ViDeDup: An Application-Aware Framework for Video De-duplication

Atul Katiyar
Windows Live
Microsoft Corporation, Redmond, WA 98052
atulk@microsoft.com

Jon Weissman
Department of Computer Science and Engineering
University of Minnesota Twin Cities, MN 55455
jon@cs.umn.edu

Abstract—**Key to the compression-capability of a data de-duplication system is the definition of *redundancy*. Traditionally, two data items are considered redundant if their underlying bit-streams are identical. However, this notion of redundancy is too strict for many applications. For example, for a video storage platform, two videos encoded in different formats would be unique at the system level but redundant at the content level. Intuitively, introducing application-level intelligence in redundancy detection can yield improved data compression. We propose *ViDeDup* (Video De-Duplication), a novel framework for video de-duplication based on an application-level view of redundancy. The framework goes beyond duplicate data detection to *similarity-detection*, thereby providing application-level knobs for defining acceptable level of noise during replica detection. Our results show that by trading CPU for storage, a 45% reduction in storage space could be achieved, in comparison to 8% yielded by system level de-duplication for a dataset collected from video sharing sites on the Web. We also present tradeoff analysis for various tunable parameters of the system to optimally tune the system for performance, compression and quality.**

I. Introduction

THE emergence of the *internet as a platform* has provided its users new opportunities for data and information sharing. User generated content in its various forms contributes to redundancy in Web data. Redundancy in Web data has been studied extensively in the past [1]. There has been less focus, however, on exploiting redundancy to reduce storage cost. With the evolution of large scale datacenter storage and data-clouds [2], we envision that storage for Web data may become more centralized in the future. We further envision the possibility of data-clouds provided by a few prominent vendors to strength this belief. Storage of this highly redundant Web data within a single administrative domain would make inefficient use of storage resources. Even today, in the context of *user data sharing platforms* (like YouTube), the challenges for large-scale, highly redundant Web data storage are high.

Storage for this increasingly centralized Web data can be optimized by its de-duplication. De-duplication [3] is a method for eliminating redundant copies of duplicate data and replacing them with a pointer to the unique copy. Considering the prevalent redundancy in Web data and its increasing centralization, the prospects of storage optimization via de-duplication are promising.

From the storage perspective, it is important to distinguish here the case of *managed* versus the *unmanaged* redundancy. We refer to redundancy as *managed* when the underlying storage-system is aware of the replicas and replication is performed for specific goals (e.g., high-availability, performance, QoS etc.). However, in case of *unmanaged* redundancy, the storage-system is unaware of the replicas and their existence does not specifically contribute towards improved characteristics of the system. Unmanaged redundancy in large scale storage systems poses data-management challenges.

In this paper, we address the unmanaged redundancy in Web data by introducing the notion of *application-aware de-duplication* which gauges redundancy in content rather than at byte-level (which we call system level de-duplication). We present the design and architecture of ViDeDup*, an application-aware video de-duplication system for compressing videos. The framework is novel in introducing the notion of application-aware de-duplication. It incorporates application-level intelligence in various stages of the de-duplication process. This framework in addition provides *similarity-detection*, hence providing application-level knobs for defining acceptable noise during replica detection. Why would loss ever be tolerated? For video-sharing websites, e.g. YouTube, the cloud administrator may make such decisions when storage pressure is highest. The clients of such services would be aware of such policies. Or there may be classes of users willing to pay for a better QoS for video retrieval and request no loss. For users that want free service, they may have to be willing to suffer some degree of loss (particularly since the cloud is not storing an authoritative copy, but rather a version for dissemination).

We implemented a functional prototype of ViDeDup and exercised its compression capabilities on videos collected from video sharing websites. We demonstrate that by trading CPU for storage, ViDeDup can reduce storage by as much as 45% when system level de-duplication only yielded 8% space savings.

*Refer to http://videdup.cs.umn.edu for the dataset used in this paper and examples of video de-duplication.

