A FLEXIBLE STARTING POINT BASED PARTIAL CACHING ALGORITHM FOR VIDEO ON DEMAND

Lian Shen, Wei Tu, and Eckehard Steinbach

Institute of Communication Networks, Media Technology Group,
Technische Universität München, Munich, Germany
shenlian@mytum.de, {wei.tu, eckehard.steinbach}@tum.de

ABSTRACT

In this paper, we propose a novel proxy caching scheme for Video on Demand (VoD) services. Our approach is based on an observation we have made during subjective VoD performance evaluation tests. We have found that when users are seeking for some specific content, they pay most attention to the initial delay, while a small shift of the starting point is acceptable. Based on this observation as well as the dynamic popularity of video segments, we propose an efficient segment based caching algorithm. The approach is applied on a proxy and minimizes the average initial playout delay at the clients. Our experimental results show a significant reduction of the average initial waiting time compared to conventional caching approaches.

1. INTRODUCTION

Proxies are widely used in today's web browsing and video streaming services to decrease the data traffic from the remote server and to minimize the client waiting time. In our work, we consider the scenario shown in Fig. 1. The proxy is close to the clients and has a high speed and robust connection to all the clients. If the requested video content has already been accessed by one user in the local network and is cached on the proxy, the initial delay for the second and later users is significantly decreased compared to the case where content has to be requested from the remote server. Since the proxy has only finite storage capacity, a dynamic cache management scheme has to be used that decides which videos or which parts of a video to cache.

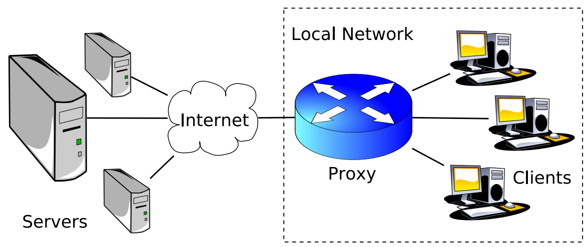


Fig. 1. Server-Proxy-Client network structure

A number of studies have been performed to explore the benefits of a proxy for video streaming applications. In [1]–[3], partial caching algorithms are proposed to minimize the data traffic between the server and clients. Miao and Ortega consider in their work [4] also the efficient control and usage of the client buffer. As the data volume of the video content is large, it is impractical to store all the videos on the proxy. Proxy cache update algorithms [5] [6] have been proposed to dynamically manage the proxy's cache. The lifetime of video is considered in [7]. [8] and [9] propose cache update algorithms based on the popularity of entire videos. [10] considers a popularity based caching algorithm for layered video where the popularity is managed for individual layers.

Most proxy caching algorithms proposed in the literature assume that video files are always played from the beginning and continuously towards the end. Generally, this assumption does not hold in practice. The log file of a real video system reported in [11] shows that for Video on Demand (VoD) services most of the video content is randomly accessed instead of being played sequentially. Hence, proxy caching and dynamic cache management algorithms should take random access into account.

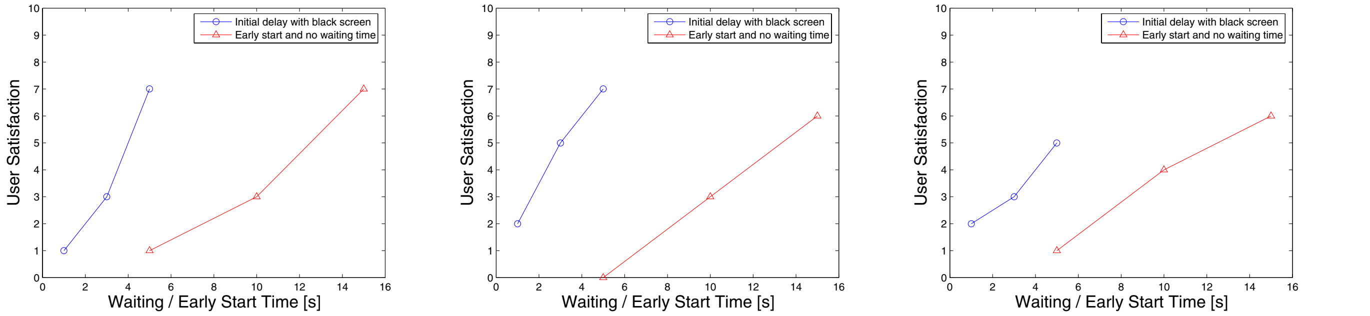
In VoD services, the initial delay is one of the most important aspects for service quality. If the users have to wait a long time before they can start playing the video content, they will lose patience. We have set up an online subjective test to examine the impact of initial delay on user satisfaction. In this test, three content types are used, namely sport highlights, news reports and action movies. Each clip is played using the following two modes:

1. The playout starts after an initial delay of 1, 3 or 5 seconds. A black screen with a red indication " Buffering..." in the middle of the screen is played during the initial waiting time.
2. The playout starts immediately without any initial delay. However, the playout starts 5, 10 or 15 seconds earlier than the starting point selected by the user.

