Efficient Storage Support for Real-Time Near-Duplicate Video Retrieval

Zhenhua Nie, Yu Hua, Dan Feng, Qiuyu Li, and Yuanyuan Sun

Wuhan National Lab for Optoelectronics (WNLO),
School of Computer Science and Technology,
Huazhong University of Science and Technology,
Wuhan, Hubei 430074, China
{niezhenhua,csyhua,dfeng,liqiuyu,sunyuanyuan}@hust.edu.cn

Abstract. Near-duplicate video retrieval in a real-time manner is important to offer efficient storage services, and becomes more challenging due to dealing with the rapid growth of multimedia videos. Existing work fails to efficiently address this important problem due to overlooking the storage property of massive videos. In order to bridge the gap between storage system organization and application-aware videos, we propose a cost-effective real-time video retrieval scheme, called FastVR, which supports fast near-duplicate video retrieval. FastVR has the salient features of space- and time-efficiency in large-scale storage systems. The idea behind FastVR is to leverage space-efficient indexing structure and compact feature representation to facilitate keyframe based matching. Moreover, in the compact feature representation, FastVR transforms the frames into feature vectors in the Hamming space. The indexing structure in FastVR uses Locality Sensitive Hashing(LSH) to support fast similar neighboring search by grouping similar videos together. The conventional LSH unfortunately causes space inefficiency that is well addressed by a cuckoo hashing scheme. FastVR uses a semi-random choice to improve the performance in the random selection of the cuckoo hashing scheme. We implemented FastVR and examined the performance using a real-world dataset. The experimental results demonstrate the efficiency and significant performance improvements.

1 Introduction

According to the report of International Data Corporation (IDC) in 2011, the amount of data in our whole world reaches 1.8 Zettabytes and its growth rate is about doubling every two years. The amount of data produced will increase to 40ZB by 2020 [5]. The unstructured data, typically represented by videos and images, is growing faster than structured data. The percentage of unstructured data is about 90 percent of all data created in the next decade [5].

The number of online videos has experienced an exponential growth in recent decades, especially when social video sharing sites appear. In YouTube, over 6 billion hours of video are watched each month, and there are 100 hours of video being uploaded each minute in 2014 [1]. These videos in a variety of formats contain duplicate copies and exhibit similarity from the content view [20]. Near-duplicate video retrieval is important in the era of big data.

X.-h. Sun et al. (Eds.): ICA3PP 2014, Part II, LNCS 8631, pp. 312–324, 2014.

In order to improve storage performance when handling massive videos, data centers are utilized by service providers and developers. However, the centralized data storage model becomes potential performance bottleneck when dealing with the highly redundant videos. Although the cache techniques on chip-multiprocessors [25] are used, data centers still consume substantial energy and system resources to offer storage services. In the meantime, highly redundant videos increase query delay and jeopardize the quality of storage services. In practice, existing work proposes efficient query schemes to deal with metadata management in storage systems, such as Spyglass [14], SmartStore [6], flash-based multiple Bloom filters [19] and BloomStore [16]. These retrieval methods mainly address the exact-matching query in the metadata search, rather than the near-duplicate retrieval for videos in a real-time manner.

In general, near-duplicate video retrieval contains the following operations. First, the extracted keyframes of videos are classified by time sampling, shot based detection algorithms or sliding window so that every video is represented by a sequence of keyframes [8]. Second, these keyframes are represented by their visual features including global features and local features [9,24]. Specifically, a global feature consists of color and spatial and temporal features extracted from a keyframe. A local feature is represented by local keypoints extracted from a keyframe in a high dimensional vector. Third, the system constructs an indexing structure via organizing extracted visual features (global and local features), like the prevalent inverted index [22]. Fourth, we compute the similarity between the queried videos and the video library by using visual features. The whole extracted visual features are taken as the representation of video library. The final results include the most similar videos that are compared with the query video. In this paper, we address two challenges, i.e., inefficient feature representation and performance bottleneck of indexing, to support real-time near-duplicate video retrieval.

