SOFT CACHING: WEB CACHE MANAGEMENT TECHNIQUES FOR IMAGES

A. Ortega, F. Carignano
Integrated Media Systems Center
University of Southern California
Los Angeles, CA, USA

S. Ayer and M. Vetterli[1]
Visual Communications Lab
EPFL
Lausanne, Switzerland

Abstract - The vast majority of current Internet traffic is generated by web browsing applications. Proxy caching, which allows some of the most popular web objects to be cached at intermediate nodes within the network, has been shown to provide substantial performance improvements. In this paper we argue that image-specific caching strategies are desirable and will result in improved performance over approaches treating all objects alike. We propose that *Soft Caching*, where an image can be cached at one of a set of levels of resolutions, can benefit the overall performance when combined with cache management strategies that estimate, for each object, both the bandwidth to the server where the object is stored and the appropriate resolution level demanded by the user. We formalize the cache management problem under these conditions and describe an experimental system to test these techniques.

INTRODUCTION AND MOTIVATION

The explosive growth in Internet traffic has made it critical to look for ways of accommodating the increasing number of users while preventing excessive delays, congestion and widespread blackouts. Increased transmission capacity and more sophisticated pricing will help in the long run relieve current bottlenecks but an immediate goal in both research and commercial environments has been to define methods which can provide a more efficient utilization of existing resources. For the past several years a large proportion of Internet traffic has been generated by web-browsing. This has led to a great deal of effort being devoted to studying *web caching* techniques [1].

Consider the basic interaction in web browsing, where a client requests an object[2] that is stored at a server host. While caching is useful both at the server (for example some pages might be kept in RAM memory) and the client (where recently accessed files are saved to disk), we concentrate here on *proxy* based caching. In this environment clients can designate a host to serve as a proxy for all or some of their requests (e.g., http, ftp, etc). The proxy acts as

[1] Martin Vetterli is also with Department of EECS, University of California, Berkeley, CA 94720, USA

[2] In the web context an object is a file (text, image, audio, executable) which is available to be downloaded by a client.

0-7803-3780-8/97/$10.00 ©1997 IEEE

a cache by temporarily storing on local disks, or memory, objects which were requested by the clients. When one of the clients sharing the proxy generates a request, the proxy searches its local storage for the requested object. If the object is available locally (hit) it is sent to the client, otherwise (miss) the request is passed on to the remote server (or to another proxy server, if the proxy cache belongs to a hierarchy of caches.)

Obviously caching will improve the overall performance of the system as long as the *hit ratio*, i.e. the ratio of locally available information to total volume of requests, is sufficiently high. However, unlike traditional low level caching, as used in most current computer architectures, a relatively low hit ratio suffices to make using a web caching system worthwhile. This is true because the overhead of a miss (getting the object from the remote server) can be very high compared to the speed of a local search and transfer and thus the savings on a few hits are sufficient to make up for the overhead needed for searching the cache storage first.

The potential benefits of caching have sparked commercial and research interest. Several companies offer a proxy server among their products [2, 3] while government-funded projects have resulted in a freely available cache implementation [4, 1]. Web caching can be divided into two main classes, namely, *push*, where the server places information in the network caches [5], and *pull*, where proxies operate independently of the servers and store information as a function of clients requests. In this work we concentrate on pull-based environments, which are also the most popular in terms of implementation and research interest [2, 3, 4].

There are several aspects which clearly differentiate web caching from traditional caching environments. For example the above mentioned low hit ratio requirement, but also the fact that computation and memory at the proxy come relatively cheap and thus sophisticated cache management strategies are possible, including algorithms with different approaches for each class of objects. Another significant difference is that the bandwidths to the various servers are different (and indeed can change over time) and thus the cost of a miss does not depend on the size of the object alone. The goal of this paper is to motivate that increased levels of performance can be achieved with *web caching strategies specifically geared towards images.*

SOFT CACHING

A web cache design addresses two main questions (see [1] for details):
1. Given a requested object that is not locally available, should it be fetched and stored for future requests? The answer depends for example on the object type, since dynamic objects (e.g. counters) may not be suitable for caching.
2. Given the current state of the cache, i.e. number of objects and their characteristics (size, number of times they have been accessed, last access time, etc), and given the limited storage available, what objects should remain in the cache and what objects should be erased? This "garbage collection" or object removal policy (see for example [6]) will be the focus of our study.

Current design efforts have provided answers to the above questions which are valid for generic objects, while assuming that object integrity has to be preserved, i.e. objects cannot be modified. Thus an object is either present or not in the cache, but cannot be *available in part*. Here we propose that caching proxies should be able to perform recoding of the images in the cache so that lower resolution versions of the images can be stored and made available to the clients. We call this strategy *Soft Caching*, since we now allow a lower resolution version of an image object to be stored; a soft decision is made (how many bits to use for a certain image) instead of a hard one (is the image left in the cache or not).

