Caching strategies based on popularity prediction in content delivery networks

Nesrine Ben Hassine, Dana Marinca, Pascale Minet, Dominique Barth

Caching Strategies Based on Popularity Prediction in Content Delivery Networks

Nesrine Ben Hassine*, Dana Marinca†, Pascale Minet* and Dominique Barth†
*Inria Paris, 2 rue Simone Iff, CS 42112, 75589 Paris Cedex 12, France
Email:nesrine.ben-hassine@inria.fr, pascale.minet@inria.fr
†DAVID, University of Versailles, Versailles, France
Email: dana.marinca@uvsq.fr, dominique.barth@uvsq.fr

***Abstract*—In Content Delivery Networks (CDNs), knowing the popularity of video content helps the manager to take efficient decisions about which video content should be cached near the end users and also about the duplication degree of each video to satisfy the end user Quality of Experience. This paper focuses on predicting the popularity of video content, in terms of the number of requests. For that purpose, different software entities, called experts, compute the popularity value of each video content. Each expert uses its own prediction method. The accuracy of expert's prediction is evaluated by a loss function as the discrepancy between the prediction value and the real number of requests. We use real traces extracted from YouTube to compare different prediction methods and determine the best tuning of their parameters. The goal is to find the best trade-off between complexity and accuracy of the prediction methods used. Finally, we apply these prediction methods to caching. Prediction methods are compared in terms of cache Hit Ratio and Update Ratio with the well-known LFU caching strategy.**

***Index Terms*—CDN, popularity prediction, caching, LFU, YouTube, video content.**

I. Introduction

A. Context

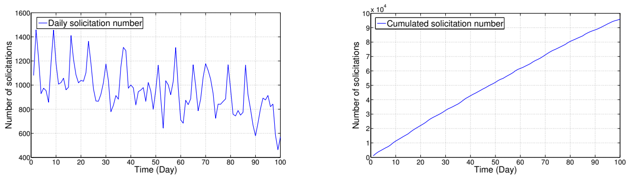
The number of Internet users is continuing to grow at a spectacular rate and the diversity of the services it offers has certainly contributed significantly to this huge increase. Among the currently most popular Internet services is multimedia content. Accordingly, there has been a substantial growth in network traffic particularly due to the large demand for content being accessed over the Internet. Frequently requested content experiences network bottlenecks, which results in decreases in service quality which implies new challenges in managing content delivery. In some cases, the server holding the content can become temporarily unavailable.

To cope with such a problem, Content Delivery Networks (CDNs) provide a method for a more efficient delivery of content to end users [1]. The distinguishing factor of CDN is that the node identifier is no longer the key aspect but the content to be retrieved. A CDN is a network of geographically dispersed servers that are strategically placed at various locations. The typical components of a CDN are the Origin server and the Surrogate servers. The content to be distributed over the Internet is stored in the Origin server. Surrogate servers located across the globe either cache a replication of the Origin server or the most requested contents [2]. Thus, the users retrieve contents from the Surrogate server close to them. Since data is cached on several servers, requests are directed to the closest one that sends the requested video content along the shortest route to the end user. This reduces latency and load on the Origin server, allowing for faster load times as well as improved streaming quality. Furthermore, when a server is highly loaded, redundancy enables the requested content to be available in another server. This distributed infrastructure can neutralize the impact of the dysfunction of a particular server. Hence, CDNs improve the network performance and hasten the delivery of requested contents to any end user location.

Nonetheless, cache management is a challenging task. Given the finite storage capacity of a server, it is crucial to determine whether content is requested sufficiently often for it to be worth caching. Thus, all stored content has to be continuously monitored so that it is removed from the cache once it is no longer required. Moreover, the user request pattern is not always uniform. There may be sudden changes in demands from users all over the world. This issue requires an in-depth analysis of the dynamics of content requests.

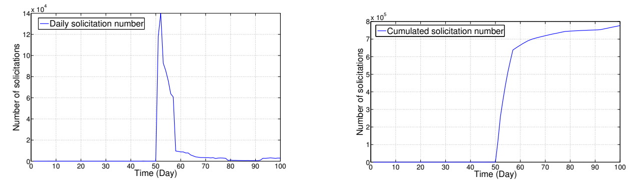
To better suit these fluctuations, we need to estimate the demands. The behavior of video content over time reveals distinct patterns of popularity evolution [3]. A vast amount of content is viewed only a few times. Some content remains popular for a long period of time whereas other content ceases to be popular after only few days or even hours. Hence, it is possible to estimate the future popularity of content by examining the types of popularity growth behavior that content displays over time. Therefore, demands can have different behaviors depending on the content and the time interval (Figures 1 and 2).

