CRATES: A Cache Replacement Algorithm for Low Access Frequency Period in Edge Server

Pengmiao Li[†], Yuchao Zhang[†*], Huahai Zhang[†], Wendong Wang[†], Ke Xu[§], Zhili Zhang[‡],
[†]*State key Laboratory of Networking and Switching Technology,*
Beijing University of Posts and Telecommunications, Beijing, China
Email:{pengmiaoli, yczhang, zhanghuahai, wdwang}@bupt.edu.cn
[§]*Tsinghua University, Beijing, China.* Email:xuke@tsinghua.edu.cn
[‡] *University of Minnesota, Minneapolis, US.* Email:zhzhang@cs.umn.edu

***Abstract*—In recent years, with the maturity of 5G and Internet of Things technologies, the traffic in mobile network is growing explosively. To reduce the burden of cloud data centers and CDN network, edge servers that are closer to users are widely deployed, caching hot contents and providing higher Quality of Service (QoS) by shortening access latency. Storage resources on edge servers are much limited compared with CDN servers, so the research on cache replacement strategy of edge servers is critical to edge computing and storage area. Many efforts have been made to improve caching performance on edge servers. Existing caching strategies only focus on the high access frequency period to solve the caching problem, they ignore low access frequency period with two characteristics, including that hot contents are difficult to predict and hot topics usually change unstably, which makes it inefficient to improve the hit rate on edge servers.**

In this paper, we deeply analyzed the real traces from ChuangCache and found some specific user groups are playing more important roles than general users during low access frequency period, and the contents accessed by these specific user groups have a much higher possibility to become hot contents. Therefore, we firstly classify such users to core users, and treat others as common users. Then we adopt the principal component analysis algorithm to analyze the relationship between hot contents and core users. On this basis, we finally propose a hot contents pre-cache protection mechanism, which is a significant part of our cache replacement algorithm *CRATES*. To improve *CRATES*'s efficiency, we extract key part of historical data by designing a sliding window method. Through a series of experiments using real application data, we demonstrate that *CRATES* reaches about 98% in caching hit rate and outperforms the state-of-the-art algorithm LRB by 1.4X.

***Index Terms*—Edge caching, hot contents, core user, and hit rate.**

I. Introduction

In recent years, with the maturity of 5G and Internet of Things technologies, the traffic in mobile network is growing explosively. According to the report [1], the number of worldwide phone users was 5.22 billion in 2020, which introduces high requirements on content storage. To reduce the burden of cloud data centers and CDN networks, edge servers that are closer to users are widely deployed, caching hot contents and providing higher Quality of Service (QoS) by shortening access latency. However, storage resources on edge servers are much limited compared with CDN servers, so the research on cache replacement strategy of edge servers is critical to edge computing and storage area.

Many efforts have been made to improve caching performance on edge servers. LeCaR [2] uses the machine learning technology based on content popularity for caching content. LRB [3] is a new method for caching in CDNs that adopts machine learning to approximate the Belady MIN (oracle) algorithm. Although these works increase the hitting rate on edge servers during busy hours with high access frequency, they fail in low access frequency period, which has the following two characteristics. Hot topics usually change unstably, and are difficult to predict. Existing caching strategies only focus on the high access frequency period to solve the caching problem, but they ignore the low access frequency periodwith two characteristics above, making them inefficient to improve hit rate on edge servers.

To improve caching hit rate during low access frequency period, we deeply analyzed the real traces from ChuangCache [4] and found an essential observation (see Sec. III). That is, some specific user groups are playing more important roles than general users during these periods, and the contents accessed by these specific user groups have a much higher possibility to become hot contents. This observation provides us a new perspective to review the above caching problem, from the perspective of particular user groups. However, distinguishing hot contents from these user groups is not easy, there are two challenges as follows.

- How to capture the relationship between hot contents and special user groups using historical data?
- How to design a strategy with the above relationship to improve hit rate on edge server?

In this paper, we firstly classified the users with high access frequency and broader distribution on the time dimension to core users, and treated others as common users. Then we adopt the principal component analysis algorithm to analyze the relationship between hot contents and core users. On this basis, we finally propose a hot contents pre-cache protection mechanism, which can easily gain hot contents from history in static cache area, and cache & evict contents with real-time by the relationship between hot contents and core users in the

[*] Yuchao Zhang (yczhang@bupt.edu.cn) is the corresponding author.

surplus area. It is a significant part of our cache replacement algorithm *CRATES*. To improve *CRATES* efficiency, we extract key part of historical data by design a sliding window method. Through a series of experiments using real application data, we demonstrate that *CRATES* reaches about 98% in caching hit rate and outperforms the state-of-the-art algorithm LRB by 1.4X. The main contributions of this paper are listed as follows:

- We disclose the relationship between content popularity and user groups during low access frequency period in edge servers.
- We propose a caching algorithm called *CRATES* by capturing the above relationship and designing an item pre-cache protection mechanism.
- We conduct a series of experiments using real industry data and demonstrate the efficiency of *CRATES*.

The remainder of the paper is organized as follows. In the next section, we introduce the related work about replacement strategy. In section III, we deeply analysis the real traces and find an essential observation. In section IV, the design of architecture is presented. Section V conducts experiments by detailed introduction and analysis of the results. And our conclusions are presented in Section VII.

