. 95th percentile bandwidth metering);
- Predict which video files will get high peak-period popularity based on historical access information;
- Proactively push popular peak-period files in advance during off-peak hours.



Background: Video Feed Streaming Service

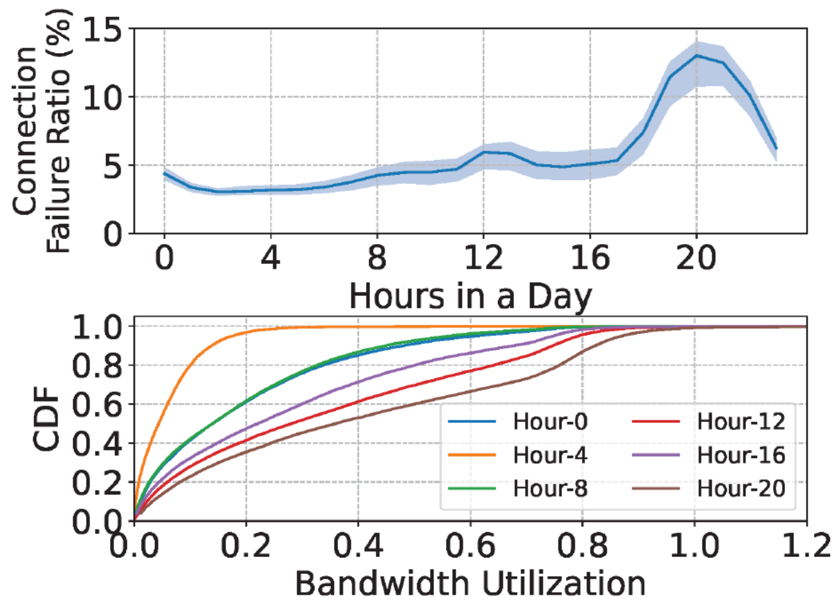
Lack of historical information!

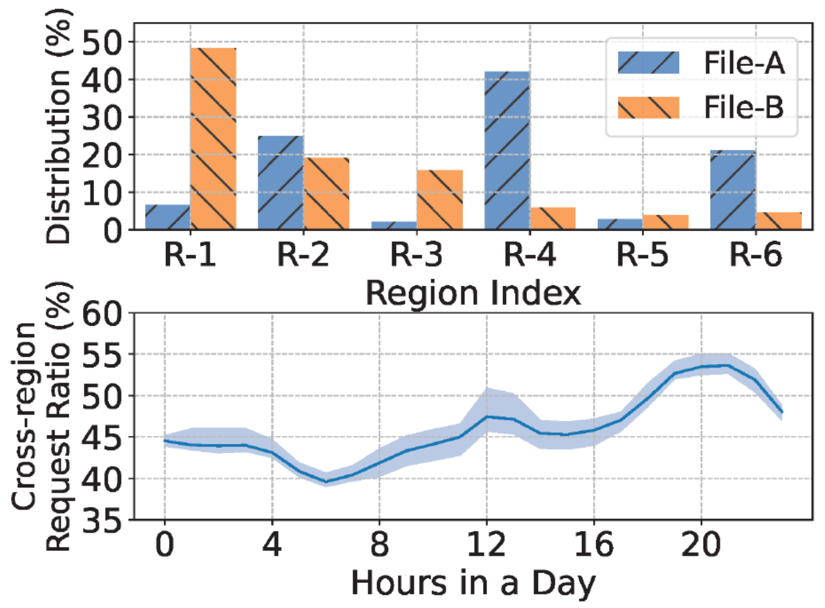


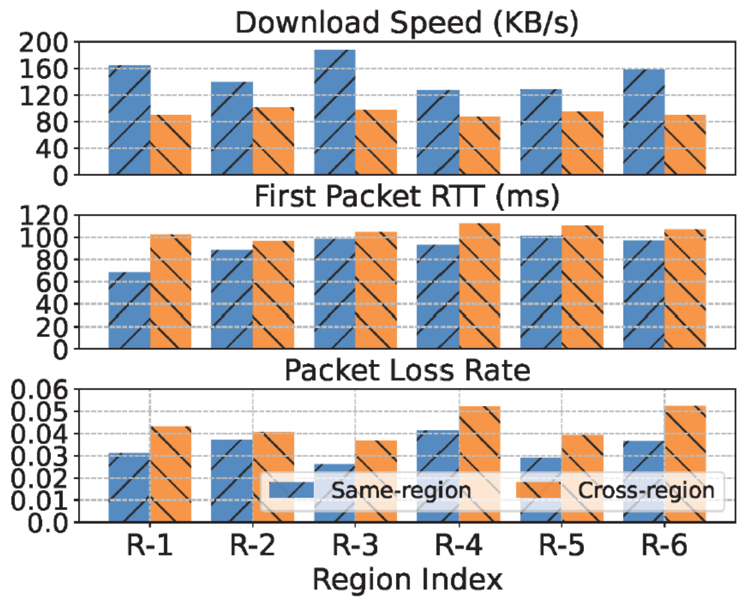
The median time between video upload by the creator and distribution to the users is around 20 hours in the video feed streaming, which is not enough to even gather the historical information from one day before, while for Video-On-Demand (VoD), the time could be days to weeks.

Background: Inaccurate Proactive Video Push

Lack of historical information ➡ Inaccurate Popularity Prediction ➡ Inaccurate Proactive Video Push







Load Imbalance Problem

Allocation Imprecision Problem

- ➢ Load imbalance limits edge network’s performance.
- ➢ Imprecise resource allocation degrades user’s experience (QoS).