In this paper, we study how popularity prediction of video contents can be used by caching. Unlike [4] where we studied the behavior of different experts to predict popularity of video contents, we show how to use popularity prediction to improve caching and compare the hit ratio obtained with this obtained using the LFU (Least Frequently Used) caching strategy. In the rest of this section, we briefly present the best-known caching strategies. In Section II, we give a theoretical framework to predict the popularity of video content. We also explain how the content is selected from YouTube. In Section III, we define the experts used and report the results obtained from real traces. We compare the performances of our experts. In Section IV, we first define the theoretical framework for caching with its evaluation metrics (e.g. Hit ratio, Update ratio) and the reference strategies (e.g. LFU and Best) for a comparative evaluation of our caching strategy based on popularity prediction. Section V comments on the results obtained. Finally, we conclude in Section VI and outline some further work.



(a) (b)

Fig. 1: A smooth profile: a) real profile, b) cumulated one.



(a) (b)

Fig. 2: A sharp profile: a) real profile, b) cumulated one.

B. Related work

Many caching strategies have been proposed in the literature. In this section, we only present the best-known strategies. These strategies, if available in the context of our experiment, will be used as a reference in the performance evaluation reported in Section V.

The Least Recently Used (LRU) caching strategy always replaces the least-recently-used object in cache [5], [6]. For that purpose, a sorted list is maintained: video contents stored in the cache are sorted in decreasing order of their last request time. When a video content is requested, two situations can appear: (1) if the content is not in the ordered list, it is inserted in the list at the first rank and the last content in the list is removed, or (2) if the content is already in the list, it gets the first rank and the ranks of the other contents are updated accordingly. LRU has been a widely used caching strategy for many years and is often used as a reference benchmark for performance evaluation of other caching strategies.

Least Frequently Used (LFU) is a more complex strategy than LRU. It keeps track of the frequency of requests for every content to evaluate its popularity. It puts in the cache the contents having the highest request frequencies within a specific time window [5], [6]. If multiple contents have the same request frequency and one of them has to be removed from cache, it is chosen according to the LRU strategy. The difficulty with LFU, however, is selecting an appropriate time window. If the window size is too large, popularity is calculated based on obsolete information. On the other hand, if the window size is too small, the popularity estimation can generate a lot of useless cache updates.

The Optimal Strategy (MIN) [7] is an optimal, offline, caching strategy in terms of cache hit ratio. When a requested video content is not in the cache and the cache is full, it replaces the content in the cache whose next request occurs the furthest in the future. If there are multiple contents that will never again be requested, one of them is chosen arbitrarily. Several studies have been presented to show its optimality [8], [9]. However, this strategy cannot be applied in a real deployment, because it requires knowledge of the future requests' profile.

II. Theoretical Framework for Prediction

In this paper we adopt a machine learning approach and propose a model to predict requests for video content on a CDN platform. The predictions will be used to store the most requested content in caches near the end-users. In the work presented hereafter we focus on two prediction methods and on the best tuning of their parameters. To validate the proposed model we use real content request traces extracted from the YouTube platform.

A. Generalities

The model is based on logical entities called *experts*. The set of video contents for which we predict the popularity is denoted $\mathcal{C}$. We denote by $\mathcal{E}$ the set of experts studied. Each *expert* $E_i \in \mathcal{E}$ computes and predicts the future number of requests for each video content in $\mathcal{C}$, using its own computation method. Two experts E_i and $E_j \in \mathcal{E}$ using the same method differ in their parameter tuning. Each expert $E_i \in \mathcal{E}$ computes at time t the value $p_{i,t+1}$, representing the predicted number of requests of a given video content, using its own prediction strategy (Section III-A.). The goal of each expert is to predict a number of requests that is as close as possible to the real number y_{t+1}.

At time t, the accuracy of each expert prediction is evaluated by a loss function (see Section II-C). The expert's *instantaneous loss* at time t represents the discrepancy between the predicted value $p_{i,t}$ and the real number of requests y_t. For some experts (see Section III-A1), we need to use the cumulated number of requests, denoted Y_t and the prediction of the cumulated number of requests $P_{i,t}$. The sum of the instantaneous losses up to time t is called *cumulated loss* at time t.

Table I summarizes our notations.

TABLE I: Notations.

$\mathcal{C}$	the set of video contents
$\mathcal{E}$	the set of experts
ow_t	the observation window at time t
	For each given video content $c \in \mathcal{C}$
y_t	the number of requests at time t
Y_t	the cumulated number of requests up to time t
$p_{i,t}$	the predicted number of requests by expert $E_i \in \mathcal{E}$ for time t
$P_{i,t}$	the prediction of the cumulated number of requests by expert $E_i \in \mathcal{E}$ for time t
$l_{i,t}$	the instantaneous loss of expert E_i at time t
$L_{i,t}$	the cumulated loss of expert E_i at time t

B. Discrete time and observation window

The prediction process is done periodically using a discretised time. The goal of the experts is to predict the number of requests for each video content studied, at each time t. At current time t, the *observation window*, denoted ow_t, is a window over the past up to time t. Let $|ow|$ denote the size of the observation window. The observation window contains all the real values collected in the interval $(t - |ow|, t]$. The experts will compute their predictions based on these values. The predictions are done at time t for time $t+1$. At time $t+1$, the loss function evaluates the accuracy of the prediction made by each expert.

