Combination Feature for Image Retrieval in the Distributed Datacenter

Di Yang, Jianxin Liao, Qi Qi, Jingyu Wang, Haifeng Sun and Shan Jiang
State Key Lab of Networking and Switching Technology, Beijing University of Posts and Telecommunications
Beijing 100876, China
yangdi.bupt@gmail.com, {liaojianxin, qiqi,wangjingyu, sunhaifeng}@ebupt.com, i@ajiangshan.cn

***Abstract*—Since the emergence of cloud datacenters provides an enormous amount of resources easily accessible to people, it is challenging to provide an efficient search framework in such a distributed environment. However, traditional search techniques only allow users to search images over exact-match keywords through a centralized index. These methods are insufficient to meet requirements of content based image retrieval (CBIR) and more powerful search frameworks are needed. In this paper, we present LCFIR, an effective image retrieval framework for fast content location in the distributed situation. It adopts the peer-to-peer paradigm and combines color and edge features. The basic idea is to construct multiple replicas of an image's index through exploiting the property of Locality Sensitive Hashing (LSH). Thus, the indexes of similar images are probabilistically gathered into the same node without the knowledge of any global information. The empirical results show that the system is able to yield high accuracy with load balancing, and only contacts a few number of the participating nodes.**

Keywords—Cloud computing; Content based image retrieval; Peer-to-peer; Locality sensitive hashing; Combination feature

I. INTRODUCTION

Nowadays, since smart phones, tablet computers and many lightweight devices have been penetrating into our lives, the digital image equipments on devices enable end-users to capture and edit their own image content, sometimes of high intellectual or commercial value. With the widespread use of devices, more and more images are being shared and circulated over the internet. One example of such an environment is the "cloud" that store a large number of resources collected from all around world. It allows users to access resources more flexibly, when storage and computation is switched from the clients to the "clouds" [1]. The paradigm offers essential properties including location independent data storage, on-demand self service, and data access independent of locations and time [2].

Cloud-based services rely on the datacenter which is a fundamental block and provides highly durable storage [3]. Therefore, it is important to choose an appropriate topology to establish the high-performance datacenters that satisfy the requirements of searching and analyzing large dispersive datasets. Most commercial cloud infrastructures are centralized. In such a model, resources in datacenters are centrally managed and the central points are vulnerable to failures caused by fires, power outages, natural disasters, etc [4]. To address this problem, decentralized datacenters are designed according to the peer-to-peer (P2P) model, which can provide better scalability and adaptability. The P2P-based datacenter can be built by connecting many individual peers, without any central monitoring or coordination component. Each peer takes charge of part of data and replicas to improve the clouds' reliability and the datacenter's resilience to correlated failures. It is well known that P2P techniques are very likely to be adopted in Clouds [5].

The performance of P2P paradigm mainly depends on the topological structure and the placement of indexes. Unstructured P2P system like Gnutella [6] has little control over the topology and the placement of indexes. This structure does not guarantee the efficiency of data search and causes high communication cost, since the query is spread without any global planning. In contrast, structured P2P has tight control over the overlay topology, and only publishes the files' indexes to special nodes through distributed hash table (DHT). Chord [7], CAN [8], and BATON [9] are classical examples of structure P2P networks. In DHTs, files' names are generally hashed into keys and the queries are forwarded to target nodes based on the keys. The number of lookup hops to locate files can be limited to the logarithmic number by using deterministic routing algorithm. The DHT structure is adopted in our work due to its great extensibility.

Considering an application scenario of image retrieval in the P2P datacenter, millions of files including images, videos and plain texts are transferred into the datacenter, as Figure 1 shown. The structure of the datacenter uses a decentralized P2P paradigm. The content providers upload their significant resources to P2P datacenters. In addition, cloud customers can also upload interesting images to their Facebooks or Microblogs which are deployed in cloud datacenters. When an authorized cloud customer issues a query request, it is sent to the datacenter, which takes charge of search processing. Afterwards, the query results are sent back to the cloud customer. However, it is challenging to locate files in such a distributed datacenter, which stores a large amount of files. As a case of study, we present an image retrieval application implemented on the P2P datacenter. Although throughout this paper we focus on image retrieval, our methods are applicable to multimedia retrieval domain where similar search is performed in a

978-1-4799-7615-7/14/$31.00 ©2014 IEEE

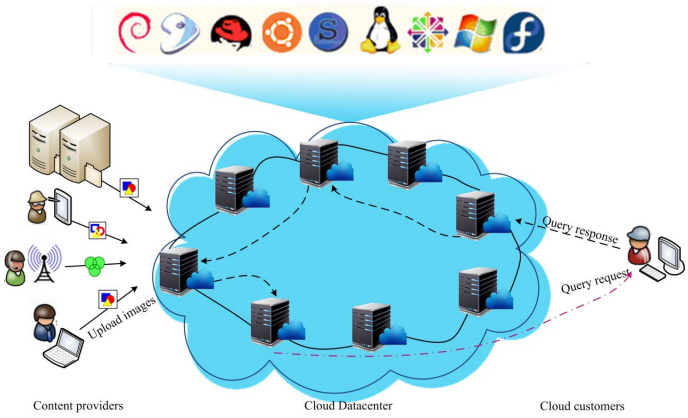

Fig. 1. Image retrieval in the P2P datacenter.

P2P paradigm.

