Analyzing Compute vs. Storage Tradeoff for Video-aware Storage Efficiency

Atish Kathpal, Mandar Kulkarni, Ajay Bakre
NetApp Inc.

Abstract

Video content is quite unique from its storage footprint perspective. In a video distribution environment, a master video file needs to be transcoded into different resolutions, bitrates, codecs and containers to enable distribution to a wide variety of devices and media players over different kinds of networks. Our experiments show that when 8 master videos are transcoded into most popular 376 formats (derived from 8 resolutions and 6 containers), transcoded versions occupy 8 times more storage than the master video. One major challenge with efficiently storing such content is that traditional de-duplication algorithms cannot detect significant duplication between any 2 versions. Transcoding on-the-fly is a technique in which a distribution copy is created only when requested by a user. This technique saves storage but at the expense of extra compute cost and latency resulting from transcoding after a user request is received. In this paper we develop cost metrics that allow us to compare storage vs. compute costs and suggest when a transcoding on-the-fly solution can be cost effective. We also analyze how such a solution can be deployed in a practical storage system using access pattern information or a variant of ski-rent [1] online algorithm when such information is not available.

1. Introduction

The way people view video content is changing, Television is no longer the only way people can view videos. Media players running on Smartphones, PCs, Laptops, Tablets and similar devices use a variety of resolutions, bitrates, encodings and containers for playing videos. Popular TV programs and movies are also offered by content providers and content distributors (e.g. Netflix) for on-demand viewing. Modern media players and content delivery systems have the ability to change the bitrate of a video stream to adapt to the bandwidth available in the network between the user's device and the media server.

Content delivery systems typically store popular videos in all possible resolutions and containers that supported user devices might use. These versions are transcoded from video masters obtained from content providers (e.g. movie or television studios). Traditional de-duplication algorithms operating on binary data in video files cannot achieve any savings in a video distribution environment because transcoded versions of a video rarely have common data blocks. A content management system can potentially eliminate some video versions from the content repository, which can be re-created using on-the-fly transcoding if accessed by a user subsequently. This approach attempts to achieve storage efficiency at the expense of compute resources required for transcoding and potential delay incurred in satisfying a user request while the requested copy is re-created.

In this paper, we explore the tradeoff between storage savings achieved by such an approach. We do not consider latency yet, leaving it for future investigation, assuming for now that a user is willing to wait until first transcoded bytes are made available to the user's media player. We experimented with real videos available in a test suite and obtained transcode times for different resolutions. We also measured the size of video versions thus generated. We chose a utility cost model for comparing transcode cost with storage savings in this analysis based on employing an Amazon EC2 instance for compute and Amazon S3 service for storing data for a certain period. The analytical model however is generic enough to be used with any other cost metrics for compute and storage. We propose a comparison metric (known as Elimination Metric) that can be used by a storage system to identify which versions are good candidates for elimination and subsequent re-creation on demand.

The remainder of this paper is organized as follows. In section 2, we develop an analytical model for comparing storage and compute costs in a storage system that has the ability of eliminating certain versions of videos and recreating them from master video if requested by a user. In section 3, we present experimental results obtained from transcoding a variety of videos and analyze them using the model developed earlier. Section 4 discusses strategies for eliminating videos based on their Elimination Metric (EM) and any available access patterns. Section 5 discusses related work and Section 6 lists our conclusions and some future work we plan to do in this area.

2 Cost Analysis and Elimination Metric

While we mentioned that there is a cycle of elimination and re-creation, the decision to eliminate the video comes at a cost of re-creating it. We would like to discuss the tradeoffs while deciding to eliminate the video. Later in the section we perform a cost analysis and de-

fine a metric called Elimination Metric, to aid in our decision of eliminating a video.

2.1 Storage, Compute and Latency Trade-offs

We identify three major factors that one needs to consider when making a decision to eliminate a given video viz. storage savings, compute overheads and latency. As we attain more storage savings, we increase the compute overheads, incurred while recreating the eliminated videos. Additionally the user accessing the eliminated video has to wait till the video is re-created, this adds to user perceived latency. It would be interesting to find a sweet spot, such that our overall cost reduces from eliminating the video. In this work, we develop a way to optimize the cost based on storage and compute tradeoffs and do not consider the latency, which is left for future work

2.2 Storage and compute cost analysis

Naturally, it would be cost effective to eliminate the video only when it's cheaper to transcode the video rather than to store it for a given period of time.

