Multimed Tools Appl
DOI 10.1007/s11042-014-2185-x

GPU-based MapReduce for large-scale near-duplicate video retrieval

Hanli Wang · Fengkuangtian Zhu · Bo Xiao · Lei Wang · Yu-Gang Jiang

Received: 7 February 2014 / Revised: 6 July 2014 / Accepted: 7 July 2014

Abstract With the exponential growth of multimedia data, people are overwhelmed with massive amount of online videos, of which Near-Duplicate Videos (NDVs) occupy a large portion. In this paper, we present a novel framework for NDV retrieval, which explores the parallel power of two promising techniques: Graphics Processing Unit (GPU) and MapReduce. With the power of the proposed framework, various key algorithms in the field of computer vision, such as K-Means clustering, bag of features, inverted file index with hamming embedding and weak geometric consistency, are applied to NDV retrieval. Experimental results on the benchmark CC_WEB_VIDEO NDV dataset demonstrate that the proposed framework can significantly speed up processing huge amounts of video repositories.

Keywords Near-duplicate video retrieval · MapReduce · GPU · Hadoop

1 Introduction

With the development of social network and mobile multimedia techniques, the amount of video data has grown rapidly. According to http://www.comscore.com, a leading company in measuring the digital world, 85 % of the U.S. Internet users (or about 188 million Americans) viewed online videos and these users viewed 49.8 billion online videos in December 2012 alone. This results in a large percentage of Near-Duplicate Videos (NDVs), and according to [31], the ratio of NDVs even reaches a surprising number of 93 % for some of the user queries searched from the Internet. To eliminate the redundancy and perform content analysis from huge amounts of videos, it is very desired to develop web-scale

H. Wang (✉) · F. Zhu · B. Xiao · L. Wang
Department of Computer Science & Technology and Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai, 200092, People's Republic of China
e-mail: hanliwang@tongji.edu.cn

Y.-G. Jiang
School of Computer Science, Fudan University, Shanghai, 201203, People's Republic of China

near-duplicate video retrieval systems. To this aim, we propose a novel framework for NDV retrieval, which utilizes two promising parallel computing technologies: MapReduce and GPU.

MapReduce [3] is a parallel programming model proposed by Google, which is very capable of processing large-scale data and well realized in Hadoop [30], the most representative open source cloud computing implementation. In recent years, much attention is paid on MapReduce owing to its simplicity and powerfulness to solve time-consuming problems. There are mainly two kinds of functions in MapReduce: *map* and *reduce*, which are respectively responsible for handling the input data and task distribution as well as synthesizing the final output. However, the traditional MapReduce model becomes gradually unsuitable when dealing with large-scale video data for the computer vision tasks. Several modules need to be refined to meet the requirement of NDV retrieval, which is to be discussed later in this work.

As far as GPU [20] is concerned, it is initially designed to accelerate computations for computer graphics, and it is now becoming increasingly common to employ GPU as a modified form of stream processor. For high-throughput computations, GPU devotes proportionally more transistors to arithmetic logic units and less to caches as compared with CPU. GPU-based high performance computers are applied to many types of tough parallel tasks and are playing a more and more significant role in large-scale modeling.

In this work, a GPU-based MapReduce framework, named as Multimedia and Intelligent Computing Cluster (or MICC in short), is proposed for large-scale NDV retrieval. MICC is currently implemented with a 12-node computer cluster configured with MapReduce and multiple GPUs on each node to strengthen the power of MapReduce. The contribution of this work is that the proposed MICC framework provides a promising solution for large-scale multimedia data processing and exploits jointly the computing power of CPUs and GPUs to speed up high-throughput computing tasks in the field of computer vision. Experimental results on the benchmark CC_WEB_VIDEO dataset demonstrate that the proposed MICC framework is able to achieve a wonderful speedup in processing large amount of video data.

The remaining of this paper is organized as follows. Section 2 reviews the related work about NDV retrieval, MapReduce and GPU. The proposed MICC framework is elaborated in Section 3. In Section 4, the implementation details of MICC for NDV retrieval are introduced. Section 5 presents the experimental results. Finally, the paper is concluded in Section 6.

2 Related work

NDV retrieval has attracted a lot of interests in recent years, with a number of approaches proposed which can be mainly divided into two categories: global or local feature based algorithms. Global feature based algorithms usually regard color and spatial characteristics as the feature. In [4], the HSV color model is used to represent color histogram signatures of keyframes. In [22], global spatial-temporal features are applied to construct a real-time large-scale NDV retrieval system. In [10], a coarse-to-fine ordinal signature is developed as the video global feature. Although the global feature based algorithms achieve better performances in rapid retrieval, as an exchange, it is difficult to deal with various challenging distortions, *e.g.*, variation in intensity and color.

There is more attention to extracting local features to represent videos for NDV retrieval. Douze et al. [5] construct a NDV retrieval system with local features being employed in the

Bag of Feature (BoF) [23] model. In [5], the locally Hessian-Affine detector [17] and Scale Invariant Feature Transform (SIFT) descriptor [16] are used for local feature detection and description. In addition, the Hamming Embedding (HE) and Weak Geometric Consistency (WGC) techniques [11] are utilized for improving NDV retrieval accuracy. Similar to [5], Zhao et al. [33] apply the techniques of BoF, HE and WGC to retrieve NDVs for annotation of web videos. In [24], Song et al. introduce a multiple feature hashing algorithm for real-time large-scale NDV retrieval. Interested readers are referred to [15] for a more complete research survey in NDV retrieval.

Regarding MapReduce, several researches are presented to address computer vision problems. In [32], a MapReduce implementation is realized for large-scale semantic concept modeling. A 60-node cluster is built with MapReduce for landmark classification [14]. In [1], a MapReduce-based similarity search system is developed on a test collection of more than 50 millions of images. The MapReduce framework is applied to multimedia data mining including K-Means clustering and background substraction [29]. In [28], MapReduce is utilized for image classification, NDV retrieval and video event detection. An efficient high-dimensional indexing algorithm with MapReduce is proposed in [18] to index and search 100M images.

On the other hand, there are scientific applications and algorithms focusing on the GPU realization. One of the popular algorithms implemented by GPU is K-Means clustering [21]. In [27], the real-time performance of visual categorization is significantly improved with GPU-based implementations. Recently, GPU clusters are realized to address large-scale computing problems. Karantasis et al. [13] boost K-Means clustering with a GPU cluster by using shared memory abstraction. In [2], an online image search engine is designed with a GPU cluster. In [9] and [25], MapReduce is implemented with GPU to improve the real-time performance for a number of traditional applications such as string matching, matrix multiplication, page view counting, etc. However, to the best of our knowledge, the joint optimization of MapReduce and GPU is not fully investigated for large-scale multimedia mining applications.

3 Proposed MICC framework

In the following, we describe the details of the proposed MICC framework with multiple CPUs+GPUs on MapReduce. In order to fit multimedia data mining tasks, the traditional MapReduce framework is refined by modifying and adding several modules.