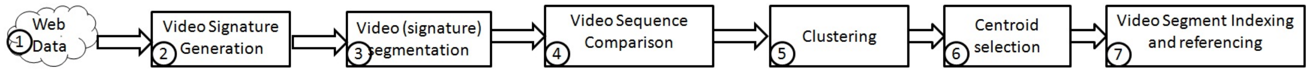
Web Data
1
Video Signature Generation
2
Video (signature) segmentation
3
Video Sequence Comparison
4
Clustering
5
Centroid selection
6
Video Segment Indexing and referencing
7

Figure 1. ViDeDup System architecture.

II. System Architecture

In this section, we describe the detailed system architecture of ViDeDup. Figure 1 shows the major steps involved in video de-duplication.

A. Video Signature Generation

The de-duplication framework needs a mechanism for comparing videos. Time efficient video-comparison is done on a compact representation of the video, called the *video-signature*. Various techniques have been proposed in the past [4]. From the proposed schemes, we chose Ordinal signature [5] to construct video-signatures. Ordinal signature is a function of both the color and spatial-temporal distribution of intensities of video and hence is robust to changes in resolution and display formats of the compared videos. Also, ordinal signatures are computationally efficient to compute and compare, thereby meeting the needs of our framework.

In the ordinal signature scheme, the original video is first down-sampled at a uniform rate of T_s frames/second to make the signature robust to different frame rates. Each keyframe is extracted and divided into $m = N_x \times N_y$ blocks, which are then sorted based on their average gray-level intensity. Summarizing, the ordinal video signature (S) is a feature vector in $\mathbb{Z}^{n\times m}$ space such that

$$\begin{aligned} &S = \{S_1, S_2, S_3, ..., S_n\}, \\ &S_i = \{(r_1, r_2, ..., r_m) \mid I(r_1) \le ... \le I(r_m)\} \end{aligned} \tag{1}$$

where $(r_1, ..., r_m)$ is a permutation of $(1, ..., m)$, $I(r_i)$ is the average gray-level intensity of the r_i block, m is the number of blocks in each extracted keyframe of the video and n equals total number of key frames in the down-sampled video.

B. Video Segmentation

It is intuitive that the comparison of complete video-sequences would ignore similar videos differing in only a smaller number of frames (due to minor video editing like prefixing/suffixing/infixing or chopped-length videos). Hence it is crucial to partition the video signatures into *segments* and compare them for similarity instead of complete sequences. However, picking the optimal segment size is difficult [3]. In our implementation, we segment the video signatures statically for ease of implementation. Compared video-sequence signatures are divided into segments of uniform size which are then compared for a match to identify redundant segments among videos. More specifically, the video signature matrix $S \in \mathbb{Z}^{n\times m}$ (defined in Equation 1) is divided into $\lfloor n/k \rfloor$ segments, S^i, ($S^i \in \mathbb{Z}^{k\times m}$), each containing k m-dimensional ordinal feature vector (except the last which contains $n\%k$ feature vectors that are stored as is during de-duplication). Comparison of video-sequences is done on segment basis as described next in Section II-C.

C. Video Sequence Comparison

In our prototyped system, we use a 2-phase video comparison scheme based on the ordinal signature, as described in [5]. This scheme is primarily for localizing a short query clip in a long target video. We adapted this algorithm so that it can find sub-sequence similarity between compared videos based on segments. The coarse comparison between two video sequences is based on the Sequence Shape Similarity (SSS) metric [5].

1) Sequence Shape Similarity (SSS) Metric: SSS measures the distance between the ordinal signatures of the compared video sequences. Let $S_X = \{X_1, X_2, \ldots, X_M\}$ and $S_Y = \{Y_1, Y_2, \ldots, Y_N\}$ denote the ordinal signatures of the compared video sequences X and Y respectively. Signature of video sequence X and Y are segmented statically into uniform segments, S_X^i, and S_Y^j respectively, each of size k such that,

$$S_X^i = \{X_{i\times k+1}, \ldots,\ X_{i\times k+k}\}, 0 \le i < \lfloor M/k \rfloor \tag{2}$$

$$S_Y^j = \{Y_{j\times k+1}, \ldots, Y_{j\times k+k}\}, 0 \le j < \lfloor N/k \rfloor. \tag{3}$$