Let us assume that at any instance, we can predict, the time interval between consecutive accesses of a given video. Let us call this time T_P *(hours)*. Hence it would be cost effective to eliminate the video when:

Cost of transcoding video < Cost of storing the video for T_P hours (1)

Next, we quantify the transcode and storage costs to be able to compare them. We define them respectively as:

C_c = Cost of compute per hour

C_N = Cost of data transfer per GB

C_s = Cost of storage per GB per hour

Assume,

Time taken to transcode a video V from its master M = τ *hours*

Size of video V = ξ *GB*

Size of master M = ζ *GB*

Transcoding V from M involves transferring M from storage node, transcoding it to obtain V, and then placing V into persistent storage.

Thus, net cost (C_T) of transcoding video V from M is given by:

$$C_T = C_c \times \tau + C_N \times (\zeta + \xi)$$

Thus, by substituting values in equation (1) we have,

$$C_T < C_s \times \xi \times T_P \quad (2)$$

$$\Rightarrow C_c \times \tau + C_N \times (\zeta + \xi) < C_s \times \xi \times T_P$$

$$\Rightarrow [C_c \times \tau + C_N \times (\zeta + \xi)] / (C_s \times \xi) < T_P$$

We define the LHS of the above equation as the *Elimination Metric (EM)* for a given video V. Thus,

$$\textbf{\textit{Elimination Metric}} = [C_c \times \tau + C_N \times (\zeta + \xi)] / (C_s \times \xi) \quad (3)$$

Elimination Metric implies that the cost of transcoding the video from its master is equal to the cost for storing the video for EM hours. Thus, higher the value of EM, higher is the cost of transcoding the video, as against storing the video, per unit time. Lower EM values would suggest that the video is heavier on storage resources as compared to compute. Hence, videos with lower EM values would be easier targets for elimination as compared to those with higher values, assuming popularity to be same across the videos.

It follows from *equations 1* and *3* that, if we can predict that the video is not expected to be accessed for next T_P hours, and T_P > EM, then it is cost effective to eliminate the video rather than to store it for coming T_P hours. Thus, given the popularity (access history in terms of T_P) of the video, we can make a cost effective decision to eliminate the video, based on its EM value.

To get an estimate of the transcoding and storage costs, we used the prices for elastic cloud compute EC2 [4] and simple cloud storage S3 [5] at Amazon's AWS. We chose per hour cost of on-demand EC2 High Compute Medium Instance and the GB per hour cost for storing first 1TB of data on Amazon S3 as C_c and C_s respectively. We chose the High Compute Medium Instance as it was the closest to the configuration of our experimental server as explained in section 3.1. On using these numbers and ignoring the network transfer cost C_N, we find that:

$$C_T = C_c = \$0.165 \text{ per hour}$$

$$C_s = \$1.73 \times 10^{-4} \text{ per GB per hour}$$

$$\Rightarrow C_c = \sim 1000 \times C_s \quad (4)$$

Thus, by substituting values in equation (3) we obtain EM for our AWS based cost model as,

$$\textbf{\textit{Elimination Metric}}\ (EM) = (\tau \times 1000)/\xi \quad (5)$$

For our discussion in section 3, we stick to using the EM values as defined in (5), based on Amazon's AWS.

3. Experiments and results

3.1 Experimental Setup

We used an Ubuntu 10.04 compute server with Intel Xeon 3.00 GHz dual core processor with 4GB of RAM as our transcoding machine. We used FFmpeg as the transcoding application installed on our compute server. For storage, we used a NetApp E-series storage controller with a 280GB SSD drive, attached to the compute server with 8Gbps fiber channel.

3.2 Dataset

We used a collection of 391 videos as our dataset. The dataset was populated using eight 1920x1080p 30 second video clips encoded with libx264 codec as masters. The masters were obtained from Testvid [10] and were carefully chosen to cover various characteristics like zoom, slow/fast motion, limited/vivid colors etc.

