Performance Evaluation of Transcoding-enabled Streaming Media Caching System

Bo Shen, Sung-Ju Lee, and Sujoy Basu

Hewlett-Packard Laboratories
Palo Alto, CA 94304

Abstract. We evaluate the performance of transcoding-enabled streaming media caching proxy system (TeC). TeC is designed for efficient delivery of rich media web contents to heterogeneous network environments and client capabilities. Depending on the connection speed and processing capability of an end user, the proxy transcodes the requested (and possibly cached) video into an appropriate format and delivers it to the user. We study the TeC performance by simulation using real traces derived from enterprise media server logs.

1 Introduction

With the popularity of the Internet and the Web, proxy caching systems have been widely deployed to store frequently requested files close to the end users. Proxy caching reduces the traffic between content origin and the proxies as well as the user perceived latency. Today we find more streaming media contents on the Internet. In order to satisfy various users with different network connection speeds, most streaming content providers encode the video at several different bit-rates. Availability of various bit-rate video is useful as the prosperity of wireless networks brings more heterogeneous client devices (e.g., laptops, PDAs, palm pilots, cell phones, etc.) that have different network connection, processing power, and storage. Popular streaming media files can also be stored in proxy caches for efficient distribution. Traditional caching system treats each client request equally and independently. This policy results in caching various versions with different bit-rates of the same video clip, which is a waste of storage.

We proposed to enable the caching proxy with the transcoding ability so that variants of a video object can be delivered with transcoding the cached content to the appropriate format, instead of accessing the object all the way from the content origin[1]. One of the advantages of having transcoding-enabled proxy is that the content origin servers need not generate different bit-rate videos. Moreover, heterogeneous clients with various network conditions will receive videos that are suited for their capabilities, as content adaptation can easily be done at the network edges. Given that the caching proxy is transcoding-enabled, new adaptive caching systems need to be developed for better utilization of the storage. We proposed a set of caching algorithms that consider the case where variants of the same video object exist in the system at any point of time [1]. We evaluate

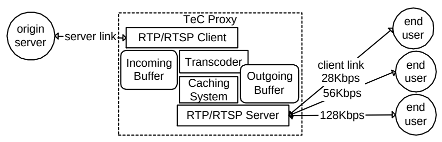


Fig. 1. System and components for transcoding-enabled caching proxy

our proposed TeC (Transcoding-enabled Caching) system using various experiments, including simulations using the proxy trace derived from a real corporate media server log. The trace reveals some degree of heterogeneous access of media objects.

The rest of the paper is organized as follows. We overview the system architecture of TeC in Section 2. Simulation results are reported in Section 3 and we conclude in Section 4.

2 Transcoding-enabled Caching

2.1 System Architecture

Transcoding-enabled caching (TeC) proxies serve various bit-rate versions of video objects to the end users with different devices or connection profiles. TeC proxy consists of the components shown in Figure 1. The proxy puts the received stream from the origin into the incoming buffer. The transcoder continuously pulls bit streams from the incoming buffer and subsequently pushes the transcoded bits out to the outgoing buffer. The proxy decides to cache the content from the incoming or the outgoing buffer while it is being produced by the transcoder. The data in the outgoing buffer is obtained either from the transcoder or from the caching system.

Given the real time transcoding capability, the TeC proxies dynamically transcode video objects to different variants to satisfy the end users in heterogeneous network environments. Each variant is a version. If version x can be obtained by transcoding from version y, we call version y a *transcodable* version for x. Conversely, version x is the *transcoded* version of y.

2.2 Caching Algorithms

We highlight the TeC caching algorithms in this section. Details of the algorithms can be found in [1]. Let us assume that the origin server has n versions at bit-rates $b_1, b_2, \ldots, b_n$ for each video object. The highest bit-rate version is b_1 and the lowest is b_n, i.e., $b_1 > b_2 > \ldots > b_n$. When version b_i is requested from the end user, and if there is version b_j ($b_j > b_i$, i.e., b_j is a transcodable version for b_i) in cache, the TeC proxy transcodes b_j to b_i instead of fetching b_i from the content origin. Therefore, it is a cache hit even though b_i is not directly available from the cache.

We proposed three different caching algorithms for TeC system. TEC_11 and TEC_12 allow at most one version of a video object to be cached at the proxy at any single time. These two algorithms operate differently when a user requests a version of the video that has lower bit-rate than the one that is already cached. For example, if the client requests version b_i of a video while b_j, where $b_i < b_j$, exists in the cache, b_i is transcoded from b_j and streamed to the user. TEC_11 refreshes the access record of b_j, but TEC_12 removes b_j and caches b_i. On the other hand, TEC_2 may cache multiple versions of the same video object. The goal is to reduce the processing load on the transcoder.

