Storage Efficiency Opportunities and Analysis for Video Repositories

Suganthi Dewakar *NetApp, Inc.*
Sethuraman Subbiah *NetApp, Inc.*
Gokul Soundararajan *NetApp, Inc.*
Mike Wilson *NetApp, Inc.*
Mark W. Storer[1] *Box Inc.*
Kishore Udayashankar[1] *Exablox*
Kaladhar Voruganti[1] *Equinix*
Minglong Shao[1]

Abstract

Conventional wisdom states that deduplication techniques do not yield good storage efficiency savings for video data. Indeed, even if the video content is the same, the video files differ from each other due to differences in closed-captioning, text overlay, language, and video resolution. Therefore deduplication techniques have not yielded good storage efficiency savings. In this paper, we look at the effectiveness of four different deduplication algorithms on different scenarios. We evaluate fixed-sized, variable-sized, and two content-aware deduplication techniques. Our study shows that content-aware and variable-sized deduplication techniques do provide significant storage efficiency savings.

1 Introduction

With the proliferation of smart devices and surveillance gadgets, the creation of video data is growing exponentially. This in turn is increasing the presence of video data in public clouds and repositories such as YouTube, Dropbox, and so on. Furthermore, most of the professionally created digital content like movies, television shows, and documentaries is stored in video repositories and is accessed using online streaming video providers like Netflix, Hulu, and others. In most of these video repositories, multiple copies of the same video exist. However, traditional fixed-sized deduplication techniques fail to identify duplicate data because there are differences in the video copies due to: 1) the presence of additional audio data, 2) the presence of audio data in different languages, 3) the presence of textual captions, 4) copies of same video have been recorded by different DVRs with different commercial/advertisement content, 5) use of different transcoding, and 6) difference in video frame rate/resolution.

[1]The work was done when the author was at NetApp.

These differences in video occur in the following use-cases: 1) cloud storage, such as Dropbox and Google drive, where multiple copies of videos are stored in a single repository by different users, 2) streaming video providers, such as Netflix and Hulu, where several forms of the same video are stored to accommodate different devices, networks, and regions, 3) video production environments, such as production houses, where multiple editors work on segments of the same RAW video, 4) cloud-based DVR, such as Comcast, where different users start recording the same movie or TV show at different start times, and 5) video content repository such as YouTube, where multiple copies of the same video exist.

Thus, conventional wisdom says that deduplication techniques will not yield high storage savings in video repositories. In this paper we address the above needs through the following contributions:

Storage Efficiency for Video Taxonomy: We present a taxonomy that organizes the related work in storage efficiency for video data in order to get better insights into existing video storage efficiency techniques.

Deduplication Algorithm for Video: Next we present two content-aware deduplication algorithms that can detect duplicates in the first four use cases listed above. We develop these algorithms based on the insight that by leveraging standard metadata encoded in a video file such as MP4 format, it is possible to overcome the perceived differences between two video file copies.

Video Deduplication Study: We conduct an experimental study to understand the deduplication possibilities in video files extracted from DVDs. We build different dataset mixes to simulate real-world scenarios. We want to see how well content-aware techniques perform in comparison to content-unaware techniques. The key results of our study are that deduplication yields storage efficiency for video data, and that both content-aware and variable-sized deduplication techniques are suitable for deduplicating videos.

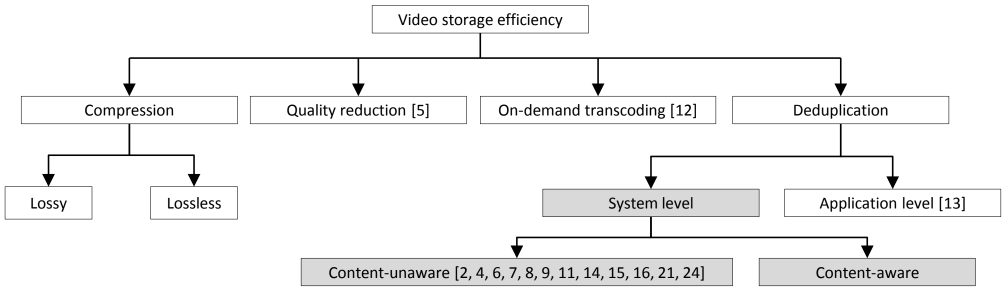

Figure 1: The classification is mainly from the storage efficiency angle. Refer to the text for details of each technique. The shaded boxes are the techniques evaluated in this paper. The content-aware deduplication at the system level is a new technique that we propose.