Inefficient Feature Representation. In order to support real-time video retrieval, existing work proposes a variety of feature extraction based schemes for efficient video representation to facilitate the near-duplicate video retrieval. These approaches can be classified into two categories: global feature based schemes [21], and local feature based schemes [28]. Specifically, the global feature schemes can extract color, spatial and temporal signatures to deal with almost identical videos, while the local feature schemes can extract the local keypoints to tolerate more photometric and spatial transformations. The global feature based methods have rapid processing speed but low accuracy, while the local feature based methods obtain high accuracy by extracting local keypoints. In practice, these schemes mainly concentrate on improving the retrieval accuracy via extracting more features from the videos or the keyframes to represent the features. Due to overlooking the property of storage, these approaches always consume too much in-memory space, thus failing to support real-time video retrieval.

Performance Bottleneck of Indexing. To support real-time query performance, it is important to construct an efficient indexing structure, which is

space-efficient and obtains low query latency. Only a few studies focus on the real-time retrieval in the large scale near-duplicate videos. Many methods are used to construct the indexing structure in real-time near-duplicate video retrieval, such as tree-structures [2], cluster, spectral hashing [23], and the variations of the search engine technology like the inverted table [4]. Locality Sensitive Hashing (LSH)[3] [10] is often used to implement the fast similarity query by mapping the approximate points into a same bucket to narrow the query scope. LSH has the salient features in efficiency of hashing computation and stabilization of data locality. LSH has the property that the similar items can be hashed into the same buckets with high probability. Although LSH can be used to maintain the near-duplicate relationships among keyframes in the indexing structure, performing real-time LSH-based near-duplicate video retrieval needs to address two main problems. *(1) LSH suffers from low space-efficiency and low-speed I/O access*, *(2) LSH has an unbalanced load in the hash table storage.*

The conventional LSH unfortunately causes space inefficiency that is well addressed by a cuckoo hashing scheme. The cuckoo hashing scheme obtains worst-case constant query time and high utilization of hash tables. Cuckoo hashing recursively kicks items out of their positions and uses multiple hash functions for resolving hash collisions during insertion operation. The random selection in cuckoo hashing incurs a large number of repeated relocations in the kicking-out processes. The reason of repeated relocations is that the item frequently kicks similar items, thus incurring repetitions and loops with a high probability.

This paper has made the following contributions.

Compact Feature Representation. In order to obtain the real-time near-duplicate video retrieval, we propose a compact feature representation. Our feature representation only extracts the local feature to obtain high accuracy, and to avoid the complex computation of the extraction operations which combine the global feature and local feature. By using a feature-aware Bloom filter, we map the local keypoints (extracted from a keyframe) to a feature vector in the Hamming space. Hence, this keyframe is represented by a feature vector. The space of feature vector is much smaller than that of local keypoints.

Semi-random Holistic Hashing. The indexing structure is constructed by the semi-random holistic hashing, which addresses the space-efficiency and load imbalance for the LSH-based method in the near-duplicate video retrieval. The conventional LSH unfortunately causes space inefficiency that is well addressed by a cuckoo hashing scheme. FastVR uses a semi-random choice to improve the performance in the random selection of the cuckoo hashing scheme. We show that the effective combination of LSH and Cuckoo Hashing can accelerate the near-duplicate video retrieval.

Real Prototype Implementation. We have implemented all the components and algorithms of FastVR in our prototype system. We compared it with state-of-the-art work, including NEST [7], ViDeDup [11] and the baseline approach. The baseline approach is the traditional LSH without cuckoo hashing. Furthermore, we use a real and large dataset collected from the popular campus networks of universities in China to evaluate the performance.

The rest of paper is organized as follows. Section 2 presents the backgrounds and related work. Section 3 shows our FastVR model. Section 4 shows the performance evaluation. We conclude our paper in Section 5.

2 Backgrounds and Related Work

This section presents the research backgrounds about feature representation in near-duplicate video retrieval, locality sensitive hashing scheme and cuckoo hashing scheme.

2.1 Feature Representation

In recent decade, various feature extraction methods are proposed by researchers to represent a keyframe. These features are mainly grouped into two categories, global feature based schemes and local feature based schemes.