Fig. 2(a) shows the result for the sport clip. The Y-axis shows the user satisfaction on a scale from 0 to 9, where 0 indicates very comfortable and 9 indicates intolerable. The average scores of the evaluation for 15 test persons are shown in the figure. We can see from Fig. 2(a) that the test persons feel more comfortable if the playout starts without delay. This means the users are more satisfied if the video starts with a very short initial delay, despite the earlier starting point. Similar results can be observed for the news and movie clips in Fig. 2(b) and Fig. 2(c). We draw the following conclusion from the test: *VoD users prefer immediate feedback from the system even if the played video does not start exactly at the desired starting point.* This conclusion can be even enforced by the fact that starting point selection is typically performed using some kind of slide bar which does not allow for very accurate starting point selection anyway.

Based on the above observation, we propose a Popularity-Aware Partial cAching (PAPA) algorithm that minimizes the average initial delay of the system, while allowing a small deviation from the desired starting point. We assume that the buffer at the client is very small and can not prefetch frames from the proxy. The video file

This work was supported by Deutsche Forschungsgemeinschaft (DFG) under grant STE 1093/3-1



(a) Sport (b) News (c) Movie

Fig. 2. Results of the subjective VoD performance evaluation test

is divided into segments of several seconds and we enforce that the playout always starts from the beginning of a segment. This means that the proxy always sends the user the first frame of the segment the starting point falls in and keeps on sending the later frames until a new access request arrives from the user. When the proxy cache is newly initialized and has enough free space, all video frames are cached on the proxy for future access by other users. When the proxy cache is full, the proposed cache update algorithm will be called. Instead of evaluating the popularity of the whole video file as presented in [9], we evaluate the popularity of every segment in our work, which gives more accurate popularity analysis and more flexible frame update. The decision which frames to cache and which to replace is then determined based on the Weighted Waiting Time (WWT), which is calculated from the popularity of the segment, the size of the video frame and the link connection rate between the corresponding server and the proxy.

In the next section, we present our proposed video segmentation structure and the popularity-aware partial caching algorithm in detail. Section 3 presents the simulation results which show the improvements achieved by our proposal compared to a GOP based frame selection approach. Our conclusions are given in Section 4.

2. POPULARITY-AWARE PROXY CACHE UPDATE

2.1. Segment-Prefix Structure

We split a video first into segments with a length of M Group of Pictures (GOPs). The first N GOPs in a segment are defined as prefix and the remaining GOPs in a segment as suffix, as shown in Fig. 3. The size of the prefix N is adjusted to the round trip time and the transmission capacity between the remote server and the proxy. The size of the prefix is always less than or equal to M.

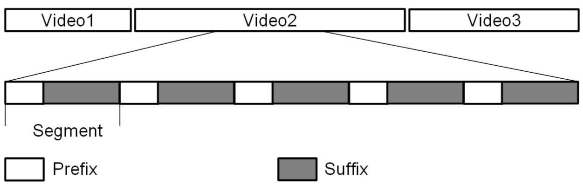


Fig. 3. Segment structure with N GOPs for the prefix and (M-N) GOPs for the suffix

2.2. Playout strategy

Every GOP is independently decodeable, which means, for VoD service, every first frame in a GOP can be the possible access point. However, each GOP should then either be cached on the proxy or be loaded from the server when it is requested, which leads to either a large required cache size or a possible long waiting time. As observed during our subjective VoD tests, users would rather have a small shift of starting point than suffering a long waiting time. Based on this, we assume that the playout of the video starts always from the beginning of a segment instead of the beginning of a GOP, where the segment contains the starting point the user wants to access. This will cause the playout to start from a time point, which might be several seconds earlier than the time point the user has selected. For instance, a user moves the slide bar of a video player, sets it to the time point at the 10^{th} minute (i.e., 00:10:00), and starts playing. But actually the playout starts from 00:09:55. As shown in Fig. 2, such an early playout will not disturb the user too much if we limit the maximum length of the segment. However, it brings us the benefit of shorter initial waiting time for the same cache size, because only N out of M GOPs need to be available to start playout with short initial delay.

