CHCF: A Cloud-Based Heterogeneous Computing Framework for Large-Scale Image Retrieval

Hanli Wang, *Senior Member, IEEE*, Bo Xiao, Lei Wang, Fengkuangtian Zhu, Yu-Gang Jiang, and Jun Wu, *Senior Member, IEEE*

***Abstract*— The last decade has witnessed a dramatic growth of multimedia content and applications, which in turn requires an increasing demand of computational resources. Meanwhile, the high-performance computing world undergoes a trend toward heterogeneity. However, it is never easy to develop domain-specific applications on heterogeneous systems while maximizing the system efficiency. In this paper, a novel framework, namely, cloud-based heterogeneous computing framework (CHCF), is proposed with a set of tools and techniques for compilation, optimization, and execution of multimedia mining applications on heterogeneous systems. With the aid of the compiler and the utility library provided by CHCF, users are able to develop multimedia mining applications rapidly and efficiently. The proposed framework employs a number of techniques, including adaptive data partitioning, knowledge-based hierarchical scheduling, and performance estimation, to achieve high computing performance. As one of the most important multimedia mining applications, large-scale image retrieval is investigated based on the proposed CHCF. The scalability, computing performance, and programmability of CHCF are studied for large-scale image retrieval by case studies and experimental evaluations. The experimental results demonstrate that CHCF can achieve good scalability and significant computing performance improvements for image retrieval.**

***Index Terms*— Data parallelism, distributed scheduling, heterogeneous computing, image retrieval, multimedia mining.**

I. Introduction

THE success of social networks makes it easy for people to capture and share various multimedia contents, leading to the generation of a huge amount of images and videos on the Internet. It is highly desired to effectively store, query, analyze, and utilize these immense data for a number of multimedia applications with the large-scale image retrieval as a golden application example, such as Google Image Search, Baidu Image Search, and Bing Image Search.

In [1], most image retrieval algorithms are based on the bag-of-feature (BoF) model that represents images with sparse vectors by extracting salient local features from images and assigning them to the nearest visual word(s). During retrieval, similar images will be returned through some ranking method. However, with the explosion of data, in order to deal with large-scale multimedia data, not only advanced algorithms but also powerful computing systems or platforms are desired and even essential [2].

An important characteristic about image retrieval is that the computations of most image retrieval related algorithms are data intensive, and the data of image retrieval (such as images, intermediate feature vectors, BoF vectors, and so on) can be easily partitioned into data chunks that are relatively independent with each other when being processed. Considering this trait, it is desired to employ high-performance computing (HPC) systems to scale out the data workloads for large-scale image retrieval. Moreover, the past decades witness a growing popularity of parallel computing frameworks that allow the horizontal scaling of large-scale workloads using HPC systems. MapReduce [3], for instance, is such a popular parallel programming paradigm deployed on computer clusters, which applies the intuition that many large-scale workloads can be processed with the map and reduce operations.

Meanwhile, the HPC world has undergone a new trend toward heterogeneity and hierarchy in the past several years. Such hybrid systems (or called heterogeneous systems), which are usually equipped with multiple graphic processing units (GPUs), are more powerful and environment friendly compared with traditional HPC systems. As one of the most representative many-core processors, GPU is usually manufactured with hundreds or even thousands of stream processors and is able to invoke a large number of threads simultaneously, which makes it suitable for data-intensive computations at fine grained level. Since the most time-consuming operations for image retrieval, such as summation or reduction of vectors, distance calculation between vectors, and so on, can be accelerated by the single instruction multiple data (SIMD) execution model implemented with stream processors, it is desirable and practicable to build a hybrid system for large-scale image retrieval in which coarse and fine grained parallelization are gained from inter- and intra-machine cooperations.

Manuscript received September 14, 2014; revised January 14, 2015 and July 29, 2015; accepted August 30, 2015. Date of publication September 11, 2015; date of current version December 3, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 61472281, in part by the National 863 Program of China under Grant 2014AA015101, in part by the Shu Guang Project through the Shanghai Municipal Education Commission and Shanghai Education Development Foundation under Grant 12SG23, in part by the Program for Professor of Special Appointment (Eastern Scholar) at the Shanghai Institutions of Higher Learning under Grant GZ2015005, and in part by the Fundamental Research Funds for the Central Universities under Grant 0800219270. This paper was recommended by Associate Editor J. Liu.

H. Wang, B. Xiao, L. Wang, F. Zhu, and J. Wu are with the Key Laboratory of Embedded System and Service Computing, Department of Computer Science and Technology, Ministry of Education, Tongji University, Shanghai 200092, China (e-mail: hanliwang@tongji.edu.cn; 1314xiaobo@tongji.edu.cn; 110_wangleixx@tongji.edu.cn; 2zhufengkuangtian@tongji.edu.cn; wujun@tongji.edu.cn).

Y.-G. Jiang is with the School of Computer Science, Fudan University, Shanghai 201203, China (e-mail: ygj@fudan.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2015.2477939

Although the performance of hybrid systems is impressive and continues to improve, there still remains several challenges to implement hybrid systems for multimedia mining applications. First, most hybrid systems confront the problem of low utility of computational resources. As far as we know, one of the most significant reasons causing low utility of computational resources is load imbalance, which can be considered as uneven distribution of computations in the system. A number of factors will result in load imbalance, for instance, the variation of the computational resources, the fluctuation of the network bandwidth, and the difference between the computation tasks. It is necessary to design effective scheduling strategies to avoid load imbalance for multimedia mining applications so as to fully utilize the computational resources of hybrid systems, which is never an easy task. Second, it is complex and less productive to implement various multimedia mining algorithms on many-core processors such as GPU [4]. Compared with CPU threads, GPU threads are massive and lightweight, which require the programs running on GPUs to be fine grained and be capable of scaling to a large number of stream processors. The memory hierarchy of GPU, which is different from that of CPU, requires a careful design of data layout and memory access in order to improve the efficiency of GPU. Moreover, users have to deal with the details such as data transfer between host and GPU memory, GPU kernel configuration and invocation, coordination of multi-GPUs, and so on.

Facing the above challenges, a cloud-based heterogeneous computing framework (CHCF) is proposed in this paper, which is designed to reduce the barriers to developing multimedia mining applications (e.g., large-scale image retrieval) on hybrid systems while maximizing the overall system performance. The main contributions of this paper include the following aspects.

1) A simple but effective programming paradigm associated with a high-level language, called CHCF language, is designed for developing multimedia mining applications. With the proposed paradigm, an application consists of a driver and a set of filters.
2) A utility library is developed, including a set of multimedia mining algorithms (such as feature extraction, k-means clustering, and BoF generation), which are provided as built-in filters. The library is implemented based on the proposed programming paradigm.
3) An adaptive partitioning policy for data parallelism is designed. This partitioner works in conjunction with the input format function to guarantee the best data partitioning when the computation begins.
4) A knowledge-based hierarchical distributed scheduler and performance estimator are proposed. In combination with the static scheduling information during the application compilation, the proposed scheduler updates the knowledge of the framework that is provided by the performance estimator to make the scheduler more intelligent and adaptive to the specific hardware conditions and diverse computation loads.

The key components of the proposed CHCF, including adaptive partitioner, knowledge-based hierarchical scheduler, and performance estimator, are designed to address the problem of load imbalance for multimedia mining applications. The proposed filter-based programming paradigm with CHCF language is able to conceal the difference of the underlying processors, such as CPU, GPU, and so on, so as to facilitate the programmability of multimedia mining applications.

The rest of this paper is organized as follows. Section II gives an overview of the related works, and the key differences between the proposed CHCF and the related works are also discussed. The architecture of the proposed CHCF is introduced in Section III, which presents the basic ideas of the proposed CHCF. The main concepts and contributions of this paper, including filters, adaptive partitioner, knowledge-based hierarchical scheduler, and performance estimator, are detailed in Section IV. Four case studies for image retrieval are provided in Section V to demonstrate the usage of the proposed CHCF. Section VI presents the experimental evaluation. Finally, Section VII concludes this paper and presents future research directions.

II. Related Work

A number of works have been developed to speed up large-scale image retrieval from the view of algorithm. The binary local features such as BRIEF [5] and ORB [6] employ simplified comparison and compression techniques to achieve high computing efficiency while still keeping robustness [7] as the traditional local features like Hessian-affine detector [8] and scale-invariant feature transform (SIFT) descriptor [9]. On the other hand, several clustering methods such as hierarchical k-means (HKM) clustering [10] and approximate k-means (AKM) clustering [11] are designed to accelerate the quantizing process and to generate high-dimensional vocabularies, which the traditional flat k-means clustering method fails to reach. As an exchange, HKM and AKM result in a fraction of accuracy loss. Moreover, most systems generate visual dictionaries using a small set of images and/or feature points due to the limitation of computational resources. In addition, the inverted file structure [1] is generally employed, which greatly reduces memory requirements and makes it extendable for large-scale image retrieval. In [12], the latent Dirichlet allocation (LDA) model is applied to image representation for large-scale image retrieval. The results show that the combination of LDA-based image representation with an appropriate similarity measure outperforms previous approaches. Wang *et al.* [13] introduce three key components for large-scale image retrieval, including a global descriptor for image representation, a hashing algorithm for feature indexing, and a distance measure for reranking retrieved images. In [14], hashing codes are employed to encode high-dimensional features into hamming space to support effective and efficient search.

