On the Merits of Popularity Prediction in Multimedia Content Caching

Jeroen Famaey, Tim Wauters, and Filip De Turck

Department of Information Technology
Ghent University – IBBT
Gaston Crommenlaan 8/201, B-9050 Gent, Belgium
Email: jeroen.famaey@intec.ugent.be

***Abstract*—In recent years, telecom operators have been moving away from traditional, broadcast-driven, television towards IP-based, interactive and on-demand services. Consequently, multicast is no longer a viable solution to limit the amount of traffic in the IP-TV network. In order to counter an explosion in generated traffic, caches can be strategically placed throughout the content delivery infrastructure. As the size of caches is usually limited to only a small fraction of the total size of all content items, it is important to accurately predict future content popularity. Classical caching strategies only take into account the past when deciding what content to cache. Recently, a trend towards novel strategies that actually try to predict future content popularity has arisen. In this paper, we ascertain the viability of using popularity prediction in realistic multimedia content caching scenarios. The use of popularity prediction is compared to classical strategies using trace files from an actual deployed Video on Demand service. Additionally, the synergy between several parameters, such as cache size and prediction window, is investigated.**

I. Introduction

The proliferation of interactive, personalized and on-demand television services is causing an increasing need for bandwidth in telecom operator networks. As broadcasting or multicasting such content is no longer a viable approach to limiting bandwidth consumption, novel techniques need to be considered. Proxy caching, which had already been widely employed in the delivery of web content, has been proposed as a way of offloading bottleneck links [1]. Caches are strategically placed throughout the network and store a subset of the available content. However, the size of such caches is usually limited, so they are only capable of storing a fraction of available content. Therefore, it is very important to accurately predict the future popularity of content, so that the most popular items, or item segments, can be offered closer to the end-users.

Over the years, many caching strategies have been proposed. Classical strategies, such as Least Recently Used (LRU) and Least Frequently Used (LFU), assume that what was most popular in the past, will also be most popular in the future. However, the popularity of multimedia content is known to be highly dynamic [2]. Consequently, caching efficiency can be further increased by taking these dynamics into account and actually try to predict future popularity instead of only using historical information.

The prediction of time series has been a topic of great academic interest for a long time [3]. In the context of multimedia caching, some early work exists on predicting content popularities [4], [5]. However, little work exists that actually links these predictions to the actual caching strategies. Additionally, the effect of important parameters, such as the prediction window size, has not yet been thoroughly evaluated.

In this paper, we contribute to the field of popularity prediction in the context of multimedia content caching in several ways. First, we propose two alternative approaches to popularity prediction, respectively based on recency and frequency principles. Second, we study the theoretical gain that can be achieved by these two popularity predicting caching strategies. This optimum is compared to both the efficiency of classical caching strategies and to the global optimal caching strategy. Third, the effect of the prediction window size parameter on caching efficiency is determined. This parameter is influenced by the cache size. Therefore, the synergy between these parameters is thoroughly evaluated. Fourth, in order to increase the applicability and validity of the presented results, all evaluations are performed using a trace of an actual deployed Video on Demand (VoD) service. This gives our evaluations more leverage and credibility than those performed on synthetically generated datasets. The ultimate goal of this work is to show that popularity prediction indeed improves caching efficiency and to determine under what parameter combinations it achieves the most optimal result.

The rest of this paper is structured as follows. Section II gives a more in depth description of existing work on caching and time series prediction in the context of multimedia content caching. Subsequently, an overview of the employed evaluation methodology is given in Section III. We describe the used VoD trace file, the evaluated caching strategies and the evaluation metrics. In Section IV, the simulation results are discussed. Finally, the paper is concluded in Section V.

II. Related Work

The large size and stringent sequential delivery demands of multimedia content have caused a push towards novel caching strategies. Classical caching algorithms have been adapted to operate on individual content segments instead of entire items [6], [7]. This allows the caches to better utilize the sequential

978-1-4244-9220-6/11/$26.00 ©2011 IEEE

nature of multimedia content demand patterns. Additionally, such techniques better map to the skewed internal popularity of multimedia content. Yu et al. argue that selecting a suitable segment size is a complex problem and therefore propose an alternative solution that models the internal popularity of multimedia streams independent of segment size [8]. Certain IP-TV services have specific properties that can be exploited by caching strategies. For example, the use of sliding-window caches has been proposed in the context of time-shifted TV services [9].

Recently, researchers have discovered the merits of using popularity prediction methods in multimedia content caching. Such methods attempt to predict either the actual future request pattern of individual content items, or their relative popularity compared to each other. Most work on this topic was performed in the context of video-sharing services such as YouTube [10], [4], [5]. Cha et al. found that there is a strong correlation between the popularity of a video after two days and after ninety days [10]. These observations were supported by a study performed by Szabo et al. [5]. An alternative approach was proposed by Avramova et al. [4]. They found that YouTube video popularity traces follow several different distributions, such as power-law or exponential. An analytical model is devised that predicts the distribution associated with specific popularity traces. In the context of VoD services, De Vleeschauwer & Laevens propose a prediction method based on a generic user-demand model derived from traces of VoD and catch-up TV services [2].

