Replacement Strategies for Quality Based Video Caching

Stefan Podlipnig and Lászlo Böszörményi
Abstract

Due to the future dominance of video data, video caching will be an important performance factor in future networked multimedia systems. A major component of video caches is the replacement strategy. This paper presents replacement strategies for video caches that incorporate quality reduction and the use of metadata given by the content provider. The strategies are evaluated by simulation.
1 Introduction

Proxy Caching is an important factor in providing fast access to web objects over the Internet. Due to the locality of reference in web request streams, popular web objects and web servers form hot spots in the network. To overcome such hot spots proxy caches have been introduced to cache popular objects near to clients. During recent years the amount of multimedia content including images and continuous media data has increased. Especially continuous media data is growing dramatically. Although currently audio data dominates, video data will be the dominating factor in the future. Because videos are several order of magnitude larger than normal web objects they introduce a high load. Therefore it is an important question how web videos can be cached near to clients.

The growing number of web videos has influenced the development of different techniques for video delivery. We distinguish two techniques for sending stored video data from a server to a client, namely download-and-play and streaming. With download-and-play video data is transferred completely to the client site before display. This results in large space usage and long start-up delay. With streaming the client software plays the video as soon as a certain amount of data has been received. It uses a small buffer for the incoming data and does not have to store the whole video. Generally, downloading the whole file is accomplished over reliable transport mechanisms. For streaming applications a number of streaming protocols and systems have been introduced. Although there exist standard protocols for streaming (Real Time Transport Protocol, Real Time Streaming Protocol) the current environment of internet streaming is influenced by a number of commercial companies which use proprietary implementations. We do not want to concentrate on one special technique. Our main aim is to present concepts that are applicable to different scenarios.

2 Related Work

Due to the growing interest in web videos a growing number of video caching strategies has been proposed recently. These strategies can be divided into two categories:

- Full video caching: The whole video is cached at the proxy.
- Partial video caching: A certain part of the video is cached at the proxy.

Full video caching can be found in many available caching products, where videos are handled like typical web objects. Special commercial solutions for caching streaming videos also cache whole videos\(^1\).

Because full video caching can be very resource consuming, research has focused on the development of new caching schemes. There exist a lot of proposals for caching only specific parts (prefix [14], prefix and selected frames [9], bursty parts of a video [17], hotspot segments [6], popularity-based prefix [11], segment-based prefix caching [16], distributed architectures for partial caching [1, 2]) of a video. Another approach to partial caching is quality based video caching. Although the whole video is stored at the proxy the quality can be changed. \(^7\)

\(^1\)Although there is no special technical information about these products, they seem to apply this kind of caching
discusses theoretical modeling and simple heuristics for periodic caching of layered encoded videos. \[10\] discusses replacement strategies for layered encoded videos where each layer of a video is treated as a separate object. A comprehensive approach to quality adaptive video caching is presented in \[12, 13\]. The quality of cached streams is adjusted according to their popularity and the available bandwidth between proxy and clients. Fine-grained replacement and prefetching of segments (parts of a layer) is proposed.

Coupling quality reduction and replacement was also proposed for images in \[8\]. In this so called Soft Caching approach unpopular images are not removed but recoded to a lower resolution. The idea is to provide a client with a lower image resolution until she asks explicitly for the original version.

3 Quality Based Caching

Quality based caching is some sort of partial caching. Although initially the whole video is stored at the proxy, the quality can be changed according to some criteria. We distinguish two forms of quality based caching:

- Quality reduction: The proxy reduces the quality of videos according to some criteria.
- Quality adaptation: The proxy reduces and enhances the quality of videos according to some criteria.

We will focus on quality reduction because it allows simple replacement strategies. Although quality adaptation is seen as the more flexible approach it introduces additional complexity. To enhance the quality of a reduced video a cache has to reload specific parts of the video. Furthermore the cache has to implement intelligent adaptive behaviour. Short fluctuations in the popularity have to be filtered out to avoid oscillation of storage allocation.