2 Classification

We present a classification of video storage efficiency techniques to better understand the landscape and define the terminology used in this paper. Figure 1 summarizes the space of video storage efficiency techniques.

Compression: Video compression is the process of compressing a large digital video source into a smaller video file by using coding algorithms. Spatial image compression removes redundancy in a frame by applying discrete cosine transform (DCT) [3] or discrete wavelet transform (DWT) [19, 23]. Temporal motion compensation identifies changes across frames and stores videos in a format of a reference frame and a sequence of transformations on this frame, which are often very small [18, 22]. Video compression can be further grouped into two categories: *lossless* and *lossy* compression, with the latter being more common.

Quality reduction: This technique reduces file size by dropping frames or reduce resolution, resulting in videos that are noticeably worse than the original but are still considered acceptable in certain applications such as surveillance [5].

On-demand transcoding: Transcoding is the conversion of a video from one encoding to another. It is used to match the video file to the task at hand: from video editing, playing on diverse hardware, playing at different size/resolutions, and for varying network bandwidth, e.g., Netflix encodes a movie as many as 120 times to handle the variety of hardware and network possible [1]. One approach is to transcode on the fly using a master copy to the requested format [12].

Deduplication: These techniques remove redundant data by comparing files in a repository. They can be applied at the *application level* or at the *system level*. The application-level approach uses knowledge of the data, by using machine learning to detect similar video content, to identify redundancy that is not detected at the system level [13]. This approach has a high computing overhead and may be *lossy* from a system viewpoint, but it saves the most space. System-level deduplication splits a file into chunks, calculates a signature, and compares with existing ones. If a match is found, the chunk is discarded else it is kept. Extensive work has been done in this area because this technique is applied for primary and archival workloads [2, 4, 6, 7, 8, 9, 11, 14, 15, 16, 21, 24]. As explained in the next section, we use these *content-unaware* techniques, as well as build two *content-aware* techniques, which leverage the knowledge of video file formats.

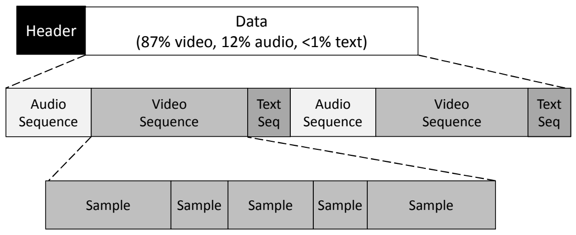

Figure 2: **Media file layout:** The file consists of *samples* and *sequences*. Sample-based chunking offer maximum efficiency while sequence-based offers low overhead.

3 Deduplication Algorithms

Deduplication splits new files into chunks, compares hashes of those chunks to existing chunks, and if there is a match, a pointer to the original chunk is stored instead of the new chunk. We compare several algorithms: *content-unaware*, such as fixed-size chunking and variable-sized chunking, and *content-aware* algorithms that leverage the structure of the file to chunk the file. The rest of this section gives an overview of the data layout of an ISO media file and discusses the algorithms.

The ISO base media file format (MP4) [ISO/IEC 14496-12] is a common file type used to store audio or video media. Other media file formats follow a similar layout to the MP4 format, to ensure efficient video playback. An MP4 file is organized with a *metadata* section and one or more *data* sections (see Figure 2). The data section of the file contains interleaved *sequences*, which contain multiple *samples* of video, audio, or text (closed-captioning). Given knowledge of the metadata structure of an MP4 file, it is always possible to construct a list of *samples* (the smallest unit of audio, video, or text data) and *sequences* (groups of samples of a single type).

In *fixed-sized* deduplication, we segment the file into equal-sized chunks of 4096 bytes. *Variable-sized* deduplication maintains a rolling hash and chunks by using the Rabin fingerprint [20]; as in LBFS [17], we use a 48-byte window and an average chunk size of 4096 bytes. We also develop two content-aware techniques: *sample-based*, where each sample within the file is considered as a separate chunk, and *sequence-based*, where groups of samples are considered as chunks. Additional metadata in the file is considered as a separate chunk for the purposes of this technique. Samples offer maximum benefit, while sequences allow efficient streaming and lower overhead – that is, if the deduplication efficiency is the same, then sequence-based chunking is better.

4 Evaluation

We evaluate the following deduplication schemes in terms of deduplication ratio and metadata overhead on two different datasets: fixed-sized (denoted as F in figures; variable-sized (V); sample-based (S); and sequence-based (Q).