The global feature schemes extract color, spatial and temporal signatures to represent the keyframes extracted from a video. For example, HSV is extracted to represent the keyframe, and this method receives fast query performance [24]. The compact spatiotemporal feature can be represented by using relative gray-level intensity distribution in a frame and temporal structure of videos [21]. The global feature based methods are less robust in the videos which has spatially and temporally variation, and the experimental performance of the global feature based methods is not as good as the local feature based methods.

Because of the shortage of the global feature based methods, the local feature based methods obtain more attentions [4][13][22]. Some of the popular methods based on local features are SIFT [15] and PCA-SIFT [12]. In [22], the local keypoints can be extracted by SIFT to represent keyframes. The visual keywords are quantized by the local keypoints, then the inverted index is used to index the visual keywords. The correlated work [4] based on visual keywords and inverted index presents excellent experimental results.

2.2 Locality Sensitive Hashing

Locality Sensitive Hashing (LSH) [10] has the property that the similar items can be hashed into the same bucket with high probability. Formally, each hash function h(v): $\mathrm{R}^{\theta} \rightarrow \mathrm{Z}$ maps a θ dimensional vector to a real number. The domain S denotes the point sets, and the distance measure D denotes the distance between two points, so an LSH family is defined as:

Definition 1. *LSH family,* $H = \{h : S \rightarrow U\}$ *is called* (r, cr, P_1, P_2)*-sensitive for distance function* D *if for any* $p, q \in S$

- *If* $D(p,q) \leq r$ *then* $Pr_H[h(p) = h(q)] \geq P_1$,
- *If* $D(p,q) > cr$ *then* $Pr_H[h(p) = h(q)] \leq P_2$.

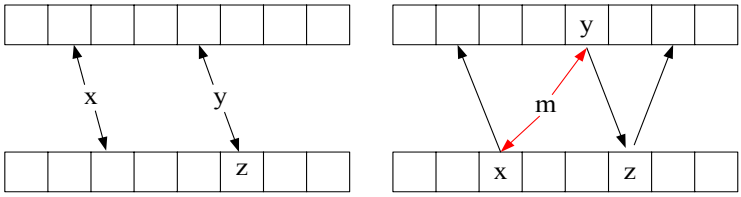


Fig. 1. An example of cuckoo hashing structure

Due to the use of LSH, the settings should be $P_1 > P_2$ and $c > 1$. We define a function family $\Gamma=\{g:S\rightarrow \mathrm{U}^k\}$. Then we need to illustrate another two parameters, d, the number of hash tables and k, the number of hash functions in function family. We have $g(v)=(\mathrm{h}_1(v),\ldots, \mathrm{h}_k(v))$, where v is a θ dimensional vector and $\mathrm{h}_i(v) \in H$.

LSH uses the hash collisions to support the approximate queries. As a result, it suffers from low space-efficiency utilization in hash tables and unbalanced load in the storage of hash tables. The multi-probe LSH [17] is proposed to illustrate the similarity with adjacent buckets. The last level caches can leverage the temporal and spatial capacity demands to narrow the gap between processor cores and main memory [26] [27]. This can partially address the time-inefficiency in real-time near-duplicate video retrieval.

2.3 Cuckoo Hashing

To address the low space-efficiency utilization and unbalanced load in hash tables of LSH, the cuckoo hashing scheme can be used to address this problem very well. Cuckoo hashing [18] is a simple dynamic dictionary for resolving hash collisions in a hash table, with achieving worst case constant query time by providing several possible locations in the hash table for each key. Cuckoo hashing can efficiently support query services and has high utilization of hash tables.

In order to describe the detailed implementation of cuckoo hashing principle, Figure 1 shows an example of the standard cuckoo hashing. Each item has two possible positions and it can be inserted directly into the hash table if either of the candidate positions is empty, as the items x, y shown in Figure 1(a). Figure 1(b) illustrates the case of inserting a new item m when both of its candidate positions are occupied(which are filled by "x" and "y"). The item m will kick either item ("x" and "y" will be selected randomly) for getting a position, likewise, the repeated kicking-out operations will be accomplished by addressing an empty bucket. A rehashing operation is required if an endless loop appears. The case of $d = 2$ has a low utilization of hash tables, hence we introduce the case of $d > 2$ to improve the space utilization to meet the needs of the real-world near-duplicate video retrieval.

