Towards a predictive cache replacement strategy for multimedia content

Jeroen Famaey*, Frédéric Iterbeke, Tim Wauters, Filip De Turck

Ghent University – IBBT, Department of Information Technology, Gaston Crommenlaan 8 bus 201, B-9050 Gent, Belgium

Abstract

In recent years, telecom operators have been moving away from traditional broadcast-driven television, towards IP-based interactive and on-demand multimedia services. Consequently, multicast is no longer sufficient to limit the amount of generated traffic in the network. In order to prevent an explosive growth in 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. Traditional 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 article, we ascertain the viability of using popularity prediction in realistic multimedia content caching scenarios. The proposed generic popularity prediction algorithm is capable of predicting future content popularity, independent of specific content and service characteristics. Additionally, a novel cache replacement strategy, which employs the popularity prediction algorithm when making its decisions, is introduced. A detailed evaluation, based on simulation results using trace files from an actual deployed Video on Demand service, was performed. The evaluation results are used to determine the merits of popularity-based caching compared to traditional strategies. Additionally, the synergy between several parameters, such as cache size and prediction window, is investigated. Results show that the proposed prediction-based caching strategy has the potential to significantly outperform state-of-the-art traditional strategies. Specifically, the evaluated Video on Demand scenario showed a performance increase of up to 20% in terms of cache hit rate.

Keywords: multimedia content delivery, caching strategies, popularity prediction

*Corresponding author, tel: +3293314938, fax: +3293314899

Email addresses: jeroen.famaey@intec.ugent.be (Jeroen Famaey), tim.wauters@intec.ugent.be (Tim Wauters), filip.deturck@intec.ugent.be (Filip De Turck)

1. Introduction

The proliferation of interactive, personalized and on-demand television services is causing an increasing need for bandwidth in telecom operator networks. Obviously, broadcasting or multicasting cannot sufficiently reduce bandwidth consumption of on-demand multimedia services. 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] in on-demand scenarios. 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. Traditional 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 directly applying historical information.

Predicting the future popularity of individual multimedia content items can be reduced to a time series prediction problem [3]. Several efforts have been made to apply this theory to the prediction of multimedia content popularity [4, 5]. However, to our knowledge, these predictions have never been integrated into an actual cache replacement strategy. Additionally, the effect of important parameters, such as the prediction window size, has not yet been thoroughly evaluated.

This article presents a generic popularity prediction algorithm. In contrast to existing algorithms, it is not tailored to a specific service. The prediction algorithm fits a set of functions to the cumulative request pattern of content items. These fitted curves are then used as an approximated model of the request pattern. Through extrapolation the future of the request pattern can then be estimated. Subsequently, we present a novel prediction-based cache replacement strategy. It uses the predicted request patterns to determine the subset of all available content to store in the cache. Additionally, to assess the theoretical maximal gain in caching efficiency that can be achieved using predictions, a theoretical variant is also presented. It assumes the future can be perfectly predicted.

The proposed cache replacement strategies are thoroughly evaluated and compared to traditional strategies that directly employ historical information. The goal of this evaluation is to determine both the theoretical and practical gain in caching efficiency that can be achieved using popularity prediction. Moreover, the effect of the prediction window parameter is assessed. This parameter is defined as the future time-frame that is predicted (i.e., the counterpart of the history window parameter of LFU). The effect of this parameter is influenced by the cache size. Therefore, the synergy between these parameters is thoroughly

evaluated. 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 of a leading European telecom operator. This gives our evaluations more leverage and credibility than those performed on synthetically generated datasets. The ultimate goal of this study is to show that popularity prediction indeed improves caching efficiency and to determine under what circumstances it achieves the most optimal result.

The remainder of this article is structured as follows. Section 2 gives a more in depth description of existing work on popularity prediction of multimedia content. Section 3 presents our proposed generic popularity prediction algorithm and cache replacement strategy that uses it. Subsequently, Section 4 evaluates the proposed cache replacement strategy using simulation results. Finally, the paper is concluded in Section 5.

2. Related work

The large size and stringent sequential delivery demands of multimedia content have caused a push towards novel caching strategies. Traditional caching strategies have been adapted to operate on individual content segments instead of entire items [6, 7]. This allows the caches to better utilize the sequential nature of multimedia content demand patterns. Additionally, such techniques better map to the skewed internal popularity of multimedia content. Yu et al. [8] 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. Kim and Das [9] further extended the work on segment-based caching, by way of an analytical model that exploits the temporal locality and popularity of content. Guo *et al.* [10] studied a prefix caching method that exploits the fact that the beginning of a movie is more popular than the ending. Their algorithm automatically determines the optimal number of segments from the beginning of each movie that should be cached, based on the internal popularity distribution of the movie. Certain IPTV 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 [11]. In line with our work, these techniques aim to improve caching efficiency. Nevertheless, they focus on a different aspect, which falls outside the scope of this article.

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 [12, 13]. Recently, wyffels et al. [14] have used reservoir computing, a form of recurrent neural networks, for time series prediction. 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 [15]. This approach was also successfully combined with reservoir computing [16].

Recently, several studies have been conducted on modelling the popularity of multimedia content. These studies can be split into two types. A first type focuses on characterizing the popularity distribution among different multimedia objects, while the second type focuses on modelling the popularity evolution of individual multimedia files.

A popularity distribution among multiple multimedia objects models the static popularity relationship between the content items offered by a multimedia service. It can be used to derive the probability that the content item with a specific popularity index (e.g., the X^{th} most popular item) will be requested. Many models have been proposed for modelling the popularity distribution of a multimedia service, including Zipf [17], Zipf-Mandelbrot [18], stretched exponential [19], Zipf with exponential cut-off tail [20], power-law with exponential cut-off tail [21], log-logistic [22] and Weibull [22].

