Video Personalization and Caching for Resource Constrained Environments

Siddhartha Chattopadhyay[†] **Suchendra M. Bhandarkar**[‡]
Yong Wei[‡] **Lakshmish Ramaswamy**[‡]
[†]Google Inc., Mountain View, California 94043, USA
[‡]Dept. of Computer Science, The University of Georgia, Athens, Georgia 30602-7404, USA

Abstract: Video playback in mobile devices is becoming increasingly popular. Since mobile devices are typically resource constrained, various video personalization strategies are used to provide personalized video content that is most relevant to the client's request. In this paper, we propose a novel video personalization server and cache architecture, which uses automatic video segmentation and video indexing based on semantic video content. A novel cache design and a novel cache replacement algorithm, specifically suited for caching personalized video files are proposed. The proposed cache replacement algorithm is shown to significantly outperform several state-of-the-art cache replacement policies.

1. INTRODUCTION

Handheld mobile computing and communication devices, such as personal digital assistants (PDAs), pocket-PCs and cellular devices have become increasingly capable of storage, rendering and display of multimedia data. The user demand for being able to view streaming video on such devices has increased several-fold. Handheld mobile devices are typically constrained by their battery power capacity, viewing time limit and, in many situations, by the available network bandwidth connecting them with video servers. Thus, the original video content often needs to be personalized in order to fulfill the client's request while satisfying simultaneously various client-side system-level resource constraints. Numerous video personalization strategies have been developed [5], [6], [7] in order to provide these resource constrained devices with personalized video content that is most relevant to the client's request while satisfying simultaneously the available power and network resource constraints.

Since mobile clients are typically not within network proximity of the personalizing server, it is often desirable to "smartly" cache portions of the video files in order to reduce the latency observed at the client end and also offshore the data load on the server to local caches. In this paper, we have proposed a novel video personalization server (VPS) [15] [16] which performs automatic video segmentation and video indexing based on semantic video content, and generates personalized videos based on client preference using a Multiple-Choice Multi-Dimensional Knapsack Problem (MMKP)-based video personalization strategy [9],[10]. Further, a novel cache design, specifically suited for efficient caching of personalized video files on the proposed VPS, is also proposed.

2. VIDEO PERSONALIZATION SERVER

The Video Personalization Server (VPS) creates and stores multiple transcoded versions of existing videos for the client based on the battery capacity of the client device. The videos on the VPS are first segmented and indexed, followed by video personalization based on client preferences.

A stochastic model-based algorithm incorporating a multi-level Hidden Markov Model (HMM) is used for video segmentation and indexing. The input video stream is classified frame by frame into semantic units [17], which is a video segment that can be associated with a clear semantic

meaning or concept, and consists of a concatenation of semantically and temporally related video shots or video scenes. The semantic units within a video stream can be spliced together to form a logical video sequence that the viewer can comprehend. In the proposed video segmentation and video indexing scheme based on semantic video content [15] [16] [17], we define six semantic concepts for TV broadcast news video, i.e. *News Anchor*, *News*, *Sports News*, *Commercial*, *Weather Forecast* and *Program Header*, and three semantic concepts for Major League Soccer (MLS) video, i.e. *Zoom Out*, *Close Up* and *Replay*. An HMM is formulated for each individual semantic concept. The optimal HMM parameters for each semantic unit are learned from the feature vector sequences obtained from the training video data. In our scheme, the HMMs for individual semantic units are trained separately using appropriate training feature vector sequences. This allows for modularity in the learning procedure and flexibility in terms of being able to accommodate various types of video data.

The search space for the proposed single-pass video segmentation and video indexing procedure is characterized by the concatenation of the HMMs corresponding to the individual semantic units. The HMM corresponding to an individual semantic unit essentially models the stochastic behavior of the sequence of image features within the scope of that semantic unit. Transitions amongst these semantic unit HMMs are regulated by a pre-specified video program model. The Viterbi algorithm is used to determine the optimal path in the concatenation of the HMMs in order to segment and index video stream in a single pass [17].