On the other hand, several researches are proposed to employ parallel frameworks (e.g., MapReduce) to meet the huge computational demands of multimedia mining, which is also the concern of the proposed CHCF on scaling out workloads across machines. These researches generally benefit from the functionalities of existing frameworks, such as fault tolerance, task scheduling, and so on. In [2], MapReduce is applied to multimedia mining, and the results of k-means clustering and background substraction have been presented. Moise *et al.* [15] propose to employ MapReduce to accelerate searching a large amount of images. MapReduce is used to find the nearest neighbors for the generation of image tags in [16]. Compared with MapReduce that abstracts the computations into map and reduce operations, the proposed CHCF is more general in terms of the operations using filter paradigm; moreover, the proposed CHCF employs adaptive data partitioning and knowledge-based hierarchical scheduling to address the challenge of load imbalance.

As far as the heterogeneous cluster is concerned, several frameworks or paradigms have been designed in recent years. In [17], StarPU is introduced for numerical kernel design to execute parallel tasks on a shared-memory machine with heterogeneous hardwares. Since the number of processors in a shared-memory machine is limited, the predefined scheduling strategy employed by StarPU improves the performance efficiently. However, the scheduler proposed by StarPU may become ineffective when ported onto distributed systems. This is the reason why we design the knowledge-based hierarchical scheduler, which is hybrid in the sense of static and dynamic scheduling policies and thus more suitable for distributed systems. In [18], a high-level model for parallel programming, compiler, and runtime, called merge, is proposed, which is similar to StarPU. Merge also aims at shared-memory machines with heterogeneous hardwares and employs the MapReduce programming paradigm to simplify the interface for users, which in turn constrains its applicable range. In [19], an elastic computing framework is proposed, which uses an adapter to hide the difference of various kinds of processors, such as GPU, Intel Phi MIC, and even field-programmable gate array. However, the elastic computing framework focuses on the design of elastic functions and has few discussions about scheduling. As inspired by the elastic computing framework, the proposed CHCF employs a similar technique called filter interface to hide the difference of the underlying processors and remain future extension to other processors. He *et al.* [20] propose a GPU-based MapReduce framework Mars, which evaluates the benefits of cooperative usage of CPUs and GPUs on a number of traditional applications such as string matching and matrix multiplication. However, the scheduling policy of Mars is not mentioned in [20].

In addition to performance and scalability, a number of works have made efforts on programmability, which share similarities with the proposed CHCF on the goal of facilitating domain-specific application development. In [21], a compiler is proposed to enable users to write hybrid CPU/GPU code by utilizing the OpenMP [22] directives. Ghoting *et al.* [23] offer a high-level declarative language SystemML for programming machine learning applications that are compiled and executed with MapReduce. Compared with SystemML, the proposed CHCF compiler is hybrid in the sense that C/C++ and Java can be utilized to develop filters. Ma and Agrawal [24] propose a code generation system, referred to as AUTO-GC, to translate data mining applications on GPU clusters. The results of two popular data mining algorithms, including k-means clustering and principle component analysis, are reported to demonstrate that a good scalability is accomplished without noticeable overheads of the translated code. Compared with AUTO-GC that is actually a code generation system for reduction operations, the proposed CHCF provides a more general programming paradigm with filters. Lee *et al.* [25] develop a similar compiler framework for translating standard OpenMP applications to Compute Unified Device Architecture-based general purpose GPU applications. However, this compiler is operating in a source-to-source translation manner.

Regarding load imbalance, the centralized paradigm is widely used by scheduling policies such as [26] and [27], which employs a master node to take charge of scheduling. Although this paradigm is simple and effective when the number of nodes of the system is relatively small, the limitations are obvious. First, the effectiveness may be sharply degraded when the number of nodes increases. Second, the scheduling made by the master is one-time assignment, but the information for making the scheduling decision is constantly changing. To overcome the disadvantage of centralized scheduling, a distributed load balancing approach, called distributed adaptive scheduling (DAS), is proposed in [28] for scientific applications expressed as iterative loops with dependencies. DAS combines the information about the run queue with the timing history of each computation worker to make scheduling. It is verified that DAS can achieve significant performance improvements over centralized approaches. However, as high-performance clusters become heterogeneous and the scale of the clusters continues to increase, DAS meets some challenges. Since the computation workers have to communicate with each other to make scheduling decisions, the overheads for network communication may hurt the overall performance and the effectiveness of scheduling when the number of nodes is large. Moreover, the efficiency of DAS may become degraded when the nodes are equipped with multiple GPUs. In contrary to DAS, the proposed knowledge-based hierarchical scheduler is masterless, which means that scheduling decisions are made by the master scheduler and node schedulers cooperatively. In another word, the master scheduler coordinates the node schedulers and the node schedulers make scheduling decisions. Therefore, the communication overheads are reduced since the scheduling efficiency is improved by distributing scheduling computations from the master scheduler to node schedulers.

III. CHCF Overview

A. Architecture

The overall architecture of the proposed CHCF is depicted in Fig. 1. As aforementioned, CHCF aims at developing and executing multimedia mining applications on hybrid systems. To achieve this, CHCF provides a high-level programming language coupled with a utility library, in which multimedia mining algorithms can be implemented for running with CPUs and/or GPUs, and a set of tools for compilation, execution, and optimization are designed. In CHCF, the popular programming paradigm, called filter stream [29], is adopted; therefore, all computing units, either atomic or compound, are represented by filters and effectively communicate with each other

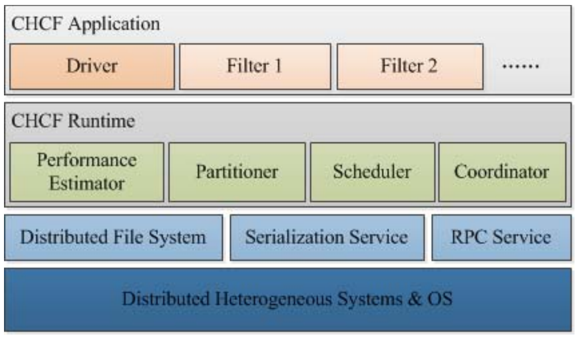


Fig. 1. Architecture of CHCF.

through streams. Another advantage of the stream mode is that it is evolved into event oriented [30] to reduce communication complexity. As a benefit, users need only to write event-processing handlers for each filter and define connectivity between filters to build an application, referred to as CHCF application as shown in Fig. 1.

During the CHCF runtime, multiple instances for each filter, called filter instance, are created as request to achieve data parallelism. An adaptive partitioning policy is designed to create data splits for the filter instances. This partitioning policy takes the following factors into consideration to balance the computation loads among the hybrid system, including the heterogeneity of the system, the information from the performance estimator, the available computational resources of the system, and the attributes of the input data.

Although the partitioner of the framework could potentially determine the best size of input data split for each filter instance when the task begins, the available computational resource and performance may vary during runtime execution. In order to reduce load imbalance while the application is running, a dynamic knowledge-based hierarchical scheduling strategy is employed. This scheduler is masterless, which consists of a master scheduler and several node schedulers. The node schedulers keep monitoring the resources and communicate with each other. If there are available resources on some node, the corresponding node scheduler will broadcast request to the system, which includes its own processing rate, for task allocation. Then, the busy nodes may respond to the request and delegate a part of their work to the idle nodes.

Since the adaptive partitioner and dynamic scheduler both make decisions based on the information about each filter's performance, it is crucial to make the information accurate enough. The performance estimator keeps tracking the execution of each filter and updating the related information. The collected performance information will be stored in an exclusive data structure, called filter performance table, for fast retrieval. The coordinator is responsible for coordinating the jobs running on the hybrid system as well as the related filter instances.