We argue that quality reduction can be an effective alternative to complex adaptive behaviour. On the one hand most of the videos will have a short period of high popularity followed by decreasing popularity. Quality reduction supports this process. On the other hand quality reduction can be coupled with explicit reloading of videos, i.e. a user can trigger a reload if she is not satisfied.

3.1 Quality Reduction

Quality based caching is based on some assumptions. At first videos should allow some sort of quality adaptation. A video should have a number of quality steps that can be obtained through operations on that video. Such quality steps can be realized through layers (base layer, enhancement layers as defined e.g. in MPEG-2 or H.263) or through object based coding (as defined e.g. in MPEG-4). In the following we do not assume a special coding technique but rely on the assumption that a video has a certain number of quality steps. Furthermore we assume that these quality steps are ordered according to some importance criteria (e.g. base layer is more important than enhancement layer). This importance criteria determines the adaptation order.

The second assumption is that there exist metadata describing the possible quality steps (as defined e.g. in MPEG-7). For each quality step these metadata describe the corresponding
operation, the resulting size and the resulting quality. The size $s_i$ and quality factor $q_i$ of a video $i$ are in the range $0 < s_i, q_i \leq 1$. For example $s_i = 0.5$ and $q_i = 0.8$ mean that the corresponding operation produces a video that has 50% of the original size and 80% of the original quality. Whereas the size reduction can be calculated exactly, the calculation of the resulting quality is not straight-forward. We assume that the content resource provides this information. Quality reduction can be implemented with different step sizes. Although it is an interesting research question we used in our simulation the same number of steps (4) to all videos.

3.2 Replacement

The replacement strategy is a major performance factor of proxy caches. A lot of proposals exist for replacement strategies ([4, 15]). These strategies are suitable for normal web caches. They do not introduce quality awareness in the replacement process. In the following we will concentrate on enhancements of two replacement strategies, namely Least Recently Used (LRU) and a simpler version of Greedy Dual Size Frequency (GDSF) [5]. We assume that the videos (data about the videos) are stored in a list. For both algorithms the list is sorted. The most valuable video is at the beginning, the least valuable at the end. LRU ensures this ordering when the requested video is inserted at the beginning. For GDSF a requested video is inserted according to its calculated characteristic value. The replacement algorithm starts at the end of a list when an insertion has caused an overflow (e.g. the size of the cached videos is above the cache size) and tries to remove videos until the size of the cached videos is below the size of the cache. Normally web caches use two marks $H$ and $L$ with $H > L$ for the replacement process. If the size of the cached videos is above $H$ the replacement algorithm deletes objects until the size of the remaining objects is below $L$. This approach has the advantage that the replacement is not triggered very often (depends on the workload). In our case replacement takes place very often. On the other hand our approach has two advantages that are very useful for video caches:

- It reduces the load in one replacement run, because we do not have to replace so many videos. The load is spread over more replacement runs and therefore we reduce replacement bursts.
- The average number of objects in the cache is higher. This is very useful for a small cache.

In the following we describe our proposals for replacement algorithms. These proposals are based on work in [10], [12] and [8].

3.2.1 Replacement with repositioning

A quality based replacement strategy chooses the last video and reduces its quality by deleting one quality step. If the video has only one quality step left, this causes the deletion of the video. Otherwise the video stays in the cache and is repositioned in the cache list. Then the video at the end of the list is chosen and the procedure is repeated. This idea of a weighted access was first proposed in [8] for LRU. We modified the proposed calculation to incorporate resulting quality and position numbers rather than time. We call this algorithm LRU-R. The
first modification assures a better link to the visual quality, the second a faster calculation. Let $L_s$ be the size of the LRU list. Let $q_r$ be the resulting quality of the chosen video $r$. After the quality reduction the new position $LRU_r$ of $r$ is calculated by $LRU_r = [(1 - q_r) \times L_s]$. If $q_r$ is near 1 (first reductions) the video is inserted near the beginning of the list. After a number of reductions the video will be inserted near the end.