Challenges & Solutions

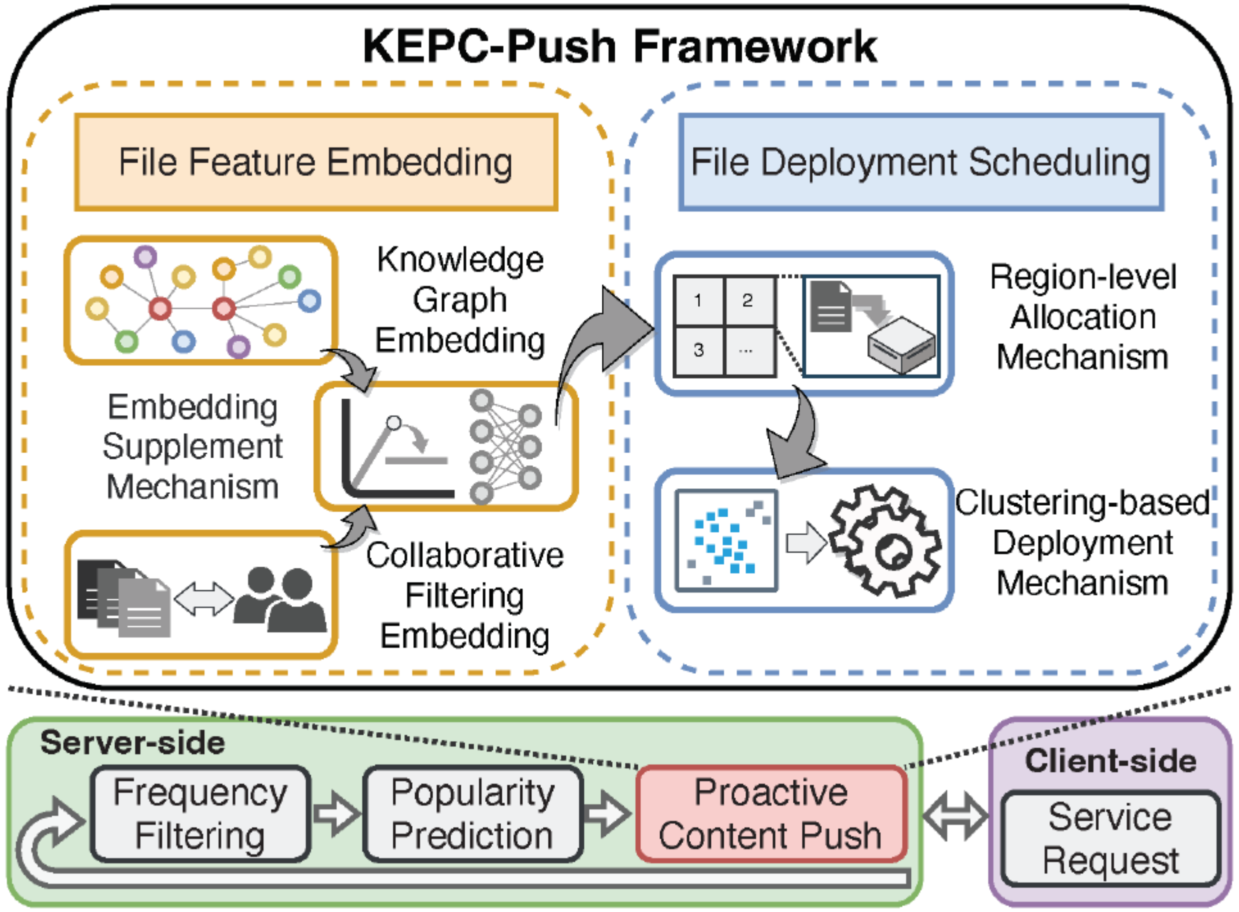
Challenges

- ✓ How to achieve load-balance proactive push for heterogeneous edge nodes without sufficient historical information.
- ✓ How to ensure that the scheduling scheme can satisfy systematic requirements for large-scale video streaming services.

Solutions： KEPC-Push (Knowledge-Enhanced Proactive Content Push)

- ✓ Our main insight is that knowledge derived from the content features of video files (such as video category and duration) can 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.
- ✓ A lightweight scheduling scheme is also necessary to ensure that the proactive content push can run in real-time in the large-scale system.

Design Overview


KEPC-Push Framework
File Feature Embedding
Knowledge Graph Embedding
Embedding Supplement Mechanism
Collaborative Filtering Embedding
File Deployment Scheduling
1
2
3
...
Region-level Allocation Mechanism
Clustering-based Deployment Mechanism
Server-side
Frequency Filtering
Popularity Prediction
Proactive Content Push
Client-side
Service Request

➢ **File feature embedding (FFE) module**

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.

➢ **File deployment scheduling (FDS) module**

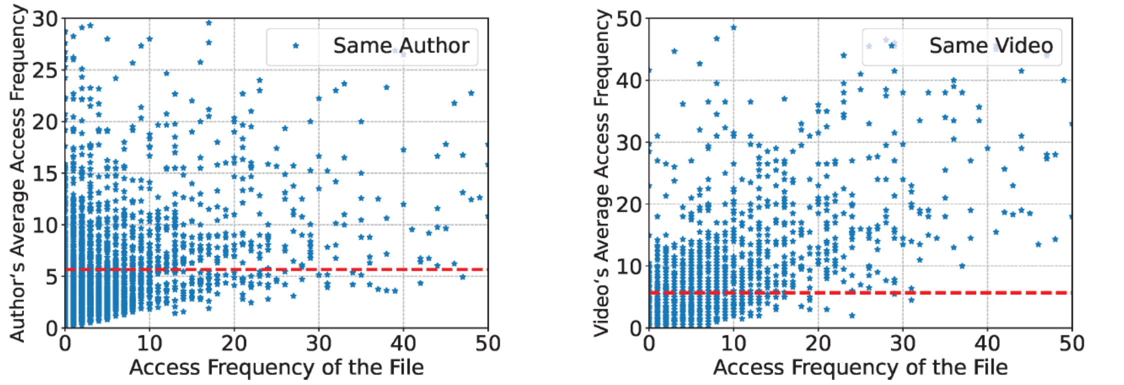
The FDS module allocates resources at the regional level and reduces the complexity of the proactive push problem with the help of a clustering-based deployment mechanism.

FFE Module – Content Feature Selection

Objective: Which content features are related to popularity trends?