The Hadoop distributed file system (HDFS) is released as a part of Hadoop [31] and is a popular distributed file system for large volume of data storage. HDFS is employed by the proposed CHCF as one of the most important infrastructures

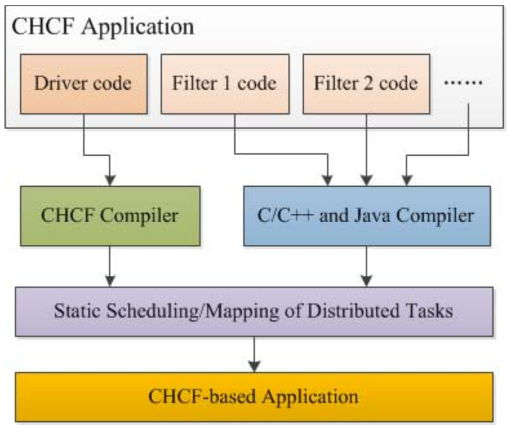


Fig. 2. Compilation of a CHCF application.

due to its simplicity and reliability. Regarding the serialization service, the ProtocolBuffer [32], which is provided by Google, is adopted by CHCF, because it is regarded as one of the most outstanding libraries for serialization in terms of the size of serialized data and the speed for serializing and deserializing. In order to lower the bar for programming distributed applications, the famous remote procedure call (RPC) middleware Internet communications engine [33] is employed for building RPC services in CHCF. The failover mechanism follows the classic heartbeat method employed by most server-client distributed applications in which the schedulers, including master scheduler and node scheduler, periodically check the status of the dispatched tasks, and a new instance will be created if there is no response from the task [34].

B. Application Buildup

With the proposed CHCF, an application is composed of a driver and a set of filters. The driver is written by the CHCF language, which is the skeleton of the application. And the filters are currently coded with C/C++ or Java for efficiency reason as well as reuse of existing codes. The compiler provided by the framework could seamlessly cooperate with C/C++ and Java compilers. A typical CHCF application and its compilation are shown in Fig. 2.

In order to facilitate users to design efficient filters on hybrid systems, the filter abstraction mode allows users spend more time on their application related workspace and leave other stuffs handled by CHCF. To create a filter, users could reuse, aggregate, or extend the built-in filters from the utility library, or develop a new one from scratch. The driver code that defines the process flow of the application will be translated into a set of distributed tasks with logical dependencies. Once the filters and driver for an application are developed, their performances will be evaluated by executing with various sizes of input data or statically analyzing the logical execution plan. The evaluation process is taken by the framework automatically, and the performance information will be stored after evaluation. Finally, the application will be executed by the framework after the users submit the computation job with specific input parameters.

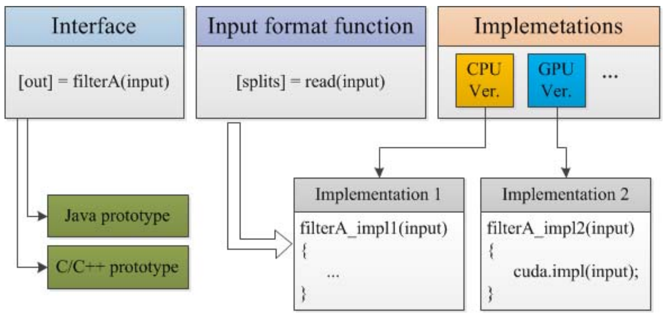


Fig. 3. Illustration of a CHCF filter.

IV. Runtime of CHCF Computing

The CHCF runtime is responsible for resource and computation management of the hybrid system. Since the system is heterogeneous and hierarchical, the CHCF runtime employs the adaptive partitioner, the knowledge-based hierarchical scheduler, and the performance estimator for executing the computation jobs to achieve high performance and system throughput.

A. Filters

A CHCF filter can be regarded as a specific computing unit defined at any level of granularity. The overall performance of an application is basically determined by its filters. To maximize the usage of a hybrid system, various kinds of implementations can be defined by each filter depending on the underlying processors, e.g., CPU and GPU. The best execution plan of an application is performed by static analysis of the compiler combined with dynamic partitioning and scheduling during the runtime according to the input parameters, available resources, and up-to-date information of the filters' performances.

As illustrated in Fig. 3, a filter consists of an interface (which is declared in driver), an input format function, and implementations. The declaration of the interface is translated into prototypes, which are used for filter implementations, in C/C++ and/or Java by the CHCF compiler. The interface acts like a bridge between the driver and filter implementations. The practice of separating an application into driver and filter takes the maximum advantage of automation tools from compiler techniques, freeing the developers from boring and fussy works on defining rules and writing the bounding code. The input format function will be further discussed in Section IV-B about adaptive partitioner.

Moreover, the proposed CHCF provides a high performance and well-tuned library, which includes a number of multimedia mining algorithms as built-in filters, such as feature extraction, k-means clustering, BoF generation, similarity search, Hamming embedding (HE) [35], weak geometric consistency (WGC) [35], RANSAC [36] for homograph calculation, and so on. Most of these algorithms own CPU and GPU implementations. This utility library facilitates developing CHCF applications. The users could utilize the built-in filters to create new filters by means of aggregation and/or inheriting.

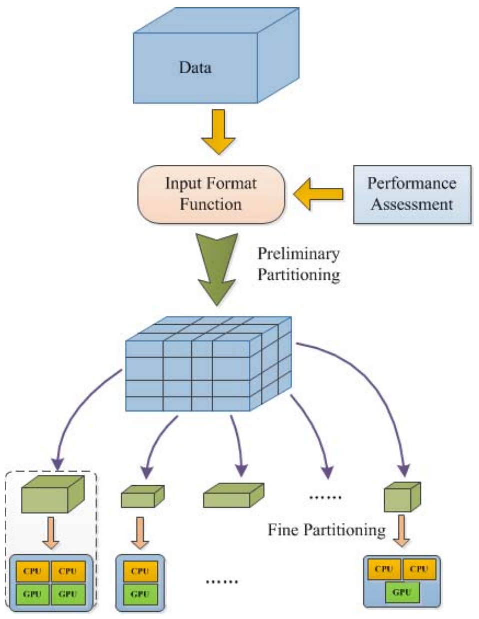


Fig. 4. Illustration of hierarchical data partitioning.

B. Hierarchical Adaptive Partitioner

In addition to the individual performance of filters, parallelism and heterogeneity conspire to influence the overall performance of an application developed on hybrid systems. Since the application latency (also known as makespan) is determined by the longest flow path during the execution, the parameters for input data (mainly including the size and the location of each data split) should be carefully chosen before the computation is executed. Meanwhile, the heterogeneity of the system imposes additional challenges for data partitioning. As a result, a hierarchical adaptive data partitioning policy is proposed for CHCF, which uses the user-supplied input format function and the up-to-date information of individual filters for different computational resources to gain the best prediction of load balance.

The illustration of the proposed hierarchical adaptive data partitioner is given in Fig. 4. As shown in this illustration, the input format function is responsible for reading data with specific structure and partitioning the data into splits. According to the data structure, the developer defines the boundary of the data split. For instance, the boundary of feature vectors, which are the input data to the built-in filter of k-means clustering, can be defined as a single feature vector or the vectors of a single image. At runtime, the partitioner will automatically split the input data for each parallel task based on the available resources, the processing rate of different computational resources, and the boundary definition. Another important feature of the input format function is that multiple boundaries with different granularities can be defined for a filter. As shown in Fig. 5, when a node in the system is idle and requests more data splits, other nodes that are busy in processing the same computing job could respond to the request with reasonable amount of workload dropped from its own using suitable boundary. This process will be further detailed in Section IV-C about dynamic knowledge-based scheduler.

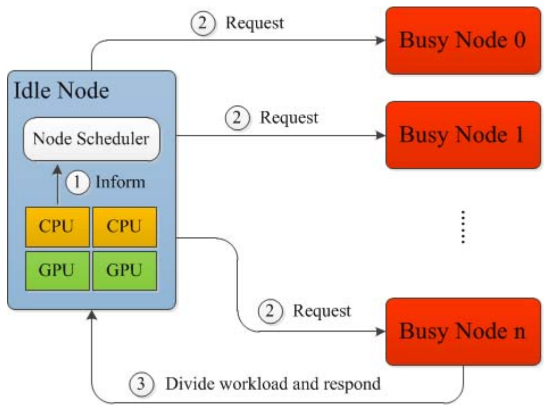


Fig. 5. Illustration of workload stealing.

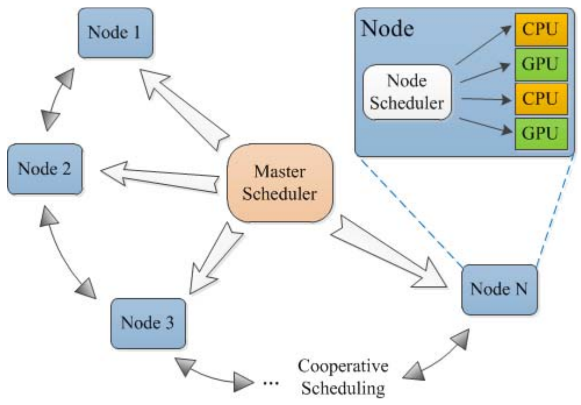


Fig. 6. Illustration of the proposed knowledge-based hierarchical scheduler.