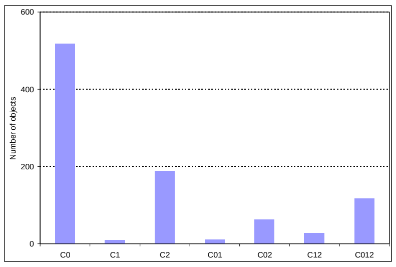
3 Simulations and Results

A stack-based implementation is developed for the performance evaluation simulation study. We conduct simulations based on an enterprise media access trace. The trace logs prolonged media content accesses in mostly a homogeneous network environment. It does however, presented some variant-based media accesses. For performance evaluation in heterogeneous network environments, please refer to [1] in which synthesized traces are used. In simulations here, the TeC proxy uses LRU as the cache replacement algorithm within the TEC caching algorithms. We compare the performance of the TeC proxy with that of a regular caching proxy. The regular caching proxy only serves as an interception proxy using the LRU algorithm without transcoding. This proxy treats each access independently even if the access is to the same media object but a different bit-rate version. We call this regular caching scheme a "reference model."

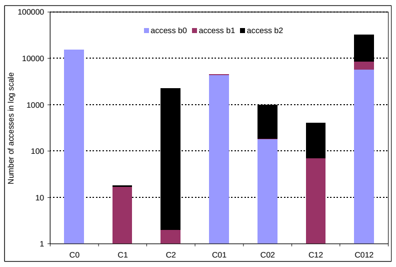
3.1 Enterprise-Trace Extraction and Analysis

The media server logs provided as input to our simulator are obtained from the servers of HP Corporate Media Solutions. We use log entries from April 1 through May 31, 2001. Since an enterprise media server stores video content at several bit-rates, we believe using the trace driven from this log can help evaluate the TeC performance. Note that the Windows Media Server is not RTP/RTSP-based as we would prefer, but the client access pattern and the video popularity statistics extracted from the server log are nevertheless useful.

To derive proxy traces from primitive server logs, we first partition the server log into separate proxy access logs of four major geographical areas based on the domain names. We then identify multiple bit-rate-variants of the same media object. They are identifiable by suffix of their URL. We find that suffixes such as *28.asf, *56.asf and *110.asf or *112.asf are used. It is verified that these objects are coded at 28Kbps, 56Kbps or above 100Kbps, respectively. We label them as low (b2), mid (b1) and high (b0) bit-rate variants, respectively. Additionally, we look further at the average bandwidth field of each entry in the server logs. We label the access as to a low, mid or high bit-rate variant based on the experienced bandwidth while delivering the object.



(a) Distribution of variant objects

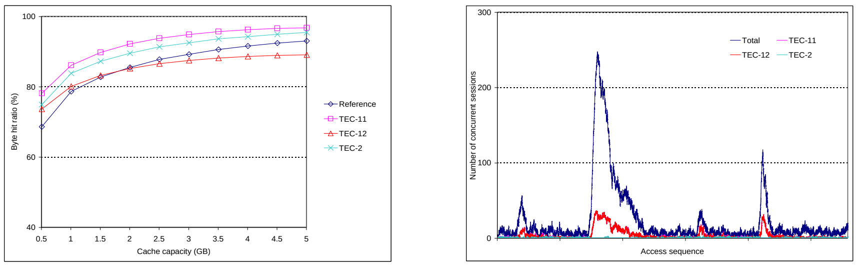


(b) Distribution of accesses

Fig. 2. Workload variance characteristics

To analyze the obtained traces, we define seven variant categories. Categories C0, C1 and C2 contain objects with one version. Categories C01, C02 and C12 contain objects with two variants. Category C012 contains objects with three variants. Figure 2 (a) shows the distribution of the number of objects in each variant category. Since the media server is targeting at corporate intranet users, the majority of the objects are coded in one variant. There are also 115 objects in category C012, and nearly 100 objects coded in various combinations of two variants (in categories C01, C02, and C12). The TeC system is most useful when there are accesses to these objects. Figure 2 (b) shows the number of accesses for each variant categories. Within each category, the number of accesses to different variants is shown. Since the TeC system improves caching performance when there are accesses to bit-rate variants, we focus on C01, C02, C12 and C012 categories. Note that the access to one variant dominates in these categories. In most cases, the highest-bandwidth variant is accessed most often. Note also that the accesses to variants are mainly to objects in C012. The access to variant b0 dominates in C012. The access to other variants is nearly 25% of the total access to objects in the category, and less than 10% of overall total access.