4.1 Experimental Setup

The datasets consist of movies and TV episodes generated using Handbrake [10], an editing and transcoding tool to generate variations. All files are in MP4 or 3GPP format. We show the results of MP4 format, although the results are consistent for both formats. The original videos are called the *Base* video set. The Base videos and the different variations combined contain 202 video files with a total size of 115GB. The variations can be classified into two categories: **Content variations:** Files generated by modifying the content of Base files. It includes four types of changes: *dubbed* - video files with different languages (variation in the audio segment), *resolution* - video files with varying video resolution (variation in the video segment), *closed-caption* - video files with an additional text segment, and *web-optimized* - the video header always precedes the content; Normally the header can be located anywhere in a video file, but web-optimized places the header at the start of the file, optimizing it for streaming across the web. **Video segment:** Contains files with different *start time*, *end time* and *subset* (both start and end times) compared to base files.

We report *deduplication ratio*, the ratio of saved storage space to total data size; the ratio is the percentage of space saved. For example, if two video files total 200MB, a deduplication ratio of 30% indicates that 60MB of data is removed and 140MB of data is stored. We also report *metadata-overhead*, which is measured using two numbers: the number of signatures, which affects lookup performance, and their storage overhead (additional storage space used). For content-aware deduplication techniques, we further break down storage savings into different content segments: sample-based/ sequence-based audio and video. For example, assume that we have two video files with the same video content but different languages. An ideal content-aware deduplication algorithm would deduplicate the video segments and mark the audio segments as unique.

4.2 Content Variations

We run our multiple deduplication algorithms over the content variation dataset and collect total file size, unique data size after deduplication, number of signatures, and storage space occupied by signatures for comparison.

Figure 3(a) shows the deduplication ratio. The x-axis shows four video variations and the y-axis shows the deduplication ratio. The first two clusters show the results obtained using the Dubbed and Closed-Caption datasets. A total of 28 movies are stored in the Dubbed dataset. The Closed-Caption dataset contains 30 movies. In both cases, the fixed-sized technique fails to identify duplicate content. This is because even a small variance in video chunks most likely alters the video header length and subsequently affects all the offsets at which the fixed-sized technique chunks the video file. Content-aware techniques differentiate between the audio, video, and text frames. Thus, in the Dubbed dataset, content-aware techniques identify the variation in the audio content, yielding a 43% deduplication ratio. Variable-sized technique yields 34%, because the audio sequences placed in between the video sequences break the continuity of duplicate data between a video file and its dubbed version. Videos in the *Closed-Caption* dataset contain an additional text segment, but it contributes only 0.00008% to the entire file size, so the content-aware and variable-sized techniques are able to achieve 49.7% and 49.4%.

The *Resolution* dataset, whose results are shown in the third cluster, is different from other datasets. Figure 2 shows the average size of audio, video, and text samples in a video file. With no common video content, deduplication is possible only on the audio and text of a video file; that is, the maximum deduplication ratio is 14%. Although audio content at the sample level is the same, it is reshuffled among sequences, which makes sequence-based deduplication less effective than the sample-based technique. In the case of the *Web-Optimized* dataset, the entire content is similar but jumbled, where the header portion is located in a different part of the file; both

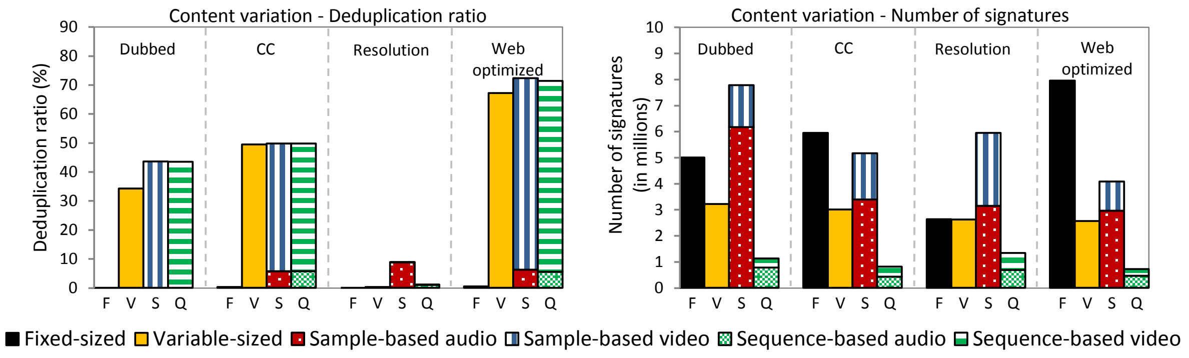

Figure 3: **Content variations:** The left figure (a) shows the deduplication ratio observed for different content variation datasets. The right figure (b) shows the corresponding metadata overhead in terms of number of signatures in the signature table.