For GDSF the repositioning can be done by recalculating the value of the video and inserting the video correspondingly. The value $H_i$ of video $i$ is given by $H_i = \frac{f_i}{s_i} + l$, where $f_i$ is the number of references to $i$, $s_i$ is the actual size of $i$ and $l$ is an inflation value. At the beginning $l$ is set to 0, at each deletion $l$ is set to the value of the replaced video. With quality based caching a recalculation of $H_i$ is necessary because the size of a video changes. We call this algorithm GDSF-R.

### 3.2.2 Replacement without repositioning

Without a weighted access the last video in the list is chosen for quality reduction successively until it is deleted or the replacement stops. If the last video is not requested immediately it will be deleted in the following replacement round. This behaviour will be similar to the original underlying replacement strategy (depending on the request sequence). We call this behaviour vertical replacement. The pattern of replacement is illustrated in figure 1. The quality steps of one video are ordered from the top to the bottom.

![Figure 1: Replacement patterns](image)

To overcome the strong similarity to the underlying strategy we propose horizontal replacement. With horizontal replacement the highest layer of all videos is removed first, then the next layer and so on. Like vertical replacement this is an extreme case of adaptation. Therefore we further propose a combination of these strategies. In this combined replacement the upper layers are removed with a horizontal pattern the lower layers are removed with a vertical pattern.

A given pattern is used in each replacement run. It is possible that different videos have different numbers of quality steps. The replacement algorithm tries to follow the given pattern. It is like a matrix traversal where the dimensions are given by the number of videos and the maximum number of quality steps. In each step the replacement algorithm removes a quality step. If it is not available, the replacement process continues with the next given quality step.

Note that these patterns can be combined with any original replacement algorithm. The only condition is that the videos are sorted according to their ”popularity”. The popularity
can be determined by different video characteristics (for example request recency, request frequency, bitrate, size).

These patterns are similar to patterns proposed in [12]. There, such patterns are used for the segments of a layered coded video. Therefore, they are used for a fine grained-replacement. To be faster we use a coarse-grained replacement and use such patterns for the quality steps of all cached videos. Normally a quality step is much bigger than a single segment of a video layer. Nevertheless it is possible to model a segment as a quality step. In the following we will reference the vertical versions of LRU and GDSF as LRU-V and GDSF-V, the horizontal versions as LRU-H and GDSF-H and the combined versions as LRU-C and GDSF-C.

4 Evaluation

The replacement strategies were evaluated by simulation. The performance or web cache replacement strategies is often described through hit-rate or byte-hit-rate. With quality based video caching these metrics do not reflect the real performance. Therefore, we consider an additional metric, namely quality-weighted-hit-rate ($QwHIT$). For this we calculate the hit-rate and the so called quality-hit-rate. The quality-hit-rate gives the average quality per hit. Let $n_i$ be the number of hits for video $i$ and $r_i$ the number of requests to video $i$. The hit-rate of a cache for a request sequence with $n$ different videos, $HIT$, is given by $\sum_{i=1}^{n} \frac{n_i}{r_i}$. Let $q_i$ be the quality of video $i$ at hit $h$. Then the quality-hit-rate $QHIT$ is given by $\sum_{i=1}^{n} \sum_{h=1}^{n_i} \frac{q_i}{h}$. Without quality reduction $QHIT$ is 1, with quality reduction < 1. Note that this definition is similar to the definition of quality-hit-rate given in [10]. But whereas [10] uses the fraction of cached layers in the numerator, we propose to use the given quality. Building on these two measures we define $QwHIT = HIT * QHIT$. Therefore, $QwHit$ gives a hit-rate that is weighted by the average quality per hit. With this definition we balance between higher hit-rate and lower quality.