In this paper, the considered time granularity is the day. Predictions are done once per day for all video contents. More precisely, for each video content in $\mathcal{C}$, each expert $E_i \in \mathcal{E}$ will predict the next day's requests using its own computation method (Section III-A).

C. Expert Accuracy

The accuracy of an expert can be evaluated by its instantaneous loss or its cumulated loss. By definition, the instantaneous loss of expert E_i at time t is given by:

$$l_{i,t} = |P_{i,t} - Y_t| \tag{1}$$

By definition, the cumulated loss of expert E_i at time t is given by:

$$L_{i,t} = \sum_{j=0}^{t} l_{i,j} = \sum_{j=0}^{t} |P_{i,j} - Y_j| \tag{2}$$

The performance of different experts on a same content is evaluated according to instantaneous and cumulated loss. If we want to compare the performance of different experts on a set of contents, we have to normalize the loss function on a content by the number of requests.

D. Best Experts

With regard to a given video content $c \in \mathcal{C}$, we define the Best Expert according to two criteria: the cumulated loss and the reward. The reward of an expert predicting for the video content c, is by definition equal to the number of times the expert minimized the instantaneous loss.

Definition 1: The Best Expert associated with c, according to the cumulated loss, is the expert minimizing the cumulated loss at the end of the simulation.

Definition 2: The Best Expert associated with c, according to the reward, is the expert maximizing its reward at the end of the simulation.

Nonetheless, it is worth noting that, with both definitions, the Best Expert can only be defined at the end of the simulations. In the presence of large discrepancies between the expert prediction and the real value, Definition 1 has the drawback of penalizing this expert for a long time, whereas Definition 2 better takes into account the dynamics of the prediction.

We can extend these two definitions of Best Expert to a set of contents $\mathcal{C}$.

Definition 3: The Best Expert for the content set $\mathcal{C}$ according to the cumulated loss, is the expert that maximizes the number of times it was Best Expert according to Definition 1, for all $c \in \mathcal{C}$.

Definition 4: The Best Expert for the content set $\mathcal{C}$ according to the reward, is the expert that maximizes the sum of the rewards obtained on each video content $c \in \mathcal{C}$. at the end of the simulation.

In the following of the paper, we select the Best Expert on $\mathcal{C}$ according to the Definition 4.

E. Real Traces from YouTube

A large set of video content profiles $\mathcal{C}$ has been constituted from real traces extracted from YouTube. We study the popularity of video contents during the 100 consecutive days preceding the extraction day.

We constituted three subsets: (1) Set_1 contains 100 videos randomly selected from $\mathcal{C}$. This subset contains videos that are greatly requested as well as videos with a small number of requests. (2) Set_2 is constituted by 60 contents which are homogeneous in terms of the number of requests. (3) Set_3 includes 50 video contents with very sharp profiles, similar to those depicted in Figure 2.

III. Analysis of Experts

Various experts belong to the set $\mathcal{E}$. Each expert is characterized by a prediction method and a parameter tuning. In this paper, we study the following prediction methods: Double Exponential Smoothing (DES) and Basic.

A. Experts definition

This section details the methods used by experts to compute their predictions. The Double Exponential Smoothing (DES) expert provides its best performances when applied on profiles with trends (i.e. increase or decrease). Hence, we propose to predict the cumulated number of requests $P_{i,t+1}$ based on the cumulated requests number, $Y_{i,t}$, for each video content and for all experts considered.

1) Double Exponential Smoothing (DES) expert: To compute its prediction for time $t+1$, the DES expert [10] E_i computes first two variables $S^{'}_{i,t}$ and $S^{''}_{i,t}$ by applying the exponential smoothing twice: first on Y_t and $S^{'}_{i,t-1}$ to compute $S^{'}_{i,t}$ and second on $S^{'}_{i,t}$ and $S^{''}_{i,t-1}$ to compute $S^{''}_{i,t}$. Both use the same smoothing factor α with $0 < \alpha < 1$.

$$S^{'}_{i,0} = S^{''}_{i,0} = Y_0.$$

$$S^{'}_{i,t} = \alpha Y_t + (1-\alpha) S^{'}_{i,t-1}.$$

$$S^{''}_{i,t} = \alpha S^{'}_{i,t} + (1-\alpha) S^{''}_{i,t-1}.$$

The method also computes the estimated value $L_{i,t}$, and the estimated trend $T_{i,t}$:

$$L_{i,t} = 2S^{'}_{i,t} - S^{''}_{i,t}$$

$$T_{i,t} = \frac{\alpha}{1-\alpha}(S^{'}_{i,t} - S^{''}_{i,t}).$$

Finally, at time t, the DES expert E_i predicts the value $P_{i,t+1}$ for time $t+1$:

$$P_{i,t+1} = L_{i,t} + T_{i,t}. \tag{3}$$

2) Basic expert: The Basic expert assumes the number of requests at time $t+1$ has the same increase or decrease as the number of requests at time t. Since $Y_t = Y_{t-1} + y_t$ by definition, the Basic expert E_i [4] predicts at time t the value $P_{i,t+1}$ given by:

$$P_{i,t+1} = Y_t + (Y_t - Y_{t-1}) = 2 * Y_t - Y_{t-1}. \tag{4}$$

Hence, we come back to the real profile:

$$p_{i,t+1} = P_{i,t+1} - Y_t = Y_t - Y_{t-1} = y_t. \tag{5}$$

B. Simulation Results for Experts

In this section, we compare the performances of the Basic and DES experts with different parameter tunings on the dataset Set_1. The smoothing factor α ranges from 0.5 to 0.99, whereas the size of the observation window ow ranges from 3 to 7. We compute the Best Expert for Set_1 minimizing the cumulated loss, according to Definition 3. Since the smoothing factor is close to 1, the size of the observation window has a small impact, which explains the similar behaviors of the DES experts with an observation window of 3, 5 or 7 days. Taking into account the fact that several experts provide close predictions, we focus on the first three experts that minimize the cumulated loss. In these conditions, the Best Expert becomes DES (7, 0.99), closely followed by DES (5, 0.99) and DES (3, 0.99), as depicted in Figure 3.

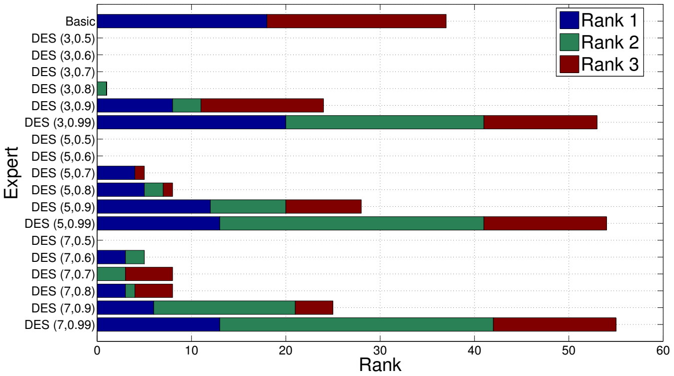


Fig. 3: Expert ranks according to the cumulated loss.

If now we use Definition 4, we focus on the first three experts that minimize the instantaneous loss most often, then the Best Expert is DES (7, 0.99) followed by DES (5, 0.99), followed by DES (3, 0.99) and finally Basic, as depicted in Figure 4.

These results allow us to tune the parameters of DES experts. Clearly, a smoothing factor of 0.99 and an observation window size of 7 provide the best performance for a DES expert. This is due to the high dynamics of requests explaining why the most recent values have a stronger impact on the prediction.

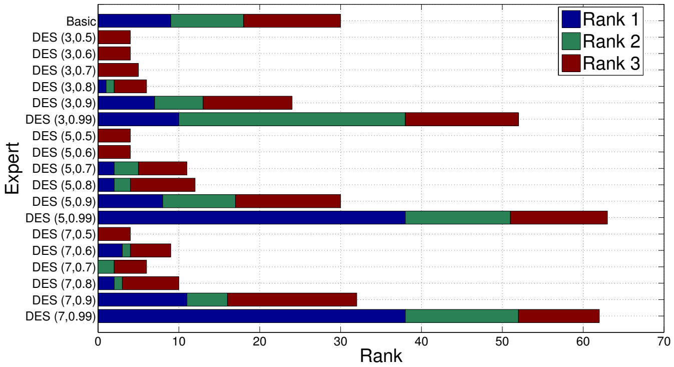


Fig. 4: Expert ranks according to the reward.

IV. Theoretical Framework for Caching

A. Caching Evaluation Metrics

Two metrics are used in the evaluation of the caching strategies: the ”cache hit ratio” (cHR) and the ”cache update ratio” (cUR). The cHR is defined as the percentage of requests that can be served from a cache. The cUR, on the other hand, represents the percentage of contents in the cache that are removed to be replaced by more popular contents. In the CDN context, the best caching technique is that maximizing the cHR as the first criterion and minimizing the cUR as the second criterion. These two metrics are evaluated each day.

B. Caching Strategies

The performance evaluation reported in the next section is based on collected traces from YouTube. Unfortunately, in these traces, the time arrival of each video content request is not known, and we only know the daily number of requests for each content. As a consequence, we cannot use the LRU strategy. MIN strategy can be extrapolated to take into account the future daily number of requests instead of the arrival time of the next request. We call Best strategy this extrapolation of MIN. That is why the caching strategies studied are LFU and Best. Best assumes knowing the future requests, so it puts in cache the most popular contents of the day. Of course, Best strategy cannot be implemented in real CDN platforms but it provides the optimal strategy for our case study. These caching strategies will be compared with our proposed Prediction based Caching Strategy (PCS). PCS puts in cache the contents with the highest predicted popularities.

C. Caching Assumptions

In our performance evaluation, we adopt the following assumptions:

- All the video contents have the same size.
- The cache size is a multiple of the content size.
- Each night, the cache is updated for the next day, according to the caching strategy.
- For each video content, only the number of requests per day is given.
- The observation window size of LFU is equal to one day.