Existing models, which are usually centralized, are not scalable enough to work in a large scale distributed setting. Additionally, most of them only offer keyword search, i.e. they return the images that are associated with a given set of keywords [6] [10]. However, since it is difficult to annotate images very exactly, identifying images in this way is inaccurate and may be insufficient to users' requirement in some cases. In some applications, users may request inexact queries such as "find the top-*k* images which are most similar to a given sample". However, it is difficult for humans to describe how an image is similar to the given sample with keywords [11]. The content based image retrieval (CBIR) can find similar images through the sample images instead of keywords. In our previous work, we present a CBIR system called LRFIR [35] based on a single feature for the distributed environment. It only adopts the texture feature to represent images. However, such a feature can hardly describe the content of images at various angles.

In this paper, we present a novel CBIR system called LSH-based and Combination Features for Image Retrieval (LCFIR) for the distributed P2P datacenter. LCFIR supports content similar search leveraging the early feature combination algorithm. The efficient index construction service (ICS) and query processing service (QPS) are proposed for LCFIR.

In the index construction service, LCFIR leverages color difference histogram (CDH). It combines edge orientation, color, and the spatial layout to generate a smaller but more effective feature, from which the content similarity can be measured quantitatively. LCFIR operates on top of a DHT, which has properties such as scalability and self-organization. Inspired by the property of the locality sensitive hashing (LSH), feature vectors of similar images are probabilistically assigned to the same hash values [12]. Then, the hash value denoted by an integer vector is mapped to a resource *ID* without destroying the locality-preserving property. In this way, the indexes of similar images are more likely published into the same nodes in the DHT layer. When a query is propagated to datacenters, the query processing service produces a set of query messages based on the same LSH functions. The messages are selectively routed to nodes which are more likely responsible for the results. Finally, we implement a prototype system based on Next Generation Service Overlay Network [13]. It is used to evaluate the performance of our algorithm in Corel10000 image dataset [14]. The experiments show that our scheme achieves high accuracy with only a small number of lookup hops.

In this paper, our main contributions are as follows: (1) the combination feature is integrated into DHT for implementing an efficient indexing and locating approach. The index construction is based on image content represented by the CDH instead of a single feature; (2) LCFIR clusters the indexes of the similar images to the same node by exploiting the property of LSH, without any global information. In this way, a query can be answered by visiting a few numbers of nodes. That brings down the communication cost without degrading query accuracy; (3) The work is a novel approach to bridge the gap between the computer vision domain and the distributed systems domain.

The rest of this paper is organized as follows. Section II shows an overview of related work. Section III presents the framework of LCFIR. Section IV describes the index construction service and the query processing service. Section V evaluates the performance of our approach. Finally, Section VI concludes our discussion.

II. Related Work

The P2P network is categorized into three models: unstructured, hybrid and structured. The organizing structures and routing mechanisms for information retrieval in the P2P network also hold for the area of image retrieval.

Searching over unstructured P2P system like Gnutella [6] floods the query to all neighbor nodes. Lv et al. [15] propose random walks to improve the search performance of flooding. At each step, random walks randomly choose one of neighbor nodes to forward query messages to the neighbors, without considering the resource statistical information of neighbor nodes. To overcome the blind search, the concept of "Routing Indexes" is introduced by Crespo and Garcia-Molina [16]. Its basic idea is that query messages are forwarded to the neighbor nodes that are more likely to have the required answers. To avoid the search to be trapped around the local optimum, Gaeta & Sereno [17] choose the neighbor node to forward the query, according to the probability function of the number of connections and the distance from the query originator. However, these algorithms do not guarantee the lookup time and consume too much network resources. They are only suited for multimedia retrieval based on their names or short textual description. Therefore, the search accuracy is limited to the accuracy of text tags and the content of the multimedia is ignored.

Since unstructured P2P has little control over network topology, the hybrid infrastructures are proposed, which gather peers storing relevant files in the same community to reduce the unnecessary traffic. Many methods employing this model can be found, such as SETS [10], metric space [18], interesting-based location solution [19], source selection [20], DISCOVIR [21], P2P-CBIRM [22], and SWIM [23]. Bawa et

al. propose a topic-segmented overlay which assigns nodes with similar contents (topics) to the same group. But this method needs center nodes to manage topics segment and may suffer the single point of failure. Vlachou et al. propose that peers sharing similar data are linked to the same super node, while super nodes are organized as an M-Tree structure. But this method still needs centralized management within the community. The decentralized interesting-based location solution loosely organizes peers into an interesting-based structure for fast content location, where each peer creates an interesting-based shortcut to another peer with interesting content. But it still relies on the message flooding when there is no shortcut available. Eisenhardt et al. focus on the study of source selection, where each peer gathers its own data into clusters. When queries are issued, these clusters serve as the potential sources for the selection of interesting data. DISCOVIR links peers with similar data using attractive connections, which is independent of message flooding. However, when a new peer joins DISCOVIR, it has to broadcast its signature messages through attractive connections to find out peers sharing the similar content with the new one. P2P-CBIRM adopts the similar way of grouping peers, but extends DISCOVIR to support the capability of knowledge discovery and image data mining. The small world indexing mine (SWIM) creates a small world network for images which are connected according to the MPEG-7 descriptor similarities. However, due to the lack of global information, it is difficult for these methods to discover the new topics no longer belonging to the current cluster, without broadcasting signature messages to the overall network. And the query may not be forwarded to the most similar clusters.