In our framework, specific caching strategies are derived for images and other media objects, for which preserving the object integrity is not completely necessary. While incomplete delivery of text data may render it useless, for images it may be sufficient to provide the end user with a lower resolution version of the image stored at the remote server, especially if this can be done in significantly less time than it would take to download the full resolution image. This is one of the reasons for the continuing popularity of progressive image formats such as Progressive JPEG [7] nowadays, or pyramids [8] and wavelets [9] in the near future.

For access over a fast link, progressive transmission may not offer significant advantages, but over a slow link a progressive format allows a usable, albeit reduced resolution, picture to be available even if the transfer is terminated early. This allows users to stop the transfer when a sufficient quality image has been downloaded. A study of such a multiresolution imaging system under simple assumptions can be found in [10, 11] and motivates the advantages in terms of average access delay of using a progressive transmission.

Real-time distillation [12, 13] has been proposed to allow proxies to extract (distill) a low resolution image to serve it to slow clients (e.g. clients connected to the network via dial-up). While this approach is similar in philosophy, we consider here a more general case where a shared resource (the cache memory) is taken into account and other configurations are considered (including for example a fast local network and a remote access to the server).

CACHE MANAGEMENT STRATEGIES FOR IMAGES

In our framework the cache management task consists of determining both which images to maintain in the cache *and* the level of resolution at which they should be stored. As for generic objects, the objective of the cache management algorithm should be to minimize the average access time per image.

We consider a scenario where the user is served each image first at whatever resolution is available in the cache (or full resolution if the image is not available there) and then can request the full resolution image to be fetched by pressing "reload". From an implementation standpoint this can be achieved by filtering all the requests involving images and handling them separately from other objects. Serving the reload request can be done by requesting

from the server either (A) the full resolution object or (B) the missing layers to complete full resolution decoding.[3]

Information available at the proxy Assume a total of N different images have been accessed recently. Denote R_i the size in bits of the i-th image and let A_i be the number of times that the image has been requested during the observation period. Let r_i be the size in bits of the reduced resolution version of the i-th image currently stored in the cache and let $\mathcal{R}_i$ be the set of all available resolutions for this image. In a general scenario the cache serves J clients $(C_1, \ldots, C_j, \ldots, C_J)$, which access information stored in K servers $(S_1, \ldots, S_k, \ldots, S_K)$. Let $D_i(r_i)$ represent the number of bits required to obtain the full resolution image given that the r_i bits are available in the cache. Under scenario (A) above $D_i(r_i) = R_i$, while in the more efficient case (B), $D_i = R_i - r_i$. Among the data available to the cache manager will be the estimated bandwidth in bit/sec to a given server or client, b_{S_k} and B_{C_j}, respectively. This information can be obtained by recording the time the corresponding sockets remain open and the image sizes (both are available in log files of typical proxy caching software). Bandwidths associated to each server have been considered and proven to be useful for management of general web caches [14, 15]. Here we consider the particular case of image caches.

Single object case Assume that an image object is present at the cache and we would like to determine at what resolution it should be kept there. Assume that for any value of r_i we can estimate $p_r(r_i)$, the probability that a reload will occur if the image is stored at resolution r_i. Note that we expect $p_r(r_i)$ to become small as r_i gets close to R_i. Conversely, $p_r(r_i)$ should increase as r_i decreases.

We can define, for a choice of r_i, the expected download time for a given object as

$$E[t(r_i)] = \frac{r_i}{b_c}(1 - p_r(r_i)) \quad + \quad \frac{D_i(r_i)}{b_{s_k} + b_c} p_r(r_i)$$

where we have assumed that object i is stored in server k and we use the $D_i(r_i)$ value defined above. Further we have simplified things by considering that all clients have the same access bandwidth to the proxy. This is a realistic assumption for caches located at the bottom of a hierarchy, for example proxies that serve a set of clients within a company or campus network.

A question of interest is to find r_i^* such $E[t(r_i^*)]$ is minimal among all possible choices of r_i. This formulation is similar to the one considered in [10, 11]. The optimal value r_i^* will depend on the form of $p_r(r_i)$ and the relative values of B_{s_k} and b_c. Under scenario (A) the solution can sometimes be $r_i^* = R_i$, i.e. it is best to cache the whole image to minimize the delay.

[3] Note that while (B) is clearly more attractive, since only the additional information is downloaded, it requires that the server be able respond to these types of requests, something that is currently not supported in most practical systems. This may become possible as new formats and protocols specific to images become available (e.g. FlashPix from Kodak).

However, even if this is the case, global storage limits for the set of N images will mean that a lower resolution image may have to be stored.

Dynamic behavior We have so far assumed that there is knowledge of the bandwidths (B_{s_k} and b_c) and the $p_r(r_i)$ function. This information has to be gathered as images are accessed. It is possible for example to use methods such as those described in [14] or [15] to estimate the bandwidths. Estimating $p_r(r_i)$ is not as simple. If r_i were maintained constant it would be possible to count the instances when a reload is requested and $p_r(r_i)$ could be approximated for that particular r_i. However in an actual scenario the value of r_i would change and so would the bandwidth to the proxy. The best alternative might thus be to have a predetermined family of curves for $p_r(r_i)$ from which a specific curve can be chosen. For example curves of the form

$$p_r^m(r_i) = (1 - (r_i/R_i))^m$$

for a given parameter m can be used, with the parameter m chosen to best match the observed characteristics.