The SSS metric for comparison between i^{th} segment of $S_X (= S_X^i)$ and j^{th} segment of $S_Y (= S_Y^j)$ is defined as,

$$\begin{aligned} &SSS(S_X^i, S_Y^j) = \\ &\qquad \textstyle\sum_{l=1}^{k} I(d(X_{i\times k+l}, Y_{j\times k+l}) \le \epsilon_{frame\text{-}threshold})/k \end{aligned} \tag{4}$$

where, $d(.) = L1$ distance metric defined on the ordinal measure, and $I(x)$ equals 1 if x is true; and zero otherwise. $\epsilon_{frame\text{-}threshold}$ is the predefined distance threshold to account for possible noise added due to temporal resampling, and inherent minor differences between the compared videos. Two segments, S_X^i and S_Y^j are regarded as similar if,

$$SSS(S_X^i, S_Y^j) \ge T_{segment\text{-}threshold} \tag{5}$$

2) Video Similarity score: Based on the SSS metric computed above, each pair of compared videos is assigned a similarity score (SS). SS measures the similarity between the compared videos X, Y as

$$SS(X,Y) = \left(\frac{(\mid S_X \cap S_Y \mid)}{(\mid S_X \mid)} + \frac{(\mid S_Y \cap S_X \mid)}{(\mid S_Y \mid)}\right)/2 \tag{6}$$

where $\mid S_X \cap S_Y \mid$ equals number of similar segments as measured by Equation 5 and $\mid S_X \mid$, $\mid S_Y \mid$ represent number of segments in video sequences X and Y, respectively.

D. Clustering

At the completion of coarse-comparison phase, we obtain a similarity matrix (SM) such that,

$$SM \in \mathbb{R}^{N\times N} \; and \; SM(i,j) = \begin{cases} SS(i,j) & if \; i \neq j \\ 1 & if \; i = j \end{cases} \tag{7}$$

where N =Total Number of videos.

Note that $SM(i,j)$ is a symmetric square matrix as $SS(i,j) = SS(j,i)$. Next, videos are clustered based on their similarity score. To cluster the videos, we need a distance matrix, DM, instead of similarity matrix, SM. Distance matrix DM can be obtained as,

$$DM = (\mathbf{1}\mathbf{1}^T – SM) \tag{8}$$

where $\mathbf{1}$ is an N-dimensional vector of all ones. The videos are now clustered using the K-MEANS clustering algorithm based on DM. The optimal value for the number of clusters in the dataset was determined using gap statistics [11].

E. Centroid Selection

In ViDeDup, we store the highest perceptual-quality (defined in Section II-E2) representative video, that we call the *centroid-video*, of the cluster in its entirety. Similar segments of other videos can then be *derived* from the centroid-video's segments in a lossless manner using the standard video transformations like down-sampling video-sequence, down-scaling video-frames etc. This centroid-based segment storage and indexing deviates from the traditional scheme for segment based de-duplication (e.g., LBFS [12]) which stores all of the unique data segments in a segment-pool. The segment allocation and indexing scheme in these systems does not differentiate which of the two duplicate segments actually gets stored in the system. This is because the segments are identical and hence the choice is immaterial. However, in the case of video de-duplication, this choice is not so trivial! Storing the higher perceptual quality video segment instead of its lower quality correspondent, would allow lossless regeneration of both the segments but not vice versa.

1) Centroid Selection: Once the videos are partitioned into disjoint clusters, a centroid-video is chosen for each cluster. Informally, the centroid-video is the representative of the cluster and is the video which is *most like the others*. The clustering algorithm returns a virtual-centroid which may not necessarily map to a physical data-point (i.e., the video) in the cluster. It might be intuitive to pick the video closest to the virtual-centroid as the centroid of the cluster. But as we discussed earlier, lossless reconstruction of remaining videos of the cluster when compressed w.r.t the centroid might not be possible, as the chosen centroid-video may not necessarily be of the highest quality. On the other hand, merely selecting the video with highest quality as the centroid would compromise on compression as it might not be the best representative of the set. Hence, there is a tradeoff between the compression and the quality of compressed videos. The centroid-selection algorithm (Algorithm 1) balances this tradeoff by minimizing the compression-ratio and maximizing the quality of compression for each cluster.