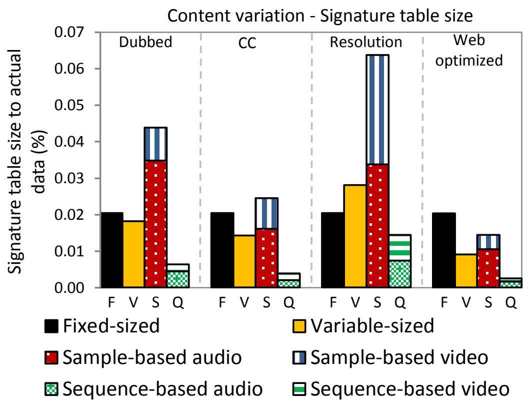

Figure 4: **Content variations:** Ratio of the size occupied by the signature table to the dataset size.

content-aware and variable-sized techniques find duplicates, while the fixed-size technique finds none.

Figure 3(b) shows the metadata overhead of the deduplication algorithms corresponding to Figure 3(a). The metadata overhead is expressed as number of signatures stored in the signature table for each dataset. The signature count provides a better understanding of lookup performance. The sequence-based technique has a very low signature count of 996,941, while other schemes are higher by at least a factor of two. Figure 4 shows the ratio of metadata size to the entire data size. Each signature table entry records a signature, an offset, and length; fixed-size chunking does not store length. An entry occupies 116 bytes (20 bytes for signature, 32 bytes for length, and 64 bytes for offset). In all datasets, the sample-based technique has the highest overhead, followed by fixed-sized and variable-sized. The sequence-based technique has the least overhead (82% lower than sample-based and 65% lower than variable-sized) and has a deduplication ratio comparable to the sample-based technique.

4.3 Video Segments

Next, we run experiments on videos that vary in their start times, end times, and both (subset of videos). There are 18 video files in the dataset.

Figure 5(a) shows deduplication ratio. The first cluster shows the results for videos with different start times. This dataset has an average of 45.9% overlapping data. Since the starting times are different, the reference frame from which the first few samples are encoded also differ; therefore not all of the overlapping content is deduplicated. Sample-based and variable-sized techniques perform the best. The sequence-based technique does not identify all of the duplicates because the samples within the sequences are rearranged. The second cluster shows a dataset containing video pairs that have different end times. The dataset contains an average of 46.1% overlapping data. The starting times are same for the videos, so the reference frames are identical. Sample-based, sequence-based, and variable-sized techniques perform equally well. Specifically, we note that between the two video files, the sample arrangements within the sequences are the same, so sequence-based deduplication performs as well as sample-based. The third cluster shows results obtained for a dataset containing pairs of video in which one video is a subset of the other. The behavior of this dataset is similar to that of the dataset with different start times.

Figure 5(b) shows metadata overhead. It is important to note that the sequence-based content-aware technique has the least amount of metadata overhead – 83.4% less metadata compared to the sample-based technique and 65% less compared to the variable-sized technique.

5 Conclusions and Future Work

We provide a classification of video storage efficiency techniques and concentrate on the effect of deduplication on video files. We overturn the conventional wisdom that deduplication does not yield benefits for video data

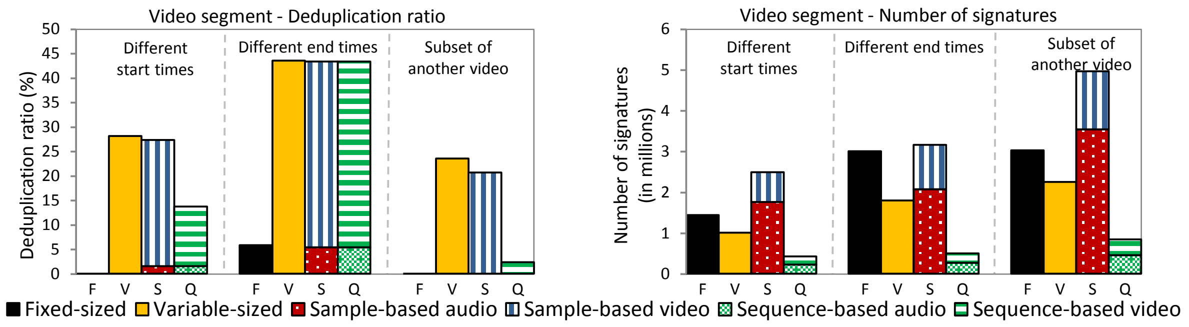

Figure 5: **Video segments:** The left figure (a) shows the deduplication ratio. The right figure (b) shows the corresponding metadata overhead in terms of number of signatures.