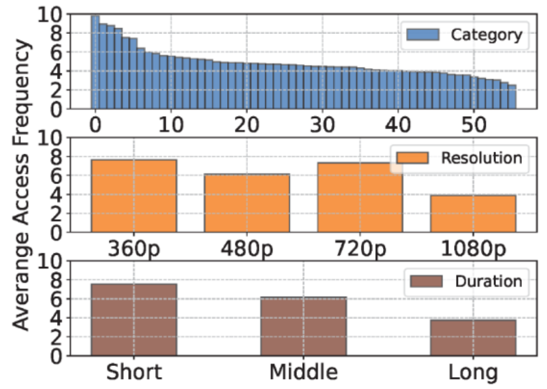
- When an author (or a video) becomes popular, other files created by that author (or belonging to that video) will also be requested more frequently.
- The popularity of files can skew significantly in some characteristics, such as category, resolution and duration.
- The files published at the same time tend to reach peak popularity at similar times, so they should be stored separately in the edge network.

Video files with <u>similar content features</u> should be <u>stored separately on different edge nodes</u> to avoid excessively high or low workloads in a short period of time.

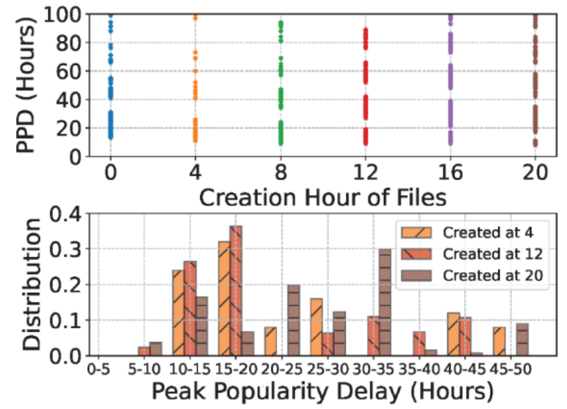


(a) Created by the same author (b) Belonging to the same video

The popularity radiation effect



The popularity skewness effect



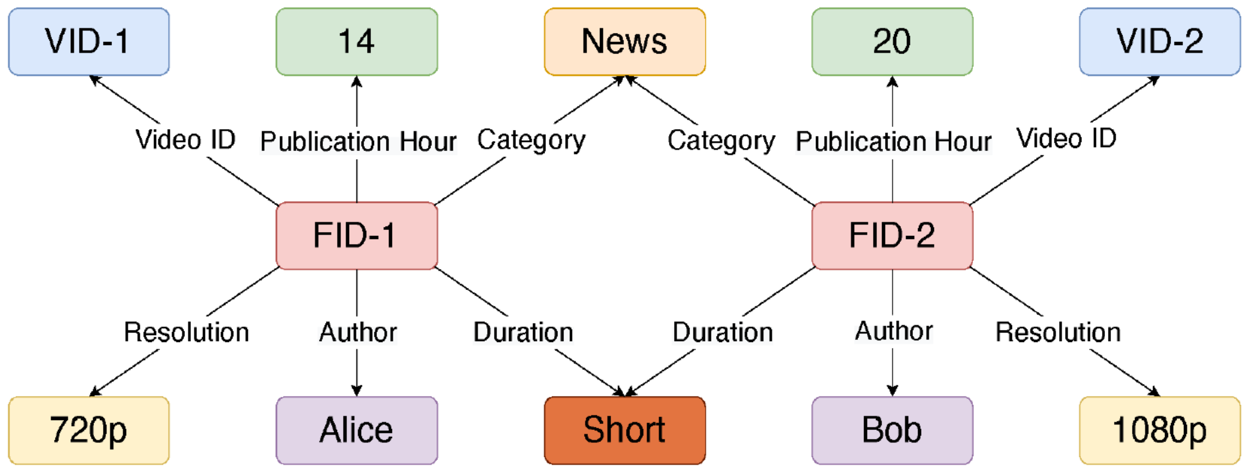
The peak popularity delay effect

FFE Module – Knowledge Graph Embedding

Objective: How to assess the similarity of files on the selected content features?

- Considering the success of knowledge graphs in understanding mutual relations, we utilize the knowledge graph embedding method to learn representation vectors for different items.
- We choose six content features (namely Video ID, Author, Category, Resolution, Duration and Publication Hour) to construct the knowledge graph based on the TransR model.

An illustration of video file knowledge graph



TransR Model

- Score function:

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

- Margin-based score function:

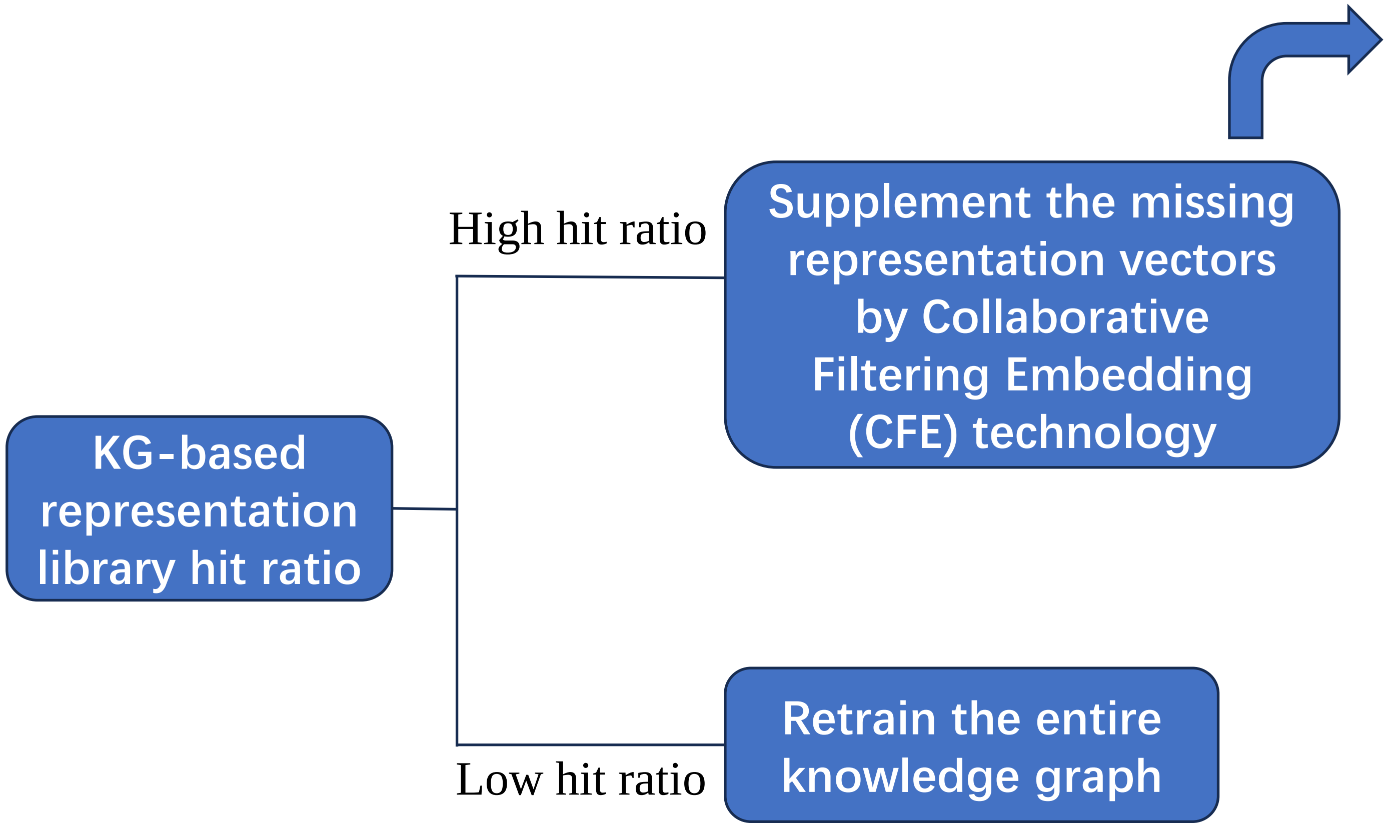
$$L = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} \max\left(0, f_r(h,t) + \gamma - f_r\left(h',t'\right)\right),$$