Other research has focused on modelling the dynamic popularity evolution of individual content items. It thus allows the request evolution of content to be estimated, based on historical request information. This latter type of research is also the focus of our work. Most work on this topic was performed in the context of video-sharing services such as YouTube. Cha *et al.* [20, 23] found that there is a strong correlation between the popularity of a video after two days and after ninety days. These observations were supported by a study performed by Szabo *et al.* [5]. More recently, Chatzopoulou *et al.* [24] studied the correlation between popularity and a wider range of metrics. They found that the popularity of a video is highly correlated with the amount of posted comments, ratings and favourites. Figueiredo *et al.* [25] characterised the popularity evolution of YouTube videos. They found that several different categories exist that exhibit distinct popularity evolutions. For example, copyright protected videos got most of their views early in their lifetime. Additionally, they identified and quantified the main referrers that lead users to videos, as they are key mechanisms in attracting users and thus highly influence popularity evolutions. 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. Jamali and Rangwala [26] studied the evolution of popularity on the social news website Digg. Based on comment data and the co-participation network, they are able to accurately predict the future popularity of any news item shortly after it has been posted. In the context of VoD services, De Vleeschauwer & Laevens [2] propose a prediction method based on a generic user-demand model derived from traces of VoD and catch-up TV services. Wu *et al.* [27] adapted the previously mentioned reservoir computing approach to the popularity prediction of multimedia content. Niu *et al.* [28] employ time-series analysis techniques to predict future content popularity, online population, peer upload and server bandwidth consumption in peer-to-peer VoD streaming systems. Specifically, they use an algorithm based on the Box-Jenkins approach [29] to predict fu-

ture popularity and regression methods to infer initial popularity evolutions of flash-crowds. In summary, some work has been done on the modelling and prediction of content popularity evolution. However, these previous studies have not applied this work to content caching. This application is the focus of our work.

This article extends previous work by ourselves [30], where the theoretical variant of the predictive cache replacement strategy was presented. Here, a practical prediction algorithm is proposed and combined with the previously introduced strategy. This additionally allows us to characterise the influence on performance of prediction errors.

3. Prediction-based caching

This section describes the proposed generic prediction-based caching strategy. The strategy consists of two parts; an algorithm to predict content popularity (cf. Section 3.1) and a caching strategy that determines what content to cache based on these predictions (cf. Section 3.2). The remainder of this section gives an in-depth overview of these two parts.

3.1. Predicting content popularity

This section presents a generic algorithm to predict the content popularity evolution of multimedia content. Concretely, the prediction algorithm estimates the evolution of the content's cumulative request pattern, based on historical information. It uses non-linear optimization techniques to fit a given set of models to the historical input data. These fitted models can subsequently be used to extrapolate the request pattern's future evolution. The remainder of this section provides a more in depth and formal overview of the algorithm.

Given is a cumulative request pattern $R_c(t)$, which is a function of time representing the total number of perceived requests of content object c up to time t. The goal of the popularity prediction algorithm is to estimate the value of $R_c(t_1)$ at some future point in time $t_1 > t$, given the value of $R_c(t_2)$ for (a representative sample of) all moments in time $t_2 \leq t$. The algorithm approximates the request history up to time t_2 with a set of popularity distribution models $\mathcal{D}$. A popularity distribution $D(t) \in \mathcal{D}$ is a function of time that mathematically models request patterns. It is characterized by a set of parameters P_D. The presented algorithm supports an arbitrary set of popularity distributions. However, we have identified four that cover a wide range of request patterns:

- **Constant**: This distribution is capable of modelling unpopular content that receives no or very few requests over long periods of time. Additionally, it supports the modelling of a constant request rate. Its parameter set P_D consists of two parameters a and b, which respectively represent the slope and intercept. The associated cumulative distribution represents the linear function and is expressed as follows:

$$D(t, a, b) = a \times t + b \tag{1}$$

- **Power-law**: Allows the modelling of steep changes in popularity. It has been previously proposed in literature as a model for cumulative request patterns of multimedia content [4]. It is characterized by two parameters, C and α, which respectively represent the normalization constant and scaling factor. The cumulative distribution function is defined as follows:

$$D\left(t, C, \alpha\right) = C \times t^{\alpha} \tag{2}$$

- **Exponential**: In contrast to power-law, this distribution is used to model more rounded and slow changes in popularity. Its parameter set consists of the scale parameter α and rate parameter λ. The scale parameter α specifies the value that the distribution asymptotically approaches, while the rate parameter λ influences the steepness of the slope (and thus popularity increase). The cumulative distribution function is expressed as follows:

$$D\left(t, \alpha, \lambda\right) = \alpha\left(1 - e^{-\lambda t}\right) \tag{3}$$

- **Gaussian**: This distribution represents an S-shaped pattern: a steep increase in popularity pre- and succeeded by a constant request rate. It is characterized by the mean μ and standard deviation σ. The cumulative distribution function is expressed as follows:

$$D\left(t, \mu, \sigma\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{t} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \tag{4}$$

