COST-DISTORTION OPTIMIZED CACHING OF STREAMING MEDIA

Anshul Sehgal

University of Illinois
asehgal@uiuc.edu

Philip A. Chou

Microsoft Research
pachou@microsoft.com

ABSTRACT

Caching of Internet content in caches local to users is beneficial to both users and the network, because it reduces retrieval times for the user as well as reduces load on the network. However, it comes at the cost of storage in the cache, which can be large, particularly when multimedia objects are cached. In this paper, we address the problem of caching multimedia objects such as audio and video streams, in a cost-distortion optimized way. That is, we seek caching policies that minimize the cost of both storage and transmission while minimizing the distortion seen by the user (or equivalently, maximizing the quality seen by the user). Simulation results demonstrate that caching in a cost-distortion optimized manner leads to significant improvements over the case where no caching is performed.

1. INTRODUCTION

Web caching refers to temporary storage of web objects for future retrieval by users. Web caching is beneficial for both users and the network, because it allows users to access web objects with relatively small retrieval times and it simultaneously reduces network traffic. However, this comes at the cost of storage in the cache.

There exists a significant amount of literature on caching of web objects such as HTML documents. Most of this work is related to the design of cache replacement policies for finite size caches. Central to the design of these policies is the mechanism used to estimate the popularity of web objects. The problem of caching of images has also started receiving attention in recent years [1, 2]. This problem is distinguished from caching of documents by the loss-resilience properties of images. In this paper, we address the problem of caching streaming media objects, such as audio and video streams. Media streams present additional challenges for caching. Besides being loss-resilient, media streams are typically large, can be cached or accessed in portions, and have real-time delivery constraints.

For caching of media streams, two problems need to be addressed: 1) for each portion (e.g., each data unit) of the media stream, the caching proxy needs to estimate the number of users who will be requesting that portion, and 2) having estimated this quantity the proxy needs to decide whether or not to cache the portion. In this paper, we tackle the second problem, i.e., assuming that an estimate of the number of requests for each data unit is available, we devise an algorithm to optimally select data units to cache based on the cost and distortion characteristics of the data units. The problem of caching media streams has also been tackled in [3, 4, 5, 6]. In [3], the authors propose to cache the first few minutes of the stream. When a user requests a stream, the caching proxy transmits the stored prefix of the requested stream *and simultaneously requests the remainder of the stream from the* server. In this way, chances that the user is starved (i.e., buffer underflow) are reduced. In [5], the authors propose to cache those parts of a VBR stream where the rates are higher than the nominal bandwidth of the server-proxy-user link, thus smoothing the bandwidth required. They call their scheme "video staging." The work of [6] is the closest to ours. In that work the authors design algorithms for caching based on a "robustness" function. They maximize the robustness function by selectively storing parts of the media stream.

We examine two caching scenarios: prefetch caching and passive caching. Prefetch caching refers to the scenario where the caching proxy proactively requests data from the server during non-peak times and caches the data for future use. Passive caching refers to the scenario where the caching proxy intercepts data sent from the server to the first user and caches the data for future use. In either case the caching proxy must have a policy for deciding whether or not to cache the data. Caching the data increases storage costs, but it also reduces transmission costs and reduces the delay and error experienced by the average user. Hence there is a trade-off to be made.

This paper addresses the problem of optimizing the caching policy in a cost-distortion optimization setting. Our objective is to minimize the cost of both storage and transmission while minimizing the distortion seen by the user (or equivalently, maximizing the quality seen by the user). Central to our modeling of the problem is the introduction of a notion of prices. We assume that a "rent" c_s must be paid for every byte of data stored in the cache per unit time, and that a fee c_t must be paid for every byte of data transmitted from the server to the cache. Furthermore, we assume that data transmitted from the server to the cache may be lost or delayed. On the other hand, we assume that communication between the cache and its users is free, reliable, and instantaneous. We detail this model more precisely in the next section.

The paper is organized as follows: Section 2 describes our abstraction of the transmission scenario. Section 3 describes the optimization problem for caching a single data unit. Section 4 solves the optimization problem for caching an entire multimedia stream. Experimental results are presented in Section 5. Conclusions are drawn in Section 6.

