"Mirror segmentation": A New Segmentation Strategy for Video Objects inside Video Caches

Claudiu Cobârzan
"Babeş-Bolyai" University, Computer Science Department,
Mihail Kogălniceanu 1,
400084 Cluj-Napoca, Romania
claudiu@cs.ubbcluj.ro

Abstract

Multiple strategies for caching video objects have been proposed in recent years in order to maximize byte-hit ratio and better cope with large sizes of video objects. Usually, a video proxy-cache will use the same strategy when caching all its objects. We propose a new approach, which enables a proxy-cache to decide which strategy to use on a per object basis. In this context we introduce a new segment-based strategy called "mirror segmentation" which is to be deployed only for select objects, depending on clients' access patterns. Some initial measurements regarding the new segmentation strategy's performance using byte-hit ratio as a metric are also presented as well as comparisons with other caching strategies.

1. Introduction

The demand for multimedia (especially video) objects on the Internet has greatly increased in recent years. This tends to stress the existing network infrastructure a lot, since video objects are much larger than, for example, web objects and they also require certain quality of service guaranties. In this context video caching has become extremely important because efficient strategies could ensure good use of external bandwidth through increased byte hit ratio and also intelligent use of local storage resources. This in turn would lead to benefits on the client side like increased availability and small latency.

Numerous video cache replacement strategies have proposed some form of segmentation for the stored objects. When the cache comes under storage constraints and space has to be freed in order to accommodate newly requested and retrieved objects, only the last segments of the selected objects are discarded and not the whole object. For those situations we propose a new segmentation strategy which is applied only for some special objects: the objects providing hints they might again become popular. The aim is to try to correctly identify those objects and if they are selected for cache replacement, try for a limited period of time to minimize the amount of data that is discarded from them.

This approach is new in the sense that unlike most of the previously proposed strategies it does not treat all the cached video objects the same way. For the objects that are considered for cache replacement but might be requested sometime in the near future, a new segmentation strategy we have named "mirror segmentation" is applied. This segmentation strategy uses up to a certain point P inside the object, segments that increase in length as they move away from the beginning of the object (each segment has double the length of the previous one). After the considered point P is reached, the size of the segments decrease until de end of the object (each segment is half the size of the previous one).

Following, we make a short review of existing approaches to video caching, as well as a detailed presentation of the "mirror segmentation" strategy. Some initial measurements in a simulated environment regarding its performance using *byte hit ratio* as metric, as well as conclusions and intended future work conclude the paper.

2. Related work

Most approaches to video caching consider partial video caching strategies, meaning that only specific parts of the video objects are considered. Examples of such proposals include caching of a prefix [15], caching of a prefix and of selected frames [8], caching of a prefix combined with periodic broadcast [17] or caching of hotspot segments [7]. Caching of a prefix based on popularity [11], segment-based prefix caching [16] and variable sized chunk caching [2] have also been considered.

Another approach is to perform cache replacement con-

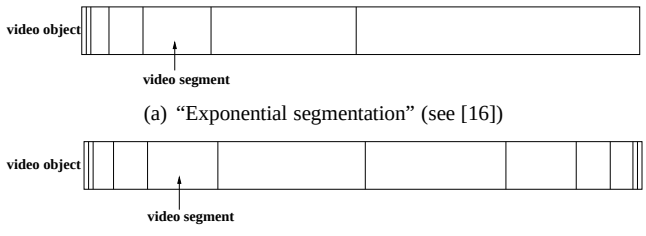
sidering the quality of the objects. Proposals include periodic caching of layered coded videos [9], adaptive caching of layered coded videos in combination with congestion control [13], quality adjusted caching of GoPs (group of pictures) [14] and also simple replacement strategies (patterns) for videos consisting of different quality steps [12].

There have been also distributed approaches to video caching: in [3] multiple video servers accessible via the web managing tertiary storage systems are proposed while [1] introduces a cooperative caching video server. In [4] and [5] a dynamic distributed architecture able to vary the number of participating nodes depending on client request patterns, network load and locally available resources is introduced.

The work we are presenting is based on the work in [16] and is intended to be integrated in the system proposed by [4] and [5].