II. Related Work

A. Base Strategy

1) Traditonal replacement: Many traditional replacement algorithms have been proposed in different scenarios, such as First Input First Output (FIFO), Least Recently Used (LRU), Least Frequently Used (LFU) and Greedy-Dual-Size-Frequency (GDSF) [5]–[8]. These traditional caching algorithms with low complexity are easy to be implement and have been widely used. The most commonly deployed caching replacement algorithms by some CDN providers, such as LRU, LFU, or their simple variants [9], are simple but do not explicitly consider the future popularity of content when making caching decisions.

2) popularity prediction: Significant efforts have been devoted to exploring item popularity prediction due to the potential business value [10]. [11] proposed a novel popularity prediction method SABER for online popularity prediction of video segments, which classified video segments into head segments and tail segments according to their positions in videos and proposed each class of segments as a specific offline predictor. [12] proposed a dynamic popularity prediction model that includes topic migration popularity for news comments and extended the reinforced Poisson process model by introducing the weak tie theory and the competitive matrix. [13] proposed a multi-head attention-based popularity prediction model to predict the content popularity, whose input includes historical popularity, social relationships, and geographic features, to better extract the multivariate features of popularity and reduce the prediction error. [14] proposed a graph convolution neural-based video popularity prediction algorithm called GraphInf, which clusters the countless short videos by region and formulates the problem in a graph-based way, thus addressing the explosive quantity problem.

The forecasting popularity of online content has been extensively studied that can provide support for caching.

B. Recent Strategy

On the studies of cache replacement strategy, researchers have proposed many solutions in different perspectives, such as location, size, and new scenarios, et.al, which are mainly designed by content popularity. Li et al. [15] proposed a pop-Caching strategy for social networks. Instead of predicting the individual popularity of each content, the algorithm assumes that the content popularity is similar when the context features are similar, and uses four historical hits as the current context features. To solve the underlying 0-1 Knapsack problem, size-weighted popularity (SWP)-based caching framework [16] was proposed, where both content popularity and content size are taken into account when determining the contents to be cached. Yang et al. [17] used the location features of edge nodes to represent the user preferences of that location and proposed an online prediction algorithm. Instead of trying to find a single dominating caching policy for all the caching scenarios, Zong et al. [18] employed an ensemble of constituent caching policies and adaptively selected the best-performing policy to control the cache. [2] used the machine learning technology of regret minimization combines LFU and LRU based on content popularity to improve on existing cache replacement strategies in small storage sources.

The explosive growth of Internet data and the emergence of new networks, such as edge computing, IoT, and 5G network, have put forward higher requirements for cache strategies. Therefore, the research work in recent years mainly proposes corresponding cache optimization algorithms for new network scenarios, especially in the social network. [3] focused on caching in CDNs and proposed a new approach called LRB that uses machine learning to approximate the Belady MIN (oracle) algorithm. For better adaptation to the time-varying popularity patterns, Forecast-Based cache replacement policies had been proposed for mobile video streaming [19]. In recent years there has been a rapid increase of short video traffic in CDN, [20] presented AutoSight, a distributed edge caching system for short video network, which consists of two main components including the CoStore predictor and a caching engine Viewfinder that adjust automatically future horizon according to video life span.

These solutions increased the hitting rate on edge servers during busy hours with high access frequency. They only focus on the high access frequency period to solve the caching problem, but they ignore low access frequency periodwith two characteristics that will be introduced detailedly in section III, making them inefficient to improve hit rate on edge servers.

III. Motivation

A. Characteristics of hot contents

- **Hot topics usually change unstably.**
  It is widely believed that hot content varies with the number of requests and contents, but this is inaccurate at some periods, as can be seen in in Figure 1 and line *different 1* of Figure 2. Figure 1 illustrates the number

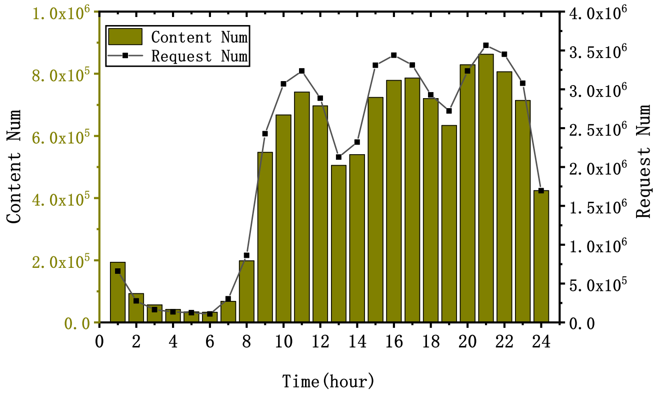


Figure 1: The number of requests and contents in every hour.

of requests & contents, and that line *different 1* is the different proportions of hot contents in two continuous unit times. Hour 1-8 are set to low access frequency period when the request number is significantly reduced than others. In hour 6, the number of requests & contents is the lowest one and the different proportions of *different 1* is higher than other hours of low access frequency period. However, hot content varies with the number of requests and content in other periods.

Therefore, in low access frequency period, the change of hot topics is often more unstable than that in other periods.