V. Analysis of Prediction-based Caching strategies

In this series of simulations, the performance evaluation is done on three random sets of video contents: Set_1, Set_2 and Set_3 defined in Section II-E. According to the study reported in Section III-B, the Prediction based Caching Strategy (PCS) is based on the predictions made by the DES (7, 0.99) expert.

A. Impact of cache size and distribution of content popularity

In the first series of experiments, we want to evaluate the optimal bound obtained with the Best strategy. We recall that this strategy knows in advance the popularity of each video content for the next day. No real strategy can do better. Hence, it represents the optimal for the context of our study. We use this strategy to determine the best cache size, that is the size maximizing the cHR. Taking into account our caching assumptions given in Subsection IV-C, the cache size is expressed as the percentage of video contents that are cached.

First, we consider Set_1. The results are depicted in Figure 5. As expected, the cHR increases with the cache size. An interesting conclusion is the large improvement obtained when the cache size increases from 5 to 10%. The improvement is slight for larger cache sizes. A cHR higher than 99% is already obtained with a cache size of 10%. We also notice that for a cache size of 5%, the cHR varies from 82 to 90% depending on the day considered. For larger cache sizes, the Hit Ratio is smooth: it depends very little on the day.

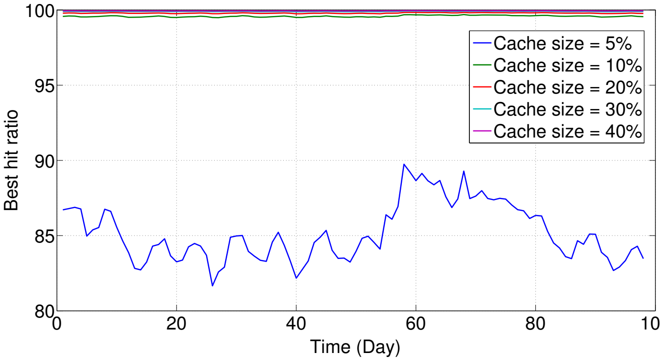


Fig. 5: Best hit ratio for Set_1.

The second test series show the impact of caching strategies on homogeneous popularities. Any smart caching strategy puts all the video contents with the highest popularities in the cache. Also all the video contents with very few requests are never put in the cache. The caching decision rather concerns the contents which have a moderate popularity (Set_2), if cache size is available. Clearly, these contents will show the differences between the caching strategies. The results depicted in Figure 6 show that large improvements are obtained by increasing the cache size up to 40%.

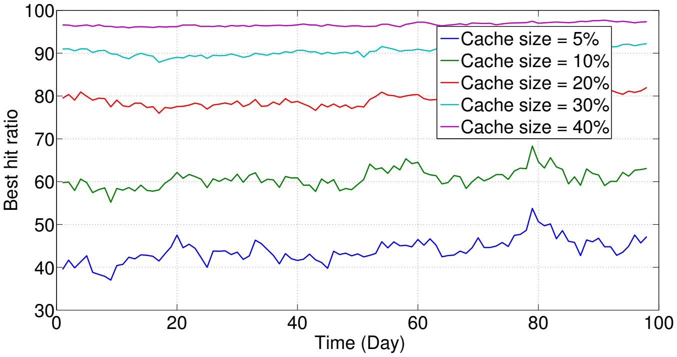


Fig. 6: Best hit ratio for Set_2.

In the third test series, we consider video contents with very sharp profiles (Set_3). Because of this sharpness, the cHR greatly depends on the day considered, even with a cache size reaching 40%, as depicted in Figure 7. This suggests that managing such profiles requires high cache sizes.

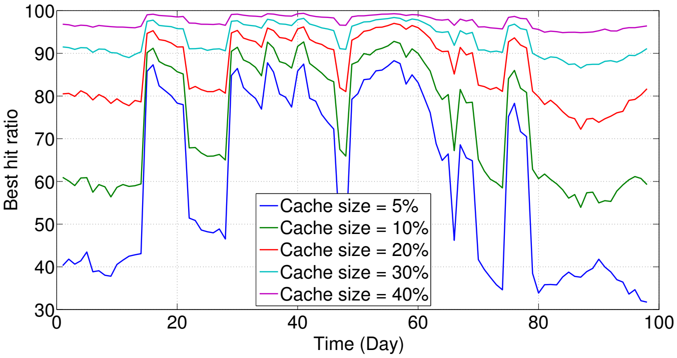


Fig. 7: Best hit ratio for Set_3.

B. Comparative evaluation with LFU

LFU can also be considered as an expert that predicts for day $t+1$ a value $p_{lfu,t+1}$ equal to the value observed at day t. That is

$$p_{lfu,t+1} = y_t. \tag{6}$$

Abusively, in the following we do not distinguish between the LFU expert and the LFU caching strategy that are denoted LFU. According to Equation 5 and Equation 6, LFU and Basic predict exactly the same value. Consequently, we have proved that:

$$Basic = LFU. \tag{7}$$

We now compare LFU with PCS, a caching strategy based on the predictions made by the DES (7, 0.99) expert. On the data set Set_1, we evaluate the benefit obtained in terms of cHR for different cache sizes. For a cache size of 20% depicted in Figure 8, LFU and PCS are very close to Best. However, at day 49, there is a strong discrepancy with Best: there is a sudden decrease in the cHR for both LFU and PCS. This is due to the very strong increase in the number of requests for a particular video content that had previously been very little requested. This decrease in the cHR is always present with a cache size equal to 40%, as depicted in Figure 9. For a cache size higher than or equal to 20%, the cHR is very smooth, except around day 49.