The standard cuckoo-driven locality-sensitive hashing design constructs the indexing structure [7], called NEST, to support real-time near-duplicate video retrieval and partially addresses the load imbalance and low space efficiency of hash tables. NEST leverages the LSH and cuckoo hashing to find similar items

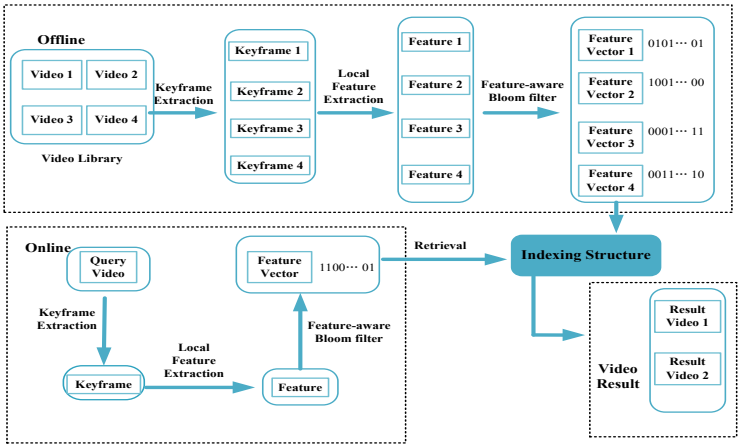


Fig. 2. The framework of our proposed FastVR

that are placed closely to obtain balanced load in hash tables. NEST uses random selection in cuckoo hashing among its kicking-out process. The random selection increases the time overhead, which is unsuitable in real time near-duplicate video retrieval.

3 The Design of FastVR

Our proposed framework is shown in Figure 2, which consists of two main parts. The first part depicts our compact feature representation, which includes the offline and online modules. Using the feature-aware Bloom filter, the local keypoints extracted from a frame are mapped into the feature vectors in Hamming space. The second part is the indexing structure using the semi-random holistic hashing. It uses the semi-random selection to accelerate the insertion of the cuckoo hashing.

The important problems to be addressed in this paper are *(1)How to construct the feature representation, (2)How to build an efficient indexing structure to accelerate query and keyframe matching.* In this section, we show the details of the two problems.

3.1 Compact Feature Representation

Our compact feature representation uses the feature-aware Bloom filter to construct the feature vectors as shown in Figure 3. Unlike the conventional hash function, the feature-aware Bloom filter is a Bloom filter which using LSH as hash function. The feature-aware Bloom filter is used to handle the hundreds or even thousands keypoints in a frame. A feature vector produced by the feature-aware Bloom filter belongs to one keyframe, not all keyframes. In our work, we use SIFT to extract local keypoints of 128 dimensions from a keyframe. The numbers of local keypoints in a keyframe may be hundreds or even thousands, and the number of local keypoints between keyframes are different.

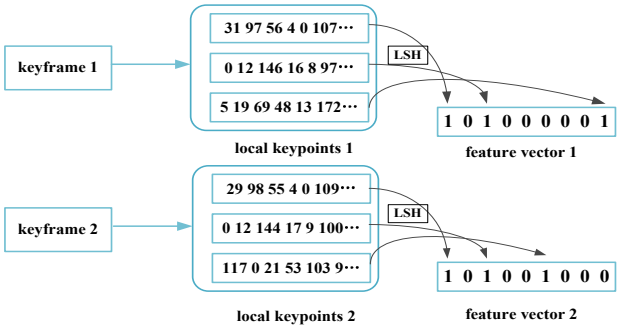


Fig. 3. Feature-aware compact representation

In order to adapt to the real-time manner, the feature-aware Bloom filter uses LSH hash functions that they can map the similar local keypoints to the same position. In the example denoted in Figure 3, we use the first three local keypoints to present the problems. The first two local keypoints of the keyframes are similar, and it is confirmed by the expressions in the local keypoints in Figure 3. Considering the first three dimensions of the local keypoints, the first local keypoints in the two keyframes are "$31, 97, 56$" and "$29, 98, 55$" and the corresponding values("31" and "29", "97" and "98", "56" and "55") are similar to each other respectively. Hence we use LSH hash functions to get the same hash values from the similar local keypoints. We introduce the feature vectors, that all the local keypoints in a frame can be represented by the resulting feature vectors thus the similarity in videos will be transformed into the same bit in feature vectors, that the first bit and the third bit in the two feature vectors are set "1". However, the third local keypoints in the two keyframes are not similar so the values obtained by LSH hash functions are different. Hence the feature vectors hashed by LSH hash functions are efficient to represent the similarity of the keyframes.