In this paper, we focus on the information retrieval in the decentralized structured P2P paradigm. There are many studies in this issue, such as MCAN [24], M-Chord [25], Psearch [26], Prism [27] and iDISQUE [28]. MCAN using CAN as the underlying structure adopts a pivot technique to map data objects to an *N*-dimensional vector space. But the chosen pivots are preprocessed in a centralized fashion, and then distributed to peers. M-Chord takes the advantage of the iDistance which maps objects into one-dimensional space. But its data clustering and mapping are still completed in a centralized manner. In Psearch, the Latent Semantic Indexing (LSI) is used to generate a semantic space. Then, this space is mapped to a multi-dimensional CAN which has the same dimension as the data space. However, different overlays may have different dimensionalities, since the dimensionality of CAN depends on the dimensionalities of various datasets. In Prism, it stores multiple indexes for one object in many Chord peers based on the distances between the object's vector and reference vector, so that indexes of similar objects are clustered to the same peer. But reference vectors are still chosen in a centralized fashion, which is not well-suited for large datasets. Zhu et al. [29] generate the same index for semantically close files by LSH and Vector Space Model (VSM), with the purpose of answering queries by visiting a small number of nodes. But these hash values are directly used as resource keys, which destroy the load balancing of Chord. In the iDISQUE framework, the data on each peer is clustered, and then LSH functions only map cluster centers to Chord resource keys. The key of the cluster center represents the data in the cluster. However, the hash values of queries may not be equal to these of cluster centers.

Considering fusion features, Wang et al. [30] propose a CBIR method based on an efficient integration of color and texture features. The integration provides a robust feature set for color image retrieval. Snoek et al. [31] study of the two classes of fusion schemes, namely early fusion and late fusion. And they compare the performance of the two kinds of fusion. Tian et al. [32] construct the Edge orientation difference histogram (EODH) descriptor for each edge pixel. And they integrate EODH with Color-SIFT to build weighted codeword distribution.

III. SYSTEM FRAMEWORK

In this section, we present an overview of the LCFIR framework. It is expected that the novel framework should support CBIR in P2P datacenters storing a large number of data. Under such an environment, the simple solution of travers in g all the participating nodes for an image query is impractical, due to the high communication cost. Similarly, establishing and maintaining a central index of all the shared images can lead to scalability and reliability concerns [27]. Departing from conventional approaches, we propose a scalable scheme, called LCFIR, which supports content similar image retrieval in a distributed network, utilizing routing indexes based on feature information.

In the underlying layer of LCFIR, a large number of nodes are organized into the structured P2P network (Chord [7]), to offer the routing service. Inside the network, the *ID* space generated through uniform hash functions is ranged from 0 to 2^s-1.where s=160. Each node is assigned an *ID* drawn from this space, and is responsible for the object key range between its *ID* and the *ID* of the previous node on the ring. In an n-node network, Chord on average routes a message to its destination in $O(\log n)$ hops. LCFIR can support efficient routing, due to the DHT layer, where indexes publishing and queries routing are automatically accomplished.

Each node in LCFIR maintains its own data objects. To publish resources to the network, for each object, LCFIR constructs a few index messages, each of which contains the resource *ID*, the image feature vector and the *nodeID* of the data owner. The indexes are sent to the nodes responsible for the resource *ID*. When a query message is submitted, it is only forwarded to nodes which are likely to responsible for the indexes of similar objects.

The challenge is how to establish the distributed index, and probabilistically gather the index of content similar images to the same node, due to the lack of global information. To solve the problem, we build image indexes through a family of LSH functions. Due to the property of LSH's locality sensitive, it is more likely to assign similar features to the same buckets of the hash table. Then they are mapped into resource *IDs*, without loss of local sensitive. Afterwards, these indexes including resource IDs are published to the target nodes. In this way, the indexes of similar images are clustered to the same node with

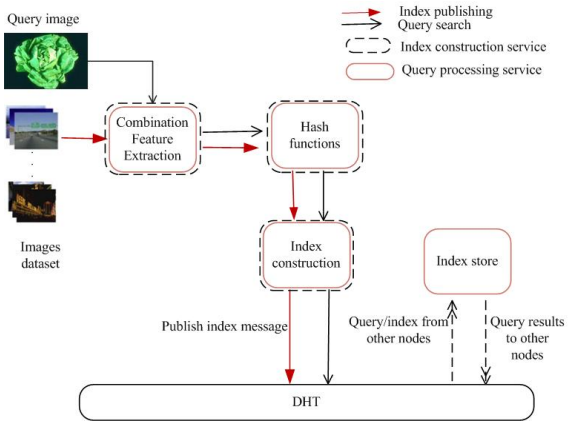

Fig. 2. Components of a LCFIR node.

high probability. Instead of flooding queries, they are routed to special nodes which may answer the queries. Another problem is how to effectively represent the content of an image. To obtain a pattern representation, we rely on the early feature fusion approach. LCFIR analyzes the content from different ways. It extracts image color, shape and texture feature, and then fuses them into a single representation.

Figure 2 shows the interactions among key components of LCFIR. A client uses LCFIR to share local objects or query the system. LCFIR is located between users and the DHT layer, which contains two services of index construction and query processing. We first discuss the index construction service. Each node has a local image database, where images are shared with others. In addition, each node also has an image feature combination which generates the combination feature specific to different image formats. The combination feature extractor accesses images in the local database, which adopts various algorithms to fuse color and edge features in order to analyze the content of images. For each feature vector, a set of LSH functions are adopted to generate resource *IDs* based on the hash values. They also specify the number of index replicas of shared objects and the number of lookup requests to be sent out for queries. At the time of the system start, these hash functions are generated and used in all nodes. For each hash value, the index construction constructs the index messages and publishes them to the DHT layer. Once receiving an index message from the DHT layer, the node inserts it in the index storage according to resource *IDs*. In addition, the image indexes in local database are refreshed after a period of time to ensure the validity of resources.