3.1 MICC overview

The overall structure of the proposed MICC framework is illustrated in Fig. 1. Currently, a 12-node computer cluster is configured to actualize MICC as the hardware infrastructure for distributed computing and storage. Above the 'Cluster & Distributed System' is the Hadoop Distributed File System (HDFS) [8] and MapReduce layer. The functionality of HDFS is to automatically manage all of the data and preserve replications of the files to backup all multimedia data to avoid data loss. Hadoop [30] as the most representative MapReduce implementation is employed herein since it is skilled in automatic task management, inter machine communication and fault tolerance. Another layer is CUDA [19] which is a popular parallel computing platform and programming model for GPU invented by NVIDIA. CUDA is employed to enhance the computing performance by utilizing the power of GPU. The top layer is the core of the proposed

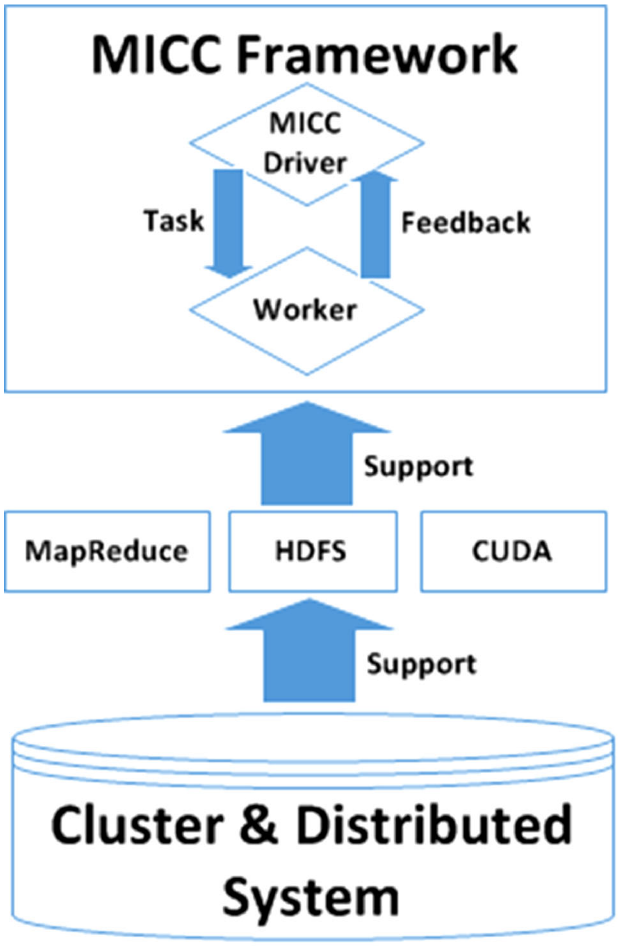


Fig. 1 Overview of the proposed MICC framework

MICC framework, including the MICC Driver and Worker which will be introduced next.

3.2 MICC driver

When dealing with large-scale video data for NDV retrieval, such as millions of video images or billions of features, the traditional MapReduce framework becomes a little bit incapable. We observe the following three aspects for further improvement.

1. *Data loading and transmission.*
   In the traditional MapReduce framework, the input dataset of a job is divided into several pieces when the job is launched based on the principles of data locality and load balancing [3]. Each piece of the dataset will be handled by a *map* task and read by the input module. In Hadoop, MapReduce is implemented with Java, but GPU is usually

realized with C/C++ for the best computational efficiency. Moreover, most of the computer vision algorithms are desired to be achieved with C/C++ owing to the excellent high-performance computing power of C/C++ programming. As a result, redundant data loading and transmission are required between the input module which is implemented with Java and the *map* module realized with C/C++. A similar problem is also with the data output. Obviously, the data transmission between *map* and input/output modules is a time waste and unnecessary.

2. *Partition and sort procedures.*
   In the traditional MapReduce framework, intermediate data outputted from *map* tasks is processed by the *partition* and *sort* procedures at first. These two procedures are designed and used to simplify the *reduce* process and thus relieve the workload. However, considering the characteristics of multimedia applications, these two procedures are of little use. Therefore, it is desirable to remove these unnecessary procedures to reduce computations.
3. *Computing power.*
   In fact, a computer cluster with only CPUs (which usually carry a limited number of processing cores) is generally not powerful enough in handling large-scale multimedia data as compared with GPU clusters, since a GPU is typically embedded with thousands of processing cores and takes much higher memory bandwidth, especially applicable in the field of data-intensive computations with NDV retrieval as a good representative.

In order to address the problems mentioned above, the MICC Driver is introduced in the proposed MICC framework as illustrated in Fig. 2. In the MICC Driver, the data is not directly loaded or transmitted via the input module; instead, small-size Filelist splits are transmitted to the *map* modules. To reduce the unnecessary transmission and improve the throughput capacity of the whole system, we utilize HDFS native shared library to read data from HDFS directly according to the Filelist received before. The huge amount of intermediate data created by the Worker (which will be introduced next) will be written to HDFS in the same manner. This method significantly reduces the amount of memory required by the whole system and improves the performance. Therefore, the *reduce* module is much simplified. As aforementioned, the *partition* and *sort* modules are removed from MICC Driver to reduce unnecessary computations.

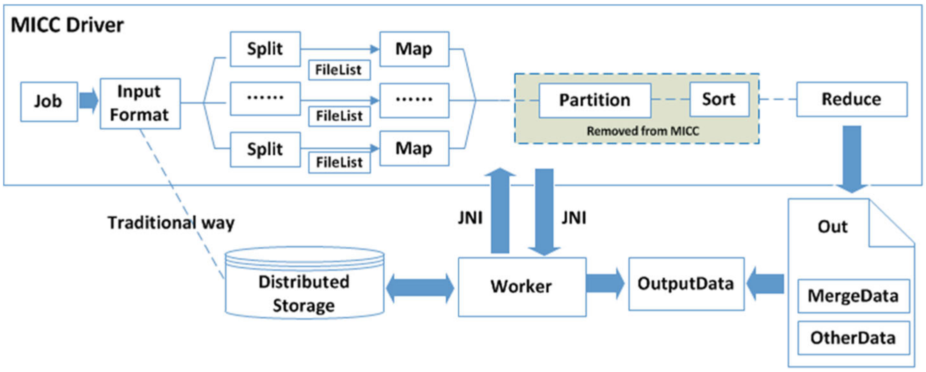


Fig. 2 MICC flowchart

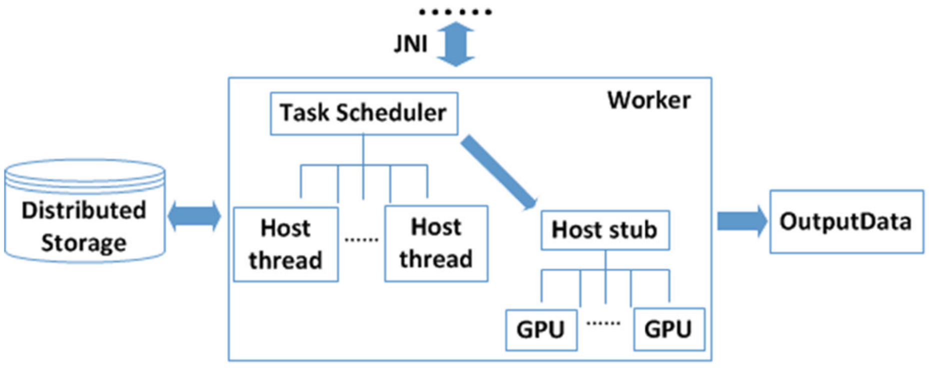


Fig. 3 Illustration of worker

3.3 Worker

Worker is another key part of the proposed MICC framework as illustrated in Fig. 3. It is responsible for all of the time-consuming tasks dispatched from MICC Driver, including feature extraction, K-Means clustering and so on. It is also well designed to be capable of extending to other multimedia data mining applications besides NDV retrieval.