The objective of video personalization is to present a customized or personalized video summary that retains as much of the semantic content desired by the client as possible but within the system-level constraints imposed by the client and its environment. The video segments are indexed using semantic terms. Each video segment is assigned a relevance value based on the client's preference with regard to video content. Each indexed video segment is summarized at multiple levels of abstraction using content-aware key frame selection and motion panorama computation algorithms. Each video summary consists of a set of key frames and motion panoramas. For each video segment, its original version is assumed to contain the greatest amount of detail; whereas its summary at the highest level of abstraction is assumed to contain the least amount of detail.

The client typically wants to retrieve and view only the contents that match his/her content preferences. In order to generate the personalized video summary, the client preferences, the client usage environment and client-side system-level constraints need to be considered. The personalization engine selects the optimal set of video contents (i.e., the most relevant set of video summaries) for the client within the system-level constraints imposed by the client device and its environment.

In the video content database, each video segment and summary is assigned a relevance value based on the client's content preferences, as computed in equations (1) and (2) respectively. We have used a Multiple-Choice Multi-Dimensional Knapsack Problem [8], [9] (MMKP)-based video personalization strategy, which can be solved using the branch and bound integer programming (BBIP) algorithm described in [10]. For further details on the MMKP-based video personalization strategy, the interested reader is referred to [15], [16] and [17].

3. VIDEO PERSONALIZATION CACHE

The Video personalization scheme described above is used to generate personalized video content based on the client requests and resource constraints. In order to reduce the latency observed by the client, we employ a specialized cache which resides between the clients and the VPS.

3.1 Cache design

The cache acts as a buffer between the VPS and the multiple clients, in order to reduce latency observed by the clients. It also acts as a generating server, which generates videos of varying quality, in order to increase the cache efficiency. Each video segment that the cache receives to be stored is used to create two more versions, or layers, of the same video. The original video segment, or layer, V_{org}, is used to create two videos, V_{mid} which is a lower-quality video compared to V_{org}, and with a correspondingly smaller file size, and V_{base}, which is of lowest quality and smallest file size. The lower-quality videos V_{mid} and V_{base} have smaller file sizes compared to V_{org}. As will be seen in Section 3.3., the lower-quality videos play an important role in the cache replacement policy. In the next section, we describe in detail how V_{mid} and V_{base} are generated. The multi-client, cache and video personalization system is depicted in **Figure 1**.

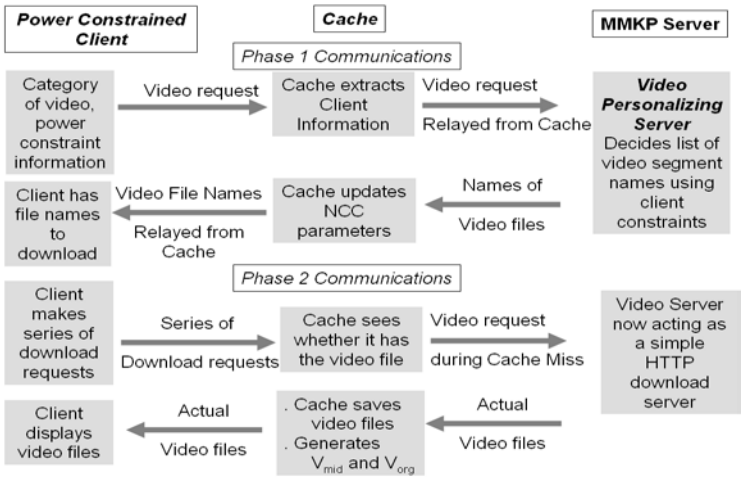

Figure 1. The client-cache-server communication protocol. The protocol is to be read from top to bottom. NCC stands for number of common clients. V_{mid} and V_{org} are lower quality videos generated at the cache from each original video segment.

