Balancing MPEG Transcoding with Storage in Multiple-Quality Video-on-Demand Services

Minseok Song, Jeong Seop Sim, Jaedoo Go, Bumsun Lee, and Soo Jun Park

ABSTRACT—To match the requirements of heterogeneous mobile devices, video objects may be transcoded, which requires considerable CPU resources. Alternatively, multiple versions of the same video may be stored on servers, but this requires a lot of disk space. We formulate the trade-off between the versions that are stored on disk and the need for transcoding. We propose an optimal solution to this formulation based on dynamic programming. Experiments show that our scheme allows up to 68% more clients to be admitted than conventional schemes when a reasonable amount of storage is available.

Keywords—MPEG transcoding, video streaming, storage management, streaming in wireless network.

I. Introduction

It has recently become possible to access video services over the Internet at any time using devices such as personal digital assistants (PDA), personal multimedia players (PMP), and so on [1]-[3]. These devices have different processing capabilities, energy budgets, display sizes, and network connectivities. To support such heterogeneity, video content needs to be converted to suitable forms by the conversion process of transcoding.

Conventional transcoding can be dynamic or static [2]-[4]. In the dynamic scheme, only the highest-quality version is stored, and lower-quality versions must be extracted online. This approach can quickly exhaust the CPU resource due to the computational requirements of transcoding. In the static scheme, the server creates multiple versions at different quality levels and stores them offline, which requires a lot of disk space. In some works, transcoding costs are considered to determine caching location or cache replacement [5], [6].

We propose a generalized analytical model to select the versions of a video to be stored on disk, with the aim of minimizing the CPU demands made by transcoding. We then propose an optimal algorithm to select versions from those stored on disk.

II. System Model

A server is able to transcode an original video object into different variants, each of which is called a transcoded version. In video transcoding, a higher bit-rate version can be transcoded to a lower bit-rate version, but increases in bit rate are not supported [2].

Table 1 summarizes the important symbols in the generalized model of our system. Let us assume that each video V_i has NR versions: the original version V_i^1 and transcoded versions V_i^2 to V_i^{NR} $(i=1,\cdots,NV)$. The highest bit-rate version is V_i^1, and the lowest is V_i^{NR}. We will assume that the access probability of every video is known in advance. Let p_i^k be the access probability of version V_i^k $(i=1,\cdots,NV)$, where $\sum_{i=1}^{NV}\sum_{k=1}^{NR} p_i^k = 1$. Upon receiving a client's request, the server searches its disk to find an appropriate version. If the requested version is stored on disk, then it is sent to the client directly; otherwise, transcoding is needed. If sufficient CPU resource is available, then the server starts transcoding and sends the resulting stream to the client; otherwise, the request is blocked. Therefore, our idea can be easily applied to current transcoding servers because transcoding is only needed for versions that are not stored on disk.

Manuscript received Dec. 12, 2008; revised Feb. 16, 2009; accepted Feb. 25, 2009.
This work was supported by Inha University Research Grant.

Minseok Song (phone: + 82 32 860 7441, email: mssong@inha.ac.kr), Jeong Seop Sim (email: jssim@inha.ac.kr), Jaedoo Go (email: goldedit@gmail.com), and Bumsun Lee (email: mrbumm@gmail.com) are with the School of Computer and Information Engineering, Inha University, Incheon, Rep. of Korea.
Soo Jun Park (phone: + 82 42 860 6899, email: psj@etri.re.kr) is with the IT Convergence Technology Research Laboratory, ETRI, Daejeon, Rep. of Korea.

Table 1. Notations used to describe the system model.