In order to obtain high performances, Worker fully combines the power of CPU and GPU, and balances the computational load between them in a joint automatic manner. Currently, Worker is realized with C/C++ programming language due to the following reasons. First, as compared with Java, the optimized C/C++ implementation is more efficient due to its native property. Java has to translate the code into the Java virtual machine language and package the code to adapt running on different platforms (*e.g.*, Windows or Linux). In addition, Java usually suffers from the larger memory usage than C/C++. Second, the GPU implementation is developed on CUDA-C environment, in which programming in C/C++ is a prerequisite. Third, in order to reduce unnecessary data transfer operations between Java (which is deployed in Hadoop) and C/C++, the HDFS native shared library is employed herein which provides a C/C++ interface to HDFS. In the current implementation, the interaction between MICC Driver and Worker is achieved by JNI, which is an effective method for information switching between Java and C/C++ codes.

4 MICC implementation on NDV retrieval

In this section, we will present the details of the proposed MICC framework for NDV retrieval, including keyframe extraction, feature extraction, K-Means clustering, inverted file index generation, Hamming Embedding (HE) and Weak Geometric Consistency (WGC), which are illustrated in Fig. 4. As aforementioned, the merits of the proposed MICC framework lie in its ability of handling large-scale multimedia data and the power of rapid processing.

As shown in Fig. 4, keyframes are firstly extracted from video streams and uploaded to HDFS for keyframe dataset construction. Then, local feature detection and description are performed on keyframes to obtain feature vectors. Then, the K-Means clustering is

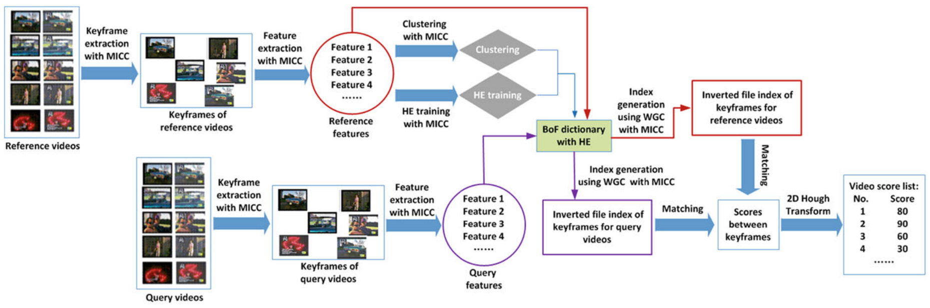

Reference videos
Keyframe extraction with MICC
Keyframes of reference videos
Feature extraction with MICC
Feature 1
Feature 2
Feature 3
Feature 4
......
Reference features
Clustering with MICC
Clustering
HE training
HE training with MICC
BoF dictionary with HE
Index generation using WGC with MICC
Inverted file index of keyframes for reference videos
Matching
Query videos
Keyframe extraction with MICC
Keyframes of query videos
Feature extraction with MICC
Query features
Index generation using WGC with MICC
Inverted file index of keyframes for query videos
Matching
Scores between keyframes
2D Hough Transform
Video score list:
No. Score
1 80
2 90
3 60
4 30
......

Fig. 4 Illustration of NDV retrieval with MICC

carried out on the feature vector points to generate visual words according to the BoF [23] model, followed by the generation of BoF-based feature vectors. Due to the high-dimensional space, visual words are usually sparse which can be reorganized by the inverted file index for more efficient computing. However, the BoF representation fails to reveal the spatial and geometry information and only preserves the accumulated count of each visual words. Therefore, the HE and WGC techniques [11] are also considered by MICC.

Considering query on the other hand, the keyframes of a query video are firstly extracted followed by feature extraction. Based on the visual dictionary trained with BoF and HE, new features will be produced for query keyframes and then the corresponding inverted file indices are generated optionally with the aid of WGC. Next, similarity scores are computed from the inverted file indices between query and reference keyframes. Finally, the similarity scores between the query video and reference videos are produced by aggregating the scores of each keyframes using the two-dimensional Hough Transform which is employed in [5] and [33].

4.1 Keyframe extraction

For video representation, keyframes are extracted by uniform sampling, *e.g.*, to extract a representative frame per second. In fact, there is another approach to extracting keyframes from each video shot. However, as discussed in [22], the shot-based keyframe extraction method produces less frames than uniform sampling and breaks the temporal continuity of videos resulting in loss of video information. For the current MICC implementation, we feed videos as the input to computer nodes, and extract a keyframe per second in the corresponding *map* task, then the output of the keyframes are saved into HDFS.

4.2 Feature extraction

Feature extraction is one of the most important procedures for most multimedia data mining tasks. In this work, the toolkit of the Hessian-Affine detector [17] which is available at [26] is utilized to detect interest points. Because only the executable program is offered at [26], MICC performs only CPU-based realization for this task. In our implementation, keyframes are passed to corresponding computer nodes, and *map* functions receive the data and generate a new copy of the data in the local disk. Then, the feature extraction proceeds in parallel without interfering with other nodes. The executable programs are invoked directly and natively to process the local temporary data and produce the desirable features.

4.3 K-Means clustering

K-Means clustering is used in the current implementation to produce visual vocabularies for BoF and inverted file index generation. The clustering Worker for this task is implemented as follows with Fig. 5 illustrating the procedure of K-Means clustering with the proposed MICC framework.