In the field of time series prediction, a wide range of techniques have been developed for forecasting all sorts of time series. Recently, machine learning techniques, such as support vector machines and artificial neural networks have been applied to this problem [11], [12]. Recently, wyffels et al. have used reservoir computing, a form of recurrent neural networks, for time series prediction [13]. Additionally, time series often exhibit repeating trends and periodical effects. For example, multimedia content request patterns often show repeating effects on a daily and weekly basis. The use of wavelet decomposition has been proposed to decompose time series into signals with dynamics in different scales. This has been shown to simplify prediction with neural network based techniques [14]. This approach was also successfully combined with reservoir computing [15]. Wu et al. adapted the reservoir computing approach to the popularity prediction problem in the context of multimedia content caching [16]. However, they have not yet compared their prediction method with classical caching strategies. Additionally, their approach has, for now, only been validated on traces of popular YouTube videos.

III. Evaluation Methodology

This section presents the methodology used for evaluating the performance of prediction-based caching strategies. First, an in depth overview is given of the VoD dataset used for the evaluation. Second, the different caching strategies used in the comparison are briefly described. Third, the different algorithmic parameters that influence behaviour are identified. Finally, the metrics used to evaluate the strategies are further explained.

A. Video on Demand Dataset

The dataset employed in the evaluation consists of a request trace of the VoD service of a leading European telecom operator, measured over a period of 32 days between Friday February 5 2010 and Monday March 8 2010. Within this period, a total of 75013 requests were sent by 8392 unique users for 4971 different movies. These users were spread across 12 city regions.

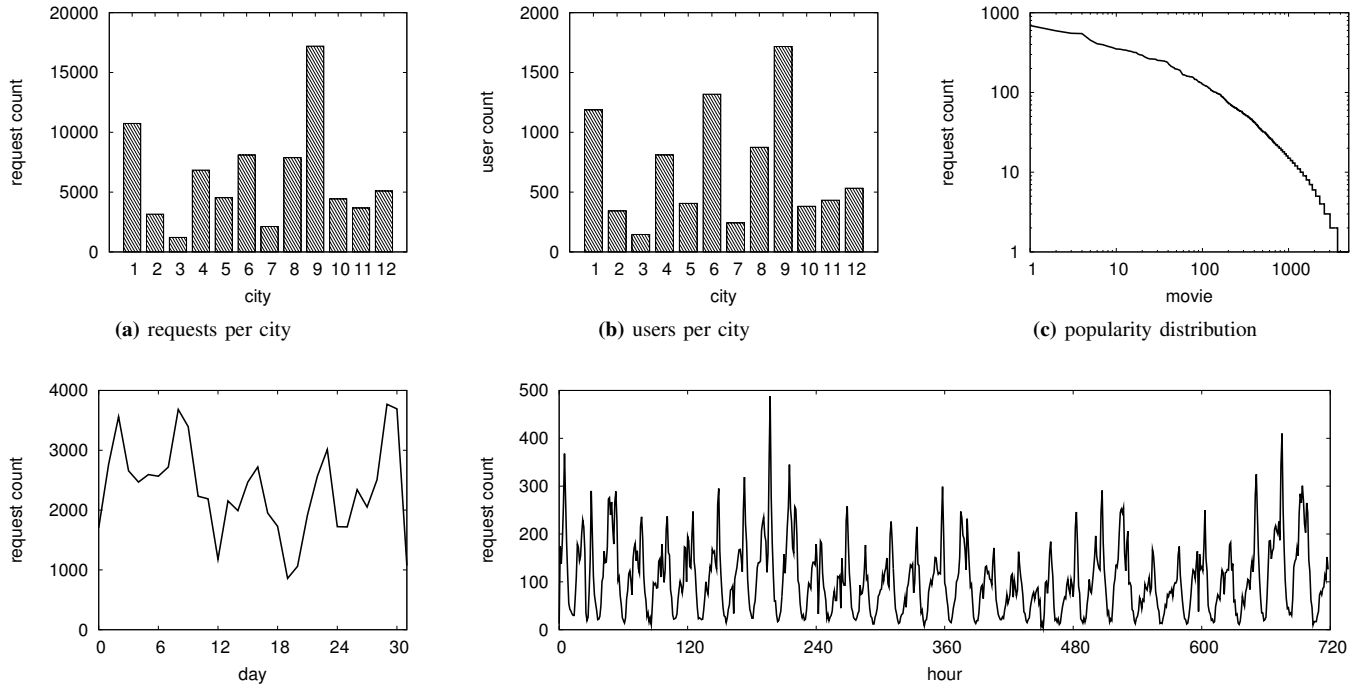
Figure 1 graphically depicts the properties of the dataset. The distribution of requests over the different city regions is shown in Figure 1a. It is apparent that requests are far from uniformly distributed among cities. The distribution of users across cities is shown in Figure 1b. Although the two figures show some correlation between request and user count, they are not entirely parallel. For example, city 6 has the second highest user count, but only the third highest request count. Subsequently, the popularity distribution over the movies is presented in Figure 1c. The popularity distribution is highly skewed. A total of 691 requests were measured for the most popular movie, while 10 or less requests were received for over 72% of all movies. Figure 1d depicts the request count per day. The figure clearly shows the weekly trend in the dataset. The five peaks represent the five weekends part of the trace. Finally, the hourly request rate is shown in Figure 1e and explicitly depicts the daily trend. On weekdays, two peaks are observed. A first, smaller, peak starts as early as 1 pm and lasts until about 5 pm. The second peak occurs during the evening from approximately 8 pm until midnight. On Saturday and Sunday, high request rates persist from 9 am until midnight.