During the simulation, 20 GB of content resides on the origin server and there are total of 27,926 accesses. The accesses are highly temporally located. For contents generated each day, nearly 6 GB of content is accessed at least once more before the end of the measured period. Figure 3 (a) shows the byte hit ratio. The results illustrate the performance improvements gained by the TeC system. TEC_11 provides 4∼9% better byte hit ratio over the reference model. Considering that the access to variants is only 10% of the total access, the improvement indicates that 50∼90% of the variant-based access can be served from TeC directly. Since the dominant version in enterprise environment is most likely the highest bit-rate version, algorithm TEC_12 that caches lower bit-rate variants



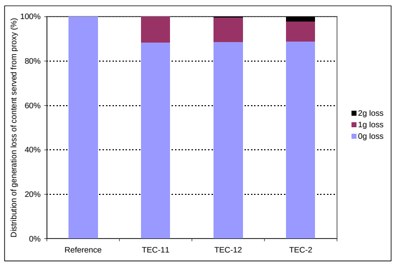
(a) Byte hit ratio (b) Transcoding load

Fig. 3. Performance Results

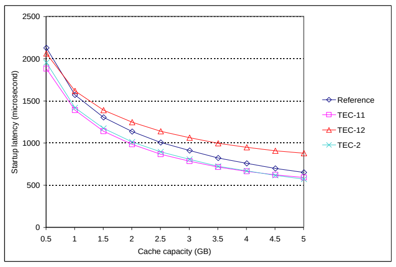
produces worse result than TEC_11. TEC_2 achieves less improvement compared with TEC_11. Nevertheless, the transcoding load is significantly reduced in TEC_2 since it caches multiple variants of the same object. This is evident from Figure 3 (b) that shows the number of conconcurrent sessions when each request arrives. For the cache size at 10% of total object size, the average concurrent transcoding session required for TEC_2 is orders of magnitude less than that for TEC_11. The peak transcoding load for TEC_11, TEC_12, and TEC_2 is 36, 36, and 3 concurrent sessions respectively, and on average 3.35, 3.25, 0.13 concurrent sessions respectively.

Bit-rate reduction transcoding often introduces quality degradation less than 1dB. If resolution reduction is considered, the degradation is negligible. Successive transcodings introduce further degradation and this is characterized as a generation loss. Figure 4 (a) shows the distribution of generation losses for the object served from TeC when the cache capacity is 1 GB. For all the TEC algorithms, only 10% of the requests are served with generation losses. There are at most one generation loss for all the requests if TEC_11 is used since it caches variant at higher bit-rate. For TEC_2, only 2% of the requests are served with two generation losses.

We now investigate the startup latency. We define startup delay as the time gap between the instance when the proxy receives a request and the instance when the first packet is served. Assuming a conservative sustained disk I/O bandwidth of 37 MB/s, the startup delay of an exact hit is 37 μsec, which is spent in retrieving of a maximum RTP payload of 1400 bytes. We measured and use the startup delay of 1500 μsec for a transcode hit. A transcode hit is where a cache has a transcodable version of the requested version. Finally, if a miss occurs, the proxy establishes a RTSP/RTP session with the origin server. Assuming an intranet backbone connection (10 Mb/s) between the proxy and



(a) Quality degradation



(b) Delay

Fig. 4. Performance Results

the origin, we set the session setup time as 8000 μsec. Using the above values, the simulator calculates the startup delay for each request. Figure 4 (b) shows the results. The startup delay decreases when cache capacity increases since more requests are served directly from the proxy. Since TEC_11 always caches the higher bit-rate variant, requests to the lower bit-rate variant of the same object are served with smaller delay as no server access is needed. However, higher bit-rate variant occupies larger space, which reduces the number of exact hits that require the smallest delay. On the other hand, as the startup delay resulted from a transcode hit is much smaller than the delay from a miss, the transcoding-enabled schemes are still beneficial in terms of latency . TEC_11 and TEC_2 generally produces 200 μsec less in startup delay than the reference model. TEC_12 shows the worst delay as most of the user requests is to higher bit-rate variants.

4 Conclusion

We evaluated the TeC performance using the trace derived from an enterprise network. Simulation results showed that TeC system provides better byte hit ratio and less startup latency than traditional caching systems. We also showed that transcoding introduces negligible quality degradation, and requires only small additional processing load on the proxies.

References

1. B. Shen and S.-J. Lee, "Transcoding-Enabled Caching Proxy System for Video Delivery in Heterogeneous Network Environments," Proceedings of IASTED IMSA 2002, Kauai, HI, August 2002.