First, the entire feature vectors which are the input to K-Means clustering are split into pieces by the MICC Driver. Each piece is processed by one *map* task. The proposed framework can run a number of tasks simultaneously across the computer nodes with the number of tasks up to the *map* task capacity pre-configured. That is the coarse-grained parallelization in the proposed system. After being invoked by the MICC Driver, the clustering Worker gets the initial centroid and the piece of feature vectors from HDFS and loads them to GPU memory if available. This process is optimized by the DistributedCache technology by which the initial centroid is cached to the local file system on each node when the job is launched. The calculation of locating the nearest centroid for each feature is executed by a single GPU thread. Therefore, all features of this piece

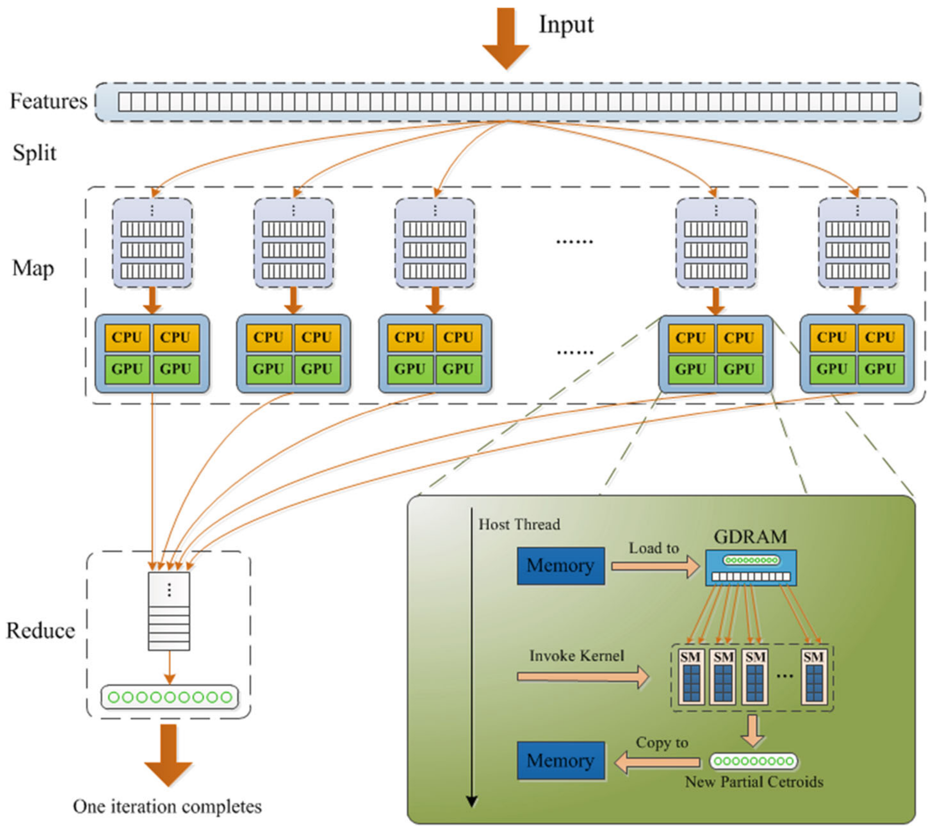


Fig. 5 Illustration of K-Means clustering with MICC

Algorithm 1 K-Means Clustering with MICC.

```
Input: All feature vectors
Output: Cluster centroids
 1: submit K-Means clustering job to MICC
 2: split job into $m$ tasks based on the input parameters of
    the total size of feature vectors and the size of each piece
 3: do
 4:    parallel for $i = 0, \cdots, m - 1$ on $N$ nodes do
 5:       cache current centroids on each node
 6:       task tracker on node $k$ receives $task_i$, clustering
          Worker invoked by MICC Driver
 7:       download feature vectors with size $L$ from HDFS for
          current task
 8:       if accessing GPU succeed then
 9:          load current centroids and feature vectors from
             host memory to device memory
10:          parallel for $j = 0, \cdots, L - 1$ on $L$ features do
11:             find the nearest centroid for feature vector $j$,
                executed by a single GPU thread
12:          end parallel for
13:       else
14:          parallel for $j = 0, \cdots, L - 1$ on $L$ features do
15:             find the nearest centroid for feature vector $j$
16:          end parallel for
17:       end if
18:       calculate membership and update partial centroids
19:       clustering Worker put new partial centroids to HDFS
20:    end parallel for
21:    reduce task gets all partial centroids from HDFS and
       calculates new centroids
22: while difference of current centroids and new centroids
    is larger than threshold and iteration loop is not over do
23: return centroids
```

can be processed in parallel logically. Actually, if the number of features in a piece is larger than the number of threads executed simultaneously by the GPU, the calculations will be executed in batch mode. Then, the partial centroid will be updated according to the new membership of the features. After all clustering Workers complete the tasks, the new centroids obtained in the current iteration are calculated by combining all partial centroids by the *reduce* task. The process of K-Means clustering is also detailed in Algorithm 1.

4.4 Inverted file index generation

The Inverted File Index (IFI) is a popular structure to represent the high-dimensional BoF vector owing to its inherent sparse property. IFI can prune the zero position and reduce both the memory of online video retrieval and the computations on matching BoF vectors, especially helpful in large-scale multimedia data applications. A typical IFI structure is composed of k lists of descriptor entries, each corresponding to a visual word with each entry containing the image id and other information if available (*e.g.*, binary signature, orientation and scale when HE and WGC are enabled, which will be discussed in the next subsection).

The implementation of IFI generation with MICC is similar to that of K-Means clustering, which can be achieved by the IFI *Worker*. First, all the feature vectors are split into several pieces and each piece will be handled by a *map* task as an individual job which is launched by the MICC Driver. The IFI Worker reads the related feature vectors and loads the visual dictionary from HDFS; alternatively, the DistributedCache technology can be employed to pre-cache the data, which is usually a better choice due to its efficiency. Then, for each feature vector, a brute-force searching strategy is used to locate the most similar visual word which can be processed in parallel logically by GPU when GPUs are available. If the number of feature vectors in a piece is larger than the number of threads executed simultaneously by the GPU, the calculations will be executed in batch mode. When all IFI Workers complete their jobs, all the inverted file indices produced by each *map* task are merged by the *reduce* task.

4.5 HE and WGC

HE and WGC [11] are two advanced techniques to improve image/video retrieval performances. For HE, the weighting based algorithm [12] on Hamming distance is employed in this work to improve the retrieval accuracy. Regarding WGC, it utilizes intrinsic scale and orientation information of interest points to filter out dissimilar matching and preserve similar matching based on geometric validation. Both HE and WGC can be utilized for IFI generation with a detailed introduction presented in [5]; specifically, the IFI structure is composed of k lists of descriptor entries, each corresponding to a visual word. Each entry contains the image id, the binary signature $b(x)$, the quantized dominant orientation $qa(x)$ and the quantized scale $qs(x)$, where x stands for a feature descriptor (or feature vector). The binary signature $b(x)$ is used for HE based calculation; the orientation $qa(x)$ and scale $qs(x)$ are utilized for WGC based computation.

During the IFI generation with the proposed MICC framework as mentioned above, after locating the most similar visual word for a feature vector x, the *map* task will calculate its binary signature $b(x)$ according to [11], which can be accelerated by GPU because the matrix multiplication involved in HE computation can be easily processed by GPU. Specifically, in this work, HE performs matrix multiplication by transforming a 1024-byte feature descriptor (*i.e.*, 128×8, 128-dimension SIFT descriptor with double data type) to a 64-bit binary signature. As for WGC, the orientation and scale information are stored per feature descriptor in the IFI structure which are processed by the *map* task. Since only simple operations are involved, GPU is not employed for inserting the orientation and scale information into the IFI structure. The process of IFI generation with the proposed MICC framework when both HE and WGC are enabled is illustrated in Algorithm 2.