B. Caching Strategies

In order to thoroughly evaluate prediction-based caching strategies, we compare a perfect predicting algorithm with several other caching strategies. This section gives an overview of the strategies used in the evaluation.

1) Least Recently Used (LRU): The LRU caching strategy always replaces the least-recently-used object in cache [17]. A sorted queue is maintained that indicates the request order of objects. When an object is requested, it is added to the queue, or its position is updated. If the queue is longer than the size of the cache, the last object is removed from it. 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.

2) Least Frequently Used (LFU): LFU is a more elaborate but also complex strategy than LRU. It keeps track of the frequency of every object and keeps the objects in cache with the highest request frequencies within a specific time window [17]. If multiple items have the same 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 a suitable time window. If the window size is too



(a) requests per city **(b)** users per city **(c)** popularity distribution

(d) requests per day **(e)** requests per hour

Fig. 1: A graphical representation of the Video on Demand dataset properties

small, the popularity dynamics are not sufficiently caught. If, on the other hand, the window size is too large, popularity is calculated based on stale information.

3) Optimal Strategy (MIN): The MIN algorithm is an optimal caching strategy in terms of cache hit rate [18]. When an object is requested that is not part of the cache and the cache is full, it replaces the object in the cache whose next request occurs furthest in the future. Several proofs have been devised to show its optimality [18], [19]. However, this strategy is impractical in an actual deployment, as the request trace needs to be known entirely in advance.

4) Predicting Least Recently Used (P-LRU): The P-LRU strategy is the prediction-based variant of LRU. It works much the same way as MIN, as it replaces the object whose next request occurs furthest in the future. However, we assume that the strategy is only capable of accurately predicting requests within a certain time window. Everything beyond this window is unknown. If multiple cached objects are not requested within the specified time window and one needs to be selected for replacement, the classic LRU strategy is used to select one of them.

5) Predicting Least Frequently Used (P-LFU): Predicting the actual order in which requests will occur, as is needed for P-LRU is very difficult. Predicting actual or relative request frequencies within a certain time frame is expected to be easier. Therefore, we propose a second prediction-based heuristic that uses the LFU principles. In contrast to LFU, it predicts request frequencies of objects within a time window in the future, instead of keeping track of those in the past. If multiple objects have the same frequency, a differentiation is made using the classic LRU strategy.

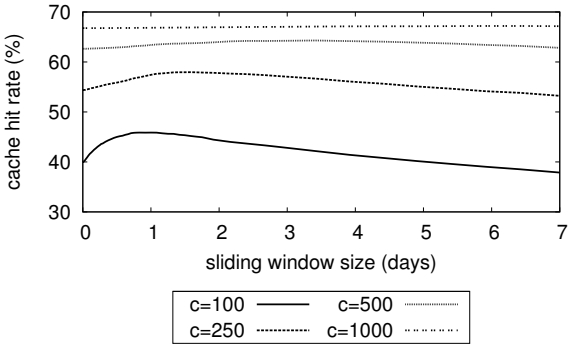
C. Evaluation Metrics

Two metrics are used in the evaluation of the caching strategies. They are the *cache hit rate* and *cache update rate* respectively. The cache hit rate is defined as the percentage of requests that can be served from a cache, as opposed to from the origin content server. The cache update rate, on the other hand, represents the percentage of requests that cause a change in the cached objects.

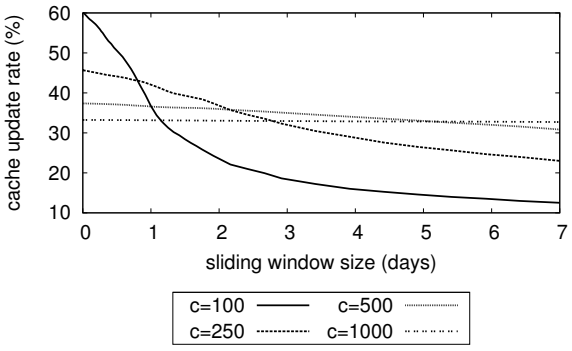
IV. Results & Discussion

In order to evaluate performance of the devised prediction-based caching strategies (i.e. P-LRU and P-LFU), they were compared to both the optimal caching strategy (i.e. MIN) and classic strategies (i.e. LRU and LFU). Note that the devised prediction-based strategies are capable of perfectly predicting the future within a certain period of time. Obviously, an actual strategy cannot achieve such accurate predictions. Nevertheless, this allows us to determine the theoretical optimum that can be achieved by prediction-based strategies under specific conditions.

The interaction between two parameters is studied in the evaluation. First, several of the presented strategies use a *sliding window* parameter of some sort. For LFU, this window determines how far back the algorithm will look when calculating frequencies. For P-LRU and P-LFU, it indicates how far in the future the strategies are capable of predicting requests.