Subsequently, every distribution $D\left(t\right)$ is fitted to $R_c\left(t\right)$ using an unconstrained non-linear optimization algorithm [31, 32, 33], which is capable of finding the parameter values that minimize the error between $D\left(t\right)$ and $R_c\left(t\right)$. Throughout the rest of this article the Levenberg-Marquardt algorithm [31] is used, as it has been shown to be faster and more robust than other existing approaches [34]. The optimization algorithm is used to find the optimal values $P_{D,c}^{\text{opt}}$ for the parameters P_D that allow $D\left(t\right)$ to best approximate $R_c\left(t\right)$ according to some metric. We call this metric the *fitting metric*. In this article, we use the mean squared error (MSE) as a fitting metric. It measures the average of the squares of the errors. Using this metric, the fitting algorithm minimizes the following function when finding the optimal parameter values $P_{D,c}^{\text{opt}}$:

$$f\left(P_D\right) = \frac{\sum_{t_2=0}^{t}\left(R_c\left(t_2\right) - D\left(t_2, P_D\right)\right)^2}{t} \tag{5}$$

Based on the optimal parameter values $P_{D,c}^{\text{opt}}$ of every distribution $D\left(t\right) \in \mathcal{D}$, a distribution $D_c^{\text{opt}}\left(t\right)$ needs to be selected for predicting the future request evolution of object c. This selection is made based on the quality of the fit to the historical trace. The metric used to assess this quality is called the *selection metric*. The MSE metric could for example be used. However, as our goal is to determine the theoretical performance gain that can be achieved, we define

the optimal (OPT) selection metric. It selects the distribution that results in the best absolute prediction within the prediction interval. This is obviously a theoretical metric as it uses information only available in the future. As such, it provides the theoretical upper performance limit of the presented prediction algorithm (i.e. it assumes the best candidate distribution is always chosen). It is calculated using the following formula:

$$\left| R_c\left(t_1\right) - R_c\left(t\right) - \left(D\left(t_1, P_{D,c}^{\text{opt}}\right) - D\left(t, P_{D,c}^{\text{opt}}\right)\right)\right| \tag{6}$$

Finally, the selected distribution $D_c^{\text{opt}}\left(t\right)$ and its optimal parameters $P_{D,c}^{\text{opt}}$ are used to predict the future. More specifically, the estimated value of $R_c\left(t_1\right)$ is defined as $D_c^{\text{opt}}\left(t_1, P_{D,c}^{\text{opt}}\right)$.

Note that time is a continuous variable. To reduce the total number of data points in the historical request pattern, we introduce the concept *time granularity*. The time granularity θ defines the interval of the sampled data points in the request pattern. Concretely, the historical request pattern $R_c\left(t\right)$ contains a sampled value for time instants $\{0, \theta, 2\theta, ..., t\}$. On one hand, reducing the granularity will allow the algorithm to make more fine-grained predictions. On the other hand, this will also increase its execution time. Throughout the rest of this article, a value of 1 hour is used for θ.

The algorithm is graphically illustrated by way of an example in Figure 1. It shows the request trace of an actual video in a deployed VoD system over the course of 448 hours. The prediction algorithm was applied to the first 400 hours of the trace (the known history) using the exponential and gaussian distributions. The parts of the curves before the vertical line represent the fits to known history, while the parts after the line represent the predictions. The actual number of requests that occur in the interval $[400, 448[$ is 15. The exponential distribution predicts 14.55 requests within that interval, which is very close to the real value. On the other hand, the gaussian distribution predicts only 1.25. This is also reflected in the figure, which show that the gaussian distribution poorly approximates the start and end of the request pattern. Note that when using the prediction algorithm in combination with a cache replacement strategy (cf. Section 3.2), the absolute prediction errors are not always a good indicator for the caching performance. Instead, the relative ordering of content items implied by the prediction algorithm's output will determine the eventual caching efficiency.

3.2. Predictive cache replacement strategy

The predicted future popularity of content can be used as input for a predictive cache replacement strategy. This section describes a cache replacement strategy that uses predicted popularity to make more efficient replacement decisions. In contrast to traditional strategies, it uses the expected future popularity of content as a measure instead of the known historical popularity.

The proposed strategy is called *Predictive Least Frequently Used* (P-LFU). It is a predictive version of the LFU caching strategy. LFU keeps track of the

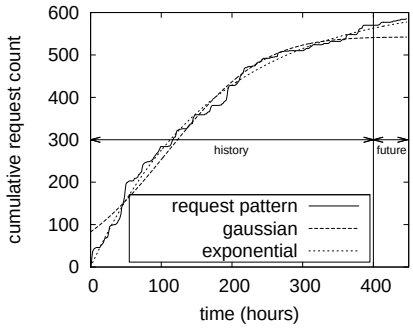


Figure 1: Optimal fit of the exponential and gaussian distributions to an example cumulative request pattern; the vertical line represents the current point in time t, on its left is the 400-hour known history, on its right the 48-hour predicted future

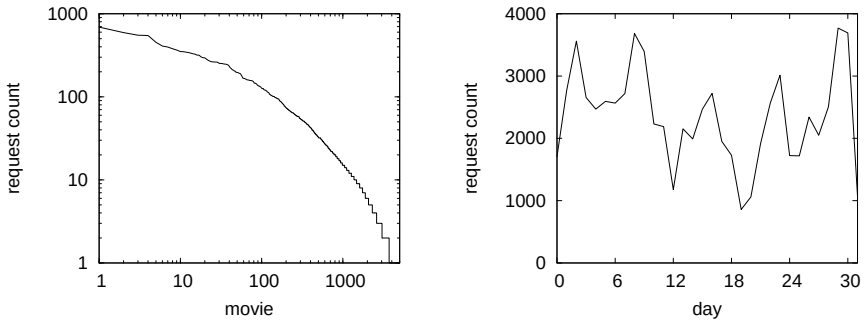
number of times that every object is requested. The objects that receive the highest number of requests within a specified time frame (i.e., the history window) are kept in cache [35]. In contrast, the P-LFU strategy uses the predicted number of requests for each object, instead of the known number of requests in the past. The objects with the most predicted number of requests in the prediction window W, are kept in cache. The prediction window is a configurable parameter. A larger value will allow the algorithm to take into account longer term popularity variations, but is also more prone to prediction errors. We introduce two variants of the strategy. The first assumes the future can be perfectly predicted. The other uses the prediction algorithm presented in Section 3.1. Throughout the rest of this article they are referred to as *Perfect Predictive Least Frequently Used* (PP-LFU) and *Optimal-Selection Predictive Least Frequently Used* (OP-LFU) respectively. Using the notations introduced in Section 3.1, we can define the number of estimated requests for object c at time t up to time $t+W$ as follows for PP-LFU:

$$r\left(t, W, c\right) = R_c\left(t+W\right) - R_c\left(t\right) \tag{7}$$

The PP-LFU strategy thus uses the actual request pattern R_c to achieve a perfect estimation. On the other hand, OP-LFU uses the optimal estimator D_c^{opt}:

$$r\left(t, W, c\right) = D_c^{\text{opt}}\left(t+W, P_{D,c}^{\text{opt}}\right) - D_c^{\text{opt}}\left(t, P_{D,c}^{\text{opt}}\right) \tag{8}$$

Whenever a request arrives for object c_i that is not currently cached, it replaces the cached object c_j if and only if $r(W, c_j) < r(W, c_i)$ and $\forall c \in \mathcal{C}$: $r(W, c_j) \leq r(W, c)$, with $\mathcal{C}$ the set of all cached objects. Or in other words, at every time t, the cache contains the subset of objects with the highest estimated request count within the interval $[t, t+W]$.



(a) popularity distribution (b) requests per day

Figure 2: A graphical representation of the Video on Demand dataset

4. Results & discussion

This section evaluates the presented popularity prediction algorithm (cf. Section 3.1) and predictive caching strategies (cf. Section 3.2). More specifically, the prediction algorithm is evaluated in terms of accuracy and execution time. Additionally, the effect of the prediction window parameter W on caching efficiency is studied in more detail. Finally, the predictive cache replacement strategies are compared to the traditional LFU caching strategy [35], as well as the theoretical optimal caching strategy MIN [36]. The MIN strategy replaces the object in the cache whose next request occurs furthest in the future. It has been proven to be optimal in terms of cache hit rate [37]. It thus gives a theoretical upper bound on performance. However, it has no practical use, as the time of the next request cannot be known in advance. Throughout this evaluation a history window of 12 hours is used for LFU. We have previously shown this to be a near optimal value for small cache sizes in combination with the dataset used in this article [30]. The *cache hit rate* is used as an evaluation metric. It is defined as the percentage of requests that can be served from a cache, as opposed to from the origin content server.

4.1. Evaluation scenario

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. Figure 2 graphically depicts the properties of the dataset. The popularity distribution over the movies is shown in Figure 2a. 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 2b 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, with increased activity on Friday, Saturday and Sunday. In addition to the weekly pattern, there is a daily pattern (not depicted in the figure). 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.

The evaluation scenario considers a single content server and proxy cache. The topology thus consists of a content server directly connected to the proxy cache. This cache is then directly connected to all end-users. The content server is assumed to have enough disk space to host all content. The available disk space of the proxy varies throughout the experiments and is expressed in terms of number of cached content items. The depicted cache hit rate results were measured in the intermediary proxy cache.

4.2. Prediction algorithm evaluation

This section evaluates the accuracy and performance of the prediction algorithm presented in Section 3.1. All results are depicted as a function of the history length. This is the length (i.e., number of data points) of the historical request trace used for the curve fitting step of the algorithm.

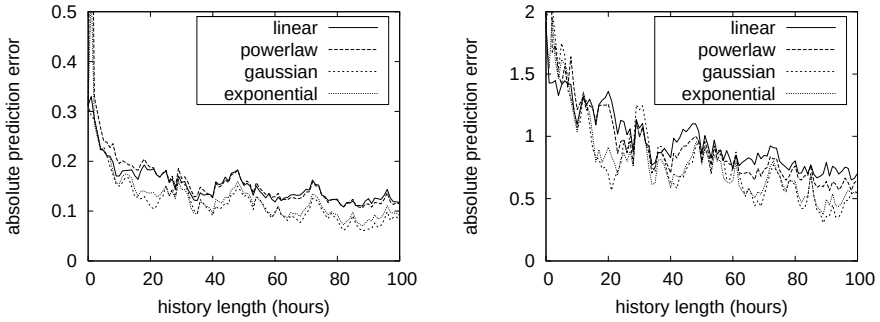
4.2.1. Prediction accuracy

The prediction accuracy represents the error of the predicted future request frequency compared to the actual future request frequency. As a metric for accuracy, the *absolute prediction error* is used. For a popularity distribution $D \in \mathcal{D}$, prediction window W and distribution parameters P, it is calculated as follows:

$$|R_c\left(t+W\right) - R_c\left(t\right) - \left(D\left(t+W, P\right) - D\left(t, P\right)\right)| \tag{9}$$

In other words, the absolute prediction error is defined as the difference between the actual number of requests and the predicted number of requests in the interval $[t, t+W]$. The goal of this section is to assess the effect of the history length and prediction window W parameters on the prediction error. Figure 3 depicts the absolute prediction error as a function of the history length for the four different popularity distributions. Figure 4 depicts the same for the exponential distribution only, but for multiple values of the prediction window W.