Similarly, the training tasks for HE are delegated to the HE *Worker*, and the image features as well as the visual vocabulary words in terms of centroids are loaded from HDFS. The matrix multiplication operations required by HE can be dramatically parallelized by GPU in each *map* task. Then, the intermediate results outputted from *map* tasks are aggregated to get the final result in the *reduce* task.

5 Experimental results

In order to evaluate the proposed MICC framework on NDV retrieval, the benchmark CC_WEB_VIDEO [31] dataset is employed for experiments, which consists of 13,137 videos downloaded from Google Video, Yahoo! Video and YouTube. These videos may differ in the photometric variations, slightly editing with logo insertion, adding borders around frames, superposition of overlay text, etc. The CC_WEB_VIDEO videos are grouped into 24 topic sets for video query. The information of the CC_WEB_VIDEO dataset including the number of extracted keyframes and feature descriptors is given in Table 1.

Algorithm 2 IFI Generation with MICC when HE and WGC are enabled.

```
Input:  All feature vectors
Output:  IFI structure
 1: submit IFI generation job to MICC
 2: split job into $m$ tasks based on the input parameters of
    the total size of feature vectors and the size of each piece
 3: parallel for $i = 0, \cdots, m-1$ on $N$ nodes do
 4:     task tracker on node $k$ receives $task_i$, IFI Worker
        invoked by MICC Driver
 5:     download feature vectors with size $L$ from HDFS for
        current task
 6:     if accessing GPU succeed then
 7:         load feature vectors from host memory to device
            memory
 8:         parallel for $j = 0, \cdots, L-1$ on $L$ features do
 9:             find the nearest centroid for feature vector $j$,
                executed by a single GPU thread
10:             generate binary signature $b(j)$ for feature vector
                $j$ by matrix multiplication, executed by a single
                GPU thread
11:             insert binary signature $b(j)$, orientation $qa(j)$ and
                scale $qs(j)$ of WGC to generate IFI structure $I(ij)$
12:         end parallel for
13:     else
14:         parallel for $j = 0, \cdots, L-1$ on $L$ features do
15:             find the nearest centroid for feature vector $j$
16:             generate binary signature $b(j)$ for feature vector
                $j$ by matrix multiplication
17:             insert binary signature $b(j)$, orientation $qa(j)$ and
                scale $qs(j)$ of WGC to generate IFI structure $I(ij)$
18:         end parallel for
19:     end if
20: end parallel for
21: $reduce$ task merges IFI structure $I(ij)$ produced by each
    $map$ task until all $map$ tasks complete
22: return IFI structure
```

Table 1 CC_WEB_VIDEO dataset information

#videos	#images	#descriptors	descriptor size
13,137	1,975,688	868,606,544	575.1G

5.1 Experimental setup

5.1.1 MICC Configuration

The proposed MICC framework is implemented with a 12-node computer cluster equipped with multiple GPUs. The detailed information about these computer nodes is given in Table 2. Each node is equipped with 1 CPU and 2 GPUs, and gigabit-nics are used among them for communication. In addition, we select one machine with middle-level configuration to run a single version of all the following evaluations and it is not included in the implementation of MICC. The Java 1.7 version and Hadoop 1.0.4 version are set on all the nodes with the GNU/Linux Ubuntu 12.04 operating system.

5.1.2 Implementation details and notations

As mentioned above, the toolkit of [26] is used to extract SIFT descriptors [16] with the Hessian-Affine detector [17] from keyframes. The visual dictionary is generated from all of the extracted descriptor points using K-Means clustering. For NDV retrieval, a standard term-frequency weighting scheme and cosine distance are used to calculate the similarity of two keyframes by employing IFI to eliminate the redundant computations at the zero positions owing to the sparsity of the high-dimensional vector space. Each keypoint is further embedded with binary signatures, scale and orientation information when performing HE and WGC. When calculating the scores between all reference keyframes and query keyframes, the proposed framework employs the 2D Hough Transform which is used in [5] and [33] to aggregate all frame-level scores into video-level scores. All the intermediate and result files are saved as a Protocol Buffer (PB) [7] format which encodes structured data in an efficient and extensible format to minimize the disk storage and reduce the heavy burdens

Table 2 Configuration of MICC computer nodes

Hardware	Amount		
	2	6+1(single)	4
CPU	Intel Core i5-3450	Intel Core i5-3470	Intel Core i5-4430
CPU Cores	4	4	4
Memory	24 GB DDR3	24 GB DDR3	24 GB DDR3
GPU	GeForce GTX 660Ti $\times 2$	GeForce GTX 660 $\times 2$	GeForce GTX 760 $\times 2$
CUDA Cores	1344×2	960×2	1152×2
Network		1Gbps LAN	

on HDFS. Moreover, we evaluate the MICC framework under a number of test conditions with several notational terms defined in Table 3.

5.2 Computational performance

In this subsection, we analyze the computational performance of the proposed MICC framework on NDV retrieval, with the comparative results shown in Table 4. Three kinds of evaluation are performed in this work, including a single computer node with only one CPU core used (‘S’), a 12-node computer cluster with GPUs disabled (‘C’) and a 12-node computer cluster with multiple GPUs enabled (‘C+G’). Here, it is necessary to mention that when performing the evaluation under the ‘S’ condition, it is impossible to deal with such a huge amount of data with the current single machine configuration within a reasonable period. Therefore, we use a relatively small number of data to run different tasks under the ‘S’ condition, then estimate the total execution time considering the relation between the amounts of running data and entire data. This approximation is reasonable since the running time of these tasks including feature extraction (ImgProc), K-Means clustering (Clustering), HE training (HeTrain), IFI generation with HE and WGC (IfiGen) is proximately propotional to the amount of data being processed. The same strategy for running time estimation also applies to the ‘C’ condition in some cases as shown in Table 4.

From the results, it is obvious that the proposed MICC framework with ‘C+G’ is able to achieve dramatically significant speedup as compared with the ‘S’ and ‘C’ configurations for the tasks including Clustering, HeTrain and IfiGen. As for Clustering, HeTrain and

Table 3 Implementation notations

Evaluation condition (Eval.)	
S	version run on a single computer node with only one CPU core used
C	version run on MICC with GPUs disabled
C+G	version run on MICC with GPUs enabled
Performance criteria	
SR	speedup ratio compared with version S
mAP	mean Average Precision
NDV retrieval procedures (Procs.)	
EtrFrm	keyframe extraction
ImgProc	image processing and feature extraction
Clustering	K-Means clustering
HeTrain	HE training
IfiGen	IFI generation
Search	Online retrieval
Approach (Apch.)	
Baseline	evaluating without HE and WGC
HE	evaluating with HE enabled
WGC	evaluating with WGC enabled
HE+WGC	evaluating with both HE and WGC enabled

Table 4 Comparison of computational performance for offline training achieved by MICC with different sized visual dictionaries