When a request from the client reaches the proxy, if all prefix frames are in the cache, the proxy sends the first frame of the requested segment immediately to the client. Otherwise, it waits until all frames of the corresponding prefix are downloaded from the remote server. The proxy sends another request to the server to download the unavailable suffix frames. After a round trip time delay, when the client still plays the prefix frames, the first suffix frame reaches the proxy and can be continuously forwarded to the client, which leads to an uninterrupted playout at the user side.

2.3. Weighted Waiting Time

Because of the random access of the video content, each segment in the same video file may have dramatically different popularity. For example, segments with goals in a football match will be visited more frequently than other segments. If we take the mean popularity over all segments in the video file, it makes the cache algorithm inefficient. Therefore, instead of evaluating the popularity for the entire video as in [8] and [9], we measure the popularity for every segment, which is defined here as the total number of requests of the frames in that segment during a predefined time interval.

If a frame needs a long time to be downloaded from the server, we'd like to store the frame on the proxy to reduce the client's initial waiting time. Similarly, if a frame is frequently requested because of its high popularity, we'd also like to store that frame on the proxy to reduce the expected initial waiting time.

Our popularity-aware cache adaptation algorithm uses the Weighted Waiting Time (WWT) to decide which frames should be dropped from the proxy cache. The WWT of a frame is defined as:

$$WWT_i = P_j \cdot \frac{S_i}{R_k} \, , \tag{1}$$

where P_j denotes the popularity of segment j, which frame i belongs to. S_i gives the frame size of frame i and R_k is the transmission rate between the proxy and the corresponding server k, where video frame i is stored. Higher WWT means the frame either has a higher popularity or needs a long time to be downloaded from the server, thus has a higher priority to be cached on the proxy.

2.4. Partial Caching Algorithm

Because of the huge size of video files and the limited cache size on the proxy, it is impossible to cache all the video content on the proxy. At the same time, new video content has to be added and the popularity of the video content also changes over time. Therefore, the proxy cache should be updated periodically, leaving out some unpopular frames and prefetching those frames with large WWT. According to our proposed segmentation structure and playout strategy, suffix frames are firstly removed so that with a small shift of the starting point, the initial delay becomes very small. However, all suffix frames in one segment have the same popularity and which frames to cache on the proxy has no influence on the initial delay, because as long as all prefix frames are available on the proxy, the same initial playout delay will be experienced. We here give an example way to select the suffix frames according to the so called Weighted Distortion (WD). The distortion when one frame is absent for the decoding can be calculated as in [12]. We take the product of the frame's distortion and its popularity to select the frames to be replaced. By doing so, the expected distortion is reduced when a transmission error occurs between the server and the proxy. Of course, other more sophisticated replacement algorithms can also be employed for the removal of suffix frames.

If all suffix frames have been dropped and even more storage capacity is required, some prefix frames have to be dropped. An incomplete prefix must be filled up before the proxy sends the first frame to the client. Hence, we drop the frames in the prefix with the smallest WWT, so that the expected initial waiting time can be minimized.

Algorithm 1: Popularity-aware partial caching algorithm

```
begin
    while ∑_{k=1}^{K} S_k ≥ CacheCapacity do
        if There is still a frame in the suffix then
            forall Remaining frames in the suffix do
                Choose the frame with minimum WD;
            Drop the chosen frame;
        else
            forall Remaining frames in the prefix do
                Choose the frame with minimum WWT;
            Drop the chosen frame;
end
```

The algorithm is shown in Algorithm 1. K is the total number of frames stored in the proxy cache plus those newly downloaded from the server. S_k is the frame size of the k^{th} frame.

3. SIMULATION RESULTS

In order to evaluate the proposed cache update algorithms, we have built up a test environment with three video streams. Video 1 is a news report out of date with 6496 frames, which is considered to be unpopular. Video 2 containing 47382 frames is an indoor football match and we assume it is very popular. Video 3 is an intraday news report consisting of 22398 frames, which has a medium popularity. Furthermore, segments of one video can also have different popularity and some segments in video 3 have a higher popularity than some segments in video 2. All the videos in our experiments are encoded using H.264/AVC[1], with QCIF resolution and a frame rate of 30 fps. Every GOP consists of 30 frames and 2 B frames are set between two I or P frames. We assume that the local network has a very high speed so that the transmission delay between proxy and client can be ignored. This means that if the video frame is on the proxy, it can be played out without any initial delay. The round trip time between the server and the proxy is assumed to be 1 second, which corresponds to 30 frames. For simplicity, here we assume that the link rate between the servers and the proxy has a constant rate which is equal to the mean rate of the video stream asked for, which is 187kbps, 641kbps and 268kbps for video 1, video 2 and video 3, respectively. Another assumption is that the prefix has a length of 30 frames, corresponding to 1 GOP, which means that if the user wants to start playout within one segment, all the frames in the prefix GOP must be available on the proxy before the first frame is sent to the client. We here also assume that the segment length is 3 GOPs. Random user requests from the clients are simulated to validate the accuracy of our estimation. 1000 requests are used in our experiment with the popularity of individual requests taken from a Zipf distribution [13].