- **Hot contents are difficult to predict.**
  It is not difficult to understand that the greater the differences between the frequency of requests for different contents, the easier it is to select content as hot content. The special is the differences between the frequency of requests for different contents are practically indistinguishable in some periods. In Figure 3, the T1 (see red line) and T2 (see black line) are the request number from the top 400 content in low access frequency period and other time respectively. We can find that the request number is 2 for nearly 50 contents after Id 50 in T1, so it is difficult to select content as future hot contents in the same access frequency. However, the differences between the frequency of different content requested are greater in T2 than T1, which means hot contents are difficult to predict in low access frequency period than the other one.

Due to the above characteristics, the existing strategies are inefficient enough to improve the hit rate on edge servers. To improve hit rate during low access frequency period, we deeply analysis the real traces from ChuangCache and find an essential observation about users who is mainly reason to generate the access traces for any period and any applications. Therefore, the next subsection taps the potential to tackle the above problems by analyzing the users' features in real traces.

B. Potential of users-based attribution.

In some applications, the number of requests is not only high, but the number of users is also high. Table I is the

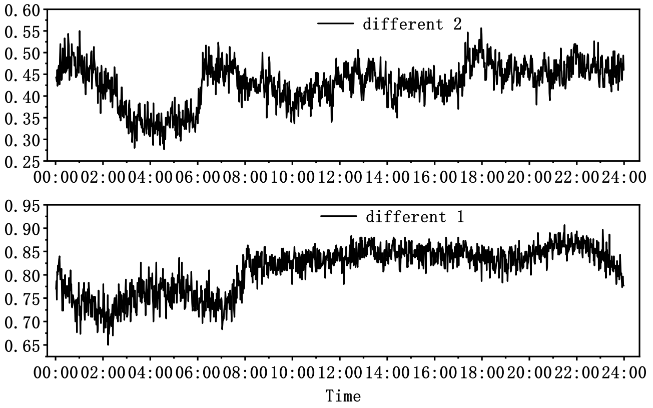


Figure 2: The difference of future hot contents with hot contents from history and special users groups. (a) *Different 1* is the different proportions of hot contents in two continuous unit times. (b) *Different 2* is the percentage of hot contents for special user groups that are different from the overall hot contents.

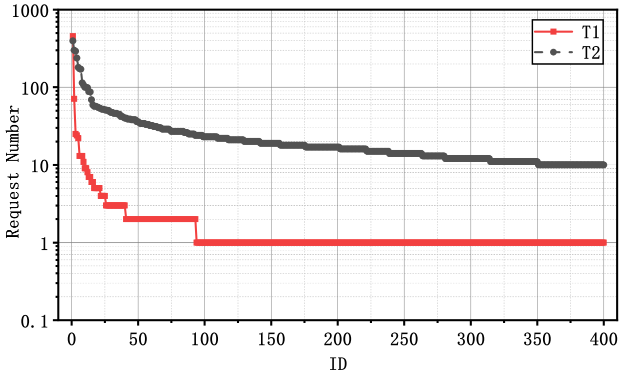


Figure 3: The access number of contents in different period time.

information about users from real trace, where the user number is reach 277524, the maximum number of user requested is 425976 and the minimum one is 1 that means the request number by different users is uneven. More specially, the number of specific user groups are tiny but with a high request frequency, such as the request number range [100000,425976] with 759 users, where the user ratio is 0.27%, and the request ratio is as high as 34.33%. We assume that these users play an important role in cache replacement for improving hit rates. Therefore, we capture the characteristics of specific user groups by analyzing the real traces are as follows.

- **More important**.
  Specific user groups are throughout the application's life cycle usually. Taking one day of the trace as an example, we plotted the request information about special user groups and content shown in Figure 4. The red line is the percentage of the user number of special user groups among all users, and the black line is the average number

Table I: Information about users.

request number range	***user number***	***user ratio***	***request ratio***
[1,10)	86352	35.25 %	0.90%
[10,100)	136502	46.24	9.34 %
[100,1000)	48571	16.45	29.18 %
[1000,10000)	5304	1.80 %	26.25 %
[100000,425976]	795	0.27%	34.33%
[1,425976]	277524	100%	100%

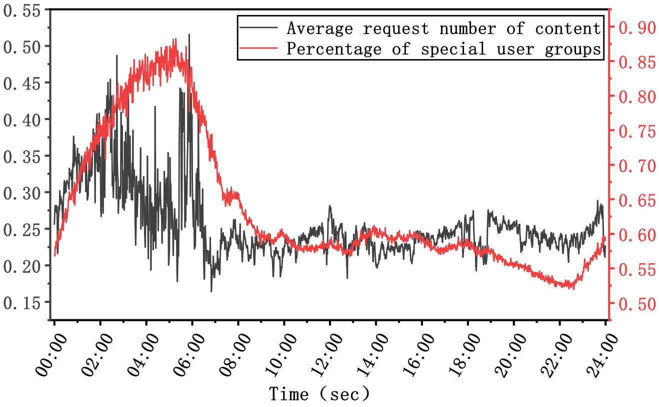


Figure 4: The request information of Special user groups and content.

of content requested by all users. As seen in red and black line, specific user groups are distributed differently throughout the day, and their requests to contents are more concentrated than other user groups during low access frequency period, which is attributed to their higher average number of content requests than in other periods. Therefore, it's obvious that specific user groups are playing a more important role than general users during low access frequency period.