The advantage of compact feature representation is that the local keypoints of a frame are mapped to a feature vector, and then a keyframe corresponds to a feature vector. In our FastVR, the indexing structure directly organizes the feature vectors as items. This indexing structure can significantly reduce the storage space and can be more suitable in the large scale near-duplicate videos. Moreover, the keyframe matching in Hamming space is faster than in Euclidean space, hence the compact feature representation can significantly improve the performance of keyframe matching.

3.2 The Semi-random Holistic Hashing

Our FastVR combines the LSH and cuckoo hashing to construct the indexing structure and we denote MaxLoop as the maximum kicking-out count, and the kicking-out count is initialized to 0. Then we optimize the efficiency of kicking-out process in cuckoo hashing. First, when the kicking-out count is under MaxLoop/2, we use the random selection in cuckoo hashing to address the

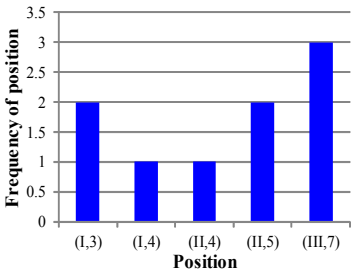
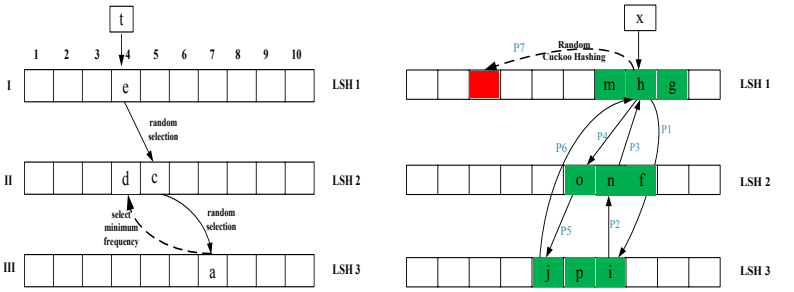


Fig. 4. The frequency of positions occurrence in a kicking-out path

kicking-out process. Meanwhile, we record the count of position occurrences. Second, when the kicking-out count is greater than MaxLoop/2 and under MaxLoop, we do not use the random selection to select the item to be moved if the potential positions are all occupied for current item. We pick the minimum frequency of the potential positions of the current item for the next "kicking out". Third, our indexing structure uses a random cuckoo hashing in the last step, i.e. the MaxLoop step, to jump out the similar group.

For example, we assume that the size of 3 hash tables, I, II and III, is 10, and we use (II, 7) to represent the 7th position in Table II. The MaxLoop is 5. The count of position occurrence is added to 1 when the position is one of the candidate positions in a kicking-out path. Figure 5(a) presents the kicking-out process in the indexing structure. Suppose item t will be inserted into the indexing structure, and its candidate positions are all occupied. Then the kick-out operations have experienced the path $e \rightarrow c \rightarrow a$, and the candidate positions of e, c, a are [(I,4)(II,5)(III,7)], [(I,3),(II,5),(III,7)], [(I,3),(II,4),(III,7)] respectively. Hence, the count of position occurrences has a statistic in Figure 4. Until now, the kicking count is 2, and the kicking operations use the random selection. The next step's kicking count is 3 > MaxLoop/2, and we will select the minimum frequency of the candidate positions of item a. From Figure 4, the count of position occurrences of a are 2,1 and 3, and the a will select the position (II,4) as the next position to kick out. Then the item a will kick the item d in position (II,4), and so on.



(a) The kicking-out process in the indexing structure.

(b) The random cuckoo hashing in last step.