C. Knowledge-Based Hierarchical Scheduler

As discussed before, load imbalance could significantly degrade the overall performance of an application running on hybrid systems. In order to achieve high scheduling performances on hybrid systems, a knowledge-based hierarchical scheduler is proposed herein to overcome the limitations of traditional load balancing methods as mentioned in Section II.

The proposed scheduling approach is illustrated in Fig. 6, where it can be seen that the master scheduler is responsible for coordinating the node schedulers, while the node schedulers take the real scheduling cooperatively during the execution. Compared with previous distributed scheduling schemata, the proposed scheduler is masterless, which means that the scheduling decisions are made by the master and node schedulers jointly, and therefore the overheads for network communication can be effectively reduced and the scheduling efficiency is improved. The information about the individual performance of the computation task, which is used for scheduling, is constantly updated by the performance estimator (which will be introduced in Section IV-D) during application execution. As a consequence, the scheduling decisions made by the up-to-date information are more likely accurate.

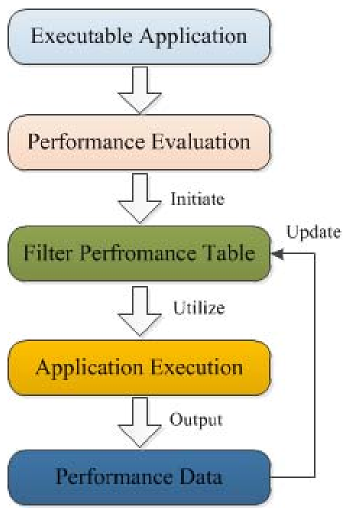


Fig. 7. Illustration of the proposed performance estimation process.

D. Performance Estimator

The performance of each filter under different situations is essential for the proposed CHCF, because it affects the efficiency and accuracy of the adaptive partitioner, hierarchical scheduler, and some other modules. It is achievable to model the system performance by either static analysis of the filter code or measuring the execution time of filters on different processors with different input data. In this paper, we employ the latter to evaluate the performance by means of an exclusive module of the framework, called performance estimator. In order to obtain the most comprehensive information of the performance, it is better to perform an exhausted evaluation with different sized input data. However, this is clearly not feasible. Instead, the proposed performance estimator employs a lazy strategy, which executes a limited number of evaluations at first and updates the information when the application is running for real. The information regarding the performance of each filter will be stored in a data structure, called filter performance table. The process about performance estimation is shown in Fig. 7.

V. Case Study of CHCF for Image Retrieval

In this section, we describe four case studies for image retrieval to demonstrate the usage of the proposed CHCF and show its programmability. We dissect the inner mechanism of filters in the first three cases and interpret the procedures for building a baseline image retrieval system by reusing built-in filters in the last case.

A. Case Study: k-Means Clustering

k-means clustering is one of the most famous algorithms in many fields, including machine learning, multimedia mining, and pattern recognition. It aims to group a set of objects in such a way that the objects belonging to one group are closer to each other than those in other groups. The k-means clustering algorithm is iterative and computationally laborious. However, its workload is divisible, which makes it an ideal computation

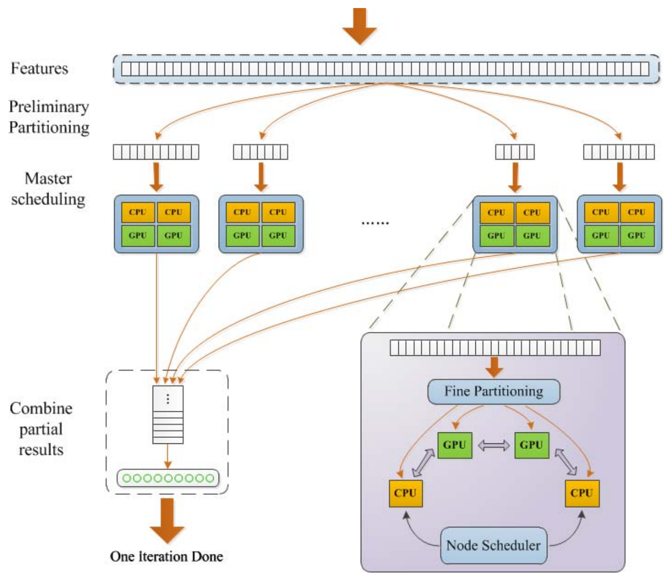


Fig. 8. Overview of the k-means clustering filter.

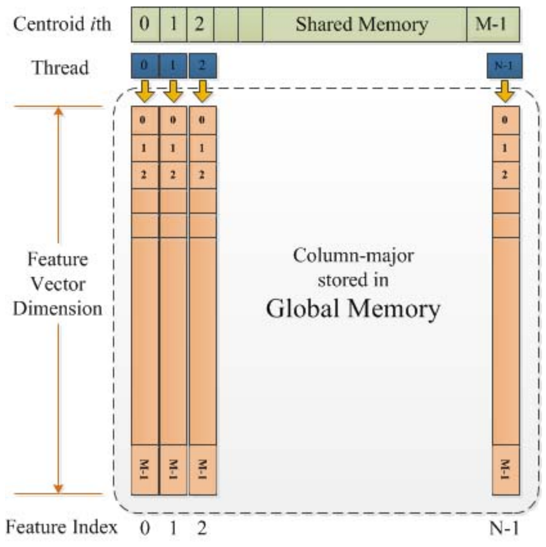


Fig. 9. Illustration of GPU computing for k-means clustering.

for data parallelism. Moreover, the most time-consuming calculation of finding the nearest object can be greatly accelerated when executed by the SIMD execution model. Therefore, it may achieve high efficiency by employing hybrid systems to perform the computation. In this paper, we implement k-means clustering as a built-in filter in the utility library. The overview of this built-in filter is illustrated in Fig. 8.

When the k-means clustering filter is invoked by the driver with input features, the implementation with either CPU or GPU is not actually executed immediately. Instead, the partitioner performs the preliminary partitioning with the help of input format function for the input features according to the information of each node. The node schedulers receive the data splits, which are distributed by the master scheduler, subdivide the data according to the processing speed, and then inform the filter instances to perform the real computation. During the execution, the nodes with higher processing speed than expected could ask the node scheduler to steal workload from other nodes when they become idle before the entire computation task is finished. After all filter instances complete the task, a special instance could combine all partial outputs by executing the reduce step in the filter. Since k-means clustering is iterative, the procedures except the preliminary partitioning will be executed a number of times.

Moreover, the most tremendously computational part is to locate the nearest centroid(s) for each feature vector. Assuming that the dimension of the feature vector is M, the number of centroids is N, and the number of feature vectors being processed by the filter instance is K, then the computational complexity is $O(M \times N \times K)$. However, the real operation, which involves the calculation of distances between feature vectors, is quite simple. Therefore, it is ideal for fine grained parallelization by GPU. The GPU implementation for performing this calculation is shown in Fig. 9.

As illustrated in Fig. 9, before the GPU kernel is invoked, the data of centroids and features in the host memory will be transferred to the GPU global memory. The centroid data are stored in column major in order to achieve coalesced read, which can dramatically increase the efficiency of GPU. The calculation of locating the nearest centroid for a feature vector is executed in a thread block. The data of the feature vector will be cached into GPU shared memory to reduce read latency. This implementation of k-means clustering filter can gain the speedup of more than 130 times over a single CPU thread implementation, running on Intel Core i5-3470 with NVIDIA GTX 660 Ti GPU.

B. Case Study: SMK/ASMK/ASMK-Binary

The selective match kernel (SMK) and aggregated SMK (ASMK) are two advanced techniques to improve the accuracy and efficiency of image retrieval [37], inspired by the vector of locally aggregated descriptors [38]. In order to reduce the computational complexity and storage requirement, ASMK-binary that is the binary version of ASMK is also proposed in [37]. The main difference of SMK/ASMK/ASMK-binary from the traditional BoF approach is that the matching kernels to calculate feature vectors' distances are redesigned for performance improvement. All of these approaches still rely on the underlying inverted file structure [1] for efficient similarity computing. Therefore, we concentrate on introducing the generation of inverted file structure with the proposed CHCF in the following.

As we know, the inverted file is an effective structure to represent high-dimensional dictionary-based feature vectors due to its wonderful sparse characteristic. By pruning the zero position of feature vectors, the inverted file structure reduces the storage to place feature vectors as well as the computations on matching. A typical inverted file structure is composed of M (i.e., the number of dictionary vocabularies) lists of descriptor entries, each corresponding to a visual vocabulary with each entry containing the image ID and other information if required, as illustrated in Fig. 10.