3. New approach: "mirror segmentation"

Existing segmentation approaches are based, upon others, on the assumption that once a video is selected for cache replacement it's time life inside the cache is close to the end. It is considered that after segments of a video object are discarded, it is only a matter of time until the whole object is removed from the cache. Although this is true most of the times, there are situations when an object that has become unpopular and is selected for cache replacement due to storage constraints, starts to be requested again by clients, sometimes even after some of it's segments have been already discarded. In such a situation, if standard segmentation strategies are deployed (e.g. [16]) the data that has to be retrieved (after being previously discarded) is greater than in the situation in which "mirror segmentation" is used, as can easily be seen in Figure 1.



(a) "Exponential segmentation" (see [16])

(b) "Mirror segmentation"

Figure 1. Segmentation strategies: (a) "exponential segmentation" vs. (b) "mirror segmentation"

Our proposal is aimed for exactly such cases: a segmentation strategy that ensures that up to a certain point, the data quantity that is discarded at one step is small enough and can be quickly retrieved if needed from one of the origin servers (video servers that hold objects of interest).

We consider in a manner similar to [16] that a video object that is requested by a client will not be fully stored in the local cache from the first request. Instead only it's first data segment will be locally saved. As the object gets requested again, the next segment is stored. Except for the first j segments at the beginning of the object that have the same size S, it is considered that the size of segment $j+1$ is double the size of segment j. While in [16] this is true for all segments, we consider in our approach a point/segment P past which this is not true anymore. In fact all segments following point/segment P will be half the size of the previous segment.

We use the following notations:

LC - the content of the local cache

$LC = \{o_i, i = 1..q\}$, q = the number of cached video objects

A video object o is defined as:

$$o = (size(o), duration(o), bitRate(o), timeLastAccess(o), hitCount(o)) \quad (1)$$

The function $SS : LCX\mathbf{N} \rightarrow \mathbf{R}$ defines the "mirror segmentation" scheme for each object in the local cache as follows:

$$SS(o,i) = \begin{cases} S & \text{if } i \leq j, j \geq 1 \\ 2SS(o,i-1) & \text{if } j < i \leq P \\ \dfrac{SS(o,i-1)}{2} & \text{if } i > P \end{cases} \quad (2)$$

where P is the first segment for which the condition

$$\sum_{i=j+1}^{i<P} SS(o,i) \geq \frac{size(o)}{2} \quad (3)$$

is true (i represents the segment's order number within video object o; j is an arbitrary number; S the size of each of the first j segments). Equation (2) and (3) state that ascending segment sizes are used up to the middle of each object o. Past that point, the size of the segments is descending.

In order to have access to the elapsed time between two consecutive requests for an object o_i we define a function Δ as follows:

$\Delta : LCX\mathbf{N} \rightarrow \mathbf{R}$,

$\Delta(o,i) = t_i(o) - t_{i-1}(o), \forall i > 1$

where

$t : LC \rightarrow \mathbf{R}$ is a time function,

$t_i(o)$ = the number of seconds elapsed between the moment the video proxy-cache became active and the moment the object o has been requested the i^{th} time

We always take into consideration the last 3 requests when deciding if an object o selected for cache replacement should be "mirror segmented". We denote by n the total number of times the object o has been requested. Since we are interested only in the last 3 requests we will consider only the values of $\Delta(o, n)$ and $\Delta(o, n-1)$. If $\Delta(o, n-1) > \Delta(o, n)$ holds then we might have a situation similar to the one we have already described: an object that is requested again after a long(er) time - when compared in our case with the last 2 previous requests.

This condition alone does not fully justify "mirror segmentation" for object o:

- the last 3 requests for object o are not reported to the current time CT when it has been selected for cache replacement;
- the average life time of an object inside the cache is not known.

Let us consider that observations are made regarding the *average life time (ALT)* of an objects inside the cache (the time interval between the moment it begins to be locally stored and the moment it is selected for deletion) and that ALT has the value V.

If for object o we have

$$CT - t_n(o) \gg V$$

than "mirror segmentation" might be useless.

This basically means that if the time interval between the moment in which object o was selected for cache replacement and the moment in which it was last requested by a client is much larger than the observed life time for the objects in the cache then "mirror segmentation" might not be useful anymore.