3.2 Creating different layers of the video

The different video layers represent transcoded versions of the original video with varying visual quality and corresponding file sizes significantly smaller than the file size of the original video. Two lower-quality video layers, V_{mid} and V_{base}, are generated for each video segment, V_{org}. The video layer V_{mid} represents an intermediate-level video which has a more compact representation than the original video V_{org}, albeit at the cost of lower visual quality. It is generated using a novel multi-resolution video encoding technique termed as Features, Motion and Object-Enhanced Multi-Resolution (FMOE-MR) video encoding [11]. The FMOE-MR video encoding scheme is based on the fundamental observation that applying a low pass filter in the image color space is equivalent to DCT coefficient truncation in the corresponding DCT (frequency) space [12]. The various regions in a video frame are smoothed to different extents based on their visual or perceptual significance. In the interest of brevity, the interested reader is referred to [11] for further details. It must be noted

that V_{mid} is finally encoded using the MPEG H.264 standard, after preprocessing using FMOE-MR, in a manner similar to the original video segment.

The base video layer, V_{base}, is generated by first blurring each frame of the video using a Gaussian filter with smoothing parameter σ_{base} prior to MPEG H.264 encoding. This results in further dramatic decrease in the file size upon MPEG H.264 encoding, albeit with a loss in video quality. We have observed empirically that values of σ_{base} in the range [19, 25] can be used to generate the base video layer V_{base}, resulting in a very small file size, albeit at the cost of low resolution and low visual quality.

In addition to V_{org}, the files corresponding to V_{mid} and V_{base} are also stored in the cache for a given video stream. For empirically determined values of the encoding parameters σ_L, σ_H and σ_{base},, the combined size of V_{base} and V_{org} results in a 50% storage overhead for each video segment..

3.3 Client-Cache-Server communication

We have considered a setup involving a single VPS, a single cache, and multiple clients. The client-cache-server communication is done in two phases:

Phase 1: Multiple clients transmit their queries or requests to the cache, which relays the requests to the VPS. Each client request specifies one video category out of six possible categories (*News Anchor*, *News*, *Sports News*, *Commercial*, *Weather Forecast* and *Program Header*), and the available battery time on the client device. The VPS creates a list of video segments based on client requests, and sends the names of the video segments to be downloaded, in order, as metadata to the cache. The cache relays the information back to each client.

Phase 2: Each client now makes a series of requests to the cache to fetch the files belonging to one of the six categories. The files can be streamed, or progressively downloaded, depending on the type of service available at the cache and the VPS. In either case, the names of the files, and the order they appear in, are provided as metadata in *Phase 1* of the communication. When a client asks for a video segment file from the cache, it is either a hit or a miss. **Figure 1** depicts the above communication phases.

A novelty of the proposed cache design is that a cache access is classified as a hit or miss depending on the "client-type". A "client-type" defines the membership grade of the client with respect to the video personalization service. Without loss of generality, we have used two categories of clients; "paid" clients, who pay for the best quality of service and "unpaid" clients who receive videos of varying quality depending on the availability of the video in the cache. For "unpaid" clients, the best quality video is not guaranteed; whereas the best quality video is guaranteed for the "paid" client. For each query from an "unpaid" client, the cache manager checks if the requested file V_{org} is resident in the cache. If V_{org} is not resident in the cache, it checks whether the next lower quality version V_{mid} is resident in the cache. If V_{mid} is not resident in the cache, it checks whether V_{base} is resident in the cache. If neither V_{mid} nor V_{base} is resident in the cache, the cache access is treated as a miss. In the case of a "paid" client, the cache access is treated as a miss if V_{org} is not resident in the cache.

For a cache miss, the required file is relayed from the VPS, and also stored in the cache simultaneously. If there is not enough space in the cache, a cache replacement policy is used to replace existing file(s) in order to make space for the new file. If the cache cannot accommodate the new file, an existing file with minimum *retention-value* (RV) is discarded. The *retention-value* is essentially a number associated with each file in the cache; the lower the *retention-value*, the less valuable the file to the cache. Thus, if a file is to be removed from the cache in order to make space

in the cache, the file with the lowest *retention-value* is removed. The *retention-value* is specific to a cache replacement policy. For the proposed cache, we have computed the *retention-value* as follows:

$$retention\text{-}value = NCC/fileSize$$

where NCC is the *number of common clients* for that particular file, and *fileSize* is the size of the file. This cache replacement policy has been termed NCCS (Number of Common Clients - Size) algorithm. NCCS essentially marks the number of clients that will be requesting this particular file. In order to give priority to files which are requested by "paid" clients, the number of clients (NCC) is incremented by a factor of 10, instead of 1, for "paid" clients as opposed to "unpaid" clients. This information is obtained during the first phase of the communication between the cache and VPS. For a miss, first, the video file with the minimum *retention-value* is identified. After that, an attempt is made to remove the original video segment; i.e., V_{org} corresponding to that video file. If V_{org} had already been removed previously, then V_{mid} is removed; if V_{mid} has also been removed, then V_{base} is removed. This top down approach ensures that the layer which takes up the largest space in the cache is removed, so that space for some more, more useful video segments, can be made.

4. EXPERIMENTAL RESULTS AND DISCUSSION

4.1. Experiment setup

In order to simulate multiple clients, we have considered 100 clients which send requests to the VPS with a time lag determined by the following equation:

$$TimeLag(clientID) = K*(1 + rand*clientID)$$

where K is a constant, typically depicting several seconds, $0 \leq clientID < 100$, and *rand* is a random number between 0 and 1. It is assumed that around 40% of he clients are "paid" clients; the rest are "unpaid". At the VPS, the video segments and their abstractions are stored as independent video files, which can be independently downloaded or streamed from VPS as desired. The combined size of the video segments in the VPS is 140 MB. The storage space available on the cache is typically significantly less compared to that on the VPS. We chose two cache sizes; one that has 33% of the storage capacity of the VPS, and the other which has 50% of the storage capacity of the VPS, in order to note the effects of varying cache size.

4.2. Cache efficiency: hit ratio

In order to perform a comprehensive study of the cache behavior, we have compared several types of cache replacement policies; hence several ways of deriving the *retention-value*. At any point in time during its life in the cache, for the cache replacement policies considered in this paper, the *retention-value* of each video segment depends on the following parameters:

nPayedClients: The number of *paid* clients which are requesting this file. This information is derived during Phase 1 of communication when the clients make initial requests for the files to the cache. If the requesting client type is "unpaid", then the value of *nPayedClients* is incremented by 1. If the client is a "paid" client, then *nPayedClients* is incremented by 10 (the value of 10 is empirically chosen).

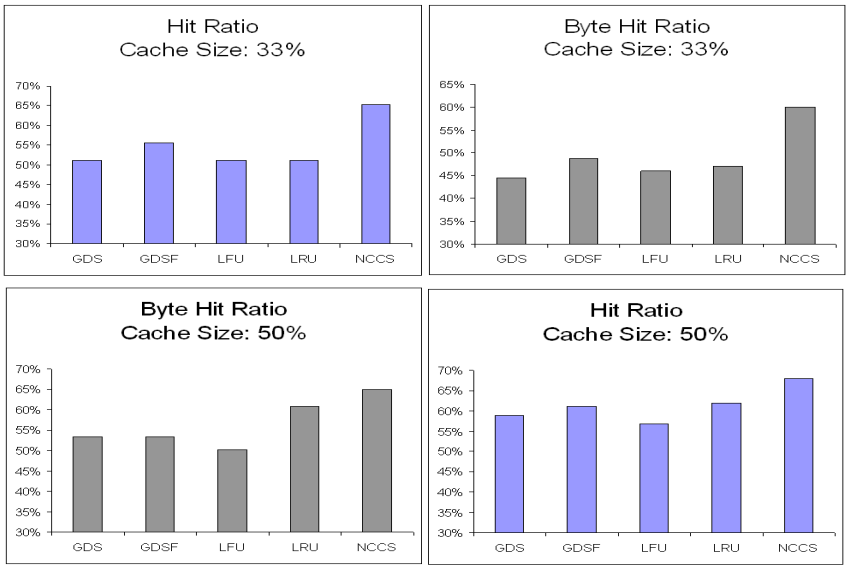

Figure 2. The hit ratio and the byte hit ratio for different cache sizes as a ratio of the server storage capacity.

nMisses: The number of misses that this video segment has had so far. The first time a video file is loaded in the cache is in response to a miss; as a result, the minimum value of *nMisses* is 1.
nHits: The number of hits this video segment has had so far.
Size: The size on hard disk of the video segment file. Typically, larger files are not preferred in the cache because the general philosophy is to have more, smaller files, instead of one, large file.
Latency: Latency incurred while fetching the video segment file. If the video segment files are distributed over multiple video personalizing servers, the latency is important as it is typically not desirable to fetch files that are far away in terms of network distance.
- *hitTime*: The time (in nanoseconds) at which the latest hit to the video segment was made.