Symbols	Meaning
NR	Number of versions of each video V_i, (i=1,⋯, NV)
p_i^k	Access probability of version V_i^k
FS_i	Set of all feasible sets of stored versions of video V_i
$FS_{i,j}$	The j-th element of set FS_i
$C_{i,j}^k$	CPU utilization needed to make version V_i^k when versions are stored as those in set $FS_{i,j}$
$CU_{i,j}$	$\sum_{k=1}^{NR}(p_i^k \times C_{i,j}^k)$
TS	Total amount of storage available
$SV_{i,j}$	$SV_{i,j} = CU_{i,1} - CU_{i,j}$
$STR_{i,j}$	Total storage requirement for the versions in set $FS_{i,j}$

III. Optimal Storage Management

A transcoding server can trade storage space against the reduced CPU usage. Our analytical model expresses this trade-off. Let FS_i be the set of all feasible sets of stored versions of video V_i. Note that the original versions of all videos must be stored on disk. Therefore, if the number of versions is NR, then there are 2^{NR-1} possible sets for FS_i. Let NE be the number of elements in set FS_i, so that $NE=2^{NR-1}$. Typically, mobile devices support several fixed resolutions, so NE is not very high.

We will assume that the elements of FS_i, that is, $FS_{i,j}$ $(j = 1,\cdots,NE)$, are sorted in ascending order of storage requirement. For example, if NR=3, then $FS_i = \{FS_{i,1}=\{V_i^1\}$, $FS_{i,2}=\{V_i^1,V_i^3\}$, $FS_{i,3} = \{V_i^1,V_i^2\}$, and $FS_{i,4}=\{V_i^1,V_i^2,V_i^3\}\}$. For transcoding, the server creates a task which consumes CPU cycles periodically. The CPU utilization associated with each task is the proportion of the CPU time allocated to the task in each period. Let $C_{i,j}^k$ be the CPU utilization needed to make a transcoded version V_i^k, when versions are stored as those in set $FS_{i,j}$. If version V_i^k is stored on disk, then $C_{i,j}^k = 0$; otherwise, the server selects the higher bit-rate version in $FS_{i,j}$ that requires the minimum CPU utilization for transcoding.

Let TS be the total amount of storage space available. Let $STR_{i,j}$ denote the total storage requirement for the versions in set $FS_{i,j}$. Let $CU_{i,j}$ denote the average CPU utilization required for storing the versions in set $FS_{i,j}$, which can be expressed as $\sum_{k=1}^{NR}(p_i^k \times C_{i,j}^k)$. Let $SV_{i,j}$ denote the saving in CPU utilization for storing the versions in set $FS_{i,j}$, instead of storing those in the original set $FS_{i,1}$, so that $SV_{i,j} = CU_{i,1} - CU_{i,j}$. Consequently, $SV_{i,1} = 0$.

Our goal is to maximize the average saving in CPU utilization per video request under the storage constraints. Let SE_i be a selection parameter which indicates that the SE_i-th element of FS_i, that is, FS_{i,SE_i}, is selected as the set of versions of video V_i. We will now define this problem formally.

Definition 1. Version Selection Problem

Find SE_i (SE_i=1,⋯, NE) for every set FS_i (i=1,⋯, NV), which maximizes $\sum_{i=1}^{NV} SV_{i,SE_i}$ such that $\sum_{i=1}^{NV} STR_{i,SE_i} \le TS$.

We propose an optimal solution to the version selection problem (VSP) called the version selection algorithm (VSA), which uses the dynamic programming technique as shown in Fig. 1. Note that the VSP is a variant of the multiple-choice knapsack problem which can be solved in pseudo-polynomial time [7].