(a) cache hit rate



(b) cache update rate

Fig. 2: Performance of the classic LFU strategy as a function of the sliding window size for different cache sizes c

Second, the *cache size* represents the number of objects that can be kept in the cache. It is expected that the optimal window size is influenced by the cache size. Therefore, the synergy between both parameters is evaluated. Note that, for simplicity, we assume all objects have the same size and entire objects are stored in cache. In multimedia content caching these assumptions normally do not hold. However, as all evaluated caching strategies are subject to these assumptions, we believe they do not influence the comparative conclusions drawn in this paper.

The simulated content delivery infrastructure consists of a single content server, which hosts all offered movies. A content proxy cache is provided for each of the 12 city regions. The end-users request content via the proxy cache of their own region, which in turn requests the content with the server if it is not cached locally. In the performed experiments, the same cache size is used for every regional cache. The results depicted in this section for cache hit rate and cache update rate are averaged over the 12 caches.

A. Influence of LFU Sliding Window

In this section, the interaction between the LFU sliding window parameter and the cache size is thoroughly evaluated. The goal is to determine suitable values of the sliding window parameter for use in the rest of the evaluation.

The cache hit and update rates of the LFU caching strategy as a function of the sliding window size (in days) for different cache sizes is shown in Figure 2. The graph depicted in Figure 2a leads to several pertinent observations concerning the cache hit rate. First, for smaller cache sizes, the LFU sliding window size affects cache hit rate to a greater extent than for larger cache sizes. For example, for a cache size of 100 objects, the cache hit rate for the optimal window size (1 day) is 45.87%, while it is only 37.86% for a window size of 7 days. On the other hand, for a cache size of 500 objects, the cache hit rate difference between the optimum and worst case is only 1.41%. Second, the optimal sliding window size is directly proportional to the size of the cache. In the depicted results, the optimum window size for cache sizes of 100, 250, 500 and 1000 objects is respectively 1 day, 1 day and 14 hours, 3 days and 10 hours and over 7 days.

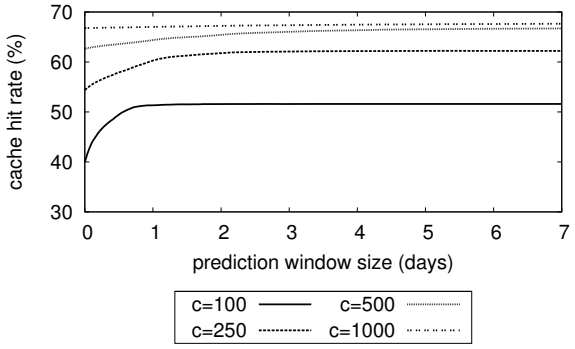
The graph shown in Figure 2b displays the cache update rate. Much like for the cache hit rate, the update rate is less influenced by the LFU sliding window size when the cache size is large. However, in contrast to the hit rate, the update rate does not achieve an optimum and keeps decreasing with a growing window size. Therefore, the larger the LFU sliding window size, the more stable the cache will become.

In conclusion, we have shown that the optimal sliding window size for LFU is highly dependant on the cache size. Additionally, its influence becomes negligible when a relatively high percentage of the total available objects can be stored in cache. However, in a real deployment the cache is expected to be able to store only a small fraction of available content. Therefore, selecting a suitable sliding window size is a crucial step when deploying a caching infrastructure in a multimedia content delivery network. In the evaluated VoD dataset, a sliding window size of 1 day was found to perform at most 0.89% less than optimal in terms of cache hit rate for all evaluated cache sizes. As such, a 1 day sliding window is used for LFU in the rest of this paper.

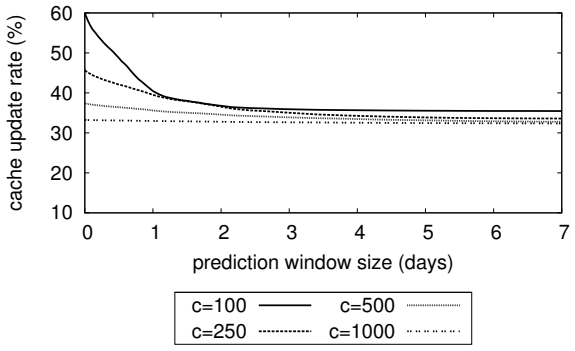
B. Influence of P-LRU Prediction Window

The two proposed predicting strategies, P-LRU and P-LFU, use a prediction window parameter to determine the amount of future information they take into account when making cache update decisions. This section explores the synergy between the prediction window size and the cache size for the P-LRU strategy.

The cache hit and update rates of P-LRU as a function of prediction window size (in days) for different cache sizes are shown in Figure 3. From Figure 3a, showing the cache hit rate, we can derive that performance for P-LRU does not degenerate when increasing the prediction window size beyond the optimum. As is the case for LFU, the optimum prediction window size increases with the cache size. For example, for a cache size of 100 objects, the cache hit rate comes within 0.1% of optimal performance for a prediction window size of 1 day and 8 hours, while for a cache size of 500 objects this is only achieved at 2 days and 22 hours. Additionally, the effect of optimizing the prediction window is directly proportional to the cache size. For a cache size of 100 objects, a gain of