- **Much higher possibility to become hot contents**. Content requested by a large number of users will become hot content, so specific user groups also have their hot ones with high request frequency. The hot contents of specific user groups are more similar in low access frequency period than other's periods by comparing the differences of hot contents from specific user groups and whole user that shown in *different 2* of Figure 2. For example, the *different 2* value of hour 4 and hour 18 are nearly 0.3 and 0.85 respectively, and the latter different value (in other periods) is far higher than the former one (in low access frequency period). This means that specific user groups' hot contents at low access frequency period are more similar to the overall hot content than at other periods. So, the contents accessed by these specific user groups have a much higher possibility to become hot content during low access frequency period.

Based on the above findings, we can provide a new perspective to examine the above caching problem for specific user groups. However, it is not easy to distinguish the popular content through specific user groups during low access frequency period, and there are two challenges: *a) How to capture the relationship between hot contents and special user groups using historical data, b) How to design strategies with the above relationship to improve hit rate on edge server.* That based on users characteristics of corresponding solutions to address these challenges above will be introduced in the next section.

IV. System Models and Design

In this section, we firstly classify the users to core users, and treat others as common users. Then we adopt the principal component analysis algorithm to analyze the relationship between hot contents and core users in IV-A. On this basis, we propose a hot contents pre-cache protection mechanism in IV-A. Finally, we describe the overview of replacement strategy of *CRATES* in IV-C.

A. *Principal component analysis*

Through analysis above in subsection III-B, special user groups with many features are an important role in this paper. To describe special user groups clearly, we firstly classify the users (whole users) with high access frequency and broader distribution on the time dimension to *core users* U, and treat others as common users.

It is well-known that hot content is generated by a large number of user requests. Whether the content is a hot one is measured by the number of requests. On this basis, the relationship between hot contents and core users can transform into the relationship between the request content number of core users and whole users' respectively. Therefore, we measure the relationship between hot content and core users by the content request number $X^t = [x_1^t, x_2^t]$, where x_1^t is a vector of length m as a sliding window from core users in time t, is presented as

$$x_i^t = [r_1^{t-m*\Delta t}, r_2^{t-(m-1)*\Delta t}, ..., r_m^{t-\Delta t}], \tag{1}$$

, each term represents the total number of user requests in per unit time Δt. x_2^t is the same structure as x_1^t, and represents the request number of whole users. We obtain the relationship between the request content number of core users and whole users' by

$$cov(x_1^t, x_2^t) = \frac{\sum_{i=1}^n (x_{1i}^t - \bar{x}_1^t)(x_{2i}^t - \bar{x}_2^t)}{n}, \tag{2}$$

where $\bar{x}_1^t$ and $\bar{x}_2^t$ are computed by

$$\bar{x}_i = \frac{\sum_{i=1}^n x_i}{n}. \tag{3}$$

The eigenvector with the largest eigenvalue in $cov(x_1^t, x_2^t)$, is the relationship value about contents of core users and whole users'. The larger an element in eigenvector E^t is the closer the relationship between hot content of core users and whole users'. We choose the larger value in the eigenvector E^t as the relationship between hot contents and core users in time t.

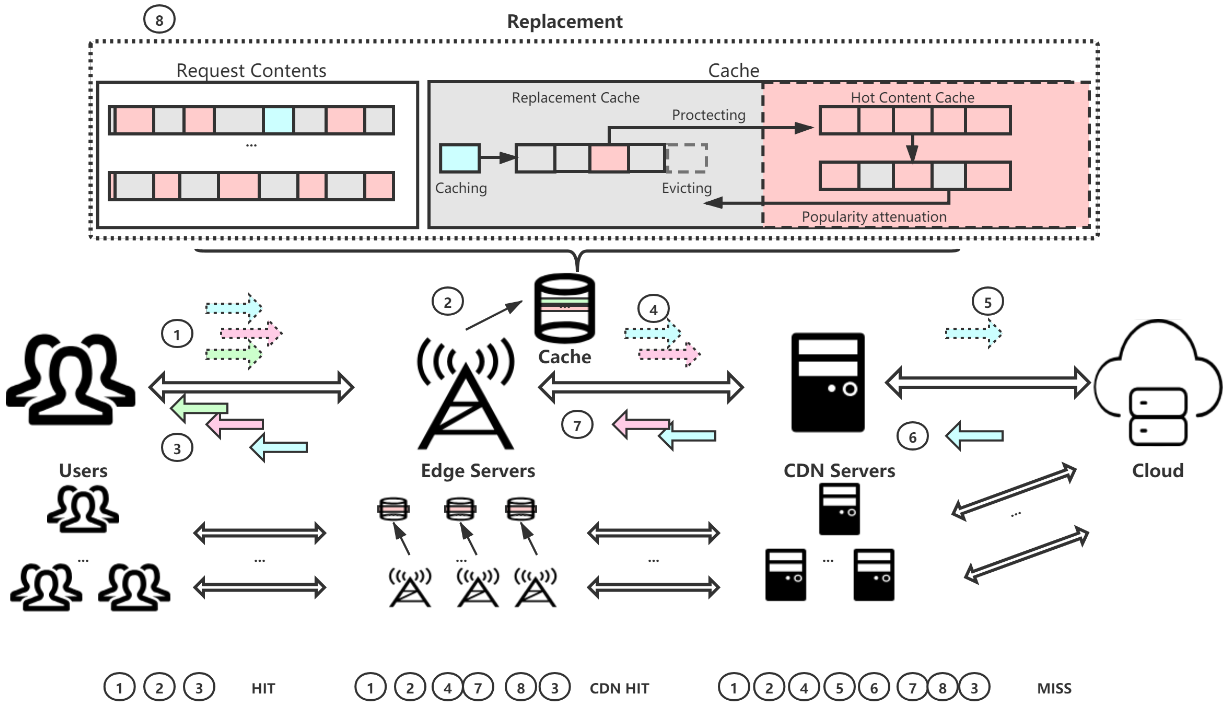


Figure 5: The framework overview.