The query processing service adopts the similar way to generate the resource *ID* and the query message. Then the query messages are forwarded to nodes responsible for the query feature. Once receiving the query message, the destination nodes do a local search to identify the top T of best matching results, and returns them to the query node. The user surface gives returned results a global rank based on Euclidean distances and presents the final results to users.

IV. DHT-BASED CBIR APPROACH

In this section, we describe in details our index construction service and query processing service designed for CBIR.

A. Combination Feature Extraction

LCFIR adopts Motion Picture Experts Group-7 (MPEG-7) descriptors, which convert visual contents to measurable feature space. The MPEG-7 standard provides the multimedia content description interface, which includes a set of descriptors, such as color, shape and texture descriptors, to support image retrieval [33]. These visual descriptors represent human visual perception as feature vectors, so that the similarity of two images in appearance can be evaluated.

However, few articles have been published on how to apply the perceptually uniform color difference between colors and edge orientations to image representation and image retrieval. To this end, Liu et.al [14] propose a new descriptor for image retrieval. This descriptor combines the use of orientation, color and color difference and considers the spatial layout without the use of any image segmentation or learning processes.

In the proposed algorithm, orientation and perceptual color information have been combined in the unified framework, and both of their spatial layouts have been considered. This algorithm can be considered as a generalized low-level feature representation without the need for any image segmentation or model training. The vector dimension of the proposed algorithm is only 108 and the algorithm is therefore very efficient for image retrieval.

We define the color difference histogram (CDH) as follows:

$$H_{color}(C(x,y)) = \{\sum\sum\sqrt{(\Delta L)^2+(\Delta a)^2+(\Delta b)^2}\} \quad (1)$$

where $\theta(x,y)=\theta(x',y');\max(|x-x'|,|y-y'|)=D$

$$H_{ori}(\theta(x,y)) = \{\sum\sum\sqrt{(\Delta L)^2+(\Delta a)^2+(\Delta b)^2}\} \quad (2)$$

where $C(x,y)=C(x',y');\max(|x-x'|,|y-y'|)=D$

The values of a quantized image $C(x,y)$are denoted as $w\in 0,1,..,W\text{-}1$. And their color index values is $C(x,y)=w_1$ and $C(x',y')=w_2$.The values of an edge orientation image $\theta(x,y)$ are denoted as $v\in 0,1,\ldots,V\text{-}1$. The angles at (x,y) and $(x',y')$are denoted by $\theta(x,y)=v_1$ and $\theta(x',y')=v_2$. For neighboring pixels, whose distance is D. Finally, the final feature vector H is: $H=[H_{color}(0),H_{color}(1),\ldots,H_{color}(W\text{-}1),H_{ori}(0),H_{ori}(1),\ldots H_{ori}(V\text{-}1)]$.

B. LSH based Index Construction

When a node wants to share an image, it constructs and publishes index messages after image features are extracted. The remaining question is how to construct resource *ID* for similar images and answer the query efficiently. LCFIR computes resource *IDs* utilizing p-stable LSH whose locality sensitive property is explored [36]. The intuition behind this approach is that the content similar images are assigned to the same resource *IDs* and the indexes of them can be stored in the same node.

1) Locality-Preserving Mapping.

The basic idea of LSH is to use a family of hash functions which map similar objects into the same value with high probability [12].

In this work, we employ the family functions of *p*-stable LSH [35], which exists for $p \in (0,2]$. Since Euclidean distance is supposed to be the most widely-used distance metric, the Gaussian distribution working for the Euclidean distance is used as the 2-stable distribution. The hash function $h_{a,b}$ is defined as follow:

$$h_{a,b}(v) = \frac{a \cdot v + b}{W} \quad (3)$$

Where *a* is a *d*-dimensional vector whose elements are chosen independently from the *p*-stable distribution. *b*, a real number, is randomly selected from the range [0, *W*]. Each hash function $h_{a,b}(v): R^d \to Z$ maps a *d*-dimensional vector *v* into an integer.

The actual indexing is done by using LSH functions and building several hash tables, in order to increase the collision probability. In fact, *m* hash tables $G=\{g_1,...,g_m\}$ are constructed, where *m* is randomly chosen. With *k* independent hash buckets $g_i(v)=(h_1(v),\ldots,h_k(v))$, the hash result is a *k*-dimensional integer vector, i.e., $G=\{g: R^d \to Z^k\}$. In this way, if two close feature vectors is hashed by more hash tables, they may get the same hash value at least in one hash table g_i, where $i=1,...,m$. As a result, similar objects are hashed to the same bucket at the higher probability, given by $1-(1-p_1^k)^m$.

Subsequently, an integer vector Z^k is obtained from one hash table g_i, where $i=1,...,m$. The resource identifier space for Chord is one-dimension, while the dimension of the feature vectors may be very high. In the next step, the *k*-dimension space is transformed to the one-dimension space, i.e., $Z^k \to N$, without destroying the locality sensitive. On the other side, the load, defined as the number of indexes stored on nodes, should be kept balanced as much as possible. To construct a resource *ID*, i.e., *fcoID*, the mapping function $\Im(v)$ is defined as:

$$fcoID_i = \Im(\sum_{j=1}^{k} h_j(v) \cdot d_j), \text{ where } h \in g_i \text{ and } i=1,...,m. \quad (4)$$

d_j is a randomly chosen non-zero integer. $\Im$ function is denoted as the consistent hash function SHA-1.