- After the training process, we can get 128-dimensional representation vectors of all video files. And the Euclidean distance between two vectors is inversely proportional to the popularity correlation of two files.

FFE Module – Knowledge Graph Embedding

Objective: How to handle constantly published new files during the push process?

- 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.



KG-based representation library hit ratio
High hit ratio
Supplement the missing representation vectors by Collaborative Filtering Embedding (CFE) technology
Low hit ratio
Retrain the entire knowledge graph

Step-1: Collect all user requests within 4 hours and count the frequency of video access;

Step-2: Screen out video files that are accessed less frequently than a preset threshold;

Step-3: Group requests into separate collections based on user differences;

Step-4: Use the item2vector method and skip-gram model to train the CF-based representation vector library from the user access set;

Step-5: Use the similarity of items in the CF-based representation vector library to supplement the KG-based representation library.

FDS Module – Region-level Resource Allocation

Objective: How many replicas to push for each video and how much replaceable cache size each edge node should provide before each round of push?

① Use XGBoost model to forecast the peak-period popularity of videos in each region;

$$v_{f,g} \leftarrow XGBoostPopularityPredict()$$

② Determine the number of to-be-pushed replicas in each region based on the predicted results and network cache information;

$$n_{f,g} = max\{\eta \cdot v_{f,g} - \sum_{e \in \mathcal{E}_g} x_{f,e}, 0\}$$

③ The replaceable cache size of each edge device is determined based on static storage size and dynamic bandwidth utilization.

```
if $U_e \geq \mu$ then
    $RC_e = 0$
else
    $RC_e = S_g \cdot \frac{C_e \cdot (\mu - U_e)}{\sum_{e \in \mathcal{E}_g} C_e \cdot (\mu - U_e)^+}$
```