Fig. 5. The kicking operations in semi-random selection

Figure 5(b) presents the random cuckoo hashing in the third step. The item x will be inserted into the hashing structure. We assume that all the items are similar items. The candidate positions of x and the adjacent positions are all occupied. Then the position which item h occupied is selected to start the kicking process in cuckoo hashing schemes. P1 means the first kicking operation in the procedure of inserting the item x. The kicking path from P1 to P6 has two circles and the last item may kick another item. The reason is that all the items are the similar items which calculated through the LSH, and all the items form a similar group. Then the items will kick each other in the similar group, and this insert operation will fail. The cuckoo hashing can only partly address the unbalanced load in the LSH based hash table. As to the problem of circle kicking in the similar group, we use the random cuckoo hashing to jump out the similar group to further address this problem.

The semi-random selection is used to optimize the kicking-out path through the history information of the kicking-out position. The random cuckoo hashing in the last step is used to jump out the similar group. Our indexing structure uses LSH to classify the near-duplicate items. Then we use the semi-random selection and the random cuckoo hashing in last step to accelerate the insertion process. It can significantly improve the utilization rate of hash tables and speed up the insertion.

4 Performance Evaluation

This Section presents the experimental results in a real cloud system in terms of multiple evaluation metrics.

4.1 Experiment Setup

We implemented a FastVR prototype on a 128-node cluster. Each node has a quad-core CPU running at 2.4GHz, with a 16GB RAM, a 500GB 7200RPM hard disk and a Gigabit network interface card. To drive the FastVR prototype evaluation, we use a real and large dataset collected from the cloud. Initially, the video dataset is randomly distributed among the nodes.

Evaluation Workload: Real Video Datasets. We collect real and openly assessable videos from the popular campus networks of universities. In order to faithfully demonstrate the property of real-world video datasets, we set certain temporal and spatial constraints on the collection. The temporal constraint defines the uploading interval to be between Sep. 30, 2012 and Oct.7, 2012, a week-long holiday season.

The spatial constraint confines the locations to the Chinese cities of Wuhan and Shanghai, with each having its own unique and popular landmarks and sceneries. While Wuhan has 10 landmarks, Shanghai has 20. We only collect videos that contain these representative landmarks, which facilitate a meaningful evaluation. The collected video dataset ultimately contains 50 thousand videos that amount to more than 2.5TB in storage size. The key characteristics of the video dataset are summarized in Table 1.

Table 1. The Properties of Collected Video Sets

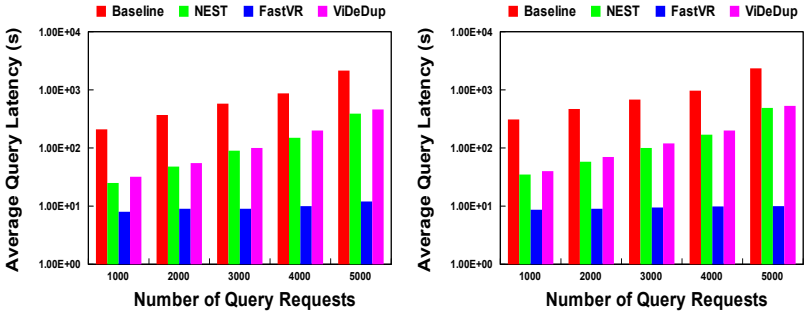
Dataset Name	No. Videos	Total Size	No. Landmarks
Wuhan	21.2 thousand	1.34 TB	10
Shanghai	28.8 thousand	1.16 TB	20

Evaluation Comparisons and Parameters. To evaluate the efficiency of our method, we compare the performance of FastVR, ViDeDup, NEST, and the baseline approach. For meaningful and fair comparisons, we mainly examine the performance of their query functionality. Since there are no complete open-source code of ViDeDup, we choose to re-implement it. ViDeDup [11] proposes a framework for video de-duplication based on an application-level view of redundancy. We implement its main components, including video signature generation, video segmentation, video sequence comparison, clustering, centroid selection and video segment indexing and referencing. Moreover, NEST [7] is the standard cuckoo-driven locality sensitive hashing scheme. The baseline approach is the traditional LSH without cuckoo hashing. All above functionalities are implemented in the Linux environment. In addition, we evaluate different parameter settings to obtain high space- and time-efficiency. By analyzing the actual experimental results, we use d=10, k=8, ω=4, and Δ=3 to obtain higher query accuracy and smaller storage space.