We call each $\Im$ a resource *ID* mapping function, and denote φ_m the function set $\{fcoID_1,...,fcoID_m\}$. Therefore, given $\varphi_m = \{fcoID_1,..., fcoID_m\}$, we can map a point *v* to *m* Chord keys $fcoID_1(v),...,fcoID_m(v)$.

Obviously, as discussed in the previous section, similar vectors should have the same *fcoID* after this mapping, without destroying locality sensitive. Given two similar vectors v_1 and v_2, we would have $g_i(v_1) = g_i(v_2)$, where $i=1,...,m$. The detail proof is described in [35].

On the other side, in order to fully utilize the Chord *ID* space and keep load balancing, the consistent hash function SHA-1 is employed to distribute indexes as symmetrical as possible.

2) Index Construction Service.

In this section, we describe the detail process of the index construction service in LCFIR. The purpose of this service is to cluster the indexes of similar features to the same node with high probability. Therefore, we propose the Index Construction Service (ICS) which adopts *p*-stable LSH to preserve image similarity and distributes the indexes to the Chord as evenly as possible. That is different from the traditional location approach [35], where DHTs access an image through the hash key of a single feature.

ICS firstly generates the index messages based on the same *fcoIDs* of the similar images, and then publish them to special nodes through the DHT layer. For each image in the local database, the Feature Fusion is firstly invoked to extract its visual feature f_v. After that, $m \times k$ hash functions are generated, which map the feature vector to *m* integer vectors, e.g., $G=\{g: R^d \to Z^k\}$. Next, each integer vector is mapped to one resource *ID*, e.g. $Z^k \to N$, without destroying the locality sensitive of LSH. And then ICS maps f_v into *m* index replicas $\varphi_m = \{fcoID_1, fcoID_2,\ldots, fcoID_m\}$, where $fcoID_i = \Im(f_v)$.

After the *fcoIDs* φ_m of an image is obtained, ICS constructs indexes in the form of $<fcoID_i, f_v, nodeID>$ where $i=1\ldots m$, *nodeID* is the address of the object owner. ICS routes each index message to the node responsible for *fcoID* through DHT layer. Once a node receives the index message from the DHT layer, ICS inserts the received message into the index storage. The indexes with the same *fcoID* in index storage are gathered into the same list to facilitate the location of local indexes.

The number of index replicas to be published for an image is a system parameter. It depends on the number of hash tables *m* and poses off between query accuracy and communication cost. As discussed above, constructing more index replicas for an object, i.e., generating hash tables, not only means more index messages to be published, but also provides better chance of finding the images that are similar to the query. So we should make a tradeoff between *m* and the query efficiency. A node can decide the value of m depending on different cases. If the object is of importance, the number of indexes can be constructed more.

The number of buckets in each hash table, *k*, is another system parameter. And it poses a trade-off between the query efficiency and the load balancing. Fewer *k* means more images cluster to the same hash value, i.e., the same *fcoID*. That can lead to fewer clusters and accordingly each cluster has more images. Once a *fcoID* is located, more relevant images can be obtained when it is in fact similar to a query. That can also cause high load of nodes in charge of the *fcoID*, if *k* is set too small.

All the indexes of shared images in the node are refreshed periodically, e.g., once a week or a month. If an image is added or deleted, its indexes are constructed or removed. Depending on the similarity between the modified image and the original one, we can determine whether or not the indexes should be reconstructed. If the similarity between the two versions is less than the threshold, the indexes remain unchanged. Otherwise, the ICS is invoked to re-construct the indexes.

A. Distributed Query Processing

The objective of the query processing service (QPS) is to

answer a query efficiently and effectively. Search efficiency is measured by the number of network hops for a query. Search effectiveness is measured by quality of search results, i.e., recall and precision.

1) Query processing

In this section, the QPS is discussed, supposing that all the image indexes are published into the DHT layer. When a node issues a query image, QPS is invoked and the content-based similar images are retrieved. It is analogous to the process of index constructing. Query images are converted to a set of *fcoIDs*. And then the query messages are published to the special nodes.

The feature vector f_q is transformed into a set of resource *IDs*= $\{fcoID_1, fcoID_2, \ldots, fcoID_m\}$, where $\text{fcoID}_i = \Im(f_q)$ by $m \times k$ *p*-stable LSH function. Note that the set of hash functions used in QPS is the same as that in ICS. Afterwards, the node sends the query messages for each *fcoID* in the form of <$fcoID_i$, f_q, *nodeID*> where i=1…m, and *nodeID* is the address of the query node. However, if the images satisfy the requirement of the query, they are more likely to be retrieved due to the same *fcoIDs*. Therefore, the similar search of LCFIR is probabilistic and relies on building several indexes to achieve highly accurate query results.

Once a query message reaches its destination through the DHT layer, the node in charge of *fcoIDs* may probably contain colliding *ID*. It checks the local index storage to find if there exists the same *fcoID* as it receives. Afterwards, the nodes that receive the request compute the similarity between the query feature and the object features in its index store. The similarity measure is specified by the Euclidean distance. To reduce the network transmission cost, it only returns the top T qualifying indexes.

Similar to ICS, the number of query messages also depends on the number of hash tables m. It impacts on not only the number of request messages to be sent but also the query accuracy. When m is increased, more query messages are routed to more candidate nodes, which causes high query cost. But it also increases the probabilities of finding the images similar to the query. In contrast, if m is too small, the query cost is reduced, while the accuracy might also be decreased.