In the pseudocode, `RelPerceptualQual`(v_i, V) computes the perceptual quality index ($quality[v_i] \in [0,1]$) for video v_i relative to the videos of cluster V (Line 1-2). The *gain-factor* (f) is computed for each video v_i considering v_i as the centroid (Line 4-7). The *gain-factor* is defined as the ratio of the relative perceptual quality of the video ($= quality[v_i]$) and the average dissimilarity of video v_i with the remaining videos of cluster ($= distance$, computed from DM as defined in Equation 8). The chosen centroid maximizes the gain-factor (Line 8), hence balancing the tradeoff. Sorting ensures that video with highest perceptual-quality is selected in case the gain-factor for two videos is the same.

2) Perceptual Video Quality : The centroid selection algorithm discussed above, needs to objectively quantify the perceptual quality of the video (`RelPerceptualQual`(v_i, V)) relative to the cluster. Objective video quality assessment is a topic of active research. In our implementation, we use heuristics to simplify the quality assessment. A video with higher frame rate and higher spatial resolution is regarded as of quality higher than others. However, other techniques for *no-reference* quality assessment, like those presented in [9], may yield better results. Exploration of these techniques is left as future work.

Algorithm 1: Centroid Computation for the given Cluster

```
input : Cluster $V = (v_1, v_2, ..., v_N)$,
        Distance Matrix $DM = [d_{ij}]_{N\times N}$
output: Index $k$ of the Centroid video $v_k$
begin
1   foreach $v_i \in V$ do
2       quality[$v_i$] ← RelPerceptualQual($v_i, V$)
    end
3   Sort $V$ descending based on quality[$v_i$]
4   foreach $v_i \in V$ do
5       distance ← $\frac{\sum_{j=1}^{N} d_{ij}}{N}$
6       if ($distance == 0$) then $f[v_i] \leftarrow Infinity$
7       else $f[v_i] \leftarrow \frac{quality[v_i]}{distance}$
    end
8   $k \leftarrow \{\texttt{min}(i) \mid f[v_i] \geq f[v_j],\ \forall v_i, v_j \in V\}$
9   return $k$
end
```

F. Video Segment Indexing and referencing

After the cluster's centroid-video is chosen, the remaining videos in the cluster are de-duped w.r.t the centroid. The segmented video-sequences of remaining videos of the cluster are compared at each coarsely matched location in centroid-video (as computed previously in Section II-C) and are aligned to the best match using Needleman-Wunsch sequence alignment algorithm [6]. Based on this, the segment-index table is built which records the segment offsets in the centroid-video (for the matched segments) and/or the compared video (for the unique segments).

III. Evaluation

We selected 12 queries chosen to retrieve the most popular videos on YouTube, to form our dataset and downloaded the videos returned by the search engine. Our dataset contains 1017 videos (over 90 hours), totaling nearly 15.81 GB. The downloaded videos were in .flv, .mp4 or .3gp formats. The dataset is heterogeneous w.r.t its encoding characteristics like frame resolution (varying from 140x96 to 854x480) and frame rate (varying from 10 fps to 59.75 fps), exercising the various design aspects of the de-duplication framework. We implemented the functional prototype of ViDeDup in Matlab. The video codec library, FFmpeg, was used for performing video-processing tasks such as extracting frames from video, encoding video from frames etc. Performance characteristics of ViDeDup were measured on Intel Core 2 Duo CPU of 2.26GHz with 4GB DDR2 System Memory and 500 GB disk.

A. Comparison with System Level De-duplication

We use the system level de-duplication simulator, used in [8], to compare ViDeDup's compression ratio with system level de-duplication. Table I and Table II describe the configuration of the two systems. ViDeDup yielded a compression ratio of 0.55 in comparison to 0.92 for system level de-duplication. The overall compression ratio of 0.55 for the dataset yielded space savings of 45% (or 7.1 GB).

Parameter name	**Parameter value**
Video-Signature Scheme's parameter	Down-sampling frequency, $T_S = 10\ fps.$
	Number of blocks per frame, $N_x \times N_y = 4 \times 4$
Signature comparison algorithm's parameters	Coarse comparison threshold, $T_1 = 0.5$
	segment size = 400 frames
	Signature matching threshold, $\epsilon_{frame\text{-}threshold} = 2$
	segment matching threshold $T_{segment\text{-}threshold} = 0.8$
Number of Clusters (K)	100

Table I
Configurable parameters of ViDeDup along with their respective initialization values.