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
            $v_{f,g} \leftarrow XGBoostPopularityPredict()$   ①
            $n_{f,g} = max\{\eta \cdot v_{f,g} - \sum_{e \in \mathcal{E}_g} x_{f,e}, 0\}$   ②
        // 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
                $RC_e = S_g \cdot \frac{C_e \cdot (\mu - U_e)}{\sum_{e \in \mathcal{E}_g} C_e \cdot (\mu - U_e)^+}$   ③
    $\mathcal{RC} = \{RC_e \mid e \in \mathcal{E}\}, \mathcal{N} = \{n_{f,g} \mid g \in \mathcal{G}, f \in \mathcal{L}_g\}$
    return $\mathcal{RC}, \mathcal{N}$
```

FDS Module – Load-Balanced Proactive Push Decision

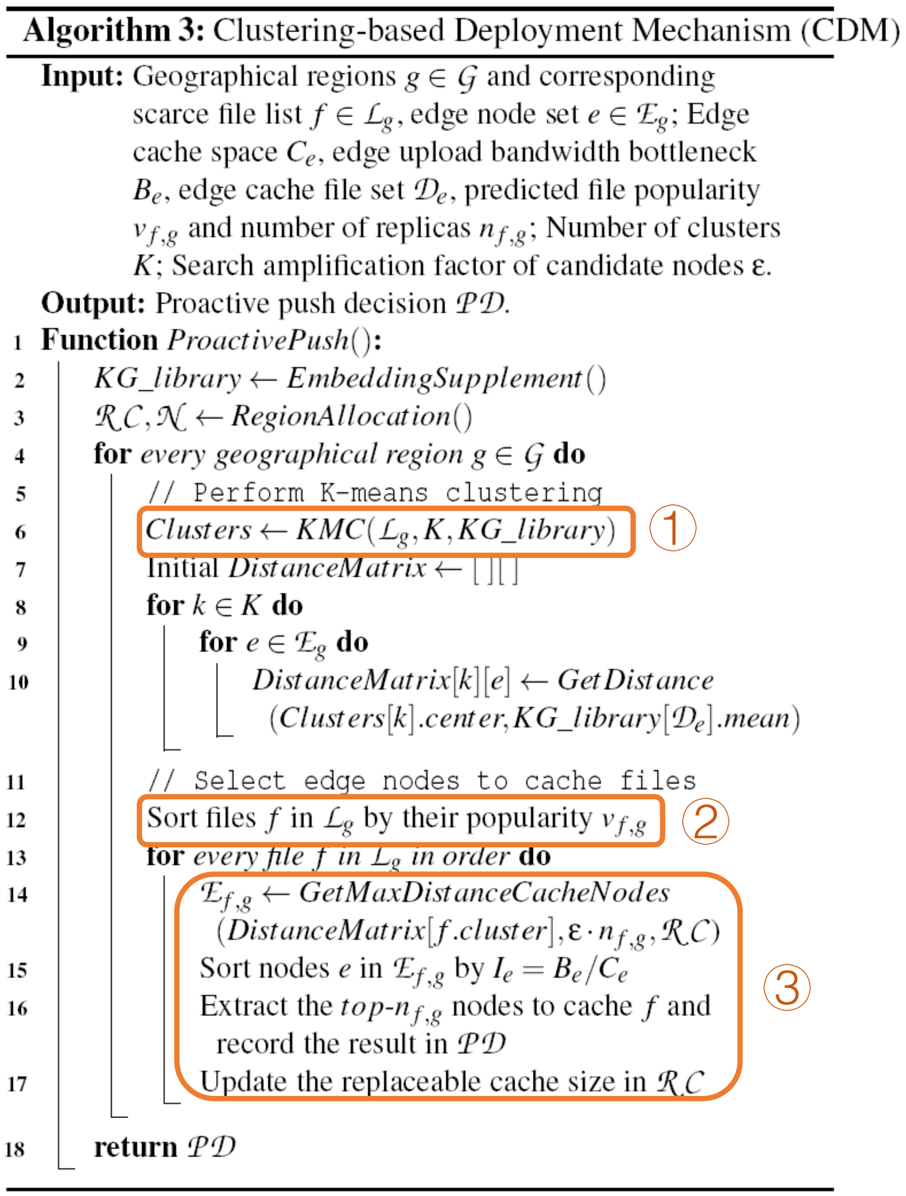
Objective: How to match the to-be-pushed replica with the edge node based on the obtained representation library in a lightweight?

① Use K-means clustering algorithm to cluster video files based on their representation vectors;

② Complete the video push decision according to the popularity priority principle;

③ Select the appropriate cache device based on the distance between video vectors and the performance of edge nodes to cache files.

Clustering-based Deployment Mechanism successfully reduces computational complexity from $O(|L_g| \cdot |E_g| \cdot |D_e| \cdot d^2)$ to $O(K \cdot |E_g| \cdot d^2)$



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 $\varepsilon$.
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
        $Clusters \leftarrow KMC(\mathcal{L}_g, K, KG\_library)$   ①
        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 $v_{f,g}$   ②
        for every file $f$ in $\mathcal{L}_g$ in order do
            $\mathcal{E}_{f,g} \leftarrow GetMaxDistanceCacheNodes$   ③
              $(DistanceMatrix[f.cluster], \varepsilon \cdot n_{f,g}, \mathcal{RC})$
            Sort nodes $e$ in $\mathcal{E}_{f,g}$ by $I_e = B_e / C_e$
            Extract the top-$n_{f,g}$ nodes to cache $f$ and
              record the result in $\mathcal{PD}$
            Update the replaceable cache size in $\mathcal{RC}$
    return $\mathcal{PD}$
```

Experimental Setup

✓ **Dataset**

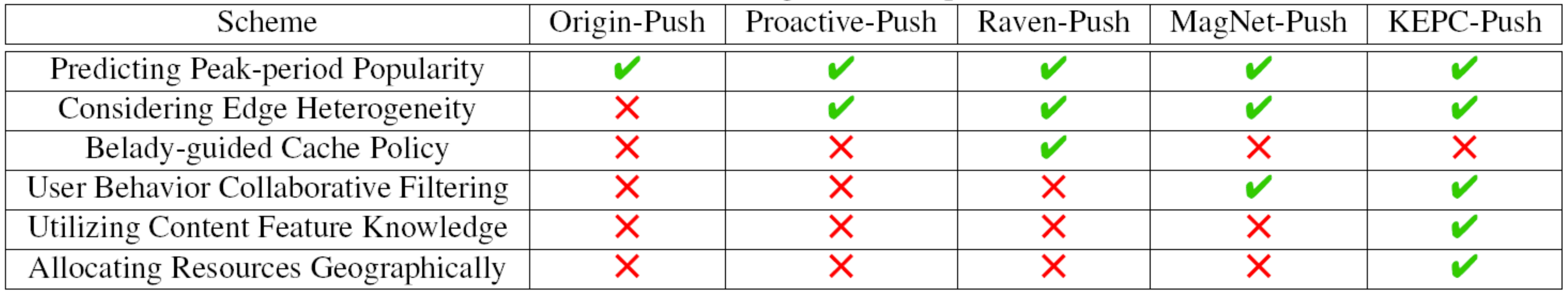
- From Douyin (a leading online video-sharing platform)
- Contains client requests, file features and device information;
- 60 million client traces spanning three weeks in September 2022;

✓ **Implementation**

- Set 5 minutes as a decision-making cycle;
- Video push period: 0:00 ~ 20:00;

✓ **Baseline Algorithms**

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	✘	✘	✘	✘	✔

✓ **Evaluation Metrics**

- **Bandwidth Cost (Goal-1):** Bandwidth Saving Ratio;
- **User Experience (Goal-2):** Average Download Speed, Packet Loss Rate, RTT / Connection Failure Ratio;
- **Load Imbalance Problem (Challenge-1):** Ratio of Overloaded Nodes / Overall Bandwidth Utilization Allocation / Number of Concurrent Requests;
- **Imprecision Problem (Challenge-2):** Cross-region Request Ratio **/** Number of Pushed Replicas