Let $OPT_{v,w}(1 \le v \le NV, 1 \le w \le TS)$ be the maximum CPU saving when the disk storage allowed for $V_1,\cdots,V_v$ is w. We first set $OPT_{v,w} = 0$, $(0 \le v \le NV, 0 \le w \le TS)$ for initialization. If v>0, $ORT_{v,w}$ can be calculated using the following recurrence:

$$OPT_{v,w} = \max_{1 \le j \le NE}\{\max\{OPT_{v-1,w}, OPT_{v-1,w-STR_{v,j}} + SV_{v,j}\}\}.$$

The value of $OPT_{NV,TS}$ is the maximum saving in CPU utilization when the available disk storage is TS. Then, SE_i can be calculated as follows. At each iteration, VSA stores the index of the element that corresponds to the largest saving in CPU utilization (line 13). Then, it finds SE_i by tracing back from $OPT_{NV,TS}$ (lines 19-24). It is easy to see that VSA runs in $O(NV \times TS \times NE)$ time. The step size used in the second loop (line 9) is 1 MB. If the step size increases, then the execution time of VSA decreases.

```
1: Temporary variables: $w$, $v$, $j$;
2: for $w$=0 to TS do
3:   for $v$=0 to NV do
4:      $OPT_{v,w} \leftarrow 0$;
5:      $OPT\_SET_{v,w} \leftarrow 1$;
6:   end for
7: end for
8: for $v$=1 to $NV$ do
9:   for $w=TS$ downto $STR_{v,1}$ do
10:     $OPT_{v,w} \leftarrow OPT_{v-1,w}$;
11:     for $j$=1 to $NE$ do
12:        if $OPT_{v-1,w-STR_{v,j}}+SV_{v,j} > OPT_{v-1,w}$ then
13:           $OPT_{v,w} \leftarrow OPT_{v-1,w-STR_{v,j}}+SV_{v,j}$;
14:           $OPT\_SET_{v,w} \leftarrow j$;
15:        end if
16:     end for
17:   end for
18: end for
19: $w \leftarrow TS$, $v \leftarrow NV$;
20: while $v$>0 do
21:   $SE_v \leftarrow OPT\_SET_{v,w}$;
22:   $w \leftarrow w-STR_{v,SEv}$;
23:   $v \leftarrow v-1$;
24: end while
```

Fig. 1. Version selection algorithm (VSA).

IV. Experimental Results and Conclusion

To evaluate the effectiveness of our scheme, we performed several simulations. We measured the CPU utilization needed to transcode five MPEG sample videos to versions with resolutions typically adopted by mobile devices, such as QVGA, CIF, and so on. We then simulated a system offering 1,000 videos by storing the original version of each sample video 200 times. The minimum storage for these 1,000 videos is 428 GB, and *NR* is taken to be 5. The arrival of client requests follows a Poisson distribution with a mean inter-arrival time of 3 seconds. The access probability follows a Zipf distribution in which $\alpha = 0.271$.

To evaluate the VSA scheme, we compare it with four other methods: OBS, PBS, VBS, and RBS. The OBS method only stores the original versions of the videos; the PBS method stores every transcoded version of the most popular video, the next most popular, and so on; the VBS method stores the highest bit-rate transcoded version of the most popular video and then that of the next most popular, and so on; while the RBS method chooses a version of a video at random and stores it on disk.

Figure 2 shows how the admission ratio depends on the total amount of storage space, *TS*, when the access probability is the same for all versions of a video. The VSA scheme admits between 26% and 50% more clients than OBS, between 5% and 6% more than PBS, between 14% and 20% more than VBS, and between 12% and 14% more than RBS.

Figure 3 shows how the distribution of requests for different versions affects the admission ratio when *TS*=800 GB. For this purpose, we define the following three situations:

- HRM: the highest-resolution versions are most popular ($\forall i$, $p_i^1 = p_i \times 0.6, p_i^2 = p_i^3 = p_i^4 = p_i^5 = p_i \times 0.1$).
- MRM: the medium-resolution versions are most popular ($\forall i$, $p_i^3 = p_i \times 0.6, p_i^1 = p_i^2 = p_i^4 = p_i^5 = p_i \times 0.1$).
- LRM: the lowest-resolution versions are most popular ($\forall i$, $p_i^5 = p_i \times 0.6, p_i^1 = p_i^2 = p_i^3 = p_i^4 = p_i \times 0.1$).