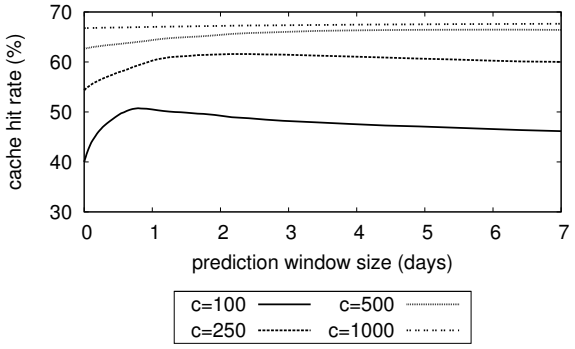


(a) cache hit rate

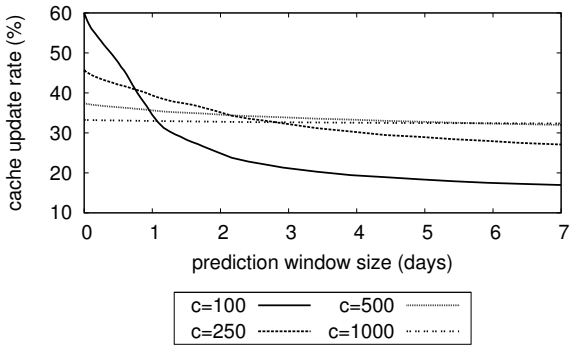


(b) cache update rate

Fig. 3: Performance of the predicting P-LRU strategy as a function of the prediction window size for different cache sizes c



(a) cache hit rate



(b) cache update rate

Fig. 4: Performance of the predicting P-LFU strategy as a function of the prediction window size for different cache sizes c

11.59% can be achieved, while for a cache size of 500 objects this is limited to a 4% gain.

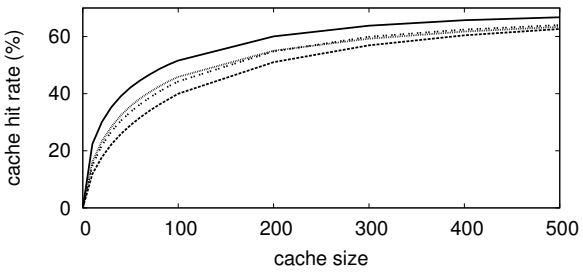
Figure 3b shows the cache update rate for P-LRU. As is the case for the cache hit rate, the update rate does not improve significantly after a certain point. For a cache size of 100 objects, there is only a 1% difference in cache update rate between a prediction window size of 2 days and 7 days. Additionally, the figure clearly shows that for a large enough prediction window (3 days or more) the size of the cache has very little influence on the cache update rate (at most 3.2%).

In summary, we have shown that a large prediction window has relatively few benefits for P-LRU, as performance does not significantly improve after a certain window size. However, it should be noted that a P-LRU-based prediction strategy should be capable of accurately predicting the relative ordering of request arrivals within a certain time window. This is very difficult compared to other approaches, such as predicting relative request frequencies, as used by P-LFU. Therefore, we believe large prediction windows cannot be accurately achieved in an online P-LRU-based caching strategy. Additionally, the results show that performance of P-LRU is less sensitive to the window size, as unlike LFU, its performance does not degenerate when the window size becomes too large. Unless otherwise stated, a prediction window size of 1 day is used for P-LRU in the rest of this paper, which achieves a cache hit rate of at most 2.3% worse than the optimum for all evaluated cache sizes.

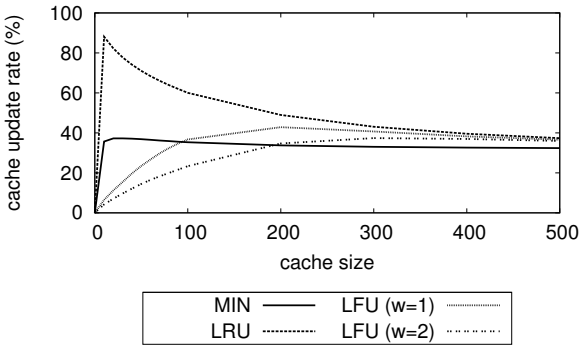
C. Influence of P-LFU Prediction Window

In this section, we evaluate the influence of the prediction window size on the performance of P-LFU in terms of cache hit and update rate. The results of this evaluation are shown in Figure 4. It depicts the cache hit and update rates as a function of prediction window size (in days) for different cache sizes. At first glance, the results for P-LFU's prediction window are very similar to LFU's sliding window. However, there are some differences. First, P-LFU's performance in terms of hit rate degenerates slower than LFU's when the window size becomes too large. For a cache size of 100 objects, the difference between the optimal window size (1 day) and 7 days is only 4.33% for P-LFU while it is 8% for LFU. Second, for a small window size the roles are reversed. For example, for a cache size of 100 objects, when increasing the window size from 1 minute to 1 day, P-LFU's hit rate increases 10.46%, while that of LFU only increases 5.89%.

In general, the conclusions drawn for LFU's sliding window also apply to P-LFU's prediction window. As such, the optimal prediction window size is influenced by the cache size, but this influence decreases as the cache size grows. However, in contrast to LFU, estimating the optimal cache size too low has more far fetched consequences for P-LFU. On the other hand, estimating it too high has less effect on performance



(a) cache hit rate



(b) cache update rate

Fig. 5: Comparison of the MIN, LRU and LFU strategies as a function of the cache size for an LFU sliding window w of 1 and 2 days

when using P-LFU rather than LFU. Unless otherwise stated, a prediction window of 1 day is used in the rest of this paper. This value achieves a cache hit rate of at most 2.07% worse than optimal.