Performance Evaluation

Improvements in CDN Bandwidth Consumption and User Experience

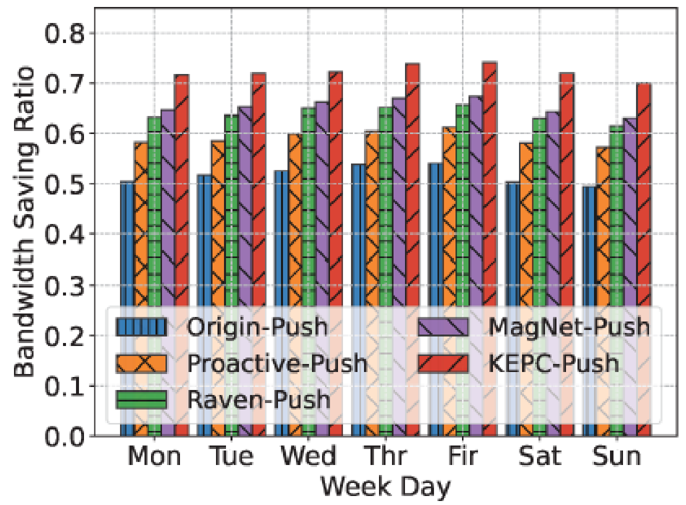
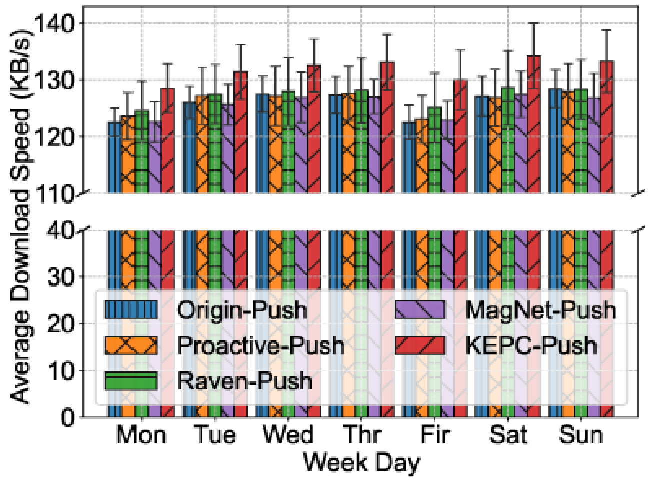
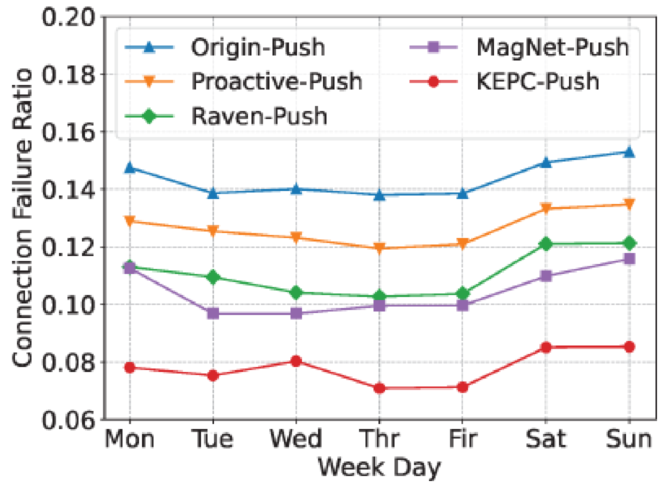


Figure 14: Evaluation of the evening peak-period CDN bandwidth saving



(a) Average Download Speed



(b) Connection Failure Ratio

Figure 15: Evaluation of the user experience improvement

- KEPC-Push saves the most CDN bandwidth, which indicates that KEPC-Push achieves excellent bandwidth saving performance through the knowledge-enhanced proactive content push strategy.
- KEPC-Push significantly improves the average download speed of user requests and reduces connection failure ratio. The improvement not only to ensure the smoothness of user viewing, but also to avoid requests being redirected to CDN due to insufficient transmission capability

Performance Evaluation

Improvements in Load Balancing Performance

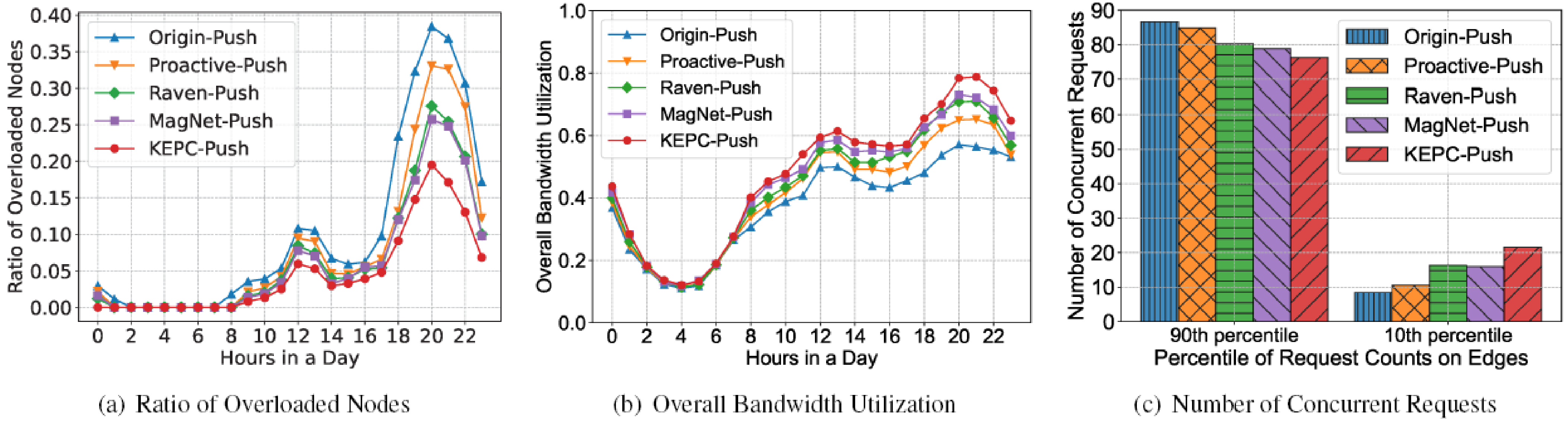
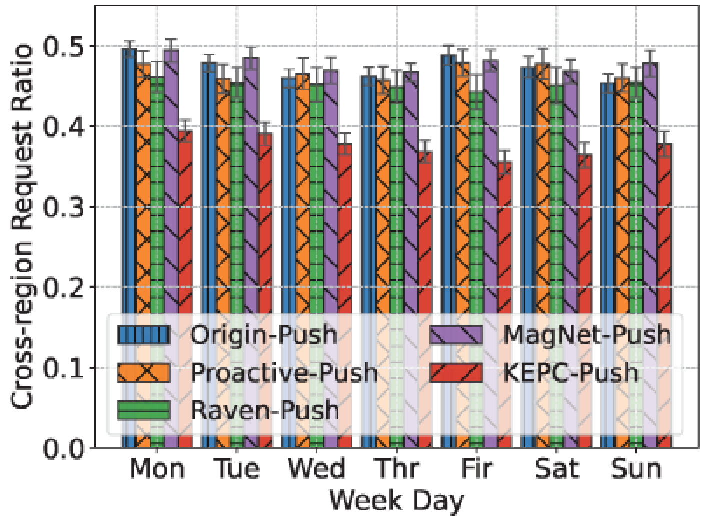