by pointing out multiple usecases where video files can benefit from deduplication. We evaluate the deduplication benefit using four algorithms: fixed-sized, variable-sized, and two new *content-aware* algorithms that leverage the video file format. We find that except fixed-sized deduplication technique, the other three can improve storage efficiency by up to 45%. Variable-sized deduplication can be applied to any video repository, regardless of the file format. When a file format is known, content-aware approaches can further reduce storage cost by 10%. In general, the sample-based content-aware technique provides the most storage efficiency. The sequence-based technique trades a small percentage of storage efficiency for an 81% reduction in signature metadata. This work has been conducted specifically on MP4 and 3GPP file formats. In the future, we can verify the effect of deduplication across file formats and different codecs. We would also like to apply deduplication algorithms to a larger video repository to further validate our results.

References

[1] Netflix Inc. http://gigaom.com/2012/12/18/netflix-encoding/.

[2] Achieving storage efficiency through EMC Celerra data deduplication. EMC white paper, March 2010.

[3] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *Computers, IEEE Trans.*, 100(1):90–93, 1974.

[4] C. Alvarez. NetApp deduplication for FAS and V-Series deployment and implementation guide. Technical Report TR-3505, NetApp Inc., 2011.

[5] B. Benight. Inventory management tracking control system. U.S. Patent 11 323 408.

[6] D. Bhagwat, K. Eshghi, D. D. Long, and M. Lillibridge. Extreme binning: Scalable, parallel deduplication for chunk-based file backup. In *MASCOTS'09*, pages 1–9. IEEE, 2009.

[7] A. T. Clements, I. Ahmad, M. Vilayannur, J. Li, et al. Decentralized deduplication in SAN cluster file systems. In *USENIX'09*, pages 8–8. USENIX Association.

[8] B. Debnath, S. Sengupta, and J. Li. Chunkstash: speeding up inline storage deduplication using flash memory. In *USENIX'10*. USENIX Association.

[9] C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, et al. Hydrastor: A scalable secondary storage. In *FAST'09*.

[10] HandBrake. The handbreak team. http://handbrake.fr.

[11] K. Jin and E. L. Miller. The effectiveness of deduplication on virtual machine disk images. In *SYSTOR'09*. ACM.

[12] A. Kathpal, M. Kulkarni, and A. Bakre. Analyzing compute vs. storage tradeoff for video-aware storage efficiency. In *USENIX'12*, HotStorage'12, Berkeley, CA, USA.

[13] A. Katiyar and J. Weissman. ViDeDup: an application-aware framework for video de-duplication. In *USENIX'11*, pages 7–7.

[14] R. Koller and R. Rangaswami. I/O deduplication: Utilizing content similarity to improve I/O performance. *ACM TOS'10*, 6.

[15] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, et al. Sparse indexing: Large scale, inline deduplication using sampling and locality. In *FAST'09*.

[16] D. T. Meyer and W. J. Bolosky. A study of practical deduplication. *ACM TOS'12*.

[17] A. Muthitacharoen, B. Chen, and D. Mazières. A low-bandwidth network file system.

[18] M. T. Orchard and G. J. Sullivan. Overlapped block motion compensation: An estimation-theoretic approach. *Image Processing, IEEE Trans.*, 3(5):693–699, 1994.

[19] M. Rabbani. JPEG2000: Image compression fundamentals, standards and practice, 2002.

[20] M. Rabin. *Fingerprinting by Random Polynomials*. Center for Research in Computing Techn., Aiken Computation Laboratory, Harvard University, 1981.

[21] K. Srinivasan, T. Bisson, G. Goodson, and K. Voruganti. iDedupe: Latency-aware, inline data deduplication for primary storage. In *FAST'12*, pages 299–312, San Jose, CA, Feb. 2012.

[22] G. J. Sullivan and R. L. Baker. Motion compensation for video compression using control grid interpolation. In *ICASSP-91*, pages 2713–2716. IEEE, 1991.

[23] B. E. Usevitch. A tutorial on modern lossy wavelet image compression: foundations of JPEG 2000. *Signal Processing Magazine, IEEE*, (5), 2001.

[24] B. Zhu, K. Lee, and H. Patterson. Avoiding the disk bottleneck in the data domain deduplication file system. In *FAST'08*, pages 269–282, San Jose, CA, USA, Feb. 2008.