2. MODEL DESCRIPTION

In this section, we define our abstraction of the encoding, packetization and communication processes and state the cost-distortion optimization problem that we are trying to solve. Our work draw heavily on the rate-distortion optimized streaming algorithm proposed in [7, 8, 9, 10]. We assume that when requesting packets, the receiver follows the rate-distortion optimized receiver-driven streaming algorithm proposed in [10]. For more details on the model, the reader is referred to [7, 8].

In a streaming media system, data are encoded, packetized into

0-7803-7402-9/02/$17.00 ©2002 IEEE

data units, and stored on a file server. Depending on the algorithm used for encoding, data units have dependencies between them that can be represented by an acyclic directed graph. Data unit l is said to be dependent on data unit l' if l cannot be decoded without first decoding l'. For example, in MPEG coded video, data units from a P frame are dependent on data units from the previous I or P frame, while data units from a B frame are dependent on data units from the closest I and P frames.

Associated with each data unit l is a quantity ΔD_l, which denotes the decrease in distortion if data unit l is decoded on time, and a quantity B_l, which denotes the size of the data unit in bytes. Also associated with each data unit is a decoding time stamp $t_{DTS,l}$, which is the time by which the data unit must be available at the decoder in order to be decoded and played back.

We model the caching scenario as follows: The client requests data units from the caching proxy. This request is communicated to the proxy without cost, without loss, and without delay. If the proxy has the requested data units in its cache, then it relays them to the client, without cost, loss, or delay. If the proxy does not have the requested data units in its cache, then it requests them from the server on behalf of the client. This request is communicated from the proxy to the server without cost, but the request may be lost or delayed. Upon receipt of a request, the server transmits the requested data units to the proxy. This transmission occurs with cost c_t per transmitted byte, and additionally the transmitted data units may be lost or delayed. If and when the proxy receives a data unit from the server, it forwards the data unit to the client, without additional cost, loss, or delay.

Specifically, the network path from the server to the proxy (the forward channel) and the network path from the proxy to the server (the backward channel) are modeled as independent time-invariant packet erasure channels with random delays. This means that each packet inserted into a channel is independently lost with probability ϵ_F for the forward channel and ϵ_B for the backward channel. If it is not lost, then it is delayed by a random time: FTT (forward trip time) for the forward channel and BTT (backward trip time) for the backward channel, which are respectively drawn from probability densities p_F and p_B. In our simulations, we use for p_F and p_B shifted Gamma distributions with parameters (n_F, α) and (n_B, α) and right shifts κ_F and κ_B, respectively. These induce a distribution on the round trip time, $RTT = FTT + BTT$, which is a shifted Gamma distribution p_R with parameters (n_R, α), $n_R = n_F + n_B$, and right shift $\kappa_R = \kappa_F + \kappa_B$. Also induced is the round trip loss probability $\epsilon_R = \epsilon_B + (1 - \epsilon_B)\epsilon_F$. We define $P(RTT > \tau) = \epsilon_R + (1 - \epsilon_R)\int_\tau^\infty p_R(t)dt$ to be the probability that a request packet sent from the proxy to the server at time t does not result in the requested data packet arriving at the proxy by time $t + \tau$, whether the packets are lost or simply delay by more than τ.

As an example, consider the case when the proxy requests a single data unit l from the server at time t_l. The probability that the server ever receives the request is $(1 - \epsilon_B)$. If it receives the request, the server transmits the data unit in a packet on the forward channel. Assuming that an amount c_t is paid for every byte transmitted on the forward channel, the expected transmission cost for data unit l is $(1-\epsilon_B)c_tB_l$, where B_l is the size of the data unit in bytes. The probability that the proxy does not receive the data unit by its deadline $t_{DTS,l}$ is $P(RTT > t_{DTS,l} - t_l)$.