Figure 16: Evaluation of the load balancing performance in the edge network

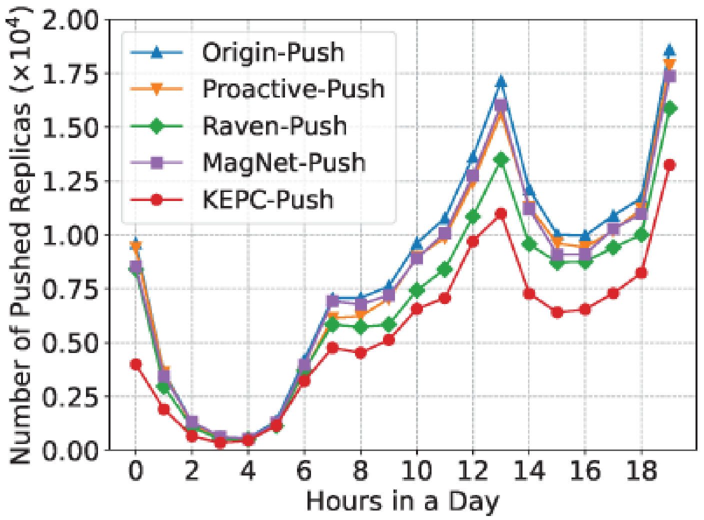
- ➢ KEPC-Push significantly reduces the ratio of overloaded nodes, especially during the evening peak period;
- ➢ The overall bandwidth utilization of KEPC-Push is higher than other algorithms during the whole day;
- ➢ KEPC-Push achieves the transfer of access traffic from high-load nodes to low-load nodes through appropriate cache configuration

Performance Evaluation

Improvements in File Allocation and Cache Hit Performance



(a) Cross-Region Request Ratio



(b) Number of Pushed Replicas

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

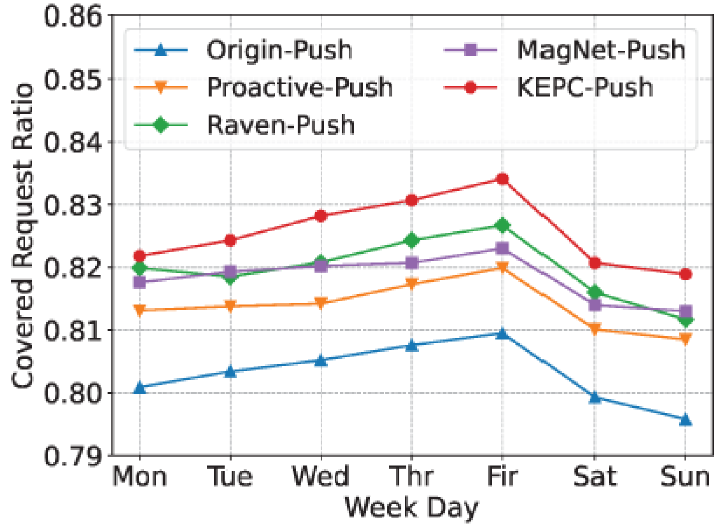


Figure 18: Evaluation of the evening peak-period cache hit performance

- ➢ KEPC-Push 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;
- ➢ 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;
- ➢ KEPC-Push's cache hit performance consistently outperforms other algorithms in different situations.