D. Traditional Strategy Comparison

In the previous sections, a suitable window size was determined for LFU, P-LRU and P-LFU. In this, and the next, section, we use previously made observations to compare the optimal (i.e. MIN), classical (i.e. LRU and LFU) and prediction-based strategies (i.e. P-LRU and P-LFU). First, we determine the efficiency of the classical strategies. These results are then employed in the next section when comparing the novel prediction-based approaches.

Figure 5 shows a comparison in terms of cache hit and update rates of the two classical strategies and the optimal MIN strategy. The graphs depict performance as a function of the cache size and for a 1 and 2 day LFU sliding window. From Figure 5a, which shows the cache hit rate, it is apparent that LRU performs worse than LFU for well chosen sliding window parameters. As was shown in Section IV-A, LFU with a sliding window of 1 day outperforms a 2 day sliding window for smaller cache sizes. However, when the cache size grows beyond 300 objects the 2 day window takes over the lead. Concretely, for a cache size of 100 objects LRU's cache hit rate is 11.61% worse than optimal, while that of LFU with a window size of 1 day is only 5.725% worse. However, LRU and LFU performance converges as the cache size increases. For a cache size of 500 objects, LRU and LFU respectively achieve a cache hit rate of only 4.1% and 3.84% worse than optimal.

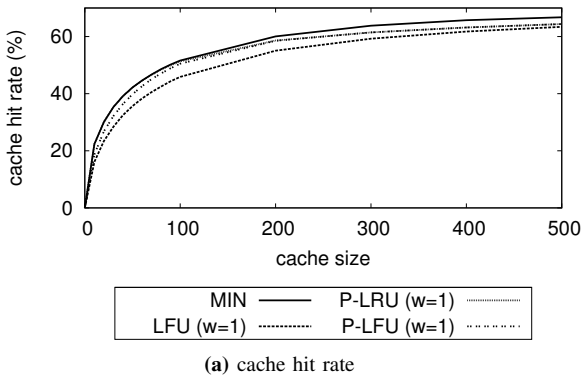
Performance in terms of cache update rate is depicted in Figure 5b. Here, LFU clearly has the edge over LRU and even MIN. For small cache sizes, LFU needs even less updates than the optimal strategy. On the other hand, LRU's peak goes up to almost a 90% update rate. This goes to show that LFU makes more long term decisions, while LRU causes the cache content to oscillate considerably. However, as is the case for cache hit rate, performance of both classical strategies converges as the cache size grows.

In conclusion, we have shown that LFU, with a well chosen sliding window parameter, outperforms the simple LRU strategy for all cache sizes, both in terms of cache hit and update rate. Additionally, the difference is larger for smaller cache sizes, which we expect will remain the norm in actual deployments. Additionally, LFU's performance in terms of cache hit rate, for a sliding window of 1 day is at most 6.87% worse than optimal (reached at a 30 object cache size). This goes to show that for the real-life VoD dataset used in this evaluation, only a small margin of possible improvement remains for prediction-based strategies. As LRU does not outperform LFU in any evaluated situation, solely LFU will be used as a benchmark of comparison in the next section.

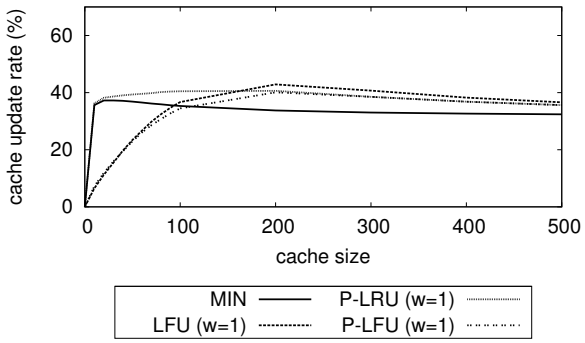
E. Predicting Strategy Comparison

This paper proposes two alternative prediction-based approaches for caching strategies in multimedia content delivery networks. First, P-LRU represents a predicting variant of the classic LRU caching scheme. The optimal MIN algorithm is actually P-LRU with an infinite prediction window. Second, P-LFU makes caching decisions based on request frequencies and is thus the predicting counterpart of the classic LFU caching strategy. As stated earlier, an online P-LFU-approximating strategy is expected to be easier to implement than one based on P-LRU. This is because predicting relative request frequencies is easier than predicting the actual order in which requests will take place. However, besides this qualitative difference, there are also quantitative performance differences between both caching strategies. In this section, these quantitative differences are evaluated.

The comparison, as a function of cache size, is shown in Figure 6. For LFU, P-LRU and P-LFU, a window size of 1 day is used. Figure 6a, depicting the cache hit rate, shows that both prediction-based strategies significantly outperform LFU for relatively small cache sizes. However, for larger cache sizes, the performance gain becomes minimal. For a cache size of up to 100 objects, P-LRU performs less than 0.27% worse than optimal, while P-LFU performs 1.1% worse and classic LFU even 5.72% worse. However, when the cache size increases to 500 objects this becomes for LFU, P-LRU and P-LFU respectively 3.34%, 2.36% and 2.36%. Their performance thus



(a) cache hit rate



(b) cache update rate

Fig. 6: Comparison of the MIN, LFU, P-LRU and P-LFU strategies as a function of the cache size for a window of 1 day

converges. Table I gives a more complete overview of the cache hit rate of the different caching strategies.