Multiple object case If storage was not an issue it would be possible to store all the images at their optimal r_i as determined by their own statistics. Obviously storage is an issue in any practical system and thus when considering the limited resources we need to select r_i for each image to ensure global optimality.

In fact, if one considers only individual images the reduction in delay achievable when caching an intermediate resolution image instead of R_i might be very small (in particular under scenario (A)) and it is tempting to conclude that soft caching has limited value. However, because soft caching allows images to be stored at lower resolution it also frees up storage for other images. Thus the performance improvement (w.r.t. hard caching) in the system we propose will come through an increase in the number of objects that can be cached for a given storage capacity.

Our goal is to find r_i for each image (note that 0 and R_i are both in $\mathcal{R}_i$) such that

$$\min_{r_i \in \mathcal{R}_i} \left(\sum_{i=1}^{N} p_i \cdot E[t(r_i)]\right) \quad \text{s.t.} \sum_{i=1}^{N} r_i \quad \leq CacheSize,$$

where $CacheSize$ is the available storage capacity. p_i is the probability of having a request for image i and can be for example approximated as $p_i = A_i / \sum_i A_i$. Note that the expected delay for each image is assumed to be independent and thus we can use well known Lagrangian techniques [16] to find the optimal solution. In this case we can minimize for each image and a given $\lambda > 0$ the cost $J_i(\lambda) = p_i \cdot E[t(r_i)] + \lambda R_i$. The overall solution can then be achieved by finding λ which provides an overall cache occupancy equal (or close) to $CacheSize$. Note that Lagrangian techniques only select points on the convex hull of the set of operating points. Thus if $E[t(r_i)]$ is minimized for a given r_i^* all the values of $r_i > r_i^*$ are automatically discarded, since they increase both the rate and the expected delay.

We are currently implementing an experimental system that will allow us to measure the reload probabilities when using a multiresolution system (this data is not available from current implementations [4]). Simulation and experimental results will be made available as they are generated and will be stored in `http://sipi.usc.edu/~ortega/SoftCaching/index.html`.

References

[1] A. Chankhunthod *et al.*, "A hierarchical internet object cache," in *USENIX Tech. Conf.*, 1996. (`http://catarina.usc.edu/danzig/cache/cache.html`).

[2] "Netscape homepage." (`http://home.netscape.com/`).

[3] "Middleware homepage." (`http://www.netcache.com/`).

[4] "Squid internet object cache." (`http://squid.nlanr.net/Squid/`).

[5] J. Gwertzman and M. Seltzer, "The case for geographical push caching," in *Proceedings of the 1995 Workshop on Hot Operating Systems.*, 1995. (`http://www.eecs.harvard.edu/~vino/web/hotos.ps`).

[6] S. Williams *et al.*, "Removal policies in network caches for world-wide web documents," in *Proc. of ACM SIGCOMM'96*, (Stanford, CA), pp. 293–305, Aug. 1996. (`http://www.cs.vt.edu/~chitra/docs/96sigcomm/`).

[7] W. Pennebaker and J. Mitchell, *JPEG Still Image Data Compression Standard.* Van Nostrand Reinhold, 1994.

[8] P. J. Burt and E. H. Adelson, "The laplacian pyramid as a compact image code," *IEEE Trans. on Commun.*, vol. 31, pp. 532–540, Apr. 1983.

[9] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. on Signal Proc.*, vol. 41, pp. 3445–3462, Dec. 1993.

[10] A. Ortega, Z. Zhang, and M. Vetterli, "A framework for optimization of a multiresolution remote image retrieval system," in *Infocom'94*, (Toronto, Canada), pp. 672–679, June 1994. (`http://sipi.usc.edu/~ortega/Infocom94.html`).

[11] A. Ortega, *Optimization Techniques for Adaptive Quantization of Image and Video under Delay Constraints.* PhD thesis, Dept. of Electrical Engineering, Columbia University, New York, NY, 1994.

[12] A. Fox and E. A. Brewer, "Reducing www latency and bandwidth requirements by real-time distillation," in *Proc. Intl. WWW Conf.*, (Paris, France), May 1996. (`http://http.cs.berkeley.edu/~fox/www96/Overview.html`).

[13] A. Fox *et al.*, "Adapting to network and client variability via on-demand dynamic distillation," in *Proc. ASPLOS-VII*, (Cambridge, MA), Oct. 1996. (`http://http.cs.berkeley.edu/~fox/papers/adaptive.ps.gz`).

[14] P. Scheuermann, J. Shim, and R. Vingralek, "A case for delay-conscious caching of web documents," in *Proc. Intl. WWW Conf.*, (Santa Clara, CA), Apr. 1997.

[15] R. P. Wooster and M. Abrams, "Proxy caching that estimates page load delays," in *Proc. Intl. WWW Conf.*, (Santa Clara, CA), Apr. 1997. (`http://www.cs.vt.edu/~chitra/docs/www6r/`).

[16] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. on Signal Proc.*, vol. 36, pp. 1445–1453, Sept. 1988.