An important fact is that the *retention-value* is computed for a video segment file V which comprises of three components: V_{org}, V_{mid} and V_{base}. The three video layers together are considered to constitute a *video bin*. The *retention-value* is computed for the V_{org} component of the *video bin*.

We have considered four other standard cache replacement policies for comparison with the NCCS cache replacement algorithm. The replacement algorithms, and their corresponding *retention-values*, are as follows:

GDS (Greedy-Dual-Size): The GDS cache replacement algorithm [13] uses a retention value given by *retention-value = Latency/Size.* In addition to removing files with the minimum retention value during a cache replacement, the GDS algorithm also subtracts this minimum *retention-value* from the *retention-value* of each of the other files in the cache. When a file is recorded as a hit, the original *retention-value* of the file is recomputed.

GDSF (Greedy-Dual-Size-Frequency): The GDSF cache replacement algorithm [14] uses a retention value given by *retention-value = nHits × Latency/ Size.* Similar to GDS, GDSF subtracts the minimum *retention-value* from the *retention-values* of each of the other files in the cache. When a file is recorded as a hit, the original *retention-value* of the file is recomputed.

LRU (least Recently Used): The retention value is given by *retention-value* = hitTime. This simple retention value is used to replace files accessed farthest in the past.

LFU (Least Frequently Used): The retention value is given by *retention-value = nHits + nMisses.* In other words, the *retention-value* is the number of accesses to the file, which can simply be used as the measure of frequency of access.

We computed the standard metrics *Hit Ratio* and *Byte Hit Ratio*, in order to measure cache efficiency. The *Hit Ratio* measures the number of requests satisfied by the proxy cache as a percentage of the total number of requests. The *Byte Hit Ratio* corresponds to the number of bytes that are transferred from the proxy cache as a percentage of total number of bytes for all the requests. **Figure 2** shows the *Hit Ratio* and Byte Hit Ratio for various cache sizea as a ratio of the VPS storage capacity. NCCS clearly outperforms the other cache replacement policies significantly.

We surmise that the reason NCCS outperforms other cache replacement schemes is due to the two phase communication initiated between the clients and the VPS. The NCCS scheme knows in advance which files will be asked for by the clients. Note that this is a particular situation that can arise only with the proposed two phase communication. Thus, the cache captures the popularity of a file by the commonality of that particular file (video segment) in future client requests. This fact is capture by NCC (number of common clients) statistic. In addition, the *fileSize* factor in the NCCS cache replacement policy makes NCCS prefer smaller, more files, compared to larger, fewer files, which should be the case for a good caching scheme. As a result, the NCCS cache replacement policy is the most successful.

5. CONCLUSIONS

A server-cache architecture has been proposed and implemented, which can disseminate personalized video contents to power constrained devices. The proposed video personalization server (VPS) perfoms automatic video segmentation and video indexing based on semantic video content, and generates personalized videos based on client preference using a Multiple-Choice Multi-Dimensional Knapsack Problem (MMKP)-based video personalization strategy. A novel cache design, dedicated to the caching of video segments generated by the VPS, has been proposed. The proposed caching scheme exploits the commonality of files across clients to result in a novel cache replacement policy termed as NCCS (Number of Common Clients Size). In addition to caching video files, the cache also generates two lower quality versions of each video file. These versions aid in the dissemination of video files to different clients having different levels of importance based on whether or not they pay for the video service. Results show that the proposed cache design, coupled with the NCCS cache replacement policy, outperforms standard cache replacement policies such as GDS, GDSF, LRU and LFU. Thus, the video-cache architecture is suitable for personalized video dissemination to power constrained mobile devices such as mobile phones, PDAs, Pocket PCS and laptop computers operating in battery mode.