		20k		200k		400k	
Procs.	Eval.	Time (s)	SR	Time (s)	SR	Time (s)	SR
EtrFrm	S	39,362	1.0	39,362	1.0	39,362	1.0
	C	18,256	2.2	18,256	2.2	18,256	2.2
ImgProc	S	≈1,092,836	1.0	≈1,092,836	1.0	≈1,092,836	1.0
	C	33,934	32.2	33,934	32.2	33,934	32.2
Clustering	S	≈21,506,699	1.0	≈214,815,724	1.0	≈430,081,780	1.0
	C	≈454,063	47.4	≈5,062,007	42.4	≈10,690,573	40.2
	C+G	17,907	1201.0	93,458	2298.5	164,826	2609.3
HeTrain	S	≈1,698,830	1.0	≈17,265,120	1.0	≈35,234,024	1.0
	C	59,152	28.7	≈568,920	30.3	≈1,093,456	32.2
	C+G	3,293	515.9	26,064	662.4	49,668	709.4
IfiGen	S	≈1,642,950	1.0	≈17,141,210	1.0	≈35,173,456	1.0
	C	58,612	28.0	≈552,450	31.0	≈1,093,458	32.2
	C+G	3,094	531.0	25,919	661.3	48,781	721.0

IfiGen, the vocabulary sizes of 20k, 200k and 400k are tested to compare the speedup ratio under different conditions. As shown in Table 4, the computational performance of ‘C’ (by using MapReduce only) is about 28 $\sim$ 47 times faster respectively than that of ‘S’ (by using a single CPU core). When both GPU and MapReduce are enabled, the computational performance is significantly increased by ‘C+G’, with about 515 $\sim$ 2609 times faster than that of ‘S’ and about 18 $\sim$ 65 times faster than that of ‘C’.

Moreover, it is seen from the results that the speedup ratio is compromised when the size of processing data is relatively small. One of the major reasons is that MapReduce needs to spend additional computing resources in task scheduling, load balancing, task startup and cleanup etc., which hold quite a bit proportion of the running time. However, with the data size increased, the proportion for these computational overheads decreases and the real-time performance of the entire system gradually improves. This trend can be easily observed by comparing the speedup ratios achieved by ‘C+G’ among 20k, 200k and 400k in the tasks of Clustering, HeTrain and IfiGen.

On the other hand, for the task of keyframe extraction, the speedup ratio of ‘C’ against ‘S’ is not very significant, just about 2 times. The reason is that under the ‘S’ condition, there is no need to upload the keyframes to HDFS, however, a large amount of time is required to transfer data between local disks and HDFS in the ‘C’ environment. In a sense, MapReduce is not very skilled in the task with large proportions of disk reading and writing operations.

The computational performances of offline training for NDV retrieval based on the proposed MICC framework are presented as mentioned above. In the following, the online query time performances achieved by ‘C+G’ are shown in Table 5, where the performances obtained by ‘S’ are also given for illustrating the speedup ratio. In the current implementation on the CC_WEB _VIDEO dataset, the average length of a query video is with about 164 keyframes and 480 keypoints per keyframe. A total of 3926 queries are tested in this work.

From the results shown in Table 5, it can be observed that the time costs consumed by ‘S’ and ’C+G’ on the EtrFrm procedure are the same. This is due to the following two reasons: (1) GPUs are not enabled for keyframe extraction. (2) Since each time only one query video

Table 5 Comparison of query time per average-length video achieved by MICC with different sized visual dictionaries

	20k			200k			400k		
Procs.	Time (s)		SR	Time (s)		SR	Time (s)		SR
	S	C+G		S	C+G		S	C+G	
EtrFrm	0.009×164	0.009×164	1.0	0.009×164	0.009×164	1.0	0.009×164	0.009×164	1.0
ImgProc	0.350×164	0.012×164	29.2	0.350×164	0.012×164	29.2	0.350×164	0.012×164	29.2
IfiGen	0.933×164	0.025×164	37.3	9.470×164	0.250×164	37.9	20.71×164	0.510×164	40.6
Search, Baseline	0.066×164	0.003×164	22.0	0.144×164	0.005×164	28.8	0.225×164	0.010×164	22.5
Search, HE	0.135×164	0.004×164	33.8	0.175×164	0.006×164	29.2	0.246×164	0.011×164	22.4
Search, WGC	0.271×164	0.010×164	27.1	0.226×164	0.007×164	32.3	0.270×164	0.011×164	24.5
Search, HE+WGC	0.361×164	0.012×164	30.1	0.265×164	0.008×164	33.1	0.284×164	0.012×164	23.7

is searched and the SR performance on the EtrFrm procedure as shown in Table 4 is not significant, only a single CPU core is used for feature extraction during online query even under the 'C+G' evaluation condition. As far as the other three procedures for online query are concerned, including ImgProc, IfiGen, Search (with Baseline, HE, WGC, or HE+WGC), the query time achieved by 'C+G' is about 22 $\sim$ 40 times faster than that of 'S'. The entire query time cost per average-length video can be calculated by summing the time costs of each procedure as presented in Table 5. Taking the search approach with WGC for an example, it takes about 256.3 s, 1649.0 s and 3499.6 s, respectively, to fulfill the retrieval for one query video when the size of visual dictionary is 20k, 200k and 400k under the 'S' evaluation condition. Under the 'C+G' evaluation condition, the corresponding entire query time becomes 9.2 s, 45.6 s and 88.9 s, with the speedup ratio over 'S' being 27.9, 36.2 and 39.4 times faster.

Regarding parallel computing, efficiency is a significant issue to consider. Generally, the efficiency of a parallel system will decline when the number of computer nodes increases. This is mainly because more computing resources are spent on node communication and task scheduling. In the following, we take the most time-consuming task K-Means clustering with different number of computer nodes as an example for illustration. In this experiment, a 200k-centroid K-Means clustering on the Flickr100k dataset [6] is performed under the 'C+G' evaluation condition with 6, 8, 10 and 12 computer nodes, respectively. The time cost results are shown in Table 6, with the parameter R calculated as $R = \frac{T_6/T_i}{i/6}$, where i stands for the number of computer nodes and T_i is the time cost when i computer nodes are employed. From its definition, R indicates the influence of the number of computer nodes on the system performance, which can be referred to the efficiency factor to some extent. The less the R value is, the less the efficiency of the system becomes. As observed

Table 6 Time cost of 200k-centroid K-Means clustering performed by C+G with different number of computer nodes on Flickr100k dataset

# of nodes	12	10	8	6
Time (s)	33,276	38,748	48,484	63,132
R	0.95	0.98	0.98	1

in Table 6, the R value becomes lower when the number of computer nodes increases from 6 to 12, which indicates that the efficiency of the system declines.

On the other hand, it is important to consider the unbalanced computational capacity of computer clusters. Based on our experiences, if the entire amount of computational power is fixed, the best configuration for a computer cluster is that each node in the cluster has equal or almost equal computational capacity. However, it is not guaranteed that each computer node has the same computational power, especially in the cloud environment. In the current work, we observe that if the number of *map* tasks for a given job is relatively large (*e.g.*, the number of *map* tasks is larger than three times of the *map* task capacity of the system), the impact of unbalanced computational capacity of the computer cluster will be limited. This is because MapReduce supports dynamic load scheduling and balance very well. In another word, the computer nodes with higher computational power will be allocated more *map* tasks during the job running. It is still an open issue to study the unbalanced computational capacity of parallel systems, and we will perform deeper investigations on this issue as one of our future research works.