So, in order for "mirror segmentation" to be performed for an object o that has been selected for cache replacement we require that the following two conditions are true:

1. $\Delta(o, n-1) > \Delta(o, n)$
2. $CT - t_n(o) \leq V$

where as we have already mentioned above

n represents the number of times object o has been requested ($\Delta(o, n)$ represents the time elapsed between the last two requests for object o);

CT is the moment in time object o has been selected for cache replacement;

$t_n(o)$ is the moment in time object o has been last requested;

V represents the average life time of the objects in the cache.

4. Measurements

We have performed a series of measurements using an enhanced version of the vpcSim [6] simulator and synthetic trace logs generated using the WebTraff [10] generator. We assumed that once requested, an object is partially/fully retrieved from an origin server (if it is not already cached) and it is viewed from start to end with no interruptions or cancellations. Also no bandwidth limitations nor transmission errors were considered.

Following we present the results obtained for a trace with 1000 requests for 300 video objects that followed a Zipf distribution with $\alpha = 0.3$. The number of objects requested only once was in this case 70% from the total of 300. The size of the objects followed a Pareto distribution with a tail index set to 1.2.

We have considered two segmentation strategies: the one in [16] we have called "exponential" and the "mirror segmentation" strategy. For both, the size of the initial segment was set to 1 MB. Additionally we took into account the case when the objects are not segmented ("web-like" caching). The objects were selected for cache replacement in all cases considering their *utility* value: if a cache replacement operation has to be performed the object having the smallest utility is selected, it's last segment is discarded and it's utility is recomputed if it is segmented, or it is fully discarded if no segmentation is applied.

The used formula for computing an object's utility is the one in [5] and considers information on the size of the object, the moment in time it has been last requested, the number of times it has been requested and it's quality value. For this set of tests we have considered only the object's *size* and *hit count*, each representing 50% of the utility value.

In the case of the "mirror segmentation" strategy we have used increasing segments up to half the size of each object o and decreasing segments from half the size up to it's end. The strategy was applied for all the objects in the cache once they have been completely retrieved (during this process "exponential segmentation" was used).

The size of the cache varied from 1% to 10% (values on the x axis in Figure 2) of the total volume of the 300 considered objects.

The obtained results can be seen in Figure 2. It is obvious that using any of the two considered segmentation strategies significantly increases the performance of the proxy-cache.

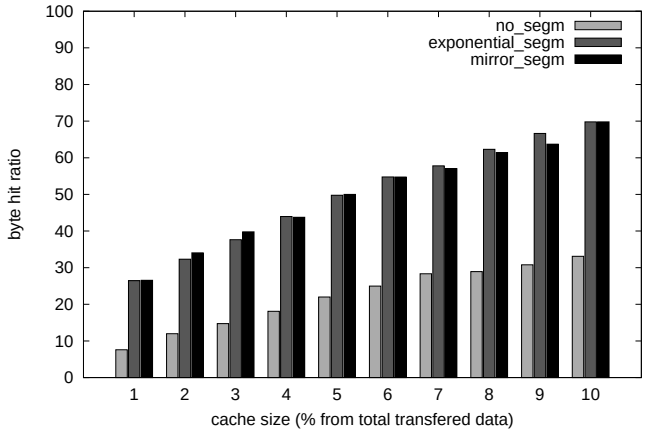


Figure 2. Byte hit ratio values for the same trace when no segmentation, "exponential segmentation" respectively "mirror segmentation" is used

For smaller cache sizes "mirror segmentation" usually performs better, while for larger cache sizes using "exponential segmentation" constitutes an advantage. This can be explained by the fact that for smaller cache sizes, replacement operations are performed more often. If objects that had some of their segments already discarded are requested again, a smaller amount of data has to be retrieved from origin servers if "mirror segmentation" is used than in a similar case when "exponential segmentation" is considered.

The big difference in terms of performance when segmentation (either "mirror" or "exponential") is used as opposed to the situation when no segmentation of objects is applied can be explained by the coarser granularity of the data on which cache replacement is performed. In the first two cases (with segmentation) the amount of discarded data whenever cache replacement is performed is (much) smaller than in the third case (no segmentation) when whole objects are discarded if cache replacement becomes necessary. The impact is seen when such an object (on which cache replacement has been performed) is requested again and a smaller or (much) larger amount of data has to be retrieved from origin servers.

Please note that those results were obtained when "mirror segmentation" was applied for all objects that were fully cached, without considering the conditions in Section 3 making thus the worst case scenario. Even in this case the results are acceptable and we expect even better ones when applying the above mentioned conditions.