4.2 Results and Analysis

The evaluation metrics include the query latency and space overhead.

Query Latency. Figure 6 shows the average query latency. We examine the query performance as a function of the number of query requests from 1000 to 5000 with an increment of 1000. The latency of Baseline, at 30min on average, is almost one order of magnitude better than 3.6min in NEST and 5.2min in ViDeDup. NEST makes use of cuckoo-driven hashing to execute flat addressing and quickly identify the queried results. ViDeDup reduces the query latency due to its similarity detection in the application level, and however, its 2-phase video



(a) Wuhan Dataset. (b) Shanghai Dataset.

Fig. 6. The average query latency

comparison exacerbates the query performance due to the increase of operation complexity.

FastVR requires the smallest latency, around 10 seconds. The advantage of FastVR is to leverage the cuckoo hashing to obtain the load balance on hash tables. The hash tables based on FastVR save the storage space and can be effectively loaded into the high-speed memory, thus alleviating frequent access to the low-speed hard disks. FastVR hence mitigates the I/O costs to obtain low the query latency. Moreover, compared with NEST, the kicking-out path of cuckoo hashing in FastVR is optimized by semi-random selection to accelerate the insertion in the hash tables. This acceleration can further reduce the query latency to make FastVR more suitable for the real-time query in near-duplicate video retrieval.

Space Overhead. Table 2 summarizes the space overheads of Baseline, NEST, ViDeDup and FastVR, normalized to that of Baseline. By reducing the number of dimensions to be processed and the use of load-balanced design, NEST achieves a space saving of about 20% from Baseline. To support application-aware deduplication, ViDeDup leverages similarity detection and trades CPU for storage, thus obtaining about 30% space savings.

Table 2. Space Overhead normalized to Baseline

Video Datasets	Baseline	NEST	FastVR	ViDeDup
Wuhan	1	0.85	0.11	0.72
Shanghai	1	0.77	0.09	0.67

FastVR makes use of the semi-random selection to significantly improve the kicking-out path of cuckoo hashing, thus obtaining the space savings. FastVR requires about 10% space overhead and is able to store more index information into the main memory, which is helpful to significantly improve the query performance.

5 Conclusion

In this paper, we proposed FastVR, a compact feature representation and an efficient indexing structure, to obtain the real-time query performance. FastVR can support fast query and consume low storage overhead in near-duplicate video retrieval. Our compact feature representation uses the feature-aware Bloom filter, and the local keypoints (extracted from a keyframe) are mapped to a feature vector in the Hamming space. The storage capacity in the feature vector is significantly decreased, compared with local keypoints. The optimized indexing structure that combines LSH and Cuckoo Hashing is more suitable to meet the needs of the real near-duplicate video retrieval. By using a real-world video dataset, the experimental results demonstrate the efficiency and efficacy of our proposed FastVR in terms of query latency and accuracy, and space overhead.

Acknowledgment. This work was supported in part by NSFC 61173043; National Basic Research 973 Program of China 2011CB302301; NSFC 61025008, 61232004; The Seed Project of Wuhan National Laboratory for Optoelectronics (WNLO).

