Multimed Tools Appl
DOI 10.1007/s11042-015-2892-y

Fusion feature for LSH-based image retrieval in a cloud datacenter

Jianxin Liao[1] · **Di Yang**[1,3] · **Tonghong Li**[2] · **Qi Qi**[1] · **Jingyu Wang**[1] · **Haifeng Sun**[1]

Received: 12 May 2014 / Revised: 24 June 2015 / Accepted: 17 August 2015

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 LFFIR, a multi-feature image retrieval framework for content similar search in the distributed situation. The key idea is to effectively incorporate image retrieval based on multi-feature into the peer-to-peer (P2P) paradigm. LFFIR fuses the multiple features in order to capture the overall image characteristics. And then it constructs the distributed indexes for the fusion feature through exploiting the property of locality sensitive hashing (LSH). We implement a prototype system to evaluate the system performance with two image datasets. Comprehensive performance evaluations demonstration that our approach brings major performance and accuracy gains compared to the advanced distributed image retrieval framework.

✉ Jianxin Liao
liaojx@bupt.edu.cn

✉ Di Yang
yangdi.bupt@gmail.com

Tonghong Li
tonghong@fi.upm.es

Qi Qi
qiqi8266@bupt.edu.cn

Jingyu Wang
wangjingyu@bupt.edu.cn

Haifeng Sun
sunhaifeng_1@ebupt.com

[1] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

[2] Department of Computer Science, Technical University of Madrid, Madrid 28660, Spain

[3] China United Network Communications Limited, Beijing 100033, China

Published online: 27 August 2015

Keywords Cloud computing · Content based image retrieval · Peer-to-peer · Locality sensitive hashing · Fusion feature

1 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 stores a large number of resources collected from all around the world. It allows users to access resources more flexibly, when storage and computation are switched from the clients to the "cloud" [27]. The paradigm offers essential properties including location independent data storage, on-demand self service, and data access independent of locations and time [8].

Cloud-based services are deployed on the datacenter which is a fundamental block and provides highly durable storage [26]. Therefore, it is important to choose 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, the central points are responsible to establish the central index for resources, but vulnerable to failures caused by fires, power outages, natural disasters, etc. [38]. 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 [14]. Each peer takes charge of part of data and replicas to improve the clouds' reliability and resilience to correlated failures. It is well known that P2P techniques are very likely to be adopted in Clouds [11].

The P2P performance mainly depends on the topological structure and the index placement. Unstructured P2P systems like Gnutella [13] have little control over the topology and the index placement. This structure does not guarantee the efficiency of data search and causes high-cost communication, since the query is spread without any global planning. In contrast, structured P2P systems have tight control led over the overlay topology, and only published the resource' indexes to the special nodes through distributed hash table (DHT). Chord [32], CAN [28], and BATON [17] are classical examples of structure P2P networks. In DHTs, the file' name is generally hashed into a key and the query is forwarded to the target node based on the key. The number of lookup hops for locate files can be limited to the logarithmic number by using the 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 distributed datacenter, millions of files including images, videos and plain texts are transferred into the datacenter which adopts the decentralized P2P paradigm, as Fig. 1 shows. The content providers upload their significant resources to the datacenter. 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 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 decentralized datacenter.