5. Conclusion and Future Work

We have presented a new strategy for segmenting video objects inside a proxy-cache which aims at minimizing (for a limited period of time) the amount of data that is discarded from some special objects. Those special objects are the ones selected for cache replacement but which might become popular in the near future. We also provide a mean to identify those objects based on their request history. The next step would be to make more performance measurements in a simulated environment considering byte hit ratio as metric and traces with various statistical properties. We also intend to integrate this approach into the distributed system described in [4] and [5] and evaluate it's performance in such a dynamic system compared with it's use on a single node.

References

[1] S. Acharya and B. Smith. Middleman: A video caching proxy server. In *Proceedings of the 10th International Workshop on Network and Operating System Support forDigital Audio and Video*, 2002.

[2] E. Balafoutis, A. Panagakis, N. Laoutaris, and I. Stavrakakis. The impact of replacement granularity on video caching. In *IFIP Networking 2002*, volume 2345 of *Lecture Notes in Computer Science*. Springer, 2002.

[3] D. W. Brubeck and L. A. Rowe. Hierarchical storage management in a distributed vod system. *IEEE MultiMedia*, 3(3):37–47, 1996.

[4] C. Cobârzan. Dynamic proxy-cache multiplication inside LANs. In *Euro-Par 2005*, volume 3648 of *Lecture Notes in Computer Science*, pages 890–900. Springer, 2005.

[5] C. Cobârzan and L. Böszörményi. Further developments of a dynamic distributed video proxy-cache system. In *Proceedings of the 15th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP 2007)*, pages 349–357. IEEE Computer Society, 2007.

[6] C. Cobârzan and D. Doruţiu. vpcsim: A video proxy-cache simulator. In *Proceedings of the Symposium Colocviul Academic Clujean de Informatică*, pages 135–140, 2005.

[7] H. Fabmi, M. Latif, S. Sedigh-Ali, A. Ghafoor, P. Liu, and L. Hsu. Proxy servers for scalable interactive video support. *Computer*, 34(9):54–60, 2001.

[8] W. hsiu Ma and D. H.-C. Du. Reducing bandwidth requirement for delivering video over wide area networks with proxy server. In *IEEE International Conference on Multimedia and Expo (II)*, pages 991–994. IEEE Computer Society, 2000.

[9] J. Kangasharju, F. Hartanto, M. Reisslein, and K. W. Ross. Distributing layered encoded video through caches. In *IEEE INFOCOM*, pages 1791–1800. IEEE Computer Society, 2001.

[10] N. Markatchev and C. Williamson. Webtraff: A gui for web proxy cache workload modeling and analysis. In *MASCOTS '02: Proceedings of the 10th IEEE International Symposium*

on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'02). IEEE Computer Society, 2002.

[11] S.-H. Park, E.-J. Lim, and K.-D. Chung. Popularity-based partial caching for vod systems using a proxy server. In *Proceedings of the 15th International Parallel & Distributed Processing Symposium (IPDPS-01)*. IEEE Computer Society, 2001.

[12] S. Podlipnig and L. Böszörményi. Replacement strategies for quality based video caching. In *IEEE International Conference on Multimedia and Expo (ICME), Vol. 2*, pages 49–53. IEEE Computer Society, 2002.

[13] R. Rejaie and J. Kangasharju. Mocha: A quality adaptive multimedia proxy cache for internet streaming. In *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 3–10. ACM Press, 2001.

[14] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara. Proxy caching mechanisms with video quality adjustment. In *Proceedings of SPIE International Symposium on The Convergence of Information Technologies and Communications*, pages 276–284, 2001.

[15] S. Sen, J. Rexford, and D. F. Towsley. Proxy prefix caching for multimedia streams. In *IEEE INFOCOM*, pages 1310–1319. IEEE Computer Society, 1999.

[16] K.-L. Wu, P. S. Yu, and J. L. Wolf. Segment-based proxy caching of multimedia streams. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 36–44. ACM Press, 2001.

[17] S. S. Yang Guo and D. Towsley. Prefix caching assisted periodic broadcast for streaming popular videos. In *Proceedings of ICC (International Conference on Communications)*, pages 2607 – 2612. IEEE Computer Society, 2002.