Parameter Name	**Parameter Value**
Segmentation mode	Variable sized segments
Range for variable segment size	4096 to 16384 bytes
Fingerprint computation function	SHA-1 hash
Size of sliding window for segmentation	32 bytes
Segment compression method (to harness intra-segment redundancy)	Ziv-Lempel

Table II
Configuration parameters for system level de-duplication simulator

B. Relationship between Compression Ratio and Quality

We use video quality assessment algorithm presented in [10] for computing the quality of compressed video in comparison to its uncompressed counterpart. Figure 2 shows the relationship between the compression ratio and quality of compression. For the overall compression ration of 0.55, the mean and median quality (Figure 3) of the compressed videos was measured to be 0.8416 and 0.8798 respectively, with a standard deviation of 0.1802. Above, quality index > 0.75 is considered to be visually undetectable.

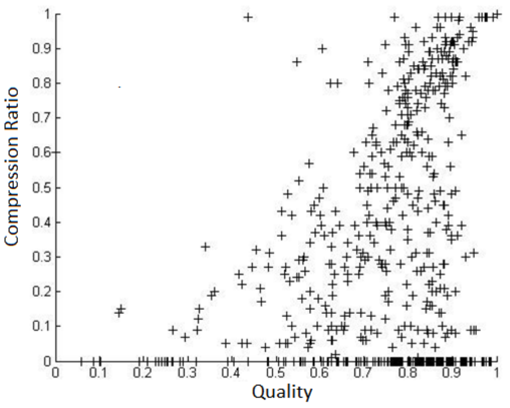

Figure 2. Compression Ratio plotted as a function of Quality of compressed video in ViDeDup

From the graph (Figure 2) it is evident that the compression ratio is directly related to the quality of compressed content i.e., higher the compression ratio (or lower the space savings), higher the quality. This however may not be true in individual cases. For example, from the graph it can be seen that there exist videos V_1, V_2 in the dataset such that $compression\_ratio(V_1) > compression\_ratio(V_2)$ but $Q(V_1) < Q(V_2)$. This happens when

1) There are *false positives* returned by the video comparison algorithm. This is analogous to hash collision in system level de-duplication. As ViDeDup targets duplicate and near-duplicate detection at the content level, replica detection via hash comparison is not possible.
2) The metadata characteristics (like frames/second and resolution) of centroid video were more closely related to V_2 than V_1. This can happen when there is no single video whose metadata characteristics are superior to all remaining videos of cluster.

We next look at the histogram plot (Figure 3) of the compression ratio and quality of compressed videos, to determine their distribution over the dataset. The histogram plot for quality of compressed videos is right skewed indicating that, most of the videos maintained high quality after compression. Another key insight is that the number of videos which have compression ratio in the interval $[0.1,\ 0.9]$ is relatively small. This insightful observation can be utilized to optimize the performance of the de-duplication framework, by favoring larger segment sizes.

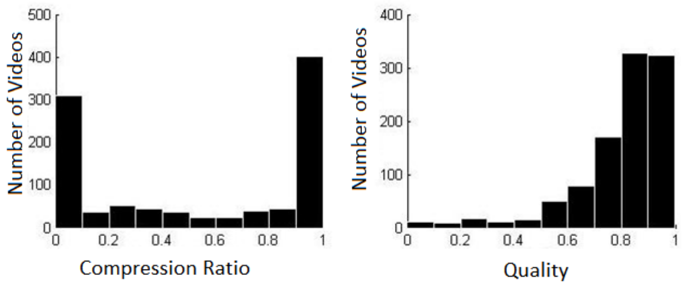

Figure 3. Histogram plot for the Compression ratio and Quality of compressed video