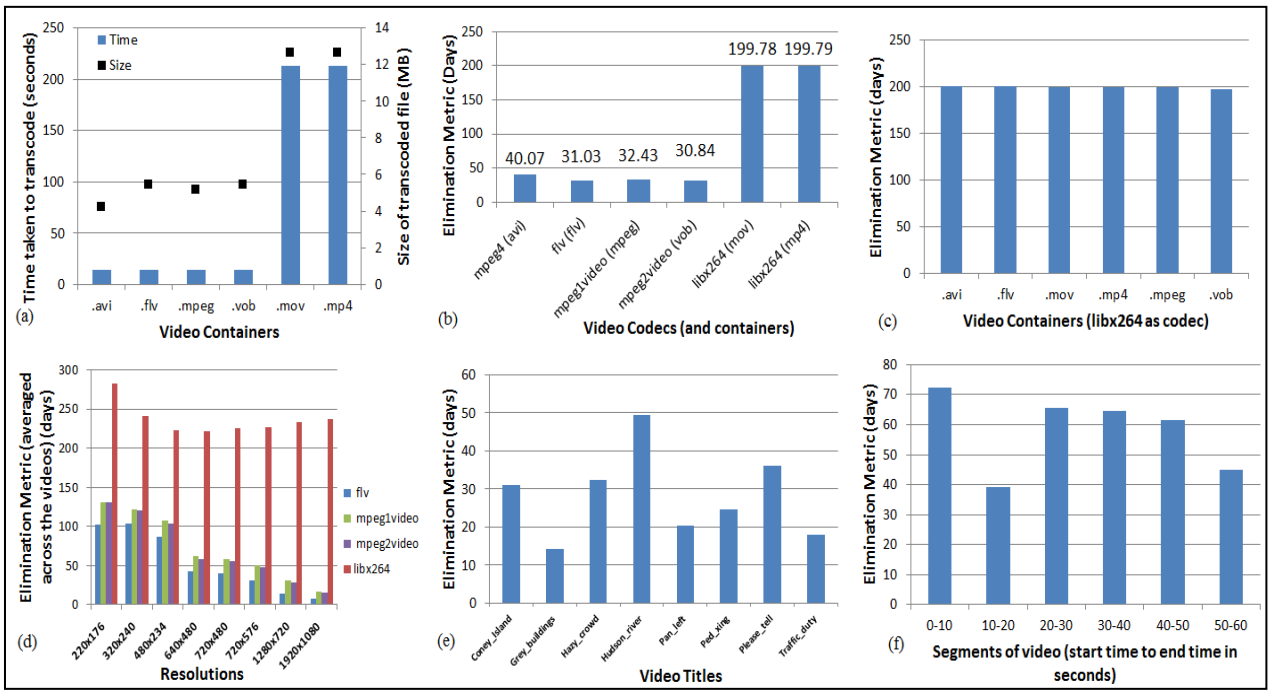

Fig 1: Results

Each of the master videos were transcoded using 8 popular resolutions and 6 popular video containers, resulting in 376 derived versions. To study relationship between containers and EM values we transcoded one master video (using libx264 encoding) to 6 different video containers. Besides, we used one more 60 second video clip from Testvid, to study dependence of EM on duration and content of a video (section 3.3.1).

3.3 Results and Observations

The transcoding jobs, run to obtain the derived versions, were found to saturate the CPU resources, with iostat [8] reporting 95% CPU utilization on an average. The memory utilization was 10% of the 4GB RAM, on an average, suggesting CPU to be the bottleneck during transcoding. The masters occupied 211 MB of space and the derived versions took up 1.77 GB space. For each of the derived versions, we obtained their corresponding

EM values, using the Amazon AWS cost model as described by *equation 5*, in section 2.2.

3.3.1 Comparison with block-based de-duplication

We performed NetApp's fixed-size (4KB) block based de-duplication, called ASIS [9], on our data set of 391 videos and found 0% space savings. Space savings using variable length block de-duplication algorithms are also reported to be low [7] for video datasets, in general.

Using transcoding based elimination of video versions that we propose; one could attain up to 95.72% space savings for our dataset (which is an upper bound, achievable on elimination of all derived videos).

3.3.2 Compute and Storage Trade-off Plots

We were interested in studying the storage and compute tradeoffs across various formats of the derived videos. For this we obtained plots of EM values for the derived videos in our dataset. We present how EM values are affected by various transcoding parameters like resolution, codec and container.

First, we show how EM plots are a reflection of the tradeoffs between time taken to transcode and the storage required to store the transcoded video. Plot (Fig 1a) compares time taken to transcode a video and size of video for various video containers (using the default associated encoding algorithms). For all video containers, the size of the transcoded file increases with transcode time. However the rate is not proportional. This can be seen from Fig 1(b), for libx264 codec, rate at which cost of compute increases w.r.t. storage cost is more than that for other codecs. Fig 1(c) shows that EM values remain same across all containers when we use the same codec (here libx264). Thus, EM values vary w.r.t. encoding algorithms and not so much w.r.t. the video containers.