This example can be generalized to the case when the proxy has N_l opportunities to request data unit l, at times $t_{0,l}, t_{1,l}, \ldots t_{N-1,l}$. At each of these times, the proxy may send a request to the server if it has not yet received the data unit. Let $a_{j,l} = 1$ indicate that the proxy will send a request to the server at opportunity $t_{j,l}$ if it has not yet received the data unit, and let $a_{j,l} = 0$ otherwise. Let $\pi_l = (a_{0,l}, a_{1,l}, \ldots, a_{N_l,l})$. This is called the proxy's *request policy* for data unit l. Under this request policy, the probability that the proxy will send a request to the server at opportunity $t_{i,l}$ is $\prod_{j<i, a_j=1} P(RTT > t_{i,l} - t_{j,l})$. Hence the expected number of times that the server receives a request is $\rho(\pi_l) = \sum_{i:a_i=1}(1 - \epsilon_B)\prod_{j<i,a_j=1} P(RTT > t_{i,l} - t_{j,l})$. Thus the expected transmission cost for data unit l incurred by this policy is $\rho(\pi_l)c_tB_l$, and the expected error (i.e., the probability that data unit l does not arrive at the cache by its deadline $t_{DTS,l}$) is $\epsilon(\pi_l) = \prod_{i:a_i=1} P(RTT > t_{DTS,l} - t_{i,l})$. Plotting the points $(\rho(\pi_l), \epsilon(\pi_l))$ for all 2^{N_l} realizations of the policy π_l for a single data unit and restricting ourselves to the policies π_l on the lower convex hull of this set of points yields a function $\epsilon(\rho)$ characterizing the optimal operational trade-off between cost and error. For more details, the reader is referred to [7, 8].

3. CACHING OF A SINGLE DATA UNIT

In this section, we consider both prefetch caching and passive caching of a single data unit. In prefetch caching, time is divided into epochs of T seconds (say of duration one day). At the beginning of each epoch, the caching proxy estimates the number of users R_l that will be requesting data unit l in a particular multimedia stream. Based on this estimate and other factors, the proxy may prefetch the data unit and cache it for time T.

If the proxy decides to prefetch and cache the data unit, then it incurs transmission cost $\rho_0 c_t B_l$, where $\rho_0 = 1/(1 - \epsilon_F)$ is the expected number of times the server will transmit the data unit while prefetching, c_t is the price for transmitting a single byte, and B_l is the size of the data unit in bytes. In addition, the proxy incurs storage cost $c_s B_l$, where c_s is the cost per byte of storage for time T. Once the data unit is in the cache, it can be communicated to its R_l users without cost, loss, or delay. Thus the total cost of transmission and storage to prefetch and cache the data unit during the epoch is $(c_s + \rho_0 c_t)B_l$. This cost is amortized over all R_l users during the epoch. Furthermore, the users experience no loss or delay when requesting the data unit from the proxy.

On the other hand, if the proxy decides not to prefetch and cache the data unit, then it incurs a transmission cost $\rho_0 c_t B_l$ for each of the R_l users during the epoch. Furthermore, each user will experience loss or delay when requesting the data unit from the proxy, since the proxy simply forwards the request to the server.

At the end of the epoch, the proxy updates its estimate of R_l and repeats the process for another epoch. Of course, if the data unit is already in the cache from a previous epoch, then it would make sense not to request the data unit once again from the server. This is the notion of passive caching. In passive caching, the caching proxy simply decides whether or not to continue caching a data unit that it has already acquired. Thus the difference between prefetch caching and passive caching is that in passive caching, the prefetch cost $\rho_0 c_t B_l$ is not incurred, or equivalently, $\rho_0 = 0$. Perhaps a more typical scenario for passive caching is the scenario where the proxy intercepts data sent from the server to a first user, and may cache the data for future users in a new epoch.

Let $\sigma_l = 1$ if the proxy decides to cache the data unit during the epoch and let $\sigma_l = 0$ otherwise. If $\sigma_l = 1$, then the expected cost over the epoch is $(c_s + \rho_0 c_t)B_l/R_l$ per user, while the ex-

pected error per user is 0. On the other hand, if $\sigma_l = 0$, then the expected cost over the epoch is $\rho(\pi_l)c_t B_l$ per user, while the expected error per user is $\epsilon(\pi_l)$. Hence to minimize the expected error subject to a constraint on the expected cost, the proxy should choose π_l and σ_l to minimize the Lagrangian

$$(1-\sigma_l)\epsilon(\pi_l) + \lambda'((1-\sigma_l)\rho(\pi_l)c_t + \sigma_l(c_s + \rho_0 c_t)/R_l) \quad (1)$$

for some Lagrange multiplier $\lambda' > 0$. Since the number of possibilities is small, this optimization can be carried out by a brute force search over all realizations of π_l and σ_l. In the next section, we use (1) as a building block for the problem of caching when data units have dependencies between them (as would be the case for most encoded audio/video).