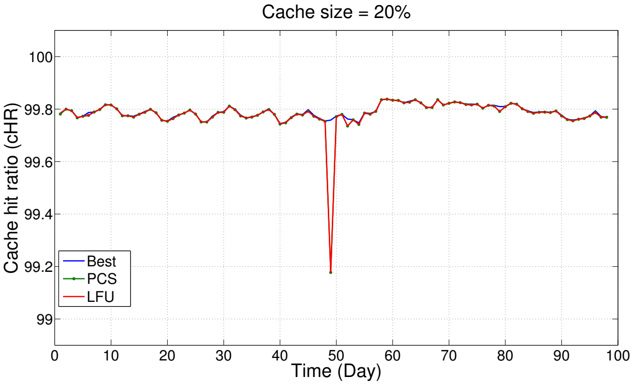


Fig. 8: cHR for Set_1, DES(7, 0.99), cache size=20%.

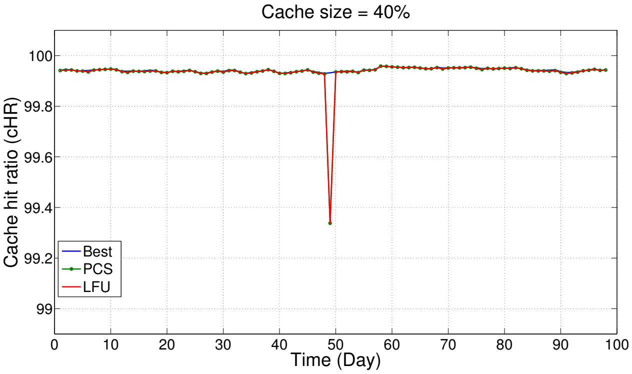


Fig. 9: cHR for Set_1, DES(7, 0.99), cache size=40%.

In terms of cUR, LFU and PCS give very similar results, whatever the cache size. We obtain cUR smaller than 0.1 for a cache size of 20% (see Figure 10) and smaller than 0.08 for a cache size of 40% (Figure 11). As expected, the cUR decreases when the cache size increases.

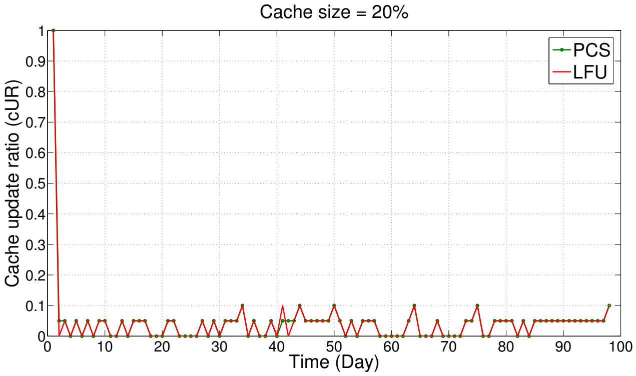


Fig. 10: cUR for Set_1, DES(7, 0.99), cache size=20%.

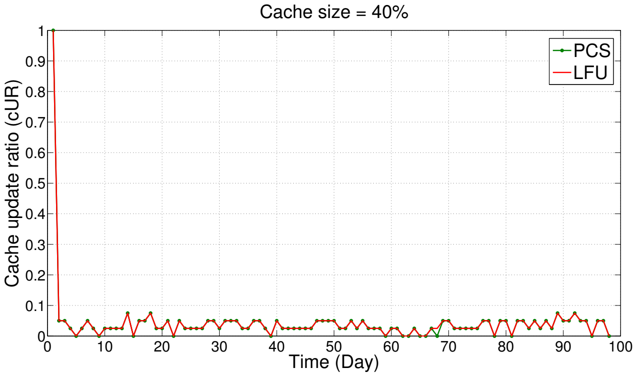


Fig. 11: cUR for Set_1, DES(7, 0.99), cache size=40%.

We now focus on the data set Set_2 and compare LFU and PCS based on DES (7, 0.99) in terms of cHR. Unlike Set_1, Set_2 provides hit ratios that vary greatly from one day to another. Even in such a case, LFU and PCS based on DES (7, 0.99) provide very close cHRs, both for a cache size of 20% and 40%, as depicted in Figures 12 and 13, respectively.