B. *Pre-cache protection mechanism*

Although there are enormous differences in hot content sets between two continuous unit times, the existing intersection is valuable for hit rate improvement, so we propose a hot contents pre-cache protection mechanism to store the intersection contents effectively.

The main idea of the mechanism is to divide cache resources into dynamic and static cache areas dynamically. The dynamic cache area stores the requested contents in real-time, and the static cache area stores contents for that intersection of hot contents without storage replacement. Since the intersection changes over time, we dynamically adjust the size of the static cache S_s^t by

$$S_s^t = \sum\nolimits_{i=1}^{k^t} size(q_i^t), \tag{4}$$

where k^t is the number of pre-cache hot contents in time $t-1$ and is calculated by equation (5). q_i^t presents the ith pre-cache hot content in time $t-1$ that is one of the pre-cache hot content set q^t can be obtain in equation (6).

$$k^t = num(H_{\mathrm{t}-1} \cap H_{\mathrm{t}-2}) * a^t \tag{5}$$

$$q^t = Top_{k^t}(Request(H_{\mathrm{t}-1} \cap H_{\mathrm{t}-2})) \tag{6}$$

H_{t-1} and H_{t-2} are hot contents sets from two continuous unit times.

To dynamically implement the division of dynamic and static cache areas, the coefficient a^t is generated automatically in time t and is based on hit rates r_s^{t-1} and r_d^{t-1} that from dynamic and static cache area in time $t-1$. The coefficient a^t is calculated by

$$a^t = \frac{a^{t-1} * r_s^{t-1}}{a^{t-1} * r_s^t + (1 - a^{t-1}) * r_d^{t-1}}. \tag{7}$$

In other words, when a cache region has a higher hit rate, more areas are allocated to that region in the next update time. This approach can dynamically provide an appropriate static cache size S_s between two continuous unit times. Dynamic cache size S_d is equal to the total size of cache subtract the static cache size S_s.

C. *Overall design*

In the introduction above, we can gain a metric to measure the content's value by the relationship between hot contents and core users and propose a protection mechanism to ensure hit rate from changeless hot contents between two continuous unit times, which provide the basis for the overall architecture design. The framework overview is shown in Figure 5, which mainly includes network infrastructures, the flow of users access contents, and a cache replacement strategy.

The network infrastructures is made up of four units that are clients, edge servers, CDNs, and Cloud. In this architecture, content statuses of users' requests include: hit, CDN hit, and miss. When users request content, it is searched in edge servers, firstly. If the content exists, we called it hit and transmit it to the user (①②③). If not, the content is found in CDNs as CDN hit (①②④⑦⑧③). Else, the requested content is only stored in the remote cloud and the statue is miss (①②④⑤⑥⑦⑧③).

The replacement strategy for caching hot contents in edge servers mainly includes 4 phases: initialization, the pre-cache of hot contents, the selection of evicting content, and the update of information.

- *Initialization.* In the initial stage, the cache replacement algorithm is LRU when the length of request content information is inconsistent with the sliding window size. (Line 1-2 in algorithm 1)
- *The pre-cache of hot contents.* When that size is matching, we divided storage resources into dynamic and static cache areas, and cached the intersection hot contents between two continuous unit times in the static cache area based on the protection mechanism above. (Line 4-7 in algorithm 1)
- *The selection of evicting contents.* With the uninterrupted request, content statuses are different. When the status is miss, the strategy chooses an evicting content v_t that is the lowest one of E in the dynamic cache area. If the total size of evicting contents and the free area is smaller than request one, we choose the smallest one in the remaining contents until the total cache size of evicts is bigger than that request content size, and then pop these contents finally. (Line 9-18 in algorithm 1)

$$v_t = f(min(E)) \tag{8}$$

- *The update of informations.* Request information is updated as well as the contents and cache size of the static cache area through the sliding window periodically. (Line 19-29 in algorithm 1)

The pseudocode of the algorithm *CRATES*is shown in algorithm 1.