Figure 3a plots the prediction error as a function of the number of historical datapoints used in the curve fitting step of the prediction algorithm, averaged over all movies in the trace. On the other hand, Figure 3b depicts the prediction error, averaged over the 250 most popular movies only. As expected, the prediction error decreases significantly as the number of historical datapoints grows. For a small history size (i.e., a few hours), the prediction error is very large. However, it quickly converges to the optimum. More specifically, there is no significant difference between the prediction after 20 hours and after 100 hours. This is obviously expected to influence performance of a predictive cache replacement strategy, as its predictions will be less accurate for newly introduced



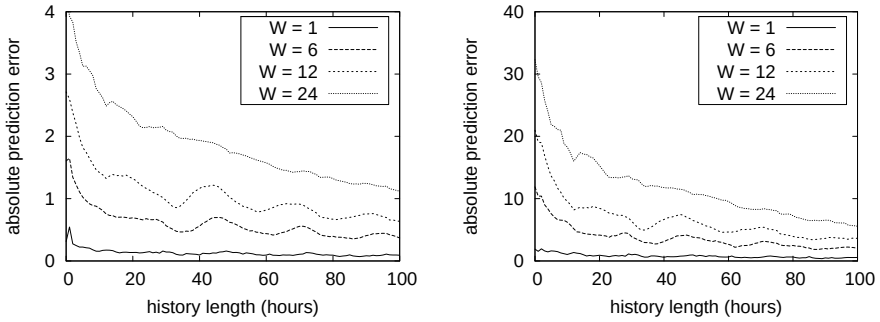
(a) Averaged over all objects (b) Averaged over the 250 most popular objects

Figure 3: The absolute prediction error averaged over all objects as a function of the history request pattern length for $W = 1$ hour

content. Additionally, comparing the two figures shows a significant different in prediction error averaged over all movies as compared to averaged over the most popular movies. This proves that it is more difficult to predict the future of popular content than that of unpopular content. Finally, the linear and power law distributions result in the worst overall predictions. Although the difference between the four distributions is insignificant when averaged over all movies, it is much clearer when looking at the 250 most popular movies only. In the latter case, the exponential and gaussian distributions clearly outperform the other two, resulting in much more accurate predictions (for a large enough history).

In addition to the history length, the prediction window is also expected to influence prediction accuracy. More specifically, a bigger prediction window is assumed to lead to larger errors, as it is easier to predict the nearby future. Figure 4a depicts the prediction error of the exponential fit as a function of the history length for different prediction window sizes W, averaged over all movies. Figure 4b depicts the same, but averaged over the 250 most popular movies only. The figures confirm our assumptions and clearly shows the direct linear connection between the prediction error and window W. However, for a request history of more than 100 datapoints, the prediction error averaged over all movies is less than 1 request even for a prediction window of 24 hours. For popular content the error increases and a prediction window of 24 hours results in an average error of less than 6 requests for a history window of 100 datapoints or more.

In summary, the above results lead to several pertinent conclusions. First, predicting the future is very difficult if the size of the known history is small. This makes popularity prediction of newly introduced content highly error prone. However, results showed that a short history trace (e.g., 20 hours) already reduces the error to a near optimal value. Additionally, there is a direct linear relationship between the prediction error and the prediction window W. Nev-



(a) Averaged over all objects (b) Averaged over the 250 most popular objects

Figure 4: The absolute prediction error of the exponential fit for different prediction windows (in hours) as a function of the history request pattern length

ertheless, even for a prediction window as large as 24 hours, the error can be reduced to less than 1 request on average as long as the historical trace is large enough.

4.2.2. Execution time

A deployed proxy cache usually operates in an online fashion. It decides whether to cache an object or not on-the-fly as requests arrive. The proposed predictive caching strategies need to execute the prediction algorithm once per time granularity interval θ for every content item that was requested during the interval. As such, it is important that the algorithm executes in a feasible time in order to support online popularity prediction. Figure 5 depicts the execution time of the curve fitting step of the prediction algorithm for the four employed distributions. As the curve fitting step of the algorithm is by far the most computationally intensive, its execution time is representative for the entire algorithm. The presented results were obtained using a test machine with a dual-core AMD Opteron$^{\text{TM}}$ 2212 processor and 4 GiB of memory.

The figure plots execution time as a function of the historical trace length. As expected, there is a linear correlation between execution time and history length. Additionally, the curve fitting algorithm's execution time is significantly different depending on the popularity distribution. The linear and power law distributions have a very low fitting time compared to the exponential and gaussian distributions. Concretely, for a trace of 400 datapoints, the fitting takes 16, 26, 188 and 1007 milliseconds for the linear, power law, exponential and gaussian distribution respectively. This is in line with the complexity of each distribution. The linear and power law distributions contain only two variables. The exponential and gaussian distributions have three variables each. Additionally, the gaussian distribution is more difficult to calculate as it contains an integral. Nevertheless, the combined execution time of all distributions for a long trace

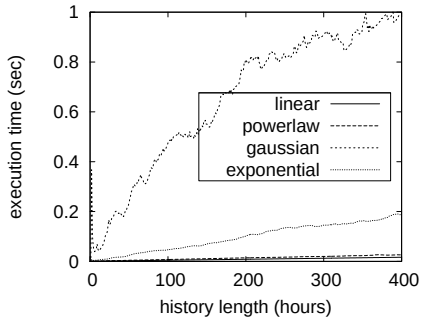


Figure 5: The processing time required to fit a single distribution to a request pattern as a function of the history request pattern length