4. CACHING A GROUP OF DATA UNITS

Our goal is to minimize the expected distortion seen by the average user, subject to a constraint on the expected cost per user, over all request policies and caching decisions for the individual data units in a media stream. However, because of the dependences between data units, it is not possible to determine the optimal request policy and the cache decision for each data unit independently of the other data units. For example, if a data unit is not cached, then it becomes less desirable to cache the data units that depend on it. Nevertheless, in this section we show that using an iterative scheme, it is possible to determine the request policy and cache decision for each data unit by minimizing (1) within each iteration, such that the resulting request policies and cache decisions are optimal, or at least locally optimal, for the overall problem.

Let L be the number of data units in the media stream. Then let $\boldsymbol{\pi} = (\pi_1, \pi_2, ..., \pi_L)$ denote the vector of request policies and let $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, ..., \sigma_L)$ denote the vector of cache decisions for all L data units. The total expected cost for all data units, per user, is therefore

$$C(\boldsymbol{\pi}, \boldsymbol{\sigma}) = \sum_l ((1-\sigma_l)\rho(\pi_l)c_t + \sigma_l(c_s + \rho_0 c_t)/R_l)B_l. \quad (2)$$

This is simply the sum of the expected costs for each data unit, per user.

The expected distortion is somewhat more complicated to express. Let I_l be the indicator random variable that is 1 if the data unit l is available to the user on time, and 0 otherwise. Note that the expectation of I_l is $\sigma_l + (1-\sigma_l)(1-\epsilon(\pi_l))$, where $\epsilon(\pi_l)$ is the probability that data unit l is not available to the user by its deadline, using policy π_l to request the data unit from the server. The product $\prod_{l' \preceq l} I_{l'}$ is 1 if data unit l is decodable on time (i.e., if data unit l and all its predecessors are available to the user on time), and is 0 otherwise. If data unit l is decodable on time, then the distortion is reduced by ΔD_l; otherwise the distortion is not reduced. Hence, the total decrease in distortion is $\sum_l \Delta D_l \prod_{l' \preceq l} I_{l'}$. Subtracting this quantity from distortion if no data units are received, and taking the expectations, we have the expected distortion

$$D(\boldsymbol{\pi}, \boldsymbol{\sigma}) = D_0 - \sum_l \Delta D_l \prod_{l' \preceq l} (\sigma_{l'} + (1-\sigma_{l'})(1-\epsilon(\pi_{l'}))), \quad (3)$$

where D_0 is the expected reconstruction error if no data units are received. Here, as in [7, 8], we have made the assumption that data packet transmission processes are independent, and are independent of the source process, in order to factor the expectation in (3).

With the expressions for the expected distortion and the expected cost in hand, we can now set up the optimization problem as minimizing the expected distortion subject to an expected cost constraint. By restricting ourselves to solutions on the lower convex hull of the set of cost-distortion pairs $(C(\boldsymbol{\pi}, \boldsymbol{\sigma}), D(\boldsymbol{\pi}, \boldsymbol{\sigma}))$, we can solve the problem by finding $(\boldsymbol{\pi}, \boldsymbol{\sigma})$ minimizing the expected Lagrangian

$$J(\boldsymbol{\pi}, \boldsymbol{\sigma}) = D(\boldsymbol{\pi}, \boldsymbol{\sigma}) + \lambda C(\boldsymbol{\pi}, \boldsymbol{\sigma}). \quad (4)$$