Algorithm 1 The pesudo code of *CRATES*

input: A new content c_{new} is requested, with size s_{new}, user id u_{new} and time t_{new}; C contents c_1 ,··· ,c_2 are already stored in the cache area, each size is s_i; Core users requests numbers sequence x_1 and wholes' x_2 are cache; r_s is the surplus area in dynamic cache.

```
1:  if Size(history traces)<Size(sliding window) then
2:      caching replacement by LFU.
3:  else
4:      if Size(history traces)=Size(sliding window) then
5:          Calculate the static cache size by equation(4)
6:          Calculate the intersection $q_t$ by equation(6)
7:          cache $q_t$
8:      else
9:          if $c_{new}$ NOT in $C$ then
10:             gain $r_s$
11:             while $r_s < s_{new}$ do
12:                 Calculate $cov(x_1^t, x_2^t)$ by equation (2) and obtain $E$
13:                 Get $c_i$ by value $v_t$ from $E$ by equation (8)
14:                 $r_s$ = $r_s$ + $s_i$
15:                 remove $c_i$
16:             end while
17:             Storage $c_{new}$ in dynamic cache.
18:         end if
19:         $x_2^{t-\delta t}$ = $x_2^{t-\delta t}$ + 1
20:         if $u_{new}$ in $U$ then
21:             $x_1^{t-\delta t}$ = $x_1^{t-\delta t}$ + 1
22:         end if
23:         if $t_{old} + \delta t = t_{new}$ then
24:             remove $x_1^{t-m*\delta t}$
25:             remove $x_2^{t-m*\delta t}$
26:         end if
27:         Calculate the static cache size by equation(4)
28:         Calculate the intersection $q_t$ by equation(6)
29:         Remove $c_1$ ,··· ,$c_2$ and cache $q_t$
30:     end if
31: end if
```

V. Evaluation

In this section, we evaluate our approach *CRATES* by real traces from ChuangCache and show the results of applying *CRATES* on them versus the existing representative policies, and analyze the results.

A. *Experiment setting.*

Algorithms: We compare with four algorithms including FIFO, LRU, LRB, and LeCaR.

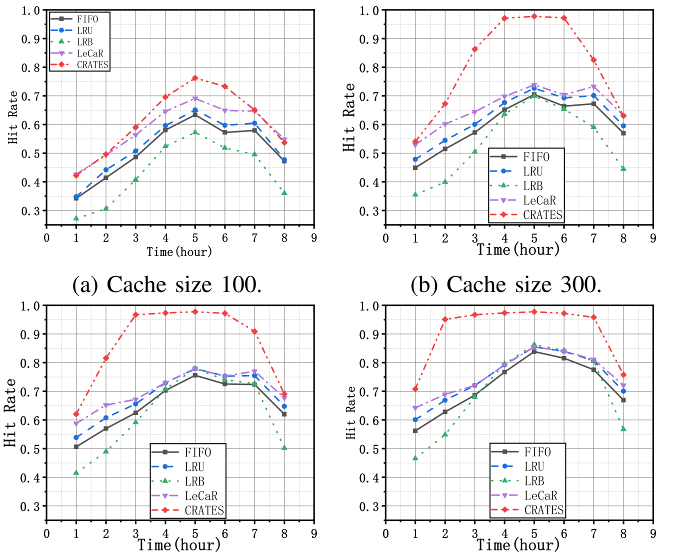
- FIFO: Removing the first requested content in the cache.
- LRU: Removing the least requested content in the cache.
- LeCaR: Using the machine learning technology of regret minimization combines LFU and LRU based on content popularity to improve on existing cache replacement strategies.
- LRB: Adopting machine learning to approximate the Belady MIN algorithm.

DateSet: The traces are from ChuangCache in China with 49,433,164 accesses to 1,158,513 contents in one day. Each trace item contains the timestamps, anonymized user ID, content ID, server ID, request size, URL, and et.al. We then deploy and evaluate *CRATES* by comparing with representative algorithms.

Parameter Settings: The sliding window size was set to 1 minute, and δt was 1 second. Each request content size was 1. The core users were obtained from the previous day's traces in the same application.

B. *Performance comparison.*

We conduct a series of experiments with different cache sizes to show the overall hit rate performance in Figure 6. Figure 6a, 6b, 6c, and 6d show that hit rates of every hour with cache size 100, 300, 500 and 800 in low access frequency period. When cache size is small, the hit rate of our algorithm has a similar performance in high access time (such as 1 and 8) but is superior to other times compared to the baselines. With the enlargement of the cache size, the performance of our algorithm becomes more prominent because more content can be cached in edge servers, so the hit rate increase as expected. Especially in figure 7 with cache size 1000, the performance improvement is nearly 0.25 than cache size 100 in hour 1 and the hit rate reaches about 98% in hour 5. From these figures,



(a) Cache size 100.

(b) Cache size 300.

(c) Cache size 500.

(d) Cache size 800.

Figure 6: [Performance comparison] The hit ratio in different cache size.

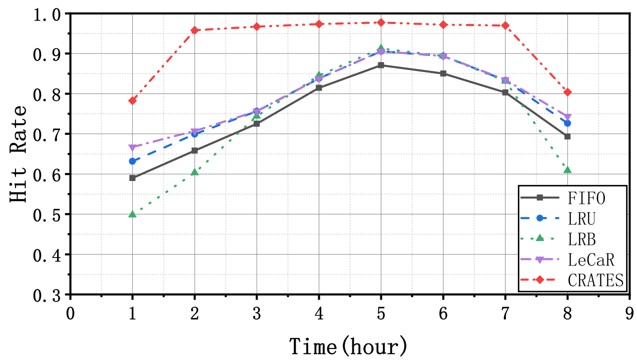


Figure 7: [Performance comparison] The hit rate in 1000 cache size from 8 hours.