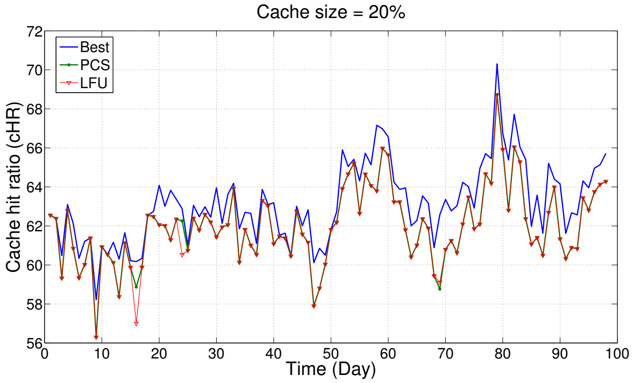


Fig. 12: cHR for Set_2, DES(7, 0.99), cache size=20%.

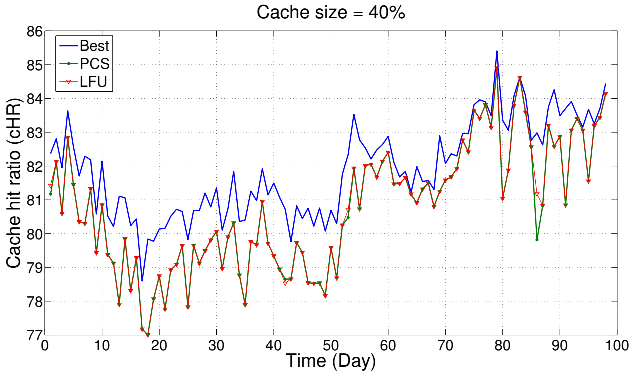


Fig. 13: cHR for Set_2, DES(7, 0.99), cache size=40%.

The cURs obtained for Set_2 are higher than the ones obtained with the previous data set Set_1. They are smaller than 0.3 for a cache size of 20% (see Figure 14) and smaller than 0.23 for a cache size of 40% (see Figure 15).

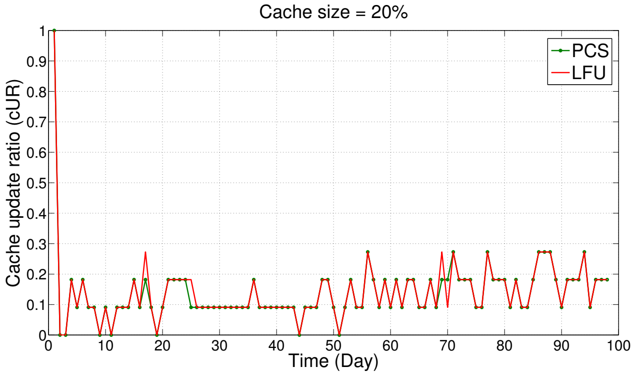


Fig. 14: cUR for Set_2, DES(7, 0.99), cache size=20%.

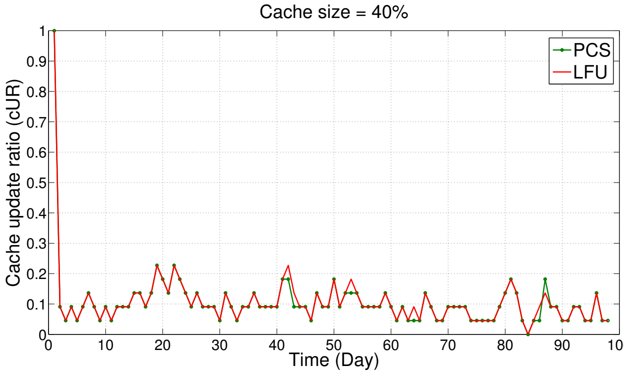


Fig. 15: cUR for Set_2, DES(7, 0.99), cache size=40%.

Finally, we consider the third data set Set_3. The cHRs obtained by LFU and PCS based on DES (7, 0.99) may be far from the optimal one given by Best, as depicted in Figure 16 for a cache size of 20% and Figure 17 for a cache size of 40%. This is due to the fact that both LFU and PCS take into account the number of requests from the day before.

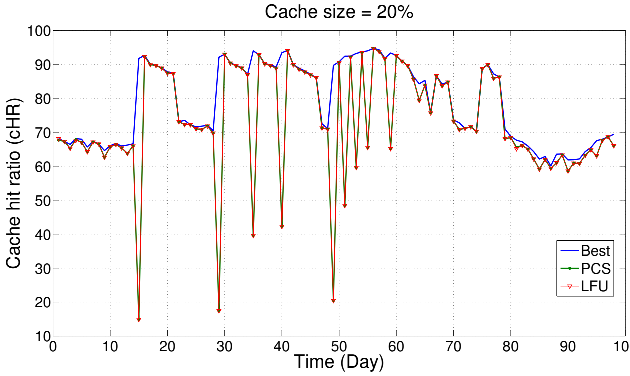


Fig. 16: cHR for Set_3, DES(7, 0.99), cache size=20%.

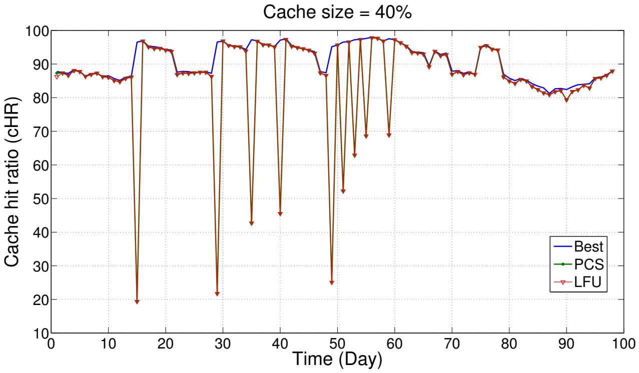


Fig. 17: cHR for Set_3, DES(7, 0.99), cache size=40%.