To show the improvements achievable by the algorithm, we compare our proposed PAPA approach with the conventionally used cache algorithm for VoD services. In this case, the first frame of a GOP is the possible starting point for the random access. When the proxy cache can not hold all the frames, the frames at the end of each GOP are first removed.

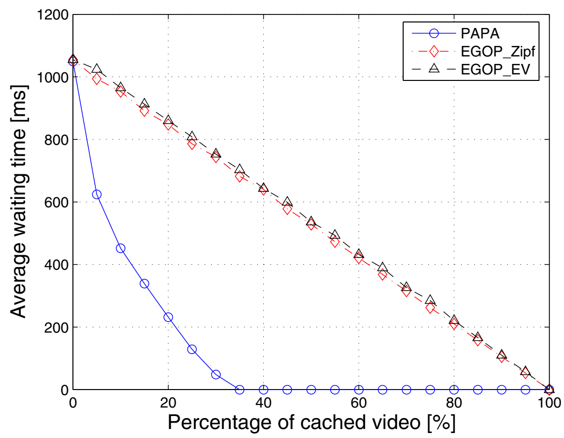


Fig. 4. Average Waiting Time for different caching strategies as a function of the percentage of all video frames being stored in the cache

In Fig. 4, PAPA shows the performance of our proposed WWT based partial caching algorithm, with $N = 1$ and $M = 3$. EGOP_EV stands for a cache update strategy where frames from the last GOP towards the first GOP of the video stream are dropped. For every GOP only the last available frame is left out in every step. If all last frames have been dropped, the procedure starts dropping the second last frames and so on. In contrast, EGOP_Zipf starts dropping frames from the GOP with the lowest popularity. Again, for every GOP only the last available frame is dropped in every step.

[1]Encoded with the software JM 10.2. http://iphome.hhi.de/suehring/tml/

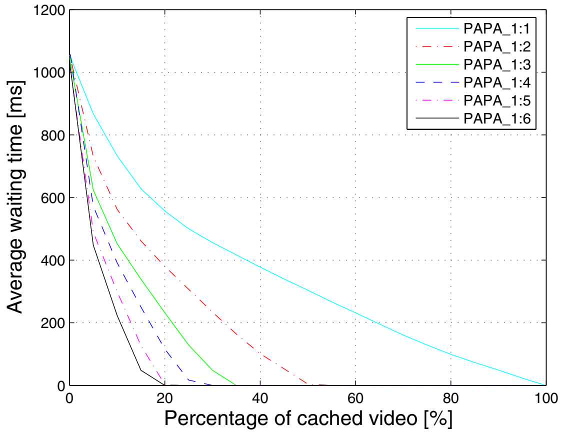


Fig. 5. Average Waiting Time by different cache capacity

As expected, PAPA achieves a much lower average waiting time compared to EGOP_EV and EGOP_Zipf at almost all cache percentages. When the caching rate is zero, the average waiting time is a bit more than 1000ms, which is due to the assumption that highly popular segments mostly have strong motion, thus higher bit rate than the mean rate. However, the initial waiting time of PAPA decreases rapidly and even reaches zero when more than 40% of the video content are cached, which is due to the "early start" playout strategy. In this experiment, the minimum early start is 0 when the requested frame is in the first GOP of the segment. Otherwise, an early start of 1 or 2 seconds are observed when the requested frame is in the second or third GOP of the segment, respectively. From the result of the subjective test in Fig. 2, an early start of 1s should lead to a similar user experience as a waiting time of 200ms in EGOP approaches.