Table 7 Details of MICC mAP for 24 queries on CC_WEB_VIDEO with different sized visual dictionaries

	20k				200k				400k	
#query	Baseline	HE	WGC	HE+WGC	Baseline	HE	WGC	HE+WGC	Baseline	WGC
1	0.997	0.996	0.997	0.996	0.997	0.996	0.997	0.996	0.997	0.997
2	0.983	0.982	0.983	0.982	0.983	0.982	0.982	0.982	0.982	0.983
3	0.944	0.945	0.944	0.945	0.945	0.945	0.945	0.945	0.944	0.944
4	0.993	0.993	0.993	0.993	0.993	0.993	0.993	0.993	0.993	0.993
5	0.997	0.994	0.996	0.994	0.997	0.993	0.997	0.993	0.996	0.996
6	0.979	0.994	0.990	0.992	0.995	0.992	0.994	0.992	0.995	0.995
7	0.990	0.993	0.993	0.993	0.993	0.993	0.993	0.993	0.993	0.993
8	0.987	0.988	0.987	0.988	0.987	0.988	0.989	0.989	0.989	0.989
9	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
10	0.946	0.955	0.947	0.954	0.946	0.954	0.947	0.954	0.944	0.947
11	0.925	0.923	0.925	0.924	0.926	0.923	0.927	0.922	0.926	0.929
12	0.993	0.990	0.993	0.987	0.993	0.988	0.993	0.988	0.992	0.993
13	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999
14	0.998	0.996	0.997	0.996	0.998	0.996	0.998	0.996	0.998	0.998
15	0.992	0.996	0.995	0.996	0.998	0.995	0.997	0.994	0.998	0.998
16	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
17	0.984	0.996	0.998	0.996	0.998	0.996	0.998	0.996	0.998	0.998
18	0.827	0.836	0.837	0.831	0.839	0.815	0.836	0.816	0.838	0.832
19	0.998	0.996	0.998	0.995	0.998	0.994	0.998	0.994	0.998	0.998
20	0.999	0.997	0.999	0.996	0.999	0.994	0.996	0.990	0.999	0.996
21	0.993	0.996	0.997	0.996	0.998	0.995	0.998	0.995	0.998	0.998
22	0.651	0.662	0.679	0.663	0.693	0.650	0.709	0.657	0.680	0.718
23	0.973	0.980	0.975	0.979	0.987	0.977	0.987	0.977	0.987	0.987
24	1.000	1.000	1.000	1.000	1.000	0.997	1.000	0.997	1.000	1.000
Average	0.964	0.967	0.968	0.967	0.969	0.965	0.970	0.965	0.969	0.970

5.3 Retrieval accuracy

In this subsection, the NDV retrieval accuracy of the proposed MICC framework is presented in terms of mean Average Precision (mAP), with the detailed information for each of the 24 queries shown in Table 7. From the results on the 20k-size and 200k-size visual dictionaries, it can be observed that WGC can slightly improve the retrieval accuracy as compared with Baseline, whereas HE is not always helpful (*e.g.*, when the size of visual vocabularies is 200k, the mAP achieved by HE is a little bit lower than that of Baseline). Therefore, we only present the mAP results obtained by Baseline and WGC when the size of visual vocabularies is 400k. In the case of 20k-size visual dictionary, HE achieves slightly higher mAP than that of Baseline. When we focus on the scenario of 200k-size dictionary, HE lowers the mAP in average possibly because the high-dimensional feature vector space is already well characterized with 200k visual words. On the other hand, WGC is useful in improving retrieval accuracy due to its usage of scale and orientation of feature points.

As observed from the results, slightly higher mAPs can be achieved by employing the 200k-size dictionary than the 20k-size dictionary, since the feature space is more appropriately segmented with more visual words in this case. However, when the size of visual words reaches to 400k, the mAP performance keeps almost the same as that of 200k. This indicates that it is not always helpful to employ larger sized visual dictionaries for improving retrieval accuracy. As shown in Table 7, the mAP results on query 18 and 22 are particularly lower than that of the other queries. This is because the original videos of the query 18 are captured by a cell phone in a bus leading to the video quality being a little vague and unrelated frames are inserted to some videos belonging to query 22, as described in [31].

Besides the presentation of mAP achieved by MICC, we also cite the mAP results of several state-of-the-art works for comparison, including the global feature with color histogram method (denoted as GF) [31], the ordinal measure based method (denoted as OM) [10], the hierarchical method (denoted as HM) [31], the spatial-temporal feature with local binary pattern based method (denoted as ST) [22] and the multiple feature hashing method (denoted as MF) [24]. The same 24 queries as used in [31] are executed with its corresponding reference set for mAP calculation. The comparative results are listed in Table 8, where it can be observed that the proposed MICC with WGC enabled and using 200k/400k-size visual dictionary can achieve the highest mAP as compared with the other methods. Moreover, since the proposed MICC framework employs almost the same set of techniques as that of [5] and [33] for NDV retrieval, the mAP performances of the proposed MICC framework, [5] and [33] should be quite similar to each other.

Table 8 mAP comparison

Apch.	mAP
GF [31]	0.892
OM [10]	0.910
HM [31]	0.952
ST [22]	0.953
MF [24]	0.954
MICC@WGC+20k	0.968
MICC@WGC+200k	0.970
MICC@WGC+400k	0.970

6 Conclusion

An efficient GPU-based MapReduce framework MICC is proposed in this paper to explore NDV retrieval tasks. In order to fully exploit the joint benefit of MapReduce and GPU, several appropriate modifications on MapReduce have been developed which make the proposed MICC very suitable for data-intensive applications. As combined with the GPU technology, the computational efficiency of the proposed MICC is much higher than traditional MapReduce-based clusters, which has been verified by the experiments. We consider MICC a successful attempt to combine advanced cloud computing and GPU technologies into the large-scale multimedia data mining field, which will become a trend to meet the needs of big-data-driven multimedia applications.

Acknowledgments This work was supported in part by the National Natural Science Foundation of China under Grant 61102059, the "Shu Guang" project of Shanghai Municipal Education Commission and Shanghai Education Development Foundation under Grant 12SG23, the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning, the Fundamental Research Funds for the Central Universities under Grants 0800219158, 0800219270, and 1700219104, and the National Basic Research Program (973 Program) of China under Grant 2010CB328101.