On the other side, when k becomes larger, the value of m has to be increased to ensure the query accuracy. Increasing k can generates more clusters that are spread on more nodes. Each cluster contains fewer indexes of similar objects. Consequently, it needs more index replicas to be sent in order to search more candidate nodes. In this way, the search efficiency is guaranteed.

After the query node receives all the results, it merges them before showing them to users. The results of the query are obtained by sorting returned indexes according to the Euclidean distance, and the top T ones are chosen to form a result list. Then connections are established between the query node and data owners to transmit the final results to the query node. Finally, the top T most similar images are shown to users.

V. EXPERIMENTAL RESULTS

We have implemented the proposed system using Java 1.6. The simulations run on a 2.83GHz Intel Core CPU with 2GB RAMs.

A. Datasets and System Settings

In our experiments, Corel 10000 is used, which contains 10000 images of various contents, such as flowers, food, wave, pills, fish and door, etc. It contains 100 categories and each category contains 100 images in JPEG format. In each category we randomly choose 20 images, so 2000 queries are drawn.

Queries are initiated at randomly chosen nodes, after all the nodes join LCFIR. The reported results are the average values over all the queries. Unless otherwise noted, the default values are W=2.0 for p-stable hash functions. The default network size N is fixed to 1000. Besides, for the image retrieval task, it is important to define suitable metrics for the performance evaluation.

In order to compare with LRFIR, we have implemented the method and parameters which are set as described in [36]. To fairly compare against this method, we ignore the processing of relevance feedback in LRFIR. We report here the best results which have achieve a fair balance as well.

B. Query accuracy

We use two metrics to verify the query accuracy of our system. As shown in Figure 3, the corresponding query accuracy is evaluated with the different number of hash functions and top images. The x-axis represents the number of top ranked images, varying from 10 to 100 for Figure 3. The y-axis denotes the average recall and the average precision measured on the top ranked images. m and k represents the number of hash tables and buckets, respectively. As shown, the average recall increases, while the average precision decreases, when the number of top images grows. The average recall and precision rapidly rise by increasing the number of hash tables m and decreasing the number of buckets k, for both datasets.

The reason is that by increasing m the collision probability of content-based similar images is grown. On the other hand, decreasing k can lead to fewer clusters, accordingly more images are gathered into one cluster. Once a cluster is located, many relevant results are searched. However, for Figure 3(a) and Figure 3(b), both m=8, k=10 and m=18, k=18 achieve the best recall rate and precision. They almost have the same value. We can choose m=8, k=10, since the computational overhead is reduced with the small number of hash functions.

Figure 3 show comparing results versus the number of hash tables. We see a big increase in accuracy for both datasets. For example, the average recall of LCFIR (m=8, k=10) achieves about 3% improvement over LRFIR (m=10, k=15) on the top 100 images in Figure 3(a). With respect to the average precision, LCFIR represents about 40% increase on the top 4 images in comparison with LRFIR, as shown in Figure 3(b).

C. Load Balancing

In this section, Figure 4(a) shows the effectiveness of load balancing with various arrangements. We define the node load as the number of index messages stored on the node. In Figure

4(a), the *x*-axis shows the percentage of nodes, where the number of nodes varies from 10 to 5000, and the correspond-

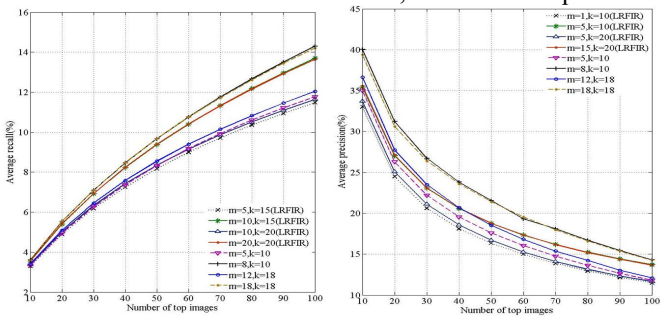

Fig. 3**.** Performance evaluation. (a)Average recall. (b)Average precision

ing values represent the percentage of indexes assigned to these nodes. Note the figure shows much less skew on load distribution as the number of nodes increases. That means the load balancing is almost achieved. This is because when the number of node increases, the interval between nodes becomes smaller and indexes are distributed to more nodes. Therefore, there are not many indexes in each node.

In Figure 4(a), when the number of node is 1000 and 5000, the load is more balanced than other cases. However, when *n* is 10, the load of LCFIR is skewed and 60% of nodes stores around 40% of indexes. The reason is that the number of nodes is so small that the interval between nodes *ID* becomes large. Therefore, some nodes store much more indexes than others. Moreover, SHA-1 distributes the indexes to a large set of possible intervals of Chord. Compared to LRFIR, the curves of LCFIR for both datasets appear to be much steadier, especially when *n*=1000 and *n*=5000. This indicates that the load balancing of LCFIR is better than that of LRFIR.

Figure 4(b) shows the load balancing versus the number of buckets, when the number of nodes in the network is 1000. As the figures reveal, when the number of buckets grows, the index distribution is much balanced. The line of *m*=8, *k*=10 is much smooth than others. In the above experiment, we reasonably choose *k*=10. That is because as the number of hash bits, *k*, increases, less index messages are gathered into a cluster, then each node correspondingly stores fewer messages.