Unfortunately, this minimization is complicated by the fact that the expression for the expected distortion cannot be split into a sum of terms. We solve this problem by using an iterative descent procedure as described in [7, 8] called the Sensitivity Adjustment (SA) algorithm. Let $\boldsymbol{\pi}^{(0)} = (\pi_1^{(0)}, \pi_2^{(0)}, \ldots, \pi_L^{(0)})$ and $\boldsymbol{\sigma}^{(0)} = (\sigma_1^{(0)}, \sigma_2^{(0)}, \ldots, \sigma_L^{(0)})$ be any initial guess of the request policy vector and the cache decision vector, respectively. Similarly, let $\boldsymbol{\pi}^{(n-1)}$ and $\boldsymbol{\sigma}^{(n-1)}$ be the estimates of these quantities at iteration $n-1$. At iteration n, select one component $l_n \in \{1, 2, ..., L\}$ (for example, $l_n = (n \mod L)$). For $l \neq l_n$, let $\pi_l^{(n)} = \pi_l^{(n-1)}$ and let $\sigma_l^{(n)} = \sigma_l^{(n-1)}$, while for $l = l_n$, let $\pi_l^{(n)}$ and $\sigma_l^{(n)}$ be chosen as $\arg\min_{\pi_l, \sigma_l} J(\boldsymbol{\pi}, \boldsymbol{\sigma})$, or equivalently, chosen to minimize

$$(1-\sigma_l)S_l^{(n)}\epsilon(\pi_l) + \lambda((1-\sigma_l)\rho(\pi_l)c_t + \sigma_l(c_s + \rho_0 c_t)/R_l)B_l, \quad (5)$$

where (5) follows from (4) with

$$S_l^{(n)} = \sum_{l' \succeq l} \Delta D_{l'} \prod_{l'' \preceq l',\, l'' \neq l} (\sigma_{l''} + (1-\sigma_{l''})(1-\epsilon(\pi_{l''}))). \quad (6)$$

Here, S_l is the sensitivity to not caching or losing data unit l. Note that (5) is the same as (1) with $\lambda' = \lambda B_l / S_l$.

For prefetch caching, the proxy runs the above optimization for each stream at the beginning of the epoch and decides which data units should be prefetched. For passive caching, the proxy runs the above optimization over only the data units that the first user requests and not the entire stream.

5. EXPERIMENTAL RESULTS

In this section we report the results of our simulations for prefetch and passive caching. The channel parameters shown in Table 1 were used for simulations. The channel has a mean RTT of 200 ms and a drop probability of 0.1. One minute of a packetized audio stream (Sarah McLachlans *Building a Mystery*) was used for simulations. The audio content was compressed using a scalable version of the *Windows Media Audio Codec*. The codec performs perceptual weighting on lapped orthogonal transform coefficients, followed by bitplane coding to produce an embedded bit stream for each group of frames (GOF) of duration 0.75 seconds. The bit stream of each GOF is partitioned into segments of length 500 bytes and packetized into data units. Twelve 500 byte data units are kept for each GOF, for a maximum bit-rate of $12 \times 500 \times 8 / 0.75 = 64$ Kbps. The twelve data units for each GOF are sequentially dependent. Each data unit l in the GOF is labeled by the decrease ΔD_l in the perceptually weighted squared error if the data unit is decoded on time and all of its predecessors in the same GOF are decoded on time. All twelve data units of each GOF receive the same decoding timestamp. A playback delay of

δ = 750ms (equal to one GOF length) was used for all simulations. The storage price c_s and the transmission price c_t were set to c_s = \$0.50 per megabyte stored per month and c_t = \$0.12 per megabyte transferred. These prices were obtained from a web space renting company. Transmitted requests and data packets are dropped at random and those not dropped receive a random delay according to the shifted Gamma distribution. The results were averaged over multiple runs to smooth out the effect of any one particular channel realization.

Next, we evaluate the performance of the proposed algorithm with the setup described above. Figures 1 and 2 plot, for prefetch and passive caching respectively, the average SNR versus cost per user curves for different values of the request rate ($R_i = R$) along with the SNR versus cost curve when no caching is performed. These plots were generated by varying λ and keeping all other parameters constant. As can be observed, even for small values of R caching leads to significant gains over not caching at all. Also, by varying λ, it is possible to obtain a trade-off between the average decoded stream quality and price by selecting only a subset of units to cache as opposed to caching the entire stream.

We also note, that unlike the solution proposed in [3], the cost-distortion optimized solution to the problem is to cache the more "important" data units from the entire stream. Roughly, these correspond to the base layers in our experiments and the I frames in MPEG coded streams.