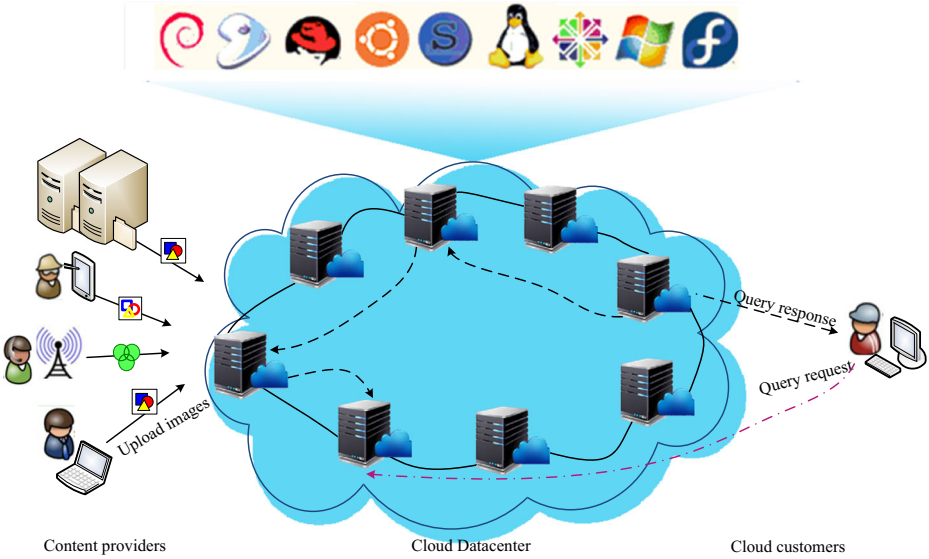


Fig. 1 Image retrieval in the distributed datacenter

Although throughout this paper we focus on image retrieval, our methods are also applicable to multimedia retrieval domain where similar search is performed in the P2P paradigm.

Existing models which are usually centralized, are not scalable enough to work in the 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 [3, 13]. However, since it is difficult to annotate images very exactly, identifying images in this way is inaccurate and may be insufficient to users' requirement sometime. In some applications, users may request inexact queries such as "find the top-*k* images which are most similar to a given sample". It is difficult for humans to describe how an image is similar to the sample with keywords [18]. However, the content-based image retrieval (CBIR) can find similar images through the image content instead of keywords. In our previous work, we present a CBIR system called LRFIR [22] based on a single feature for the distributed environment. It only adopts the texture feature to represent images, which is suitable for images containing texture. However, such a feature can hardly describe the images with various landscape. Motion Picture Experts Group 7 (MPEG-7) includes the color, texture and shape descriptors. They represent visual contents at different aspects, and suit to descript various images especially with landscape.

According to the idea of LRFIR, if the each feature is searched one by one, the lookup cost is increased. Also, it is very difficult to create a type of feature which considers color, texture and shape at the same time. In the paper, we present a novel CBIR system called LSH-based Fusion Features for Image Retrieval (LFFIR) for the distributed datacenter. LFFIR supports content similar search leveraging an early multi-feature fusion technique which integrates color, texture and shape features in order to perfectly describe image contents. Different from LRFIR, LFFIR focuses on multiple features which are suitable for searching natural landscape images containing the various landscape with rich colors. The efficient index construction service (ICS) and query processing service (QPS) are proposed for LFFIR.

In the index construction service, LFFIR leverages image feature extraction algorithms. It extracts the color, texture and shape features. Then the three types of vision features are fused

into a feature, from which the content similarity can be measured quantitatively. LFFIR 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), fusion feature of similar images are probabilistically assigned to the same resource ID [16]. In this way, the indexes of similar images are more likely published into the same nodes through overlay routing. 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 [21]. It is used to evaluate the performance of our algorithm in two image datasets, i.e., Corel [23] and the subset of Caltech 101 Object Categories [20]. The experiments show that our scheme achieves high accuracy with only a small number of lookup hops, comparing to the advanced distributed image retrieval framework.

In this paper, our main contributions are as follows: (1) We combine techniques from image multiple features and P2P computing to implement an efficient indexing and locating approach. The distributed index is constructed based on image content represented by the multiple features instead of a single feature. (2) We exploit LSH to produce the same identifiers for similar images and queries. The LSH-based approach is very effective in terms of reducing the lookup hops, which is independent of the DHT overlay size. (3) The multi-feature fusion index considers color, texture and shape feature, and brings down the lookup cost introduced by the multi-feature retrieval separately.

The rest of the paper is organized as follows. Section 2 shows an overview of related work. Section 3 presents the framework of LFFIR. Section 4 describes the index construction service and the query processing service. Section 5 evaluates the performance of our approach. Finally, Section 6 concludes our discussion.

2 Related work

The P2P networks could be 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 in unstructured P2P systems like Gnutella [13] floods the query to all the neighbors. Lv et al. [24] 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, 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 [5]. Its basic idea is that query messages are forwarded to the neighbor nodes that are more likely to have the required answers. To avoid being trapped around the local optimum, Gaeta & Sereno [12] 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 search strategies do not guarantee the lookup time and consume too much network resources. Moreover, 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 multimedia content is ignored. The decentralized interesting-based location solution [31] 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 source selection [9], where each peer gathers its own data into clusters. When queries are issued, these clusters serve as the potential selection sources of interesting data. DISCOVIR [19] 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 [4] adopts the similar way to group peers, but extends DISCOVIR to support the capability of knowledge discovery and image data mining. The small world indexing mine (SWIM) [1] creates a small world network for images which are connected according to the MPEG-7 descriptor similarity. 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.