C. De-duplication Rate

Although ViDeDup has not yet been optimized for performance, we do report its performance characteristics for three principal operations, viz.: video-signature computation, pairwise video-comparison and video-reconstruction from its index. Ordinal-signature computation time depends on the number of keyframes in the video-sequence and keyframe characteristics (like resolution and number of blocks). For keyframe characteristics listed in Table I, the prototype could compute signatures at 3.66×10^2 frames/sec. Even though this number is towards the lower end, the performance impact is not significant as it is a one time operation and the signature of different videos can be computed in parallel. Pairwise video-comparison is performed by comparing signature vector of keyframes in segments. The prototype could compare the signature vector at $\sim\ 2.88 \times 10^6$ frames/sec. Finally, video reconstruction throughput, from its unique (belonging to the original non-deduped video) and non-unique segments (belonging to the centroid-video), could be performed at 10^4 frames/sec.

The most compute-intensive step in ViDeDup is the pairwise video comparison. For our dataset of 1017 videos, this involved $\binom{1017}{2}$ pairwise video-comparisons which translated to $400 \times \binom{9721}{2}$ † pairwise signature-vector comparisons for the configuration listed in Table I, taking $\sim$ 2 hours when compared at the rate of 2.88×10^6 frame-comparisons per second. There are many known techniques for improving the performance of video comparisons [7]. Exploration of these techniques is left as future work.

†9721 equals the total number of signature segments of all videos in the dataset and multiplication by 400 is to account for segment size.

IV. Conclusion and Future Work

Data de-duplication is a powerful technique for reducing the storage needs of large scale storage systems. We presented the design of ViDeDup, a novel video de-duplication framework with an application-level view of redundancy. The framework generalizes the definition of *redundancy* by providing application-level knobs to define the acceptable level of noise in replica detection, thereby going beyond duplicate-detection to similarity-detection. We explored the potential of application-aware data de-duplication in compressing the data and analyzed the tradeoffs of performance, compression and quality as controlled by various tunable parameters of the system. From the experiments on live data downloaded from video-sharing Websites, our results demonstrate that application-aware de-duplication can reduce the storage by as much as 45% at a non-perceptual loss in quality, while system level de-duplication yielded only 8% storage savings.

Through our future work we would like to address the scalability challenges of the system through distributed implementation of framework. Also exploration of techniques to handle false-positives during video-comparison in a time-efficient and scalable manner is an interesting research problem. Through our research, we have merely scratched the surface of application-aware de-duplication and demonstrated its tremendous potential in reducing the storage needs of data-clouds.

References

[1] Dennis Fetterly, Mark Manasse, Marc Najork. On the Evolution of Clusters of Near-Duplicate Web Pages. In LA-WEB'03.

[2] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A.Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. Above the Clouds: A Berkeley View of Cloud computing. Technical Report No. UCB/EECS-2009-28, University of California at Berkley, USA, Feb. 10, 2009.

[3] Benjamin Zhu , Kai Li , Hugo Patterson. Avoiding the disk bottleneck in the data domain deduplication file system. *In FAST'08.*

[4] Julien Law-To, Li Chen, Alexis Joly, Ivan Laptev, Olivier Buisson, Valerie Gouet-Brunet, Nozha Boujemaa, Fred Stentiford. Video copy detection: a comparative study. *In CIVR'07.*

[5] X. S. Hua, X. Chen, and H. J. Zhang. Robust video signature based on ordinal measure. *In ICIP'04.*

[6] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology, 48, pp. 443-453, 1970.

[7] S.C. Cheung and A. Zakhor. Efficient video similarity measurement and search. *In ICIP 2000.*

[8] N. Park, D.J. Lilja. Characterizing Datasets for Data Deduplication in Backup Applications. *In IISWC'10.*

[9] U. Engelke and H. J. Zepernick. Perceptual-based quality metrics for image and video services: A survey," *In NGI'07.*

[10] Z. Wang, L. Lu and A. C. Bovik. Video quality assessment using structural distortion measurement. *In ICIP'02.*

[11] R. Tibshirani, G. Walther, T. Hastie. Estimating the number of clusters in a dataset via the gap statistic. J. Roy. Stat. Soc. B 63(2), 411-423 (2001).

[12] A. Muthitacharoen, B. Chen and D. Mazières. A Low-Bandwidth Network File System. *In Proc. of SOSP'01.*