D. Network Hops

The effect of network hops is depicted, as shown in Figure 5. Network hops are one of the most critical parameters in the distributed environment. The horizontal axis represents the number of nodes varying from 10 to 5000. The vertical axis is the lookup number of hops for processing a query image, when *m* varies from 4 to 18 for Figure 5(a) and *k* from 5 to 19 for Figure 5(b). As shown, the number of lookup hops mainly depends on the number of tables, *m*. As we expect, the lookup hops increase when the number of tables increases. In Figure 5(a), however, for a large number of nodes, *n*=100 and *n*=5000, the number of hops only has a slight increase.

As we expect, the number of lookup hops is independent on the number of buckets. In Figure 5(b), especially for *n*=100 and *n*=5000, the number of hops remains almost the same while the number of buckets varies.

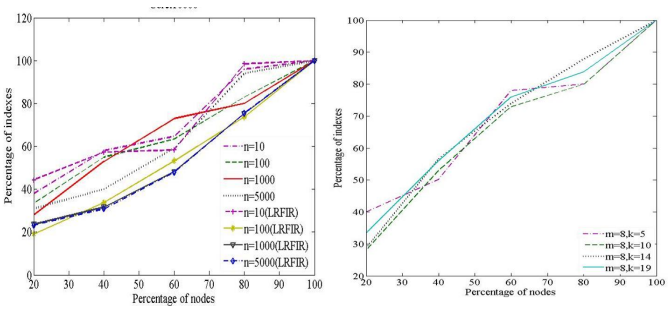

Fig. 4. (a) Effect of load on the index distribution with different nodes. (b) Effect of load on the index distribution.

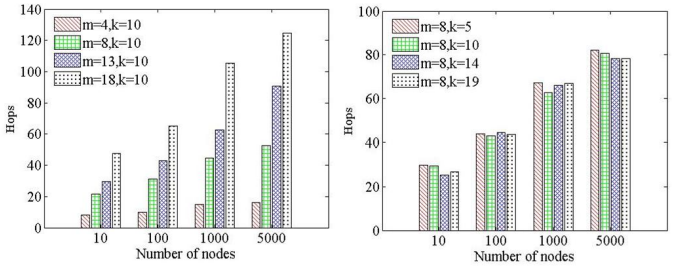

Fig. 5. (a) Network hops vs. hash tables No. (b) Network hops vs. buckets No.

We can see that the lookup process of LCFIR does not need many network hops. So *m*=8, *k*=10 can be chosen to guarantee the best query accuracy while not incurring too many network hops.

VI. CONCLUSIONS

We propose an effective framework to support CBIR in the distributed cloud datacenter. LCFIR is implemented on the DHT layer which provides efficient routing mechanisms. ICS constructs indexes based on fusion features of images and *p*-stable hash functions, where the content similar images are mapped into the same resource *ID* and distributed to the same Chord node, with high probability. QPS processes the query image and publishes the query messages. The experiments show that our approach achieves high accuracy and good load balance, and it only needs a small numbers of network hops.

ACKNOWLEDGMENT

This work was jointly supported by: (1) the National Basic Research Program of China (No. 2013CB329102); (2) National Natural Science Foundation of China (No. 61471063, 61372120，61271019, 61101119, 61121001); (3) the Key(Keygrant) Project of Chinese Ministry of Education.(No. MCM20130310); (4) Beijing Higher Education Young Elite Teacher Project (No. YETP0473); (5) the Specialized Research Fund for the Doctoral Program of Higher Education (No. 20100005110008).

REFERENCES

[1] Peng C., Kim M., Zhang Z., & Lei H. (2012). VDN: Virtual machine image distribution network for cloud data centers. *IEEE International Conference on Computer Communications (INFOCOM'12)*, Orlando, Florida, 181-189.

[2] Dikaiakos M. D., Katsaros D. , Mehra P. , Pallis G. , & Vakali A. (2009). Cloud computing: Distributed Internet Computing for IT and Scientific Research. *IEEE Internet Computing, 13*(5), 10-13.

[3] Demirkan H., & Delen D. (2012). Leveraging the capabilities of service-oriented decision support systems: Putting analytics and big data in cloud. *Decision Support Systems*.

[4] Yang Z., Zhao B. Y., Xing Y., et al. (2010). AmazingStore: Available, low-cost online storage service using cloudlets. *Proceedings of the 9th International Workshops on Peer-to-Peer Systems (IPTPS'10)*, San Jose, USA, 1-5.

[5] Forestiero A., Leonardi E., Mastroianni C., & Meo M. (2010). Self-Chord: A Bio-Inspired P2P Framework for Self-Organizing Distributed Systems. *IEEE/ACM Transaction on Networking, 18*(5), 1651-1664.

[6] Gnutella. (2000).Gnutella website. *http://www. Gnutella.com*.

[7] Stoica I., Morris R., Karger D., Kaashoek M.F.,& Balakrishnan H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. *The 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'01)*, San Diego, USA, 149-160.

[8] Ratnasamy S., Francis P., Handley M., Karp R., & Shenker S. (2001). Scalable content-addressable networks. *The 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'01)*, San Diego, USA, 161-172.

[9] Jagadish H.V., Ooi B. C., & Vu Q.H. (2005). BATON: A Balanced Tree Structure for Peer-to-Peer Networks. Proceedings of the 31st international conference on Very large data bases (VLDB'05), Trondheim, Norway, 661-672.

[10] Bawa M., Manku G., & Raghavan P. (2003). SETS: search enhanced by topic segmentation. *Proceedings of the 26th Annual International ACM SIGIR Conference (SIGIR'03)*, Toronto, Canada, 306-313.