In the utility library, a built-in SMK filter is designed to implement the SMK, ASMK, and ASMK-binary algorithms,

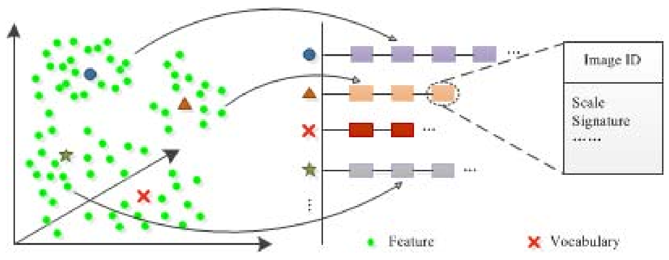


Fig. 10. Illustration of inverted file structure for image retrieval.

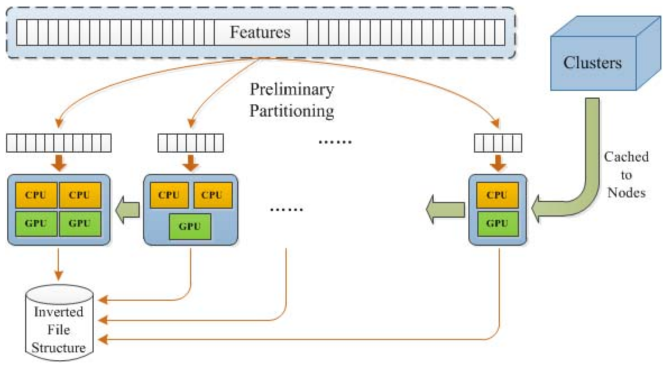


Fig. 11. Illustration of generating the inverted file structure.

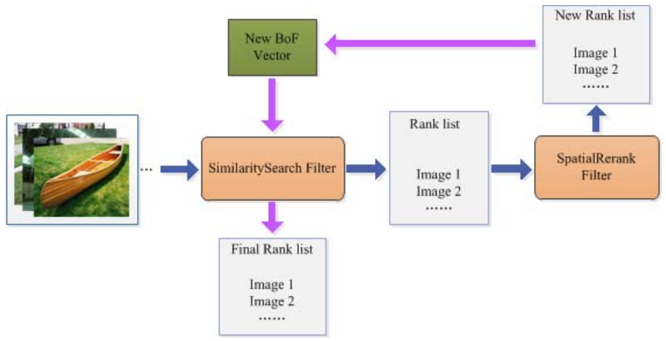


Fig. 12. Overview of the QueryExpansion filter.

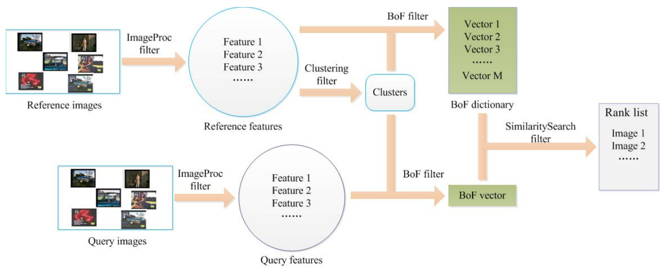


Fig. 13. Flowchart of a baseline image retrieval application.

which shares similarities with the k-means clustering filter in the design of coarse grained parallelization. The SMK filter generates the inverted file structure by finding M-nearest centroids and inserts extra data into each node. To generate the inverted file structure, the SMK, ASMK, and ASMK-binary algorithms are performed by each node with their corresponding feature generation rationale. In order to harness the power of parallel computing on hybrid systems, the CHCF hierarchical adaptive data partitioner splits the input features according to the performance of all available nodes using the input format function. Then, the master scheduler distributes the data splits to node schedulers. After invoked by the node scheduler, the SMK filter instances divide the clusters of feature vectors for fine grained parallelization and perform computations on the specified processors. Meanwhile, the node schedulers keep tracking the execution of the filter instances and update the filter performance table until the task completes. An illustration of generating the inverted file structure by the SMK filter is shown in Fig. 11.

C. Case Study: Query Expansion

Query expansion is an effective technique for improving retrieval performance. Highly ranked results from the preliminary retrieval are combined as a new query to make a fine search. Additional relevant results are then added following those highly ranked ones. There are a number of strategies to implement query expansion. For example, average query expansion (AQE) [39] is a simple but effective method. During AQE, a new query is formed by averaging the top-n verified results from the preliminary search. To filter false results from highly ranked ones, spatial verification can be employed.

In this paper, query expansion is implemented as a built-in filter called QueryExpansion in the utility library of CHCF. Compared with the other built-in filters described above, the QueryExpansion filter is a compound filter as shown in Fig. 12. Actually, this filter is composed of SimilaritySearch and SpatialRerank filters. The preliminary search is performed by SimilaritySearch filter, and then the SpatialRerank filter is used to eliminate false results. The final result is obtained by performing the SimilaritySearch filter again.

D. Case Study: Baseline Image Retrieval

Finally, we will demonstrate how to build a baseline image retrieval application using the proposed CHCF, which is illustrated in Fig. 13.

As shown in Fig. 13, the images are processed by the ImageProc filter at first to get the feature vectors. This filter is not implemented for data parallelism when processing a single image. By partitioning the clusters of features, the BoF filter distributes the tasks of processing each split of clusters among the system. The SimilaritySearch filter involves a similarity search approach, in which the calculation of the distances between the query image and the reference images takes a large amount of computations. It is obvious that this process can be parallelized by dividing the reference images into splits and scheduling the tasks for each split to different filter instances. The boundary of the input format function is defined as an image, which is a natural boundary for the reference image database.

VI. Experimental Evaluation

In this section, we evaluate the design goals of the proposed CHCF for large-scale image retrieval, which are

TABLE I
HARDWARE CONFIGURATION OF CHCF FOR IMAGE RETRIEVAL

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 ×2	GeForce GTX 660 ×2	GeForce GTX 760 ×2
CUDA Cores	1344×2	960×2	1152×2
Network	1Gbps LAN		

TABLE II
IMAGE DATASETS AND THE RELATED DESCRIPTORS

Dataset	# images	# descriptors	descriptor size
Hessian-Affine detector [8] and SIFT descriptor [9]			
Oxford5K	5,063	18,190,561	11.2GB
Flickr100K	100,000	317,565,032	195.0GB
MSR-Bing1M	1,000,000	392,528,043	242.0GB
Modified Hessian-Affine detector [47] and Root-SIFT descriptor [48]			
Oxford5K	5,063	13,759,319	8.5GB
Paris	6,392	16,402,339	10.2GB

TABLE III
SPEEDUP OF k-MEANS CLUSTERING WITH DIFFERENT SIZED VISUAL DICTIONARIES

Size	20K	60K	120K	200K	400K	600K
Speedup	1,226.3	1,685.1	2,018.2	2,366.4	2,624.1	2,678.2

presented by means of scalability analysis (Section VI-B) and high computing performance (Sections VI-C and VI-D), the speedup contribution of the proposed adaptive data partitioner and hierarchical scheduler as well as the employment of GPUs for more heterogeneous computing (Section VI-E) and the comparison with the popular MapReduce [3] and Spark [40] (Section VI-F).

A. Experimental Setup

The experiments are carried out on a 12-node computer cluster. Each node is equipped with one CPU and two GPUs and running the GNU/Linux Ubuntu 12.04 64-bit operating system. The detailed configuration of the cluster is given in Table I. As noted in Table I, a computer node with middle-level configuration is chosen to run a single CPU thread version of all the following evaluations to demonstrate the speedup achieved by employing the proposed CHCF.

A number of algorithms and techniques for large-scale image retrieval are evaluated with the proposed CHCF, including feature extraction, k-means clustering, HE [35], WGC [35], BoF histogram calculation, inverted file structure generation, and SMK/ASMK/ASMK-binary [37]. Several benchmark image datasets are utilized for experiments, including Oxford5K [41], Paris [42], Flickr100K [43], and MSR-Bing1M [44]. Flickr100K is employed as an independent dataset to train visual dictionary vocabularies so as to avoid the advantage of query set, and MSR-Bing1M is used as distracting images to evaluate large-scale image retrieval performances.

In order to verify the retrieval accuracy with the proposed CHCF, two benchmark works [37], [45] are followed for guiding our implementation of image retrieval algorithms. Specifically speaking, [45] is employed to demonstrate the performances of HE, WGC, and online image retrieval, where the Hessian-affine detector [8] and SIFT descriptor [9] are used with the toolkit of [46]. On the other hand, the SMK/ASMK/ASMK-binary algorithms are carried out according to [37], where the modified Hessian-affine detector [47] and the root-SIFT descriptor [48] are employed for interest point detection and description. The information of the image datasets and the related descriptors is shown in Table II.

B. Scalability Analysis

In this section, the scalability of the proposed CHCF is analyzed in terms of the size of input data and the size of the hybrid system. To achieve this, the k-means clustering filter is chosen to perform this evaluation. The features extracted from the Flickr100K dataset are taken as the input to k-means clustering. It is worth pointing out that the time spent for speedup calculation is the overall time of the job execution, which means that the overheads for application startup, shutdown, partitioning, scheduling, and so on are all included. The speedup results with different sized visual dictionaries (i.e., by varying the number of k-means clustering centroids) are shown in Table III.