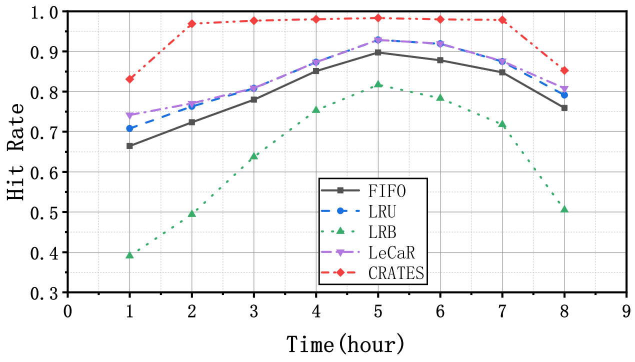
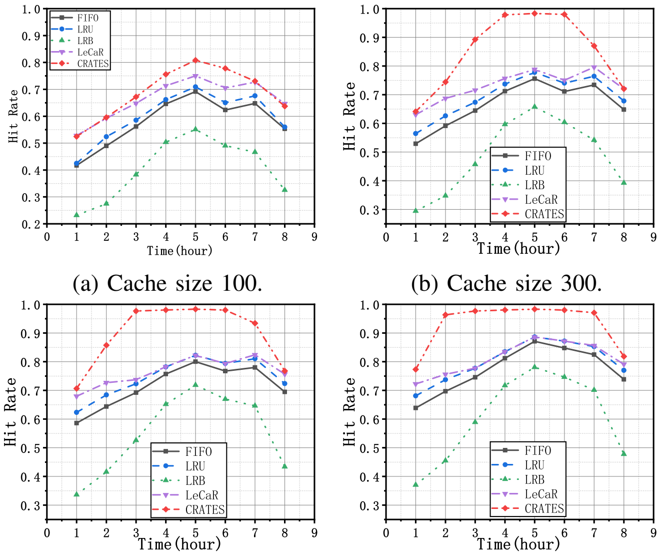


Figure 8: [Result Analysis] The hit rate from core users in 1000 cache size from 8 hours.



(a) Cache size 100.

(b) Cache size 300.

(c) Cache size 500.

(d) Cache size 800.

Figure 9: [Result Analysis] The hit rate from core users in different cache size.

we can see that *CRATES* exceeds the other four methods in every hour. The advantage is even remarkable when the percentage of core users is high (such as hour 4-6).

C. *Result Analysis.*

- *Core user's hit rate.*
  Figure 8 illustrates the hit rate from core users with cache size 1000. To compare that with the hit rate of total users, we also draw some figures (figure 9) that with the same size as it. Intuitively, the hit rate from core users is higher than from total users in the same cache size. But the hit rate of core users from LRB is lower than the hit rate of total users that leads to lower hit rate than other algorithms. Thus, it proves the importance of core users. No exception at all, as the cache size increases, the hit rate of core users also is improved as show in figure 9.
- *Replacement hot contents ratio.*
  Figure 10 describes the ratio of evicting contents with historical hot contents in cache size 1000, where hot contents are the top k (cache size) by request number. In hour 3-6, LeCaR, LRB, and LRU replacement ratio are nearly zero, but their hit rates are lower than *CRATES* which with a higher replacement ratio. That means the content is hot in history, but maybe not in the future. It also reflects that hot contents are difficult to predict. Therefore, the appropriate historical hot contents need to evict for improving the hit rate in low access frequency period for storage replacement.

To summarize, it is a good perspective of considering core users to improve the hit rate during the period. Because it's more possible that contents accessed by core users become hot, core users play a more important role than general users

VI. Future Work

Data storage equipment (such as data centers) to satisfy the demand for storing the data volume with explosive growth needs massive energy consumption, and then the economic and environmental sustainability of the earth is severely affected.

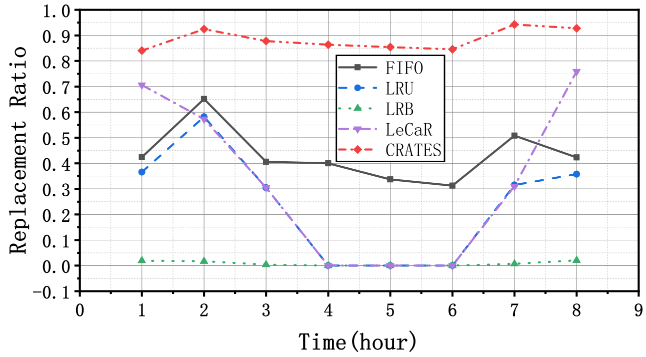


Figure 10: [Result Analysis] The ratio of replacement hot contents from history.

The power grids burn brown energy resources like coal and petroleum to meet the ever surging power demands of the data centers, thereby producing tons of CO_2 and causing global pollution [21] [22]. To make global green development, the improvement of QoS isn't the only requirement for application service providers and others right now. How to reduce waste of resources (such as edge data centers) under the premise of ensuring the QoS is a hot topic. Although this paper is to optimize the QoS for one time period, it may be a perspective to implement a green-energy-aware VM or server manager [23], [24] without affecting the QoS by the detailed research for optimum performance in different periods. This way is not only can protect the environment but also can reduce the cost of service providers.

VII. Conclusion

In this paper, we firstly classify the users with high access frequency and broader distribution on the time dimension to core users. Then we adopt the principal component analysis algorithm to analyze the relationship between hot contents and core users. On this basis, we propose a hot contents pre-cache protection mechanism, which caches the hot contents from history in static cache area, and cache & evict contents with real-time by the relationship between hot contents and core users in the surplus area. To improve *CRATES*'s efficiency, we extract key part of historical data by designing a sliding window method. Through a series of experiments using real application trace data, we demonstrate that CRATES reaches about 98% in caching hit rate and outperforms the state-of-the-art algorithm LRB by 1.4X.