Next we wanted to study the effect of resolution of the output video on its EM values. Fig 1(d) shows that for majority of codecs we used (flv, mpeg1 and mpeg2) the EM values decrease as we move from lower to higher resolutions.

Another trend to observe is how EM values change as content of the video changes (keeping rest of the parameters constant). Fig 1(e) shows how EM varies

with content across the 8 different derived videos and Fig 1(f) shows how EM values vary with change of content within the same 60s video. Thus, EM values are dependent on the content of video (Fig1e) and are independent of the duration of the video (Fig1f).

Through these results we conclude that EM values are tied with the underlying codec used for transcoding the video, its resolution and its content. EM values are independent of the duration of the video, hence the pattern of results obtained for 30s clips in our experiments, applies to videos of longer and shorter durations as well. Based on the above observations, one could map the incoming video to an expected range of EM values, depending on its codec, resolution and content. Based on our preliminary results we plan to explore such ways to estimate EM values without having to transcode the video, as part of future work.

4. Elimination Strategies

As we saw in section 2, by using EM and access pattern information, we can design an optimal scheme on whether to eliminate a video or keep it.

	Eliminate	**Cost at Access**
$T_a < EM$	*No*	$T_a \times \xi \times C_s$
$T_a = EM$	*No/Yes*	C_T *or* $(C_s \times \xi) \times T_a$
$T_a > EM$	*Yes*	C_T

Table 1 Decision policies when T_a and EM are known.

Table 1 demonstrates that if we know the actual time before next access of the video, T_a and EM, we can create an optimal decision cost matrix so that total cost of the system is minimized. Here cost to keep a video for EM hrs is $C_s \times \xi \times T_a$ and cost of recreating the video is C_T, as defined in section 2.2. If the video is accessed when $EM = T_a$, the cost of accessing the video, would be same whether we had chosen to eliminate it or to keep it for EM hours.

In practice it's unrealistic to assume that we would have the actual access pattern (T_a) information. At best, we may try to predict its value (T_P). We try to borrow some concepts from online algorithms especially the ski-rent sub-class of online algorithms explained in [1] to address the problem.

The ski-rental problem is similar to our elimination problem. Here we want to decide whether to eliminate a copy or re-create it on the fly based on the future access pattern which is unknown. Since the online algorithm gives a deterministic bound for an optimal solution without knowing the future access pattern we think it would be suitable in our case.

We propose a new scheme, where we choose to eliminate a video, only when it has not been accessed for the last EM hours. Let's see how the cost metrics change in this scenario.

	Eliminate	**Cost at access**
$T < EM$	*No*	$T_a \times \xi \times C_s$
$T = EM$	*No*	C_T
$T > EM$	*Yes*	$T_a \times \xi \times C_s + C_T = 2\ C_T$

Table 2 Decision policies without knowledge of T_a

Here we can see that if we make the above stated choice, we could at most pay twice the optimal cost for all the videos whose time since last access T > EM. This is because we have spent C_T amount while storing the video for EM days and when the access of the video comes after EM days we would have to re-create the video and pay additional C_T amount for the compute.

The important thing to note is, for accesses which fall between 0-EM ranges, "online" scheme is equivalent to the optimal scheme when access patterns are known.

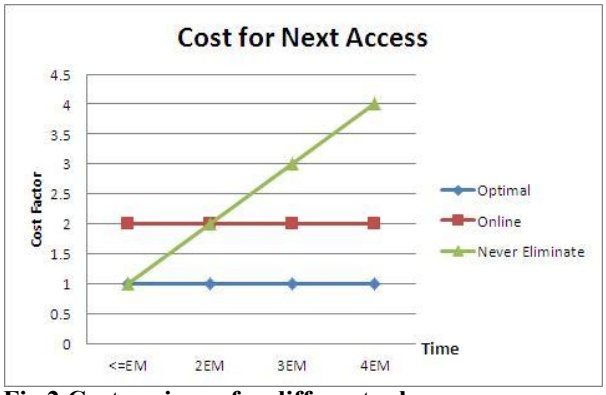

Fig 2 Cost variance for different schemes.

In Fig 2 we see that if you chose "online" scheme as discussed in table 2, it would cost 2 times optimal cost for the next access. We also note that, if we choose the traditional "never eliminate" stored video strategy, and if access lies in EM to 2EM time, "never eliminate" gives a better result than "online", although the cost for subsequent accesses scale linearly for "never eliminate" strategy. We also show that after 2EM time, "online" strategy is always better than "never eliminate". Here we don't consider the "never store" strategy as the cost for each access would incur C_T and overall access costs for the video would be factor of number of such accesses. Such a strategy would be a beneficial generic scheme for un-popular videos but a very poor strategy for popular videos.