References

1. YouTube Statistics (2014), http://www.youtube.com/yt/press/
2. Böhm, C., Berchtold, S., Keim, D.A.: Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. ACM Computing Surveys (CSUR) 33(3), 322–373 (2001)
3. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proc. Annual Symposium on Computational Geometry. ACM (2004)
4. Douze, M., Gaidon, A., Jegou, H., Marszałek, M., Schmid, C., et al.: Inria-lears video copy detection system. In: TREC Video Retrieval Evaluation, TRECVID Workshop (2008)
5. Gantz, J., Reinsel, D.: The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East. In: International Data Corporation (IDC) iView (December 2012)
6. Hua, Y., Jiang, H., Zhu, Y., Feng, D., Tian, L.: Smartstore: A new metadata organization paradigm with semantic-awareness for next-generation file systems. In: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. IEEE (2009)
7. Hua, Y., Xiao, B., Liu, X.: Nest: Locality-aware approximate query service for cloud computing. In: Proceedings of IEEE International Conference on Computer Communications, INFOCOM (2013)
8. Huang, Z., Shen, H.T., Shao, J., Cui, B., Zhou, X.: Practical online near-duplicate subsequence detection for continuous video streams. IEEE Transactions on Multimedia 12(5), 386–398 (2010)
9. Huang, Z., Shen, H.T., Shao, J., Zhou, X., Cui, B.: Bounded coordinate system indexing for real-time video clip search. ACM Transactions on Information Systems (TOIS) 27(3), 17 (2009)
10. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proc. ACM Symposium on Theory Of computing. ACM (1998)
11. Katiyar, A., Weissman, J.: ViDeDup: An application-aware framework for video de-duplication. In: Proceedings of the 3rd USENIX Conference on Hot Topics in Storage and File Systems, HotStorage (2011)
12. Ke, Y., Sukthankar, R.: Pca-sift: A more distinctive representation for local image descriptors. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004, vol. 2, pp. II–506. IEEE (2004)
13. Law-To, J., Chen, L., Joly, A., Laptev, I., Buisson, O., Gouet-Brunet, V., Boujemaa, N., Stentiford, F.: Video copy detection: a comparative study. In: Proceedings of the 6th ACM International Conference on Image and Video Retrieval, pp. 371–378. ACM Press (2007)
14. Leung, A.W., Shao, M., Bisson, T., Pasupathy, S., Miller, E.L.: Spyglass: Fast, scalable metadata search for large-scale storage systems. In: Proceedings of the Conference on File and Storage Technologies (FAST), pp. 153–166 (2009)

15. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
16. Lu, G., Nam, Y.J., Du, D.H.: BloomStore: Bloom-filter based memory-efficient key-value store for indexing of data deduplication on flash. In: Proc. IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST). IEEE (2012)
17. Lv, Q., Josephson, W., Wang, Z., Charikar, M., Li, K.: Multi-probe lsh: Efficient indexing for high-dimensional similarity search. In: Proceedings of the 33rd International Conference on Very Large Data Bases, pp. 950–961. VLDB Endowment (2007)
18. Pagh, R., Rodler, F.F.: Cuckoo hashing. Journal of Algorithms 51(2), 122–144 (2004)
19. Park, D., Du, D.H.: Hot data identification for flash-based storage systems using multiple Bloom filters. In: Proc. IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST). IEEE (2011)
20. Poullot, S., Crucianu, M., Buisson, O.: Scalable mining of large video databases using copy detection. In: Proceedings of the 16th ACM International Conference on Multimedia, pp. 61–70. ACM (2008)
21. Shang, L., Yang, L., Wang, F., Chan, K.P., Hua, X.S.: Real-time large scale near-duplicate web video retrieval. In: Proceedings of the International Conference on Multimedia, pp. 531–540. ACM (2010)
22. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV), pp. 1470–1477. IEEE (2003)
23. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Advances in Neural Information Processing Systems, pp. 1753–1760 (2008)
24. Wu, X., Hauptmann, A.G., Ngo, C.W.: Practical elimination of near-duplicates from web video search. In: Proceedings of the 15th International Conference on Multimedia, pp. 218–227. ACM (2007)
25. Zhan, D., Jiang, H., Seth, S.C.: Exploiting set-level non-uniformity of capacity demand to enhance cmp cooperative caching. In: Proceedings of the 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS), pp. 1–10. IEEE (2010)
26. Zhan, D., Jiang, H., Seth, S.C.: Stem: Spatiotemporal management of capacity for intra-core last level caches. In: Proceedings of the 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 163–174. IEEE (2010)
27. Zhan, D., Jiang, H., Seth, S.C.: Locality & utility co-optimization for practical capacity management of shared last level caches. In: Proceedings of the 26th ACM International Conference on Supercomputing, pp. 279–290. ACM (2012)
28. Zhao, W.L., Tan, S., Ngo, C.W.: Large-scale near-duplicate web video search: Challenge and opportunity. In: Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), pp. 1624–1627. IEEE (2009)