Acknowledgment

The work was supported in part by the National Natural Science Foundation of China (NSFC) Youth Science Foundation under Grant 61802024, BUPT-Chuangcache Joint Laboratory under B2020009, the Key Project of Beijing Natural Science Foundation under M21030, the Fundamental Research Funds for the Central Universities under Grant 2482020RC36, the NSFC under Grant 62072047, and the NSFC under Grant 62072048. The work of Pengmiao Li was supported in part by the BUPT Excellent Ph.D. Students Foundation under CX2019134.

References

[1] We Are Social&Hootsuite. https://wearesocial.cn.2021.

[2] Vietri G., Rodriguez L. V., and Martinez W. A., et al., ”Driving cache replacement with ML-based LeCaR,” The 10th USENIX Conference on Hot Topics in Storage and File Systems (HotStorage'18), 2018.

[3] Song, Z., Berger, D. S. and Li, K., et al., ”Learning Relaxed Belady for Content Distribution Network Caching,” 2020 USENIX NSDI, 2020.

[4] ChuangCache.https://www.chuangcache.com/.2020.

[5] N. C. Fofack, P. Nain, and G. Neglia, et al., ”Analysis of ttl-based cache networks,” the 6th International ICST Conference on Performance Evaluation Methodologies and Tools(VALUETOOLS), 2012.

[6] N. B. Melazzi, G. Bianchi, and A. Caponi, et al., ”A general, tractable and accurate model for a cascade of lru caches,” IEEE Communications Letters, vol. 18, no. 5, pp. 877–880, 2014, doi: 10.1109/LCOMM.2014.031414.132727.

[7] Chia-Tai Chan, Shuo-Cheng Hu, and Pi-Chung Wang, et al., ”A fifo-based buffer management approach for the atm gfr services,” IEEE Communications Letters, vol. 4, no. 6, pp. 205–207, 2000.

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

[9] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[10] Vasconcelos, M., J. M. Almeida, and Marcos André Gonalves, ”Predicting the popularity of micro-reviews: A Foursquare case study.” Information Sciences 325.C(2015):355-374.

[11] Z. Tan, W. Hu, and Y. Zhang, et al., ”Online Popularity Prediction of Video Segments: Towards More Efficient Content Delivery Networks,” IEEE Global Communications Conference (GLOBECOM), 2019.

[12] X. Wang, B. Fang and H. Zhang, ”Using a Dynamic Model to Predict Popularity of News,” IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2019.

[13] Liang J., Zhu D., and Liu H., et al., ”Multi-Head Attention Based Popularity Prediction Caching in Social Content-Centric Networking With Mobile Edge Computing,” IEEE Communications Letters, vol. 25, no. 2, pp. 508-512, Feb. 2021, doi: 10.1109/LCOMM.2020.3030329.

[14] Zhang Y., Li P., and Zhang Z., et al.. ”GraphInf: A GCN-based Popularity Prediction System for Short Video Networks,” International Conference on Web Services (ICWS), 2020.

[15] Li S., Jie X., and Schaar M., et al., ”Popularity-driven content caching,” IEEE Conference on Computer Communications (INFOCOM), 2016.

[16] Li, Q., Shi W., and Xiao Y., et al., ”Content Size-Aware Edge Caching: A Size-Weighted Popularity-Based Approach,” IEEE Global Communications Conference (GLOBECOM), 2018.

[17] Yang, P., Zhang N., and Zhang S., et al., ”Content Popularity Prediction Towards Location-Aware Mobile Edge Caching.” IEEE Transactions on Multimedia (TMM) 21.4(2019):915-929, doi: 10.1109/TMM.2018.2870521.

[18] Zong, T. , Li C., and Lei Y., et al., ”Cocktail Edge Caching: Ride Dynamic Trends of Content Popularity with Ensemble Learning,” IEEE Conference on Computer Communications (INFOCOM), 2021.

[19] G. Ma, Z. Wang, and M. Zhang, et al., ”Understanding performance of edge content caching for mobile video streaming,” IEEE Journal on Selected Areas in Communications(JSAC), vol. 35, no. 5, pp. 1076–1089, 2017.

[20] Zhang Y., Li P., Zhang Z., et al., ”AutoSight: Distributed Edge Caching in Short Video Network,” IEEE Network, vol. 34, no. 3, pp. 194-199, 2020.

[21] T. Bhattacharya and X. Qin, ”Modeling Energy Efficiency of Future Green Data centers,” the 11th International Green and Sustainable Computing Workshops (IGSC), 2020.

[22] Q. Lin and S. Yu, ”A Distributed Green Networking Approach for Data Center Networks,” IEEE Communications Letters, vol. 21, no. 4, pp. 797-800, 2017.

[23] G. Madi Wamba, Y. Li, and A. Orgerie, et al., ”Green Energy Aware Scheduling Problem in Virtualized Datacenters,” 2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS), 2017, pp. 648-655.

[24] E. Baccour, S. Foufou, R. Hamila and A. Erbad, ”Green data center networks: a holistic survey and design guidelines,” 15th International Wireless Communications & Mobile Computing Conference (IWCMC), 2019, pp. 1108-1114.