As shown from the results in Table III, it can be observed that as benefited from the powerful computational capacity of GPU, the application is able to achieve dramatic speedup performances. The speedup steadily increases as the size of the visual dictionaries increases, and reaches a top result of 2678.2 when the size is 600K. This trend reveals the overheads for partitioning, scheduling, and performance estimation during the job execution. When the size of visual dictionary is 20K, the time spent for computations on GPU is relatively small, usually around seconds for a data split. While increasing the visual dictionary size, the amount of GPU computing time is proportionally increased. Meanwhile, the aforementioned additional overheads keep almost the same under different situations. As a result, the overall speedup is further improved with larger dictionary sizes. This reveals the fact that more computational resources will be spent on the real calculation, which leads to a higher efficiency of the system, as the visual dictionary size increases.

Moreover, in order to verify the scalability of CHCF over the number of computer nodes, the experiments of k-means clustering are performed with a varying number of computer nodes, from 4 to 12. We choose a moderate size of visual dictionary, which is 200K, to demonstrate the analysis. The results are shown in Fig. 14 and Table IV, where the parameter R depicts the influence of the number of nodes on the

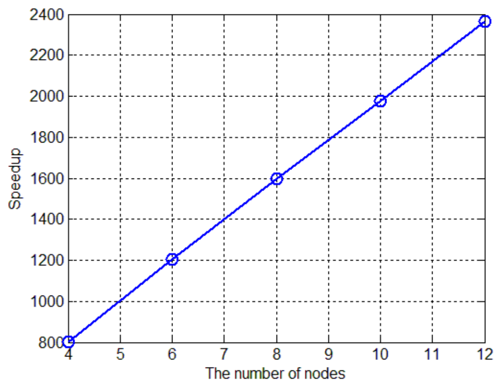


Fig. 14. Speedup relation with different number of computer nodes on k-means clustering.

TABLE IV
SPEEDUP WITH DIFFERENT NUMBER OF COMPUTER NODES WHEN THE SIZE OF VISUAL DICTIONARY IS 200K

# nodes	4	6	8	10	12
Speedup	803.4	1,205.1	1,598.7	1,975	2,366.4
R	1	1	0.995	0.983	0.982

system efficiency as defined as

$$R = \frac{S_i / i}{S_4 / 4} \quad (1)$$

where i indicates the number of computer nodes employed for computing and S_i stands for the speedup performance when i computer nodes are used. As inferred from (1), the higher the R value is, the more the efficiency of the system becomes.

As shown in Fig. 14 and Table IV, the speedup increases almost linearly as the number of nodes increases, which indicates that the proposed CHCF achieves perfect scalability in terms of the size of hybrid system under the current hardware configuration. On the other hand, the small deviation of the R value from 1 is due to the reason that there is a reduce part in the k-means clustering, which takes more time as the number of nodes increases, so that R is lower than 1 indicating that the efficiency of the system declines. Note that the scalability is currently investigated only with a relatively small-scale system. The almost linear relation as indicated in Fig. 14 and Table IV is under the current 12-node computer cluster setting, and it may be different under real large-scale system settings.

C. Image Retrieval on Oxford5K

In this section, the Oxford5K dataset is used to evaluate the computational performance as well as the retrieval accuracy in terms of mean average precision (mAP) of our CHCF-based image retrieval system. In order to realize large-scale image retrieval, the MSR-Bing1M image dataset is utilized for the purpose of distracting images. The computational speedup performances are listed in Table V for a number of typical built-in CHCF filters. In particular, the ImageProc filter aims to extract features from images; the BoF filter is employed to calculate histogram alike BoFs; the HETrain filter is used to train the HE model; and the InvFlGen filter is responsible for inverted file structure generation.

TABLE V
SPEEDUP ACHIEVED BY TYPICAL CHCF FILTERS ON Oxford5K RETRIEVAL WITH DIFFERENT SIZED VISUAL DICTIONARIES

	20K	200K	1M
ImageProc	32.2	32.2	32.2
BoF	1,432.3	2,407.9	2,687.2
HETrain	515.9	662.4	725.2
InvFlGen	531.0	661.3	731.4

TABLE VI
QUERY TIME PER IMAGE FOR A ONE MILLION IMAGE DATASET

	20K	200K
Search, Baseline	0.25 s	0.23 s
Search, HE	0.32 s	0.29 s
Search, WGC	1.23 s	0.41 s
Search, HE+WGC	1.25 s	0.43 s

TABLE VII
RETRIEVAL ACCURACY ACHIEVED BY CHCF IN TERMS OF mAP

	20K		200K		1M
Distractors	0	1M	0	1M	0
Baseline	0.397	0.330	0.461	0.284	0.502
HE	0.588	0.448	0.625	0.383	0.612
WGC	0.533	0.363	0.542	0.266	0.544
HE+WGC	0.657	0.427	0.656	0.363	0.618

As seen from the results, the speedup performances of the ImageProc filter remain the same under all test cases, because this filter is totally unrelated to the visual dictionary size. In addition, compared with other filters, the speedup of the ImageProc filter is relatively low because it owns only CPU implementation at present. The speedup of the BoF filter is commensurate with that of k-means clustering, because the procedures of these two algorithms are similar and both are ideal for data parallelism. Moreover, along with the increment of the data, a more significant speedup can be achieved by the CHCF filters.

The average query time per image with the 1M MSR-Bing1M distracting images is shown in Table VI. From the result, we can observe that the query time is usually less than 2 s owing to the processing power of CHCF.

Table VII shows the retrieval accuracy in terms of mAP achieved by CHCF for different sized vocabularies. Compared with [45] (which is used as the benchmark to guide our implementation of the corresponding built-in filters), the mAP result of CHCF is better under all test conditions, with a minimum improvement of 0.03, a maximum improvement of 0.16, and an average improvement of 0.11. This significant improvement of mAP is mainly due to the reason that much more multimedia data can be processed by CHCF for mining.

TABLE VIII
SPEEDUP AND mAP PERFORMANCES ACHIEVED BY THE SMK FILTER

		ASMK	SMK	ASMK-Binary
Speedup	Oxford5K	355.4	384.1	410.2
	Paris	347.2	390.5	424.0
mAP	Oxford5K	0.817(0.000)	0.795(0.021)	0.800(-0.004)
	Paris	0.785(0.003)	0.744(0.026)	0.788(0.018)

D. Extension With SMK/ASMK/ASMK-Binary

In this section, the performances of the SMK filter are presented by strictly following the benchmark work of [37] including the SMK, ASMK, and ASMK-binary algorithms. Two image datasets including Oxford5K and Paris are utilized for performance evaluation. As specified in [37], a 65K sized visual dictionary is generated by k-means clustering for performing image retrieval on these two datasets. The visual dictionary trained on Paris is applied for Oxford5K image retrieval and vice versa. In addition, the modified Hessian-affine detector [47], the root-SIFT descriptor [48], and the technique of multiple assignment [45] are employed for feature generation.

Both of the speedup and mAP performances are presented in Table VIII, where the mAP values in the parenthesis indicate the improvement of our implementation against that of [37]. Since our implementation in this experiment follows [37], the mAP performances of these two works should be similar to each other, which is verified by the small mAP improvements as shown in the parenthesis in Table VIII. As far as the computational performance is concerned, significant speedup ratios can be achieved by the SMK filter for all the test cases.

E. Contribution Analysis

In this section, first, we evaluate the contribution of the proposed adaptive data partitioner, knowledge-based hierarchical scheduler, and performance estimator, which are designed to work jointly to address the challenge of load imbalance. To this aim, the comparative experiments are carried out on the filters of k-means, BoF, HETrain, and InvFlGen with different sized visual dictionaries. The performance degradation of these filters caused by disabling the adaptive partitioner, knowledge-based hierarchical scheduler, and performance estimator is shown in Table IX, where the performance degradation measure ΔT is presented as

$$\Delta T = \frac{T_d - T_o}{T_o} \times 100\% \qquad (2)$$

where T_d and T_o stand for the running time of CHCF filters when disabling and enabling the adaptive data partitioner, knowledge-based hierarchical scheduler, and performance estimator, respectively. Note that when the adaptive data partitioner, knowledge-based hierarchical scheduler, and performance estimator are disabled, CHCF currently employs the fixed-size partitioning method for data split and static mapping strategy for task scheduling.