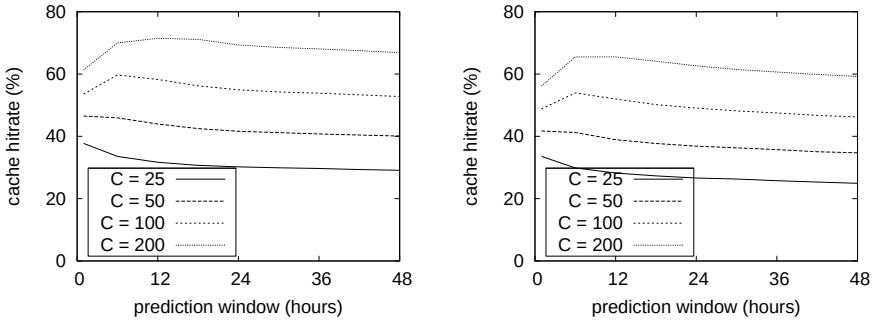
of up to 400 hours long, is only about 1.2 seconds. This allows up to 3000 popularity distribution updates in a one hour interval. Under more stringent time constraints, this time can be further reduced by either limiting the historical data that is taken into account, or limiting the maximum amount of iterations the fitting algorithm may perform. Additionally, the algorithm's execution time is significantly impacted by the number of variables in the popularity distributions. Limiting the amount of variables in the employed distributions would therefore greatly increase efficiency.

4.3. Predictive cache replacement strategy evaluation

This section evaluates the caching performance of the proposed predictive cache replacement strategies. The MIN, PP-LFU and OP-LFU cache replacement strategies all give an upper bound on performance. The MIN strategy achieves the optimal cache hit rate and represents the absolute upper bound on caching efficiency. The PP-LFU strategy provides the upper bound for frequency-based prediction strategies, as it assumes the future is perfectly predicted. Finally, OP-LFU uses a real prediction algorithm (and thus introduces prediction errors), but assumes the best popularity distribution is always chosen. It thus represents the upper bound on performance that can be achieved using the prediction algorithm presented in Section 3.1. This section consists of two parts. First, the optimal value of the prediction window parameter W is determined. Second, PP-LFU and OP-LFU are compared to to the traditional LFU and optimal MIN strategies.

4.3.1. Prediction window parameter

The prediction window parameter allows the predictive cache replacement strategy to adapt the future time frame that is taken into account. If this time frame is too short the strategy runs the risk of ignoring important future popularity fluctuations. However, if it becomes too long the prediction accuracy significantly decreases and the strategy might take into account expected popu-



(a) PP-LFU (b) OP-LFU

Figure 6: The cache hit rate as a function of the prediction window W for different cache sizes C

larity changes that are not yet relevant. The results in Figure 6 depict the cache hit rate as a function of the prediction window W for different cache sizes C.

The figure shows that there is indeed a peak in cache hitrate. Additionally, the location of this peak is directly proportional to the cache size. Comparing Figures 6a and 6b also proves that this is the case for both PP-LFU and OP-LFU. More specifically, for a very small cache of 25 objects, the optimal prediction window is 1 hour. As the cache size increases to 100 objects, the optimum increases to 6 hours. Finally, for a larger cache of 200 objects, the optimal prediction window is 12 hours. The results additionally show that estimating the prediction window parameter value both too large or too small results in decreased performance. In the depicted results, the optimal prediction window value performs up to 33% better than the worst. As such, a practical implementation should intelligently adapt its prediction window parameter to the size of the cache, in order to prevent significant drops in performance. Throughout the remainder of this section, a prediction window of 12 hours is chosen, as it achieves good performance for caches of 50 or more objects.

4.3.2. Comparison with traditional strategies

This section compares the cache hit rate of the PP-LFU and OP-LFU with that of LFU and MIN. This provides us with insights of how well predictive caching can perform compared to a good traditional strategy (i.e., LFU) and the theoretical optimum (i.e., MIN). Figure 7 depicts these results. As expected, MIN performs best, closely followed by PP-LFU, OP-LFU and finally LFU.

The results for PP-LFU represent the theoretical upper bound that can be achieved using a frequency-based predictive cache replacement strategy. As the cache size increases, its results approach the optimum more closely. For a cache size of 50 objects, PP-LFU performs 17% worse (i.e., a cache hit rate difference of 8.5%) than MIN, while for a cache size of 200 objects it only performs 3% worse (i.e., a cache hit rate difference of 2.5%). Additionally, PP-

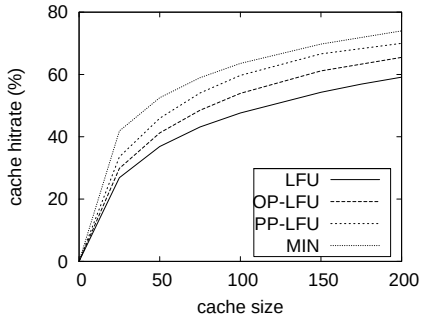


Figure 7: Comparison of the different cache replacement strategies in terms of cache hit rate, as a function of cache size for $W = 12$ hours

LFU's caching efficiency is considerably better than that of the traditional LFU strategy, performing around 20% better for all depicted cache sizes.

The OP-LFU strategy gives a theoretical upper bound on cache efficiency when using the prediction algorithm presented in Section 3.1. In contrast to PP-LFU, it is thus subject to prediction errors, which explain the reduced performance of OP-LFU. For all depicted cache sizes, its caching efficiency is about 10% worse than that of PP-LFU. Compared to MIN, its efficiency increases as the cache size grows. For a small cache of 50 objects, it is up to 25% worse, while for a larger cache of 200 objects it is only 11% worse. Additionally, OP-LFU performs considerably better than LFU. Its gain in efficiency even increases as the cache size grows, with a 5% improvement for small caches up to 50 objects and over 10% for a larger cache of 200 objects.