[11] Kalnis P., Ng W.S., Ooi B.C., & Tan K. (2004). Answering similarity queries in peer-to-peer networks. *Information Systems, 31*(1), 57-72.

[12] Indyk P., & Motwani R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. *Proceedings of the 13th ACM Symposium on Theory of computing (STOC'98)*, Dallas, Texas, 604-613.

[13] Liao J., Wang J., Wu B., & Wu W. (2012). Toward a Multi-plane Framework of NGSON: a Required Guideline to Achieve Pervasive Services and Efficient Resource Utilization, *IEEE Communications Magazine, 50*(1), 90-97.

[14] Liu G., Zhang L., Hon Y., Li Z., & Yang J. (2013). Content-based image retrieval using color difference histogram. *Pattern Recognition, 46*(1), 188-198.

[15] Lv Q., Cao P., Cohen E., Li K. & Shenker S. (2002). Search and replication in unstructured peer-to-peer networks. *Proceedings of the 16th ACM Annual International Conference on Supercomputing (ICS'02)*, New York, USA, 84-95.

[16] Crespo A., & Garcia-Molina H. (2002). Routing indices for peer-to-peer systems. *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS'02)*, Vienna, Austria, 23-32.

[17] Gaeta R., & Sereno M. (2011). Generalized Probabilistic Flooding in Unstructured Peer-to-Peer Networks. *IEEE Transaction on Parallel Distributed System, 22*(12), 2055-2062.

[18] Vlachou A., Doulkeridis C., & Kotidis Y. (2012). Metric-Based Similarity Search in Unstructured Peer-to-Peer Systems. *Transactions on Large-Scale Data-and Knowledge-Centered Systems*, 5, 28-48.

[19] Sripanidkulchai K., Maggs B.M., & Zhang H. (2003). Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03)*, San, Francisco.

[20] Eisenhardt M., Muller W., Henrich A., Blank D. & Allali S.E. (2006). Clustering-based source selection for efficient image retrieval in peer-to-peer networks. *Proceedings of the 8th IEEE International Symposium on Multimedia (ISM '06)*, Washington DC, USA, 823-830.

[21] King I., Ng C. H., & Sia K. C. (2004). Distributed content-based visual information retrieval system on peer-to-peer networks. *ACM Transaction on Information Systems, 22*(3), 477-501.

[22] Chen J., Hu C., & Su C. (2008). Scalable Retrieval and Mining With Optimal Peer-to-Peer Configuration. *IEEE Transactions on Multimedia, 10*(2), 209-220.

[23] Androutsos P., Androutsos D., & Venetsanopoulos A. N. (2006). A distributed fault-tolerant MPEG-7 retrieval scheme based on small world theory. *IEEE Transactions on Multimedia, 8*(2), 278-288.

[24] Falchi F., Gennaro C., & Zezula P. (2005). A content-addressable network for similarity search in metric spaces. *Proceedings of the 6th International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P'05)*, Toronto, Canada, 79-92.

[25] Novak D.,& Zezula P. (2006). M-Chord: a scalable distributed similarity search structure. *Proceedings of the First International Conference on Scalable Information System (INFOSCALE' 06),* Hong Kong, China, Article 19.

[26] Tang C., Xu Z., & Dwarkadas S. (2003). Peer-to-peer information retrieval using self-organizing semantic overlay networks. *The 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'03)*, Karlsruhe, Germany, 175-186.

[27] Sahin O.D., Gulbeden A., Emekci F., Agrawal D., & Abbadi A.E. (2005). PRISM: Indexing multi-dimensional data in p2p networks using reference vectors. *Proceedings of the 13rd Annual ACM International Conference on Multimedia, ACM Multimedia (MM'05)*, Singapore, 946-955.

[28] Zhang X., Shou L., Tan K., & Chen G. (2010). iDISQUE: Tuning High-Dimensional Similarity Queries in DHT Networks. *Proceedings of the 15th International Conference on Database Systems for Advanced Applications (DASFAA'10)*, Tsukuba, Japan, 19-33.

[29] Zhu Y., & Hu Y. (2007). Efficient semantic search on DHT overlays. *Journal of Parallel and Distributed Computing, 67*(5), 604-616.

[30] Wang X., Zhang B., & Yang H. (2014). Content-based image retrieval by integrating color and texture features. Multimedia Tools and Applications, 68(3), 545-569.

[31] Snoek C.,Worring M., & Smeulders A. W. M. (2005). Early versus late fusion in semantic video analysis. Proceedings of the 13rd Annual ACM International Conference on Multimedia, ACM Multimedia (MM'05), Singapore, 399-402.

[32] Tian X., Jiao L, Liu X., & Zhang X. (2014). Feature integration of EODH and Color-SIFT: Application to image retrieval based on codebook. Signal Processing: Image Communication, 29(4), 530-545.

[33] Datta R., Joshi D., Li J., & Wang J. Z. (2008). Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys, 40*(2), Article 5.

[34] Datar M., Immorlica N., Indyk P., & Mirrokni V. S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. *Proceedings of the 20th Annual Symposium on Computational Geometry (SoCG'04)*, New York, USA, 253-262.

[35] Liao J., Yang D., Li T., Wang J., Qi Q., & Zhu X. (2014). A scalable approach for content based image retrieval in cloud datacenter, Information Systems Frontiers, 16(1), 129-141.

[36] Haghani, P., Michel, S., & Aberer K. (2009). Distributed similarity search in high dimensions using locality sensitive hashing. *Proceedings of the 12th International Conference on Extending Database Technology (EDBT'09)*, Saint Petersburg, Russia, 744-755.