The VSA scheme admits between 22% and 68% more clients than OBS, between 3% and 26% more than PBS, between 10% and 51% more than VBS, and between 6% and 34% more than RBS. Note that all the schemes except VSA perform worst in the MRM situation. This is because it takes more CPU time for extraction of the medium-resolution versions, which is considered only by the VSA scheme.

We have developed a generalized analytical model to select the versions of video objects to be stored on disk and proposed an optimal solution to this formulation. Experimental results show that our scheme balancing MPEG transcoding with storage based on dynamic programming substantially reduces CPU demands and enables a server to admit many more clients.

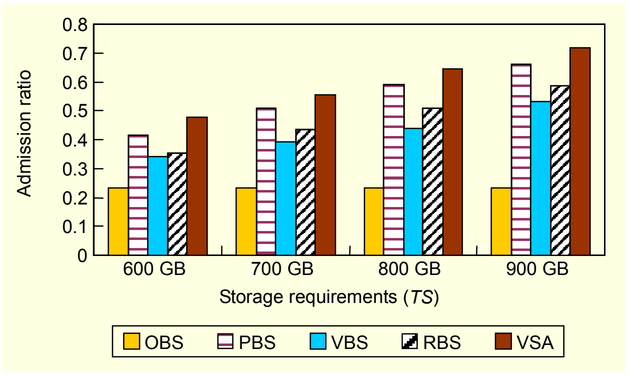

Fig. 2. Admission ratio against *TS*.

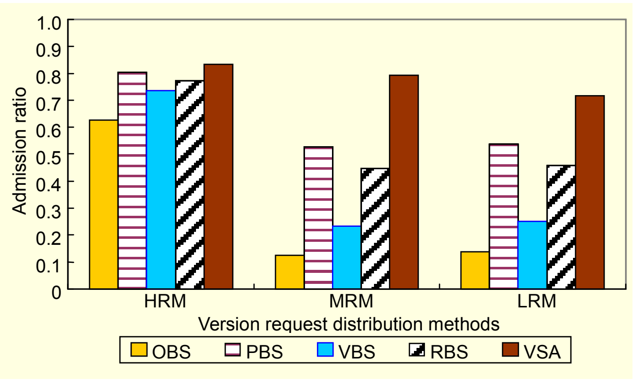

Fig. 3. Admission ratio against version distribution methods.

References

[1] R. Mohan, J. Smith, and C. Li, "Adapting Multimedia Internet Content for Universal Access," *IEEE Trans. Multimedia*, vol. 1, no. 1, Mar. 1999, pp. 104-114.

[2] B. Shen, S. Lee, and B. Basu, "Caching Strategies in Transcoding-Enabled Proxy Systems for Streaming Media Distribution Networks," *IEEE Trans. Multimedia*, vol. 6, no. 2, Apr. 2004, pp. 375-386.

[3] I. Shin and K. Koh, "Hybrid Transcoding for QoS Adaptive Video-on-Demand Services," *IEEE Trans. Consumer Electronics*, vol. 50, no. 2, May 2004, pp. 732-736.

[4] X. Tang, F. Zhang, and S. Chanson, "Streaming Media Caching Algorithms for Transcoding Proxies," *Proc. the ICPP*, Aug. 2002, pp. 287-295.

[5] H. Hung and M. Chen, "On Designing a Shortest-Path-Based Cache Replacement in a Transcoding Proxy," *ACM Multimedia Systems Journal*, vol. 15, no. 2, Apr. 2009, pp. 49-62.

[6] W. Qu et al., "An Optimal Solution for Caching Multimedia Objects in Transcoding Proxies," *Computer Communications,* vol. 30, no. 8, June 2007, pp. 1802-1810.

[7] S. Martello and P. Toth, *Knapsack Problem: Algorithms and Computer Implementations,* John Wiley & Sons, 1990.

ETRI Journal, Volume 31, Number 3, June 2009 Minseok Song et al. 335