Replacement strategies are often evaluated by request sequences from log files of operating proxies. Such log files include requests to different web objects (html, images, etc.). Although it is possible to filter out video requests and produce special traces (only video requests) the value of such traces is questionable as they represent specific workloads. Therefore, we decided to generate artificial request sequences. This approach is more flexible because we can vary different parameters to simulate different workload situations. For the generation we used a synthetic web proxy workload generator (ProWGen) described in [3]. We generated 2 request sequences with 100,000 requests to 10,000 different videos. 10% of the videos were one-timers (only one request). Popularity was modeled by Zipf distribution with parameters $\alpha = 1.0$ (strongly skewed popularity) and $\alpha = 0.3$. For the size of the videos the generator uses two distributions. Lognormal for the body of the request size distribution and Pareto distribution for the tail. Our parameters were $\mu = 7000$ KB, $\sigma = 11000$ (lognormal) and 1.2 (Pareto). We assumed no correlation between size and popularity. The whole size of all videos was 155 GByte. We assumed 4 quality steps for each video. The size-quality pairs were (1,1), (0.7,0.96), (0.4,0.82) and (0.1,0.2). Such concave quality curves are a typical assumption in video quality modeling. For combined replacement we assumed 2 horizontal and 2 vertical steps.

The results of the simulation are given in figure 2 and figure 3. For the hit-rate, horizontal
Figure 2: Results for LRU
Figure 3: Results for GDSF
replacement gives the best results for LRU and GDSF. Repositioning is not so effective for LRU but gives good performance for GDSF. The performance of vertical replacement is similar to the original algorithm (same hit-rate for GDSF, slightly smaller hit-rate for LRU). Combined replacement is slightly better than repositioning for LRU. For GDSF combined replacement is between the original algorithm and repositioning.

From the plots for the quality-hit-rate we see that vertical replacement introduces nearly no loss. Repositioning and especially horizontal replacement cause more quality loss. Indeed for horizontal replacement the average quality is near 0.2. Therefore most of the videos are stored with only one quality step. Combined replacement introduces a constant moderate quality loss.

The quality-weighted-hit-rate shows very different behaviour. Horizontal replacement is bad for $\alpha = 1.0$ and does not give an improvement with $\alpha = 0.3$. Vertical replacement and repositioning are below LRU or similar to LRU. Combined replacement is better with $\alpha = 0.3$ and with $\alpha = 1.0$ for small cache sizes. For GDSF horizontal replacement performs very bad over a broad range of the workload. Repositioning performs good for all cache sizes and both workloads. Combined replacement performs very good with $\alpha = 0.3$ and with $\alpha = 1.0$ at very small cache sizes. Vertical replacement is similar to GDSF (overlapping curves).

Horizontal replacement can be effective for not too skewed popularity distributions. Furthermore the improvement through quality-awareness is higher for LRU than for GDSF. Vertical replacement performs similar to the original algorithm when it is used for LRU. For GDSF repositioning seems to be a valuable alternative for not so skewed popularity distributions. Combined replacement will give a moderate increase in hit-rate and outperforms most algorithms in terms of quality-weighted-hit-rate when the popularity is not too skewed or the cache sizes are very small. Therefore, combined replacement and repositioning are good candidates for quality based replacement. Both algorithms are more robust to popularity fluctuations and do not exhibit the extrem behaviour of horizontal replacement. Horizontal replacement is a valid choice for users who prefer a high hit-rate and do not bother lower quality.

5 Conclusions and Future work

We presented replacement algorithms for quality based video caching. Our proposals build on the two assumptions that a video consists of different quality steps and that there exist some sort of metadata that describes this possible quality steps. The proposed strategies were evaluated by simulations. In future work we plan to evaluate further replacement strategies (based on randomization) and more differentiated workloads (e.g. different number of quality steps). Furthermore we introduce reloading in our simulation and implement the strategies in real environments.

References


Institute of Information Technology
University Klagenfurt
Universitaetsstr. 65-67
A-9020 Klagenfurt
Austria

http://www.tec.uni-klu.ac.at