As shown in Table IX, the trends of ΔT among the filters are similar. As the size of the visual dictionaries is increased, the degradation gets worse and enters a relatively stable zone after the size is greater than 120K. This is because when the amount of data being processed is increased, the percent of additional computational overheads becomes relatively small and more computational resources are spent on the real calculations as discussed in Section VI-B; on the other hand, the proposed adaptive data partitioner, knowledge-based hierarchical scheduler, and performance estimator stably work in terms of the performance of running time. Therefore, the degradation performance result ΔT becomes stable when the amount of processed data reaches to a certain level. In addition, the performance degradations under all test situations are relatively small as shown in Table IX, because the scale of the current experimental system is limited and the variation of the capacities of underlying computational resources is small, which can be discerned from Table I.

In order to further illustrate the computing power of heterogeneous computing, the comparison between CHCF with CPU+GPU and CHCF with CPU only (i.e., heterogeneous versus CPU-only) is performed on the filters of k-means, BoF, HETrain, and InvFlGen with different sized visual dictionaries. The comparison results are shown in Tables X and XI, where it can be observed that significant speedup performances can be achieved when GPUs are enabled by CHCF for more heterogeneous computing.

TABLE IX
PERFORMANCE DEGRADATION WHEN DISABLING THE ADAPTIVE DATA PARTITIONER, KNOWLEDGE-BASED HIERARCHICAL SCHEDULER, AND PERFORMANCE ESTIMATOR WITH DIFFERENT SIZED VISUAL DICTIONARIES

	20K	60K	120K	200K	400K
K-Means	7.3%	7.9%	8.6%	9.3%	10.5%
BoF	9.7%	12.9%	13.2%	13.6%	14.1%
HETrain	8.7%	9.9%	12.3%	13.1%	13.2%
InvFlGen	12.1%	14.4%	14.6%	14.6%	14.9%

TABLE X
SPEEDUP OF HETEROGENEOUS OVER CPU ONLY ON Oxford5K WITH DIFFERENT SIZED VISUAL DICTIONARIES

	20K	200K	1M
BoF	28.9	51.5	62.4
HETrain	16.7	20.8	22.0
InvFlGen	18.0	20.4	21.1

TABLE XI
SPEEDUP OF HETEROGENEOUS OVER CPU ONLY ON Flickr100K WITH DIFFERENT SIZED VISUAL DICTIONARIES

	20K	60K	120K	200K	400K
K-Means	25.3	40.2	48.7	54.2	64.9

F. Comparison With MapReduce and Spark

The comparative experiments of the proposed CHCF with MapReduce [3] and Spark [40] are also carried out to further demonstrate the advantage of the proposed CHCF.

TABLE XII
SPEEDUP ACHIEVED BY CHCF OVER MapReduce AND Spark WITH DIFFERENT SIZED VISUAL DICTIONARIES ON Oxford5K

		20K	200K	1M
BoF	MapReduce	30.2	56.8	66.8
	Spark	29.2	51.8	63.3
HETrain	MapReduce	18.0	21.9	22.5
	Spark	16.9	21.0	22.1
InvFlGen	MapReduce	19.0	21.3	22.7
	Spark	18.1	20.7	22.3

TABLE XIII
SPEEDUP OF k-MEANS CLUSTERING ACHIEVED BY CHCF OVER MapReduce AND Spark WITH DIFFERENT SIZED VISUAL DICTIONARIES ON Flickr100K

		20K	60K	120K	200K	400K
K-Means	MapReduce	26.7	43.1	51.2	56.9	67.2
	Spark	25.8	41.7	49.5	55.8	65.3

As Hadoop [31] is the most famous open source implementation of MapReduce, the algorithms required by the comparative experiments, including k-means clustering, BoF, HETrain, and InvFlGen, are implemented on Hadoop 1.0.4 with Java language. Regarding the implementation with Spark, the same set of algorithms is developed with Java language on Spark 1.1.1.

As shown in Tables XII and XIII, the speedup performance improvement achieved by CHCF over both MapReduce and Spark is significant mainly because more heterogeneously computational resources (e.g., GPUs) are harnessed by CHCF. Similar to the performance trends on speedup presented in Tables III, V, X, and XI, when the amount of data being processed increases, a more significant speedup can be achieved by CHCF compared with MapReduce and Spark. In addition, as seen from the comparisons presented in Tables XII and XIII, Spark is only slightly faster than Hadoop for the algorithms of k-means clustering, BoF, HETrain, and InvFlGen. This is because these four algorithms are CPU intensive, which means that a relatively small portion of computing time is spent on data loading. Therefore, the Spark in-memory technique [40] that can efficiently speed up data loading compared with Hadoop becomes trivial for these CPU-intensive algorithms, and thus the advantage of Spark over Hadoop is reduced.

VII. CONCLUSION

In this paper, an efficient CHCF is proposed, which is capable of executing multimedia mining applications on heterogeneous computer clusters. The proposed CHCF provides an optimized compiler, which can work in conjunction with Java and C/C++ compilers to build CHCF applications. A high-performance utility library that contains a set of multimedia mining algorithms, such as feature extraction, k-means clustering, BoF, similarity search, HE, WGC, and SMK, is provided to facilitate the development of multimedia mining applications. CHCF employs an adaptive partitioner, a knowledge-based hierarchical scheduler, and a performance estimator to maximize the system throughput and performance. With the help of input format function and the boundary abstraction, the adaptive partitioner could predict the best partitioning of the input data. By employing the hierarchical strategy, the scheduler significantly reduces redundant communication costs among the system, which in turn enhances the scheduling efficiency. The performance assessment obtained from performance estimator is important because both the partitioner and scheduler make decisions based upon it. Extensive experiments in terms of image retrieval have verified the scalability, real-time performance, and programmability of the proposed CHCF.

In the future, more research efforts and attempts will be put to enrich the utility library, optimize the scheduler and the partitioner to further eliminate computation and communication overheads, and improve the compiler for static scheduling during application compilation and releasing. In addition, it is also desired to employ CHCF to investigate other multimedia data mining applications besides large-scale image retrieval, such as video event detection, large-scale video retrieval, and so on.

REFERENCES

[1] J. Sivic and A. Zisserman, “Video Google: A text retrieval approach to object matching in videos,” in *Proc. 9th IEEE ICCV*, Oct. 2003, pp. 1470–1477.

[2] B. White, T. Yeh, J. Lin, and L. Davis, “Web-scale computer vision using MapReduce for multimedia data mining,” in *Proc. 10th Int. Workshop MDMKDD*, Jul. 2010, Art. ID 9.

[3] J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[4] NVIDIA Corp. *NVIDIA Developer Zone*. [Online]. Available: https://developer.nvidia.com/cuda-zone, accessed 2015.

[5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: Binary robust independent elementary features,” in *Proc. 11th ECCV*, Sep. 2010, pp. 778–792.

[6] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” in *Proc. IEEE ICCV*, Nov. 2011, pp. 2564–2571.

[7] J. Heinly, E. Dunn, and J.-M. Frahm, “Comparative evaluation of binary features,” in *Proc. 12th ECCV*, Oct. 2012, pp. 759–773.

[8] K. Mikolajczyk and C. Schmid, “Scale & affine invariant interest point detectors,” *Int. J. Comput. Vis.*, vol. 60, no. 1, pp. 63–86, Oct. 2004.

[9] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[10] D. Nister and H. Stewenius, “Scalable recognition with a vocabulary tree,” in *Proc. IEEE Comput. Soc. Conf. CVPR*, Jun. 2006, pp. 2161–2168.

[11] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, “Object retrieval with large vocabularies and fast spatial matching,” in *Proc. IEEE Conf. CVPR*, Jun. 2007, pp. 1–8.

[12] E. Hörster, R. Lienhart, and M. Slaney, “Image retrieval on large-scale image databases,” in *Proc. 6th ACM Int. CIVR*, Jul. 2007, pp. 17–24.

[13] H. Wang, S. Zhang, and W. Liang, “Three components for large scale image retrieval,” in *Proc. IASP*, Nov. 2012, pp. 1–5.

[14] Y.-G. Jiang, J. Wang, X. Xue, and S.-F. Chang, “Query-adaptive image search with hash codes,” *IEEE Trans. Multimedia*, vol. 15, no. 2, pp. 442–453, Feb. 2013.

[15] D. Moise, D. Shestakov, G. Gudmundsson, and L. Amsaleg, “Indexing and searching 100 M images with map-reduce,” in *Proc. 3rd ACM Conf. ICMR*, Apr. 2013, pp. 17–24.

[16] L. Kennedy, M. Slaney, and K. Weinberger, “Reliable tags using image similarity: Mining specificity and expertise from large-scale multimedia databases,” in *Proc. 1st Workshop WSMC*, Oct. 2009, pp. 17–24.

[17] C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier, “StarPU: A unified platform for task scheduling on heterogeneous multicore architectures,” in *Proc. Euro-Par*, Aug. 2009, pp. 863–874.

[18] M. D. Linderman, J. D. Collins, H. Wang, and T. H. Meng, “Merge: A programming model for heterogeneous multi-core systems,” in *Proc. 13th Int. Conf. ASPLOS*, Mar. 2008, pp. 287–296.