In summary, the presented results show that applying popularity prediction to cache replacement has the potential of significantly improving the cache hit ratio compared to traditional strategies such as LFU. Theoretically, employing the perfect prediction allowed the algorithm to improve up to 20% compared to LFU. The prediction errors introduced by an actual prediction algorithm based on curve fitting reduced the maximum performance gain to 10%.

5. Conclusion

The goal of this article was to investigate the merits of using popularity prediction techniques in cache replacement strategies. To this end, we proposed a novel generic popularity prediction algorithm. It estimates the future popularity of multimedia content by fitting a set of popularity distributions to the known request history. The set of popularity distributions can be adapted to fit the characteristics of the multimedia service to which it is applied. Additionally, a predictive variant of the Least Frequently Used cache replacement strategy, called P-LFU, is proposed. It uses the predicted future request frequency to determine what content to cache. Two theoretical versions of P-LFU are considered. PP-LFU assumes the future can be perfectly predicted. It thus gives

an upper bound on performance that can be achieved using P-LFU. OP-LFU instead employs the presented popularity prediction algorithm, but assumes the popularity distribution that best predicts the future is selected. It thus gives an upper bound on performance that can be achieved using P-LFU, when used in combination with the presented prediction algorithm.

A detailed simulation study, using an actual Video on Demand trace file, was performed in order to evaluate the merits of prediction-based cache replacement. The evaluation consists of two main components. First, the popularity prediction algorithm was validated. Second, PP-LFU and OP-LFU were compared, in terms to cache hit rate, to LFU and the theoretical optimum. This evaluation lead to several pertinent conclusions. First, the prediction accuracy is severely impacted by the number of available historical datapoints. This is especially true for very short history lengths of less than 10 datapoints. This makes predicting the popularity of newly introduced content highly error prone. Additionally, there is a direct linear relationship between the prediction window parameter and the prediction error. Nevertheless, even for a prediction window as large as 1 day, the error can be reduced to on average less than 1 request as long as the known history is large enough. For the predictive cache replacement strategies it was shown that the optimal value of the prediction window parameter is directly proportional to the cache size. Choosing a suboptimal prediction window was shown to lead to performance drops of up to 33%. A practical predictive cache replacement strategy could thus automatically adapt its prediction window according to changes in cache size in order to further optimize performance. Moreover, applying popularity prediction to cache replacement has the potential of significantly improving the cache hit rate compared to traditional strategies, such as LFU. Under the assumption that the future can be perfectly predicted (i.e., PP-LFU) an improvement of up to 20% can be achieved. However, when instead using the actual prediction algorithm, this improvement is reduced to at most 10%.

The presented OP-LFU strategy uses information about the future popularity of content to select the best popularity distribution for prediction. As this approach is infeasible in a practical deployment, our future work consists of finding a good estimator for the selection of the optimal popularity distribution based on historical information.

Acknowledgement

Jeroen Famaey is partly 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.

References

[1] K. J. Ma, R. Bartos, S. Bhatia, A survey of schemes for internet-based video delivery, Journal of Network and Computer Applications 34 (5) (2011) 1572–1586. doi:10.1016/j.jnca.2011.02.00.

[2] D. De Vleeschauwer, K. Laevens, Performance of caching algorithms for IPTV on-demand services, IEEE Transactions on Broadcasting 55 (2) (2009) 491–501. doi:10.1109/TBC.2009.2015983.

[3] S. Makridakis, Time series prediction: Forecasting the future and understanding the past, International Journal of Forecasting 10 (3) (1994) 463–466.

[4] Z. Avramova, S. Wittevrongel, H. Bruneel, 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 (INTERNET), 2009, pp. 95–100. doi:10.1109/INTERNET.2009.22.

[5] G. Szabo, B. A. Huberman, Predicting the popularity of online content, Communications of the ACM 53 (8) (2010) 80–88. doi:10.1145/1787234.1787254.

[6] K.-L. Wu, P. Yu, J. Wolf, Segmentation of multimedia streams for proxy caching, IEEE Transactions on Multimedia 6 (5) (2004) 770–780. doi:10.1109/TMM.2004.834870.

[7] S. Chen, H. Wang, X. Zhang, B. Shen, S. Wee, Segment-based proxy caching for internet streaming media delivery, IEEE Multimedia 12 (3) (2005) 59–67. doi:10.1109/MMUL.2005.56.

[8] J. Yu, C. Chou, Z. Yang, X. Du, T. Wang, A dynamic caching algorithm based on internal popularity distribution of streaming media, Multimedia Systems 12 (2) (2006) 135–149. doi:10.1007/s00530-006-0045-x.

[9] S. Kim, C. R. Das, An analytical model for interval caching in interactive video servers, Journal of Network and Computer Applications 30 (1) (2007) 384–413. doi:10.1016/j.jnca.2005.06.005.

[10] H. Guo, G. Shen, Z. Wang, S. Li, Optimized streaming media proxy and its applications, Journal of Network and Computer Applications 30 (1) (2007) 265–281. doi:10.1016/j.jnca.2005.08.008.

[11] T. Wauters, W. Van de Meerssche, P. Backx, F. De Turck, B. Dhoedt, P. Demeester, T. Van Caenegem, E. Six, Proxy caching algorithms and implementation for time-shifted TV services, European Transactions on Telecommunications 19 (2) (2008) 111–122. doi:10.1002/ett.1181.