Performance in terms of cache update rate is shown in Figure 6b. In line with earlier results, LFU and P-LFU achieve the best performance at low cache sizes. However, as the cache size increases beyond 200 objects, performance of LFU, P-LRU and P-LFU converges.

In summary, we have shown that using prediction-based caching strategies can indeed improve performance in terms of cache hit rate compared to classical strategies. For a prediction window of 1 day and a cache size of 100 objects, a gain in cache hit rate of 5.45% can be achieved when using prediction. Although this is not an excessive improvement, it goes to show that bandwidth consumption can be further decreased by devising more intelligent caching strategies, even in real VoD scenarios. Additionally, from the results it can be concluded that P-LRU achieves near optimal performance in terms of cache hit rate for very small cache sizes, even with relatively small prediction windows. On the other hand, for larger cache sizes, P-LFU with a relatively large prediction window can be used. For such large cache sizes, P-LFU achieves similar performance as P-LRU. However, predicting request frequencies is easier than the actual request pattern order, especially for large prediction windows. As such, the choice for P-LFU is obvious.

TABLE I: Summary of the cache hit rate of the MIN, LRU, LFU, P-LRU and P-LFU strategies for different cache sizes and a window of 1 day

Cache Size	Cache Hit Rate (%)				
	MIN	LRU	LFU	P-LRU	P-LFU
10	22.36	11.89	16.14	22.33	18.70
20	30.09	17.76	23.41	30.01	26.80
30	35.31	22.27	28.43	35.31	32.17
40	39.14	25.90	32.47	39.13	36.38
50	42.16	28.97	35.65	42.13	39.79
60	44.62	31.65	38.33	44.53	42.61
70	46.71	34.01	40.56	46.60	45.11
80	48.60	36.14	42.51	48.43	47.10
90	50.20	38.09	44.36	49.94	48.91
100	51.60	39.98	45.87	51.33	50.49
200	60.04	51.03	55.06	58.59	58.53
300	63.80	56.93	59.26	61.44	61.11
400	65.72	60.44	61.74	63.18	63.18
500	66.75	62.64	63.41	64.38	64.38
1000	67.81	66.77	66.85	67.02	67.02

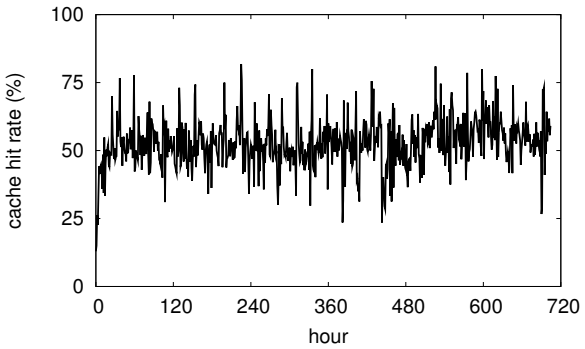


Fig. 7: Cache hit rate of the P-LFU caching strategy averaged over 1 hour intervals, for a cache size of 100 objects and a prediction window of 1 day

F. Hit Rate Evolution

In previous parts of the evaluation, the average cache hit rate was used as a basis for performance comparisons. However, when dimensioning the content delivery network, not the average, but the peak cache hit rate and bandwidth consumption are the most important indicators. Therefore, we study the evolution of the cache hit rate, averaged over 1 hour intervals, throughout the 1 month duration of the input trace. Figure 7 shows this evolution for the P-LFU caching strategy, a cache size of 100 objects and a 1 day prediction window.

The graph allows us to study the cache hit rate evolution over time, and leads to several conclusions. First, note that the average cache hit rate over the entire input trace for this scenario is around 52%. It takes the cache about 13 hours before this average performance is first reached. We thus consider the first 12 hours a warm-up period. Second, it is apparent that the cache hit rate varies greatly over time. After the warm-up period, the cache hit rate varies between 81.82% and 23.53%, with a standard deviation of 8.67%. This standard deviation means that 68.2% of one-hour intervals have a cache hit rate between 44.25 and 61.57%.

V. Conclusion

In this paper, we aim to determine the merits of using popularity prediction in a multimedia content caching scenario. As such, we have introduced and evaluated two alternative approaches to prediction-based caching in multimedia content delivery networks. First, P-LRU is the predicting variant of the classic LRU strategy. It predicts the order in which requests occur within a pre-specified future time window, and replaces objects which will be requested furthest in the future. Second, P-LFU represents the classic LFU predicting counterpart. Instead of actual request orders, it predicts only request frequencies within the specified time window. Although both these strategies require perfect knowledge of future request patterns within a specific time interval and thus have no practical applicability, they allow us to determine the theoretical gain that can be achieved by prediction-based caching strategies over traditional ones.