[19] J. R. Wernsing and G. Stitt, “Elastic computing: A portable optimization framework for hybrid computers,” *Parallel Comput.*, vol. 38, no. 8, pp. 438–464, Aug. 2012.

[20] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang, “Mars: A MapReduce framework on graphics processors,” in *Proc. 17th Int. Conf. PACT*, Oct. 2008, pp. 260–269.

[21] H.-F. Li, T.-Y. Liang, and J.-L. Liang, “An OpenMP compiler for hybrid CPU/GPU computing architecture,” in *Proc. 3rd Int. Conf. INCoS*, Nov./Dec. 2011, pp. 209–216.

[22] *The OpenMP Forum, OpenMP C and C++ Application Program Interface, Version 1.0*. [Online]. Available: http://www.openmp.org/, accessed 2015.

[23] A. Ghoting *et al.*, “SystemML: Declarative machine learning on MapReduce,” in *Proc. IEEE 27th ICDE*, Apr. 2011, pp. 231–242.

[24] W. Ma and G. Agrawal, “AUTO-GC: Automatic translation of data mining applications to GPU clusters,” in *Proc. IEEE IPDPSW*, Apr. 2010, pp. 1–8.

[25] S. Lee, S.-J. Min, and R. Eigenmann, “OpenMP to GPGPU: A compiler framework for automatic translation and optimization,” in *Proc. 14th ACM SIGPLAN Symp. PPoPP*, Apr. 2009, pp. 101–110.

[26] O. Beaumont, A. Legrand, L. Marchal, and Y. Robert, “Assessing the impact and limits of steady-state scheduling for mixed task and data parallelism on heterogeneous platforms,” in *Proc. ISPDC*, Jul. 2004, pp. 296–302.

[27] T. D. R. Hartley, E. Saule, and U. V. Çatalyürek, “Improving performance of adaptive component-based dataflow middleware,” *Parallel Comput.*, vol. 38, nos. 6–7, pp. 289–309, Jun./Jul. 2012.

[28] I. Riakiotakis, F. M. Ciorba, T. Andronikos, and G. Papakonstantinou, “Distributed dynamic load balancing for pipelined computations on heterogeneous systems,” *Parallel Comput.*, vol. 37, nos. 10–11, pp. 713–729, Oct./Nov. 2011.

[29] M. Beynon, R. Ferreira, T. Kurc, A. Sussman, and J. Saltz, “DataCutter: Middleware for filtering very large scientific datasets on archival storage systems,” in *Proc. ISMSS*, Mar. 2000, pp. 119–133.

[30] N. T. Bhatti, M. A. Hiltunen, R. D. Schlichting, and W. Chiu, “Coyote: A system for constructing fine-grain configurable communication services,” *ACM Trans. Comput. Syst.*, vol. 16, no. 4, pp. 321–366, Nov. 1998.

[31] *Hadoop*. [Online]. Available: http://hadoop.apache.org/, accessed 2014.

[32] *Protocol Buffers*. [Online]. Available: https://developers.google.com/protocol-buffers/, accessed 2013.

[33] M. Henning, “A new approach to object-oriented middleware,” *IEEE Internet Comput.*, vol. 8, no. 1, pp. 66–75, Jan./Feb. 2004.

[34] G. A. Reis, J. Chang, N. Vachharajani, R. Rangan, D. I. August, and S. S. Mukherjee, “Software-controlled fault tolerance,” *ACM Trans. Archit. Code Optim.*, vol. 2, no. 4, pp. 366–396, Dec. 2005.

[35] H. Jégou, M. Douze, and C. Schmid, “Hamming embedding and weak geometric consistency for large scale image search,” in *Proc. 10th ECCV*, Oct. 2008, pp. 304–317.

[36] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[37] G. Tolias, Y. Avrithis, and H. Jégou, “To aggregate or not to aggregate: Selective match kernels for image search,” in *Proc. IEEE ICCV*, Dec. 2013, pp. 1401–1408.

[38] H. Jégou, M. Douze, C. Schmid, and P. Pérez, “Aggregating local descriptors into a compact image representation,” in *Proc. IEEE Conf. CVPR*, Jun. 2010, pp. 3304–3311.

[39] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman, “Total recall: Automatic query expansion with a generative feature model for object retrieval,” in *Proc. IEEE 11th ICCV*, Oct. 2007, pp. 1–8.

[40] *Spark*. [Online]. Available: http://spark.apache.org/, accessed 2015.

[41] *Oxford5K Dataset*. [Online]. Available: http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/, accessed 2012.

[42] *Paris Dataset*. [Online]. Available: http://www.robots.ox.ac.uk/~vgg/data/parisbuildings/, accessed 2008.

[43] *Flickr100K Dataset*. [Online]. Available: http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/flickr100k.html, accessed 2012.

[44] *MSR-Bing Dataset*. [Online]. Available: http://web-ngram.research.microsoft.com/GrandChallenge/Datasets.aspx, accessed 2013.

[45] H. Jégou, M. Douze, and C. Schmid, “Improving bag-of-features for large scale image search,” *Int. J. Comput. Vis.*, vol. 87, no. 3, pp. 316–336, May 2010.

[46] *Toolkit of Hessian-Affine Detector*. [Online]. Available: http://www.robots.ox.ac.uk/~vgg/research/affine/, accessed 2007.

[47] M. Perd’och, O. Chum, and J. Matas, “Efficient representation of local geometry for large scale object retrieval,” in *Proc. IEEE Conf. CVPR*, Jun. 2009, pp. 9–16.

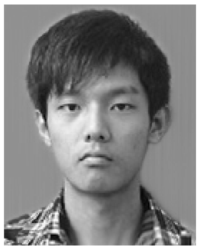
[48] R. Arandjelović and A. Zisserman, “Three things everyone should know to improve object retrieval,” in *Proc. IEEE Conf. CVPR*, Jun. 2012, pp. 2911–2918.

Hanli Wang (M’08–SM’12) 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, Hong Kong, in 2007.

He was a Research Fellow with the Department of Computer Science, City University of Hong Kong, from 2007 to 2008. From 2007 to 2008, he was a Visiting Scholar with Stanford University, Stanford, CA, USA, invited by Prof. C. K. Chui. From 2008 to 2009, he was a Research Engineer with Precoad, Inc., Menlo Park, CA, USA. From 2009 to 2010, he was an Alexander von Humboldt Research Fellow with the University of Hagen, Hagen, Germany. In 2010, he joined the Department of Computer Science and Technology, Tongji University, Shanghai, China, as a Professor. He has authored over 80 papers in his research fields. His research interests include digital video coding, image processing, computer vision, and machine learning.

Bo Xiao received the B.S. degree in thermal engineering from Tongji University, Shanghai, China, in 2004, where he is currently working toward the Ph.D. degree with the Department of Computer Science and Technology.

His research interests include multimedia data mining, video coding, heterogeneous computing, and cloud computing.

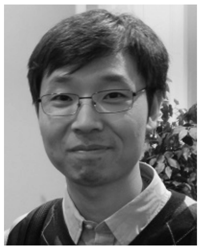


Lei Wang received the B.S. degree in information security from Tongji University, Shanghai, China, in 2012, where he is currently working toward the Ph.D. degree with the Department of Computer Science and Technology.

His research interests include computer vision and multimedia content analysis.

Fengkuangtian Zhu received the B.S. and M.S. degrees in computer science and technology from Tongji University, Shanghai, China, in 2012 and 2015, respectively.

His research interests include computer vision and multimedia content analysis.



Yu-Gang Jiang received the Ph.D. degree in computer science from City University of Hong Kong, Hong Kong, in 2009.

He was with the Department of Electrical Engineering, Columbia University, New York, NY, USA, from 2008 to 2011. He is currently an Associate Professor of Computer Science with Fudan University, Shanghai, China. He has authored over 80 papers in his research fields. His research interests include computer vision and multimedia retrieval.

Dr. Jiang's work has led to many awards, including the 2014 ACM China Rising Star Award and the 2015 ACM SIGMM Rising Star Award. He is an active participant of the annual NIST TRECVID and MediaEval evaluations and has designed a few top-performing video retrieval systems. He is an Associate Editor of *Machine Vision and Applications*.

Jun Wu (M'05–SM'14) received the B.S. degree in information engineering and the M.S. degree in communication and electronic system from Xidian University, Xi'an, China, in 1993 and 1996, respectively, and the Ph.D. degree in signal and information processing from Beijing University of Posts and Telecommunications, Beijing, China, in 1999.

He was a Principal Scientist with Huawei, Shenzhen, China, and Broadcom, Irvine, CA, USA. He joined Tongji University, Shanghai, China, as a Professor in 2010. His research interests include wireless communication, information theory, and multimedia signal processing.