6. CONCLUSIONS

A cost-distortion optimized proxy caching algorithm was described in this paper. The performance of the algorithm shows significant savings by caching only a subset of stream as opposed to not caching at all. The performance of the algorithm also suggests that a good heuristic would be to cache the more important data units in a stream as opposed to caching the first few minutes of the stream.

7. REFERENCES

[1] A. Ortega, F. Carignano, S. Ayer, M. Vetterli, "Soft caching: web cache management for images," *IEEE Signal Processing Society Workshop on Multimedia*, Princeton, NJ, June 1997.

[2] X. Yang, K. Ramchandran, "An optimal and efficient soft caching algorithm for network image retrieval,"*Proc. of ICIP*, Chicago, 1998.

[3] S. Sen, R. Rexford, D. Towsley, "Proxy prefix caching for multimedia streams," *Proc. IEEE Infocom*, New York, NY, 1999.

[4] R. Rejaie, M. Handley, H. Yu, D. Estrin, "Proxy caching mechanisms for multimedia playback streams in the internet,"*Proc.* 4^{th} *Web Cache Workshop*, San Diego, CA, 1999.

[5] Z. Zhang, Y. Wang, D. Du, D. Su, "Video staging: A proxy server based approach to end-to-end video delivery over wide-area-networks," *IEEE Trans. on Networking*, vol. 8, no. 4, pages 429-442, 2000.

[6] Z. Miao, "Algorithms for streaming, caching and storage of digital media," *Ph.D. dissertation proposal*, USC, 2001.

[7] P. A. Chou, Z. Miao, "Rate-distortion optimized streaming of packetized media," *Microsoft Research Technical Report MSR-TR-2001-35*, February 2001.

[8] P. A. Chou, Z. Miao, "Rate-distortion optimized streaming of packetized media, "*IEEE Transactions on Multimedia*, February 2001. Submitted.

[9] P. Chou, Z. Miao, "Rate-distortion optimized sender-driven streaming over best-effort networks," *IEEE Workshop on Multimedia Signal Processing*, Cannes, France, October 2001.

[10] A. Sehgal, P. A. Chou, "Rate-distortion optimized receiver-driven streaming over best-effort networks," *Data Compression Conference*, April 2002. In preparation.

n_F	2	n_B	2
κ_F	50 ms	κ_B	50 ms
$1/\alpha_F$	25 ms	$1/\alpha_B$	25 ms
ϵ_F	0.1	ϵ_B	0.1

Table 1. Parameter values used for simulations

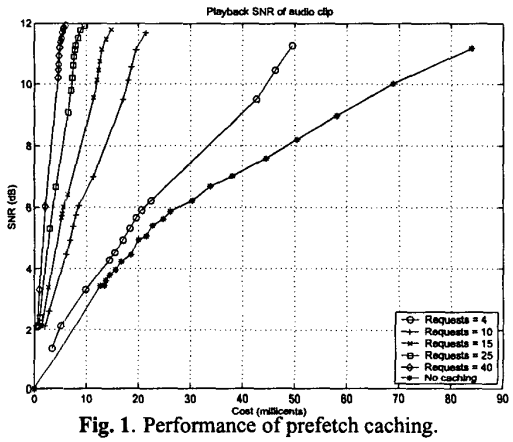

Fig. 1. Performance of prefetch caching.

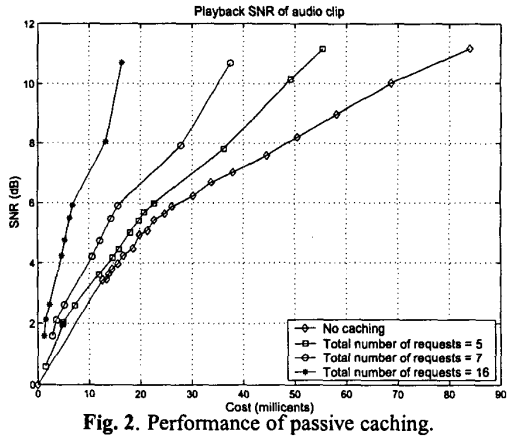

Fig. 2. Performance of passive caching.