[12] D. Verstraeten, B. Schrauwen, M. D'Haene, D. Stroobandt, An experimental unification of reservoir computing methods, Neural Networks 20 (3) (2007) 391–403. doi:10.1016/j.neunet.2007.04.003.

[13] R. Samsundin, A. Shabri, P. Saad, A comparison of time series forecasting using support vector machine and artificial neural network model, Journal of Applied Sciences 10 (11) (2010) 950–958. doi:10.3923/jas.2010.950.958.

[14] F. wyffels, B. Schrauwen, A comparative study of reservoir computing strategies for monthly time series prediction, Neurocomputing 73 (2010) 1958–1964. doi:10.1016/j.neucom.2010.01.016.

[15] S. Soltani, On the use of the wavelet decomposition for time series prediction, Neurocomputing 48 (2002) 267–277. doi:10.1016/S0925-2312(01)00648-8.

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

[17] M. Breslau, L. Baum, G. Molter, S. Rothkugel, S. P., The implications of Zipf's law for web caching, in: Third International Conference on Web Caching, 1998.

[18] W. Tang, Y. Fu, L. Cherkasova, A. Vahdat, Modeling and generating realistic streaming media server workloads, Computer Networks 51 (1) (2007) 336–356. doi:10.1016/j.comnet.2006.05.003.

[19] L. Guo, E. Tan, S. Chen, Z. Xiao, X. Zhang, The stretched exponential distribution of Internet media access patterns, in: Twenty-Seventh ACM Symposium on Principles of Distributed Computing, 2008, pp. 283–294. doi:10.1145/1400751.1400789.

[20] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, S. Moon, I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system, in: 7th ACM SIGCOMM conference on Internet measurement (IMC), ACM, 2007. doi:10.1145/1298306.1298309.

[21] X. Cheng, C. Dale, L. J., Understanding the characteristics of Internet short video sharing: YouTube as a case study, in: 7th ACM SIGCOMM Conference on Internet Measurements, 2007, pp. 28–36.

[22] A. Abhari, M. Soraya, Workload generation for youtube, Multimedia Tools Applications 46 (1) (2010) 91–118. doi:10.1007/s11042-009-0309-5.

[23] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, S. Moon, Analyzing the video popularity characteristics of large-scale user generated content systems, IEEE/ACM Transactions on Networking 17 (5) (2009) 1357–1370. doi:10.1109/TNET.2008.2011358.

[24] G. Chatzopoulou, C. Sheng, M. Faloutsos, A first step towards understanding popularity in YouTube, in: IEEE Conference on Computer Communications Workshops (INFOCOM), 2010. doi:10.1109/INFCOMW.2010.5466701.

[25] F. Figueiredo, F. Benevenuto, J. M. Almeida, The tube over time: characterizing popularity growth of YouTube videos, in: Fourth ACM International Conference on Web Search and Data Mining, 2011, pp. 745–754. doi:10.1145/1935826.1935925.

[26] S. Jamali, H. Rangwala, Digging digg: Comment mining, popularity prediction, and social network analysis, in: International Conference on Web Information Systems and Mining, 2009, pp. 32–38. doi:10.1109/WISM.2009.15.

[27] T. Wu, M. Timmers, D. De Vleeschauwer, W. Van Leekwijck, On the use of reservoir computing in popularity prediction, in: The Second International Conference on Evolving Internet (INTERNET), 2010, pp. 19–24. doi:10.1109/INTERNET.2010.13.

[28] D. Niu, Z. Liu, B. Li, S. Zhao, Demand forecast and performance prediction in peer-assisted on-demand streaming systems, in: IEEE Conference on Computer Communications (INFOCOM), 2011, pp. 421–425. doi:10.1109/INFCOM.2011.5935196.

[29] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, Time Series Analysis: Forecasting and Control, Wiley, 2008.

[30] J. Famaey, T. Wauters, F. De Turck, On the merits of popularity prediction in multimedia content caching, in: IFIP/IEEE International Symposium on Integrated Network Management (IM), 2011, pp. 17–24. doi:10.1109/INM.2011.5990669.

[31] K. Levenberg, A method for the solution of certain non-linear problems in least squares, Quarterly Journal of Applied Mathmatics 2 (2) (1944) 164–168.

[32] J. A. Nelder, R. Mead, A simplex method for function minimization, The Computer Journal 7 (4) (1965) 308–313. doi:10.1093/comjnl/7.4.308.

[33] M. J. D. Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives, The Computer Journal 7 (2) (1964) 155–162. doi:10.1093/comjnl/7.2.155.

[34] S. Katare, A. Bhan, J. M. Caruthers, W. N. Delgass, V. Venkatasubramanian, A hybrid genetic algorithm for efficient parameter estimation of large kinetic models, Computers & Chemical Engineering 28 (12) (2004) 2569–2581. doi:10.1016/j.compchemeng.2004.07.002.

[35] N. Megiddo, D. S. Modha, Outperforming LRU with an adaptive replacement cache algorithm, Computer 37 (4) (2004) 58–65. doi:10.1109/MC.2004.1297303.

[36] J. Gecsei, D. R. Slutz, I. L. Traiger, Evaluation techniques for storage hierarchies, IBM Systems Journal 9 (2) (1970) 78–117. doi:10.1147/sj.92.0078.

[37] B. Van Roy, A short proof of optimality for the MIN cache replacement algorithm, Information Processing Letters 102 (2-3) (2007) 72–73. doi:10.1016/j.ipl.2006.11.009.