For a video clip, if it is popular, we would like to have a small segment size so that more frames are cached as a prefix on the proxy. Otherwise, the segment size should be enlarged and gives a quasi-VoD option with fewer possible access points. At the same time, if the cache is lightly loaded, we can have small segment size and vice versa. In Fig. 5, we show the average initial waiting time as a function of the segment size. The prefix size corresponds to one GOP. PAPA_1:5 stands for a 1:5 ratio between the prefix and the segment, which means only around 20% of the total video content needs to be cached to enable a zero initial waiting time, with an early start playout of 2.5s on average and 5s maximum. If we decrease the size of the segment, the caching percentage for zero initial time increases, however, the early start time also decreases. When prefix:segment reaches 1:1, it turns to the case that every GOP is a possible random access point, which leads to no early playout. Please note, if we compare the PAPA_1:1 curve in Fig. 5 with the EGOP_EV and EGOP_Zipf curves in Fig. 4, both taking every GOP as an access point, PAPA_1:1 still performs much better than the EGOP dropping strategies. This is because PAPA_1:1 can drop all those frames in the unpopular segments first, while EGOP approaches have to drop frames evenly from all GOPs, although EGOP_Zipf considers also the popularity of the segments.

From the above simulation results, we can find that our proposed cache update algorithm is a very flexible approach, which can dynamically change the size of the segment according to the number of videos and available cache size to achieve a maximum user satisfaction.

4. CONCLUSION

In this paper we propose a popularity-aware partial caching algorithm for VoD. This algorithm considers random access user behavior. In order to minimize the initial waiting time, the algorithm changes the exact starting time within the bounds typically tolerated by VoD users. The users are able to seek the video and start the playout with virtually no initial waiting time for popular video segments, although only a part of the video is stored on the proxy. The simulation results show that our algorithm significantly decreases the initial waiting time at the same cache capacity.

5. REFERENCES

[1] Subhabrata Sen, Jennifer Rexford, and Don Towsley, "Proxy prefix caching for multimedia streams," in *Proc. IEEE International Conference on Computer and Communications (INFOCOM'99)*, New York, NY, Apr. 1999.

[2] Shudong Jin, Azer Bestavros, and Arun Iyengar, "Accelerating internet streaming media delivery using network-aware partial caching," in *Proc. IEEE International Conference on Distributed Computing System (ICDC'02)*, Vienna, Austria, July 2002.

[3] Hyung Rai Oh and Hwangjun Song, "Scalable proxy caching algorithm minimizing client's buffer size and channel bandwidth," *Journal of Visual Communication and Image Representation*, vol. 17, no. 1, pp. 57–71, Feb. 2006.

[4] Zhourong Miao and Antonio Ortega, "Scalable proxy caching of video under storage constrains," *IEEE J. Select. Areas Commun.*, vol. 20, no. 7, pp. 1315–1327, Sept. 2002.

[5] Markus Hofmann, T.S. Eugene Ng, Katherine Guo, Sanjoy Paul, and Hui Zhang, "Caching techniques for streaming multimedia over the internet," Tech. Rep. BL011345-990409-04TM, Bell Labs, Holmdel, NJ, Apr. 1999.

[6] Ethendranath Bommaiah, Katherine Guo, Markus Hofmann, and Sanjoy Paul, "Design and implementation of a caching system for streaming media over the internet," in *Proc. IEEE Real-Time Technology and Applications Symposium (RTAS'00)*, Washington, DC, May/Jun 2000.

[7] Carsten Griwodz, Michael Bör, and Lars C. Wolf, "Long-term movie popularity models in video-on-demand systems of the life of an on-demand movie," in *Proc. ACM International Conference on Multimedia*, Seattle, Washington, Nov. 1997.

[8] Martin Reisslein, Felix Hartanto, and Keith W. Ross, "Interactive video streaming with proxy servers," *Journal of Information Sciences – Informatics and Computer Science*, vol. 140, no. 1, pp. 3–31, Jan. 2002.

[9] Haijin Yan and David K. Lowenthal, "Popularity-aware cache replacement in streaming environments," in *Proc. International Conference on Parallel and Distributed Computing Systems (PDCS'03)*, Reno, Nevada, Aug. 2003.

[10] Reza Rejaie and Jussi Kangasharju, "Mocha: A quality adaptive multimedia proxy cache for internet streaming," in *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'01)*, Port Jefferson, NY, June 2001.

[11] Changxi Zheng, Guobin Shen, and Shipeng Li, "Distributed prefetching scheme for random seek support in peertopeer streaming applications," in *Proc. ACM Workshop on Advances in Peer-to-Peer Multimedia Streaming*, Hilton, Singapore, Nov. 2005.

[12] Wei Tu, Wolfgang Kellerer, and Eckehard Steinbach, "Rate-distortion optimized video frame dropping on active network nodes," in *Proc. International Packet Video Workshop 2004 (PV'04)*, Irvine, CA, Dec. 2004.

[13] George Kingsley Zipf, *Human Behaviour and the Principles of Least Effort*, Addison-Wesley, Cambridge, MA, 1949.