References

1. Batko M, Falchi F, Lucchese C, Novak D, Perego R, Rabitti F, Sedmidubsky J, Zezula P (2010) Building a web-scale image similarity search system. Multimed Tools Appl 47(3):599–629
2. Cevahir A, Torii J (2012) GPU-enabled high performance online visual search with high accuracy. In: Proc. ISM'12, pp. 413–420
3. Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Communications of the ACM - 50th anniversary issue: 1958-2008 51(1), 107–113
4. Dong W, Wang Z, Charikar M, Li K (2008) Efficiently matching sets of features with random histograms. In:Proc. ACM MM'08, pp. 179–188
5. Douze M, Gaidon A, Jegou H, Marszałek M, Schmid C et al (2008) INRIA-LEARs video copy detection system. In: TRECVID Workshop'08
6. Flickr100k image dataset Online available: http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/flickr100k.html
7. Google protobuf Online available: http://code.google.com/p/protobuf/
8. Hadoop Online available: http://hadoop.apache.org/docs/r1.2.1/
9. He B, Fang W, Luo Q, Govindaraju NK, Wang NT (2008) Mars: A MapReduce framework on graphics processors. In: Proc. PACT'08, pp. 260–269
10. Hua XS, Chen X, Zhang HJ (2004) Robust video signature based on ordinal measure. In: Proc. ICIP'04, pp. 685–688
11. Jegou H, Douze M, Schmid C (2008) Hamming embedding and weak geometric consistency for large scale image search. In: Proc. ECCV'08, pp. 304–317
12. Jegou H, Douze M, Schmid C (2010) Improving bag-of-features for large scale image search. IJCV 87(3):316–336
13. Karantasis KI, Polychronopoulos ED, Dimitrakopoulos GN (2010) Accelerating data clustering on GPU-based clusters under shared memory abstraction. In: Proc. CCWP'10, pp. 1–5
14. Li Y, Crandall DJ, Huttenlocher DP (2009) Landmark classification in large-scale image collections. In: Proc. CVPR'09, pp. 1957–1964
15. Liu J, Huang Z, Cai H, Shen HT, Ngo CW, Wang W (2013) Near-duplicate video retrieval: current research and future trends. ACM computing surveys 45(4). Article:44
16. Lowe D (2004) Distinctive image features from scale-invariant keypoints. IJCV 60(2):91–110
17. Mikolajczyk K, Schmid C (2004) Scale and affine invariant interest point detectors. IJCV 60(1):63–86
18. Moise D, Shestakov D, Gudmundsson G, Amsaleg L (2013) Indexing and searching 100M images with Map-Reduce. In: Proc. ICMR'13, pp. 17–24
19. NVIDIA CUDA Online available: https://developer.nvidia.com/

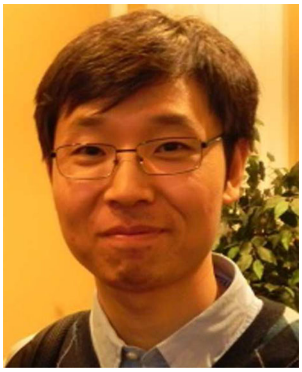
20. Owens JD, Houston M, Luebke D, Green S, Stone JE, Phillips JC (2008) GPU computing. Proc IEEE 96(5):879–899
21. Shalom SA, Dash M, Tue M (2008) Efficient K-Means clustering using accelerated graphics processors. In: Proc. DAWAK'08, pp. 166–175
22. Shang L, Yang L, Wang F, Chan KP, Hua XS (2010) Real-time large scale near-duplicate web video retrieval. In: Proc. ACM MM'10, pp. 531–540
23. Sivic J, Zisserman A (2003) Video google: A text retrieval approach to object matching in videos. In: Proc. ICCV'03, pp. 1470–1477
24. Song J, Yang Y, Huang Z, Shen HT, Hong R (2011) Multiple feature hashing for real-time large scale near-duplicate video retrieval. In: Proc. ACM MM'11, pp. 423–432
25. Stuart JA, Owens JD (2011) Multi-GPU MapReduce on GPU clusters. In: Proc. IPDPS'11, pp. 1068–1079
26. Toolkit of Hessian-Affine detector Online available: http://www.robots.ox.ac.uk/~vgg/research/affine/
27. Van De Sande KEA, Gevers T, Snoek CGM (2011) Empowering visual categorization with the GPU. IEEE Trans Multimedia 13(1):60–70
28. Wang H, Shen Y, Wang L, Zhu F, Wang W, Cheng C (2012) Large-scale multimedia data mining using MapReduce framework. In: Proc. CloudCom'12, pp. 287–292
29. White B, Yeh T, Lin J, Davis L (2010) Web-scale computer vision using MapReduce for multimedia data mining. In: Proc. MDMKDD'10, p. Article No.9
30. White T (2010) Hadoop: the definitive guide, 2nd edn. O'Reilly Media, Inc
31. Wu X, Ngo CW, Hauptmann AG, Tan HK (2009) Real-time near-duplicate elimination for web video search with content and context. IEEE Trans Multimedia 11(2):196–207
32. Yan R, Fleury MO, Merler M, Natsev A, Smith JR (2009) Large-scale multimedia semantic concept modeling using robust subspace bagging and MapReduce. In: Proc. LS-MMRM'09, pp. 35–42
33. Zhao WL, Wu X, Ngo CW (2010) On the annotation of web videos by efficient near-duplicate search. IEEE Trans Multimedia 12(5):448–461

Hanli Wang received the B.S. and M.S. degrees in electrical engineering from Zhejiang University, Hangzhou, China, in 2001 and 2004, respectively, and the Ph.D. degree in computer science from City University of Hong Kong, Kowloon, Hong Kong, in 2007. From 2007 to 2008, he was a Research Fellow with the Department of Computer Science, City University of Hong Kong. From 2007 to 2008, he also was a Visiting Scholar with Stanford University, Palo Alto, CA. From 2008 to 2009, he was a Research Engineer with Precoad, Inc., Menlo Park, CA. From 2009 to 2010, he was an Alexander von Humboldt Research Fellow in University of Hagen, Hagen, Germany. In 2010, he joined the Department of Computer Science and Technology, Tongji University, Shanghai, China, as a Full Professor. His current research interests include digital video coding, computer vision, multimedia content analysis, and machine learning.

Fengkuangtian Zhu received the B.S. degree in computer science and technology from Tongji University, Shanghai, China, in 2012. From 2012, he became a M.S. degree candidate for computer science and technology in Tongji University. His current research interests include computer vision and multimedia content analysis.

Bo Xiao received the B.S degree in thermal engineering from Tongji University, Shanghai, China, in 2004. From 2007 to 2009, He was a R&D engineer in the 2States Technology (Shanghai) Ltd. From 2010 to 2011, he was a software engineer in the Shanghai R&D center of Suzhou KEDACOM Technology Ltd. From 2011, he became a Ph.D. degree candidate for computer science and technology in Tongji University. His current interests include multimedia communication, video coding, computer vision, and cloud computing.

Lei Wang received the B.S. degree in information security from Tongji University, Shanghai, China, in 2012. From 2012, he became a Ph.D. degree candidate for computer science and technology in Tongji University. His current research interests include computer vision and multimedia content analysis.



Yu-Gang Jiang received the Ph.D. degree in computer science from the City University of Hong Kong in 2009. During 2008-2011, he was with the Department of Electrical Engineering, Columbia University, New York. He is currently an Associate Professor of computer science with Fudan University, Shanghai, China. His research interests include multimedia retrieval and computer vision.