REFERENCES

[1] Sikora, T., Trends and Perspectives in Image and Video Coding, *Proc. IEEE*, Vol. 93, No. 1, pp. 6-17, Jan. 2005.

[2] Eickeler, S., and Müller, S., Content-based Video Indexing of TV Broadcast News using Hidden Markov Models, *Proc. IEEE Intl. Conf. Acoustics, Speech Signal Processing*, March 1999, pp. 2997-3000.

[3] Leacock, C., and Chodorow, M., Combining Local Context and WordNet Similarity for Word Sense Identification. in *WordNet: An Electronic Lexical Database*, C. Fellbaum (Editor), MIT Press, Cambridge, MA, 1998, pp. 265-283.

[4] Wheeler, E.S., Zipf's Law and Why it Works Everywhere. *Glottometrics*, Vol.4, 2002, pp. 45-48.

[5] Merialdo, B., Lee, K.T., Luparello, D., and Roudaire, J., Automatic Construction of Personalized TV News Programs. *Proc. ACM Conf. Multimedia*, Orlando, FL, Sept. 1999, pp. 323—331.

[6] Tseng, B.L., and Smith, J.R., Hierarchical Video Summarization Based on Context Clustering, *Proc. SPIE*, Vol. 5242, Nov. 2003, pp. 14-25.

[7] Tseng, B.L., Lin, C.Y., and Smith, J.R., Video Personalization and Summarization System for Usage Environment. *Jour. Visual Communication and Image Representation*, Vol. 15, 2004, pp. 370–392.

[8] Akbar, M.D., Manning, E.G., Shoja, G.C., and Khan, S., Heuristic Solutions for the Multiple-Choice Multi-dimension Knapsack Problem. *Proc. Intl. Conf. Computational Science*, 2001, pp. 659-668.

[9] Hernandez, R.P., and Nikitas, N.J., A New Heuristic for Solving the Multichoice Multidimensional Knapsack Problem. *IEEE Trans. System, Man and Cybernetics*, Part A, Vol. 35, No. 5, 2005, pp. 708-717.

[10] Vanderbei, R. J., *Linear Programming: Foundations and Extensions*, Kluwer Academic Publishers, 1997.

[11] Chattopadhyay, S., Luo, X., Bhandarkar, S. M. and Li, K., FMOE-MR: Content-Driven Multi-Resolution MPEG-4 Fine-Grained Scalable Layered Video Encoding, *Proc. ACM Multimedia Computing and Networking Conference (ACM MMCN' 07),* San Jose, California, January, pp. 650404-1-11, 2007.

[12] Geusebroek, J.-M., Smeulders, A. W. M. and Van De Weijer, J., Fast Anisotropic Gaussian Filtering, *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 11, No. 3, March 2001.

[13] Jin S. and Bestavros, A., Popularity-Aware Greedy Dual-Size Web Proxy Caching Algorithms, *Proc. 20th IEEE Intl. Conf. Dist. Comp. Sys. (ICDCS),* Taipei, Taiwan, Apr. 2000, pp. 254-261.

[14] Cherkasova, L., Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Policy, HP Laboratories Report No. HPL-98-69R1, April, 1998.

[15] Wei, Y., Bhandarkar, S.M. and Li, K., Client-centered Multimedia Content Adaptation, *ACM Trans. Multimedia Computing, Communications and Applications (ACM TOMCCAP)*, in press.

[16] Wei, Y., Bhandarkar, S.M. and Li, K., Video Personalization in Resource-Constrained Multimedia Environments, *Proc. ACM Conf. Multimedia*, Augsburg, Germany, Sept. 2007, pp. 902-911.

[17] Wei, Y., Bhandarkar, S.M. and Li, K., Semantics-based Video Indexing Using a Stochastic Modeling Approach, *Proc. IEEE Intl. Conf. Image Process. (ICIP)*, San Antonio, TX, Sept. 2007.