The cache update ratio is smaller than 0.3 for a cache size of 20% (see Figure 18) and smaller than 0.2 for a cache size of 40% (see Figure 19).

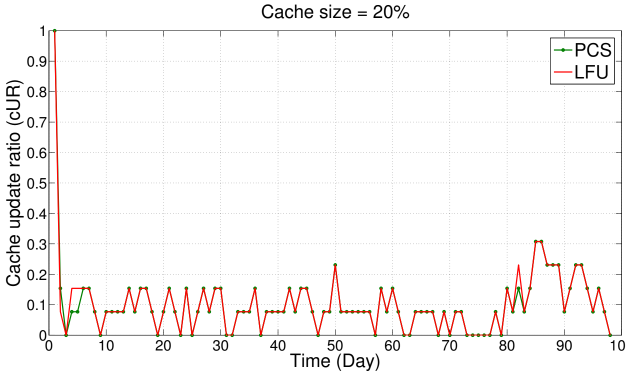


Fig. 18: cUR for Set_3, DES(7, 0.99), cache size=20%.

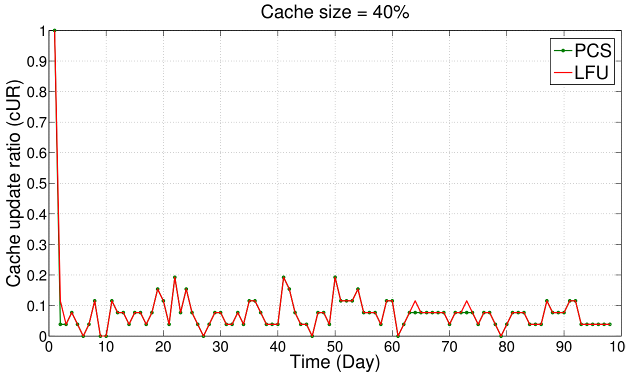


Fig. 19: cUR for Set_3, DES(7, 0.99), cache size=40%.

In conclusion, the performance of PCS is very close to that of LFU. The cache hit ratio is almost the same. However, we notice a slight improvement in the cache update ratio for PCS (3% for Set_3).

VI. Conclusion

The goal of this paper was to show that the use of prediction strategies can help improve caching strategies. We first selected the best experts in charge of predicting the popularity of video contents using real traces from YouTube. We tuned the parameters of the DES expert. We proved that the well-known LFU caching strategy can also be considered as a prediction based strategy on the Basic expert. Simulation results show that the DES Prediction-based caching strategy provides a similar Hit Ratio to LFU. These results are usually close to the optimal ones that can be achieved only when knowing in advance the popularity of each video content for the next day. The exceptions occur when a content whose popularity was very low suddenly becomes very popular (e.g millions of requests). In such a case, the accuracy of the prediction methods becomes poor. This opens up a research direction where knowledge of societal events and trends should be taken into account in order to improve the prediction.

References

[1] M. Calder, X. Fan, Z. Hu, E. Katz-Bassett, J. Heidemann, R. Govindan, *Mapping the expansion of google's serving infrastructure*, ACM IMC, 2013

[2] V. Adhikari, S. Jain, Y. Chen, and Z.-L. Zhang. *Vivisecting youtube: An active measurement study*, IEEE INFOCOM, 2012

[3] Y. Hongliang, Z. Dongdong, Ben Y. Zhao, W.Zheng, *Understanding user behavior in large-scale video-on-demand systems*, EuroSys 2006, New York, USA

[4] N. Ben Hassine, D. Marinca, P. Minet, D. Barth, *Popularity Prediction in Content Delivery Networks*, PIMRC 2015, Hong-Kong, China, September 2015.

[5] N. Megiddo and D. S. Modha, *Outperforming LRU with an adaptive replacement cache algorithm*, Computer, vol. 37, no. 4, pp. 58-65, 2004

[6] P. R. Jelenkovic and A. Radovanovic, *The persistent-access-caching algorithm*, Random Structures and Algorithms, 2008

[7] L.A. Belady, *A Study of Replacement Algorithms for Virtual Storage Computers*, IBM Systems J., vol.5, no. 2, 1966, pp. 78-101.

[8] B. Van Roy, *A short proof of optimality for the MIN cache replacement algorithm*, Information Processing Letters, vol. 102, no. 2-3, pp. 72-73, 2007.

[9] W. Vogler, *Another short proof of optimality for the MIN cache replacement algorithm*, Information Processing Letters, Volume 106, Issue 5, 31 May 2008, Pages 219-220.

[10] N. Bianchides, G. Lugosi, *Prediction, Learning, and Games*, Cambridge University, 2006