Performance Evaluation

Ability to handle potential popularity burst

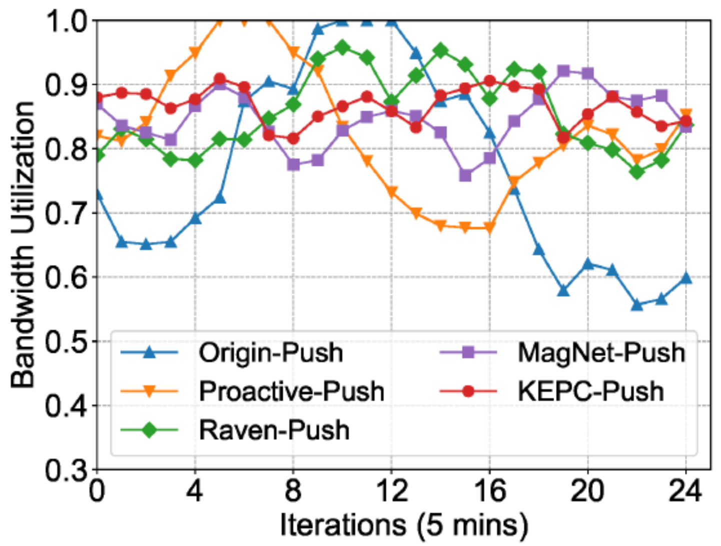


Figure 19: Temporal variation of evening peak-period bandwidth utilization

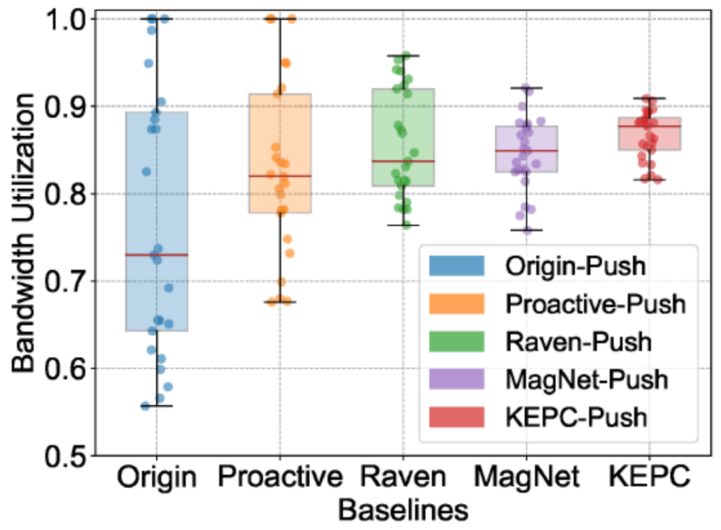


Figure 20: Boxplot graph of evening peak-period bandwidth utilization

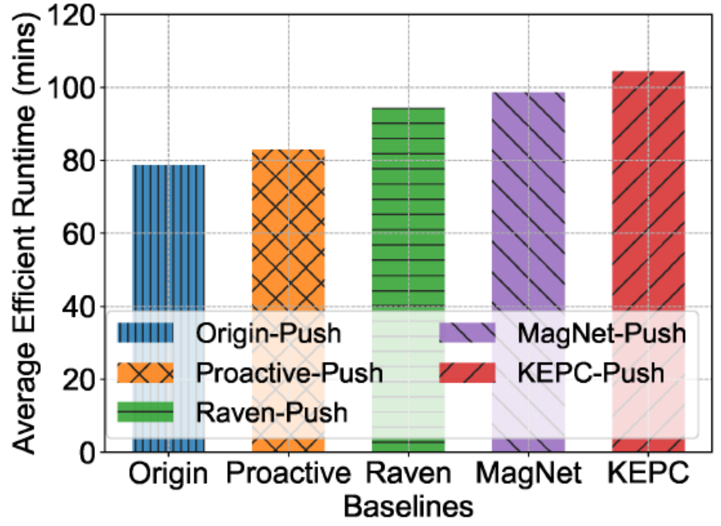


Figure 21: The average efficient runtime of edge nodes during the evening peak period

- KEPC-Push's bandwidth utilization fluctuates less and its performance is more stable, which is what we expect;
- KEPC-Push can handle the potential popularity bursts and maintain a stable and higher peak-period bandwidth utilization, thus achieving better performance;
- The more stable performance prevents edge nodes from experiencing overload or idle states, bringing more efficient runtime.

Performance Evaluation

Ablation Experimental Evaluation

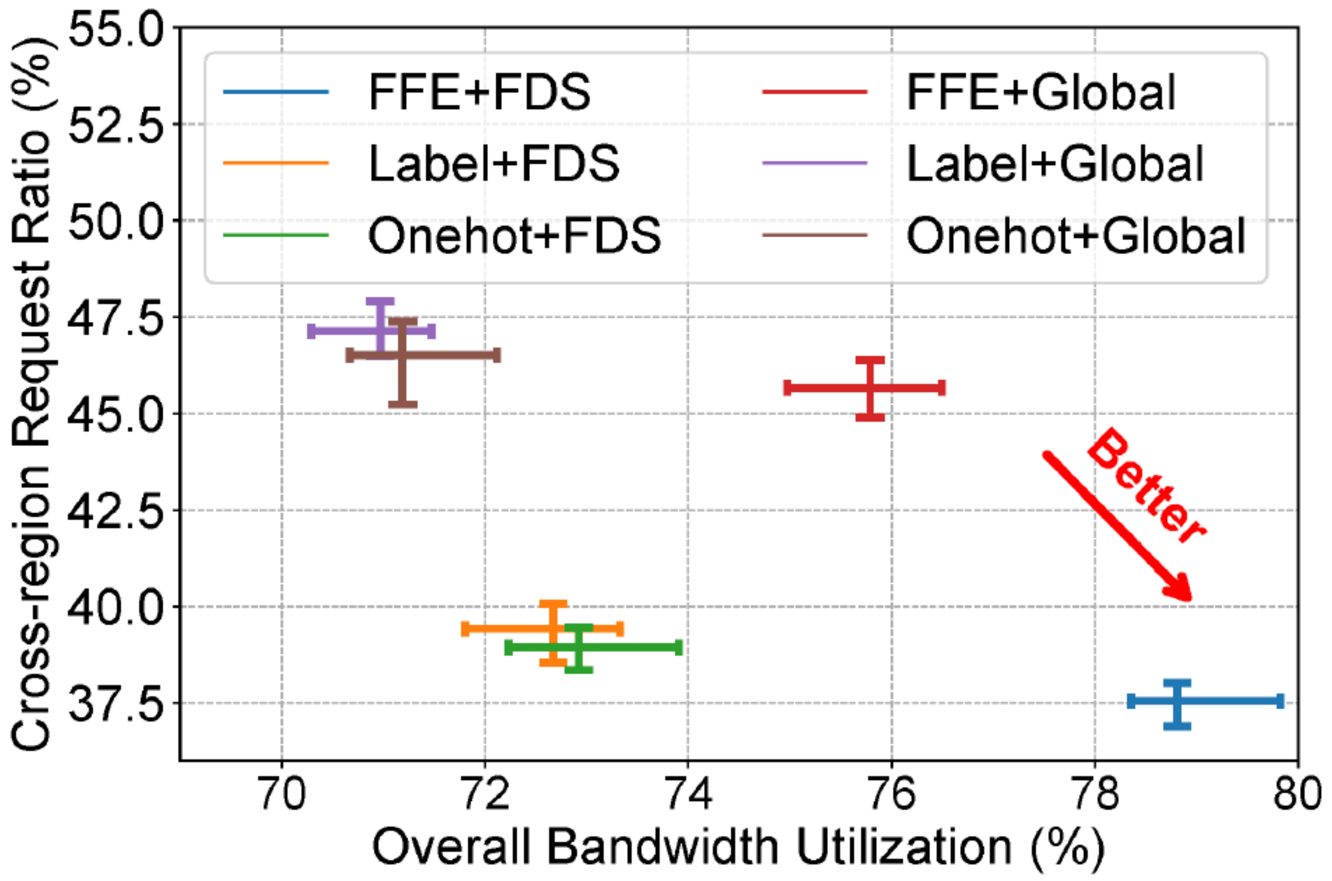


Figure 22: Ablation experiment of modules in KEPC-Push

- 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;

Summary

- ✓ KEPC-Push, a knowledge-enhanced proactive content push strategy
- ✓ KEPC-Push innovatively employs knowledge graphs to determine the popularity correlation among similar videos (with similar authors, contents, length, etc.) and pushes content based on this guidance.
- ✓ 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 simulation from real data of Douyin platform.

Thank You