Using detailed evaluations, the synergy between the cache size and prediction window size was explored. Additionally, we compared performance, in terms of cache hit and update rate, of the two prediction-based caching strategies with the classic LRU and LFU strategies and the theoretical optimum in terms of cache hit rate. These evaluations lead to several pertinent conclusions. First, it was shown that the optimal prediction window size is indeed severely impacted by the size of the cache. More specifically, the optimal prediction window size is directly proportional to the size of the cache. For P-LRU, performance only degenerates if the prediction window is chosen too small. However, for P-LFU, this is also the case if the prediction window is chosen too large. Therefore, choosing a suitable prediction window is more difficult in the case of P-LFU. Second, it can be concluded that both P-LRU and P-LFU significantly outperform LFU and LRU in terms of cache hit rate. For example, for a cache size of 100 objects, which is about 2% of the total available content, P-LFU's cache hit rate is up to 4.6% higher than that of LFU and even 10% than that of LRU, which is a relative performance increase of respectively 10 and 25%. Third, when the cache size is small P-LRU can be used with a small prediction window to achieve near optimal performance. When the cache size becomes larger, a proportionally larger prediction window is also needed. Then, P-LFU, with a large prediction window, can be used. This maps well to the advantages and disadvantages of both strategies, as P-LRU generally achieves more optimal results, but P-LFU's request frequencies are more easily approximated on the long run by online prediction algorithms.

In summary, the two prediction-based caching strategies presented in this paper assume a perfect prediction is possible within a certain time window. Although actual online strategies are not able to achieve such perfect predictions, this study nevertheless determined the theoretical optimum that can be achieved by introducing predictions. Additionally, we have shown that the overall cache hit rate can be improved by up to 6.48% compared to the traditional LFU strategy, which is a relative performance gain of 18%.

Acknowledgment

Jeroen Famaey is funded by the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT). Tim Wauters is funded by the Fund for Scientific Research Flanders (FWO). The research leading to these results has received funding from the European Union's Seventh Framework Programme ([FP7/2007-2013]) under grant agreement number 248775. The authors would sincerely like to thank Femke De Backere for her valuable feedback.

References

[1] M. Allen, B. Zhao, and R. Wolski, "Deploying video-on-demand services on cable networks," in *27th International Conference on Distributed Computing Systems (ICDCS '07)*, 2007, pp. 63–63.

[2] D. De Vleeschauwer and K. Laevens, "Performance of caching algorithms for iptv on-demand services," *IEEE Transactions on Broadcasting*, vol. 55, no. 2, pp. 491–501, 2009.

[3] S. Makridakis, "Time series prediction: Forecasting the future and understanding the past," *International Journal of Forecasting*, vol. 10, no. 3, pp. 463–466, 1994.

[4] Z. Avramova, S. Wittevrongel, H. Bruneel, and D. De Vleeschauwer, "Analysis and modeling of video popularity evolution in various online video content systems: Power-law versus exponential decay," in *First International Conference on Evolving Internet*, 23-29 2009, pp. 95–100.

[5] G. Szabo and B. A. Huberman, "Predicting the popularity of online content," *Communications of the ACM*, vol. 53, no. 8, pp. 80–88, 2010.

[6] K.-L. Wu, P. Yu, and J. Wolf, "Segmentation of multimedia streams for proxy caching," *IEEE Transactions on Multimedia*, vol. 6, no. 5, pp. 770–780, 2004.

[7] S. Chen, H. Wang, X. Zhang, B. Shen, and S. Wee, "Segment-based proxy caching for internet streaming media delivery," *IEEE Multimedia*, vol. 12, no. 3, pp. 59–67, 2005.

[8] J. Yu, C. Chou, Z. Yang, X. Du, and T. Wang, "A dynamic caching algorithm based on internal popularity distribution of streaming media," *Multimedia Systems*, vol. 12, no. 2, pp. 135–149, 2006.

[9] T. Wauters, W. Van de Meerssche, P. Backx, F. De Turck, B. Dhoedt, P. Demeester, T. Van Caenegem, and E. Six, "Proxy caching algorithms and implementation for time-shifted tv services," *European Transactions on Telecommunications*, vol. 19, no. 2, pp. 111–122, 2008.

[10] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (IMC '07)*. ACM, 2007.

[11] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, *Neural Networks*, vol. 20, pp. 391–403, 2007.

[12] R. Samsundin, A. Shabri, and P. Saad, "A comparison of time series forecasting using support vector machine and artificial neural network model," *Journal of Applied Sciences*, vol. 10, no. 11, pp. 950–958, 2010.

[13] F. wyffels and B. Schrauwen, "A comparative study of reservoir computing strategies for monthly time series prediction," *Neurocomputing*, vol. 73, pp. 1958–1964, 2010.

[14] S. Soltani, "On the use of the wavelet decomposition for time series prediction," *Neurocomputing*, vol. 48, pp. 267–277, 2002.

[15] F. wyffels, B. Schrauwen, and D. Stroobandt, "Using reservoir computing in a decomposition approach for time series prediction," in *European Symposium on Time Series Prediction*, 2007, pp. 149–158.

[16] T. Wu, "On the use of reservoir computing in popularity prediction," in *The Fifth International Multi-Conference on Computing in the Global Information Technology (ICCGI)*, 2010.

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

[18] J. Gecsei, D. R. Slutz, and I. L. Traiger, "Evaluation techniques for storage hierarchies," *IBM Systems Journal*, vol. 9, no. 2, pp. 78–117, 1970.

[19] 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.