5. Related work

There has been a lot of work around content aware algorithms and data shrinking algorithms from Ocarina [12]. They try to look inside the content and use various mathematical models to achieve optimizations. Video aware de-duplication was also discussed by Katiyar et.al [7]. They try to sample video files in a content store and create unique clusters from them by matching the signatures of each video. They propose trading storage for compute from a storage savings aspect, without discussing the cost effectiveness of the same. QuFiles [11] try to address the problem of serving the right file at the right time by providing a mechanism to dynamically resolve logical data items to correct data.

Trading storage with computation was previously discussed in the context of cloud computing [2]. There also has been interesting research around access patterns for videos [6] where they were able to classify 50% of YouTube videos as "rarely accessed" (accessed less than 180 days a year). There is also work on caching of videos based on popularity and access patterns [3].

What we are proposing is different and novel as we don't look inside the contents of a video file like Ocarina or ViDeDup. We rely on metadata and master copy relationship between the videos, which is presumably known to the service provider or content distributer. We create a cost model appropriate to system we are using and compute the EM value for the video. We then use the algorithms mentioned above to choose optimal candidates to eliminate either based on access patterns or heuristics mentioned. We potentially could use QuFiles like approach to leverage the metadata information.

6. Conclusion and Future work

Trading compute with storage is somewhat counter intuitive. What we have demonstrated from our experiments is that in certain datasets (esp. video); where traditional de-duplication fails to yield significant space savings, trading storage with compute can be cost effective. We proposed a metric called Elimination Metric (EM) that can be used to decide which video versions in a repository are good candidates for elimination and subsequent recreation on access. We also show that this decision can be taken even without prior knowledge of access patterns.

Moreover, we can conclude based on the experiments that some parameters of a transcoded video (e.g. codec) have a larger effect on EM while others (e.g. container) don't affect the EM substantially. This knowledge can be used to improve the generic elimination algorithm and its performance.

Latency: One of the crucial aspects we have not considered in this analysis is user perceived latency. When a user accesses a video that was eliminated, what is the time till the first byte / frame of the video is served to the media player. There are many ways one can resolve this problem. We could chunk the video into segments and keep enough number of segments so that user sees no latency till the remaining segments are re-created. There could be threshold, for paid customers, which gives a tighter bound on how much time a user is willing to wait. You can model the number of chunks to be stored based on such bound. If we decide to chunk the videos the EM values now would be per-chunk and could potentially add some complexity. One intuitive thing to note is for free customers such a delay due to latency is tolerated.

Different Cost models: It would be interesting to address challenges posed by different enterprise set-ups. Adding storage or compute capacity may result in additional capital outlay and a significant lead time. In a datacenter we will have limited storage and compute resources. So we need to address the question of how to model limited resources. One way to model it could be using number of compute units required to re-create the video. We just need to ensure that at given time sum of all such compute units of eliminated videos is less than total available compute resources. Another important question is how to determine cost for compute and storage in Datacenter environment as they are dependent on Opex and Capex of the resources.

References:

[1] S. Albers. *Online algorithms. In Interactive Computation:* The New Paradigm edited by D.Q. Goldin, S.A. Smolka and P. Wegner, 143-164, 2006

[2] Adams et al. "*Maximizing efficiency by trading storage for computation*," HotCloud'09

[3] Zhou et al. *A Video Replacement Policy based on Revenue to Cost Ratio in a Multicast TV-Anytime System.* Parallel and Distributed Processing Symposium 2011

[4] "Amazon Elastic Compute Cloud (Amazon EC2)"

[5] "Amazon Simple Storage Service (Amazon S3)"

[6] Gürsun et al. "*Describing and forecasting video access patterns*". Infocom 2011

[7] Katiyar et al. *ViDeDup: An Application Framework for Video De-duplication*. Usenix HotStorage'11

[8] Iostat: http://linuxcommand.org/man_pages/iostat1.html

[9] http://partners.netapp.com/go/techontap/top10_a-sis.html

[10] Testvid: www.Testvid.com

[11] Veeraraghavan et al. "*quFiles: The Right File at the Right Time*". ACM Transactions on Storage.

[12] Content-Aware Storage Optimization http://www.ocarinanetworks.com/resources/147