Since unstructured P2P systems have little controlled over network topology, the hybrid infrastructures are proposed, which gather peers storing relevant files into the same community to reduce unnecessary traffic. Many methods employing this model can be found, such as SETS metric space, source selection, DISCOVIR, P2P-CBIRM, and SWIM. Bawa et al. [3] propose the 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. [36] propose that peers sharing similar data are linked to the same super node, while super nodes are organized as an M-Tree structure.

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 [10], M-Chord [25], Psearch [33], Prism [29] and iDISQUE [39]. MCAN using CAN as the underlying structure adopts a pivot technique, iDistance, to map objects to an *N*-dimensional vector space. But the chosen pivots are preprocessed in a centralized fashion, and then distributed to peers. 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 vectors, 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 dynamic datasets. Zhu et al. [40] 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 that of 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. [37] 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. [30] 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. [34] construct the Edge orientation difference histogram (EODH) descriptor for each edge pixel. And they integrate EODH with Color-SIFT to build weighted codeword distribution.

Many work adopts iDstance to establish indexes, M-Chord is the classical one. Batko et al. [2] propose an distributed image retrieval framework based on the M-Chord [25]. The core idea of M-Chord is to map the image feature to a one-dimensional domain and navigate in this

domain using the Chord routing protocol. Its data clustering and mapping are still completed in a centralized manner. This approach exploits the idea of a vector index method iDistance [29], which partitions the data space into clusters (C_i), identifies reference points (p_i) within the clusters, and defines one-dimensional mapping of the data objects according to their distances from the cluster reference point. The M-Chord mapping works basically as follows: Several reference points are selected globally from the sample dataset. The data space is partitioned in a Voronoi like manner into C_i (each object is assigned to its closest pivot). Following the iDistance idea, the M-Chord key for an object $x \in C_i$ is defined as $mchord(x)=d(p_i, x)+i \cdot c$. Image retrieval adopts an approach of **kNN***(q, k)* queries evaluation that exploits the range search. The idea is to estimate radius r, so that the **Range***(q, r)* query returns at least k nearest objects. For each cluster, determine interval of keys to be scanned: $I_i=[d(p_i,q)+i \cdot c-r, d(pi,q)+i \cdot c+r]$. An request is firstly sent to the node responsible for the midpoint of interval I_i. If it is not responsible for the whole interval, forward the request to the predecessor and/or successor.

3 System overview

In this section, we present an overview of the LFFIR framework. The overall aim is to provide a multi-feature retrieval framework in the distributed datacenters which store a large number of resources. In such a setting, it is the naïve solution that each feature is searched at a time, then the final results are given, after all the features is traversed. Obviously, since searching a feature needs a number of hops [22], multi-feature incurs high-cost communication. Analogously, it is hard to create a feature which takes into account all the feature such as color, texture and shape [7]. Departing from conventional approaches, we propose a scalable scheme, called LFFIR, which supports content similar image retrieval in the distributed network, utilizing indexes based on multiple features information.

In the underlying layer of LFFIR, a large number of nodes are organized into the structured P2P network (Chord [32]), to offer the routing service. Inside the network, the *ID* space generated through uniform hash functions is ranged from 0 to 2^l-1,where l=160. In DHT, the identifier is usually set 160-bit strings [35]. Each node is assigned an *ID* drawn from the space, and is responsible for the object key range between its *ID* and the *ID* of the previous node on the Chord. In an n-node network, Chord routes a message to its destination in $O(\log n)$ hops. LFFIR can support efficient routing, due to the DHT layer, where indexes publishing and queries routing are automatically accomplished.

Each node in LFFIR maintains its own data objects. To publish indexes to the network, for each object, LFFIR 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 be responsible for the answer.

The challenge is how to establish the distributed index for multi-feature, 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 fuse multiple features, and then build distributed indexes through a family of LSH functions. To obtain a pattern representation, we leverage off the early feature fusion approach which analyzes the content before processing retrieval. LFFIR extracts image color, shape and texture feature, and then fuses them into a single representation. Due to the property of local sensitivity of LSH, it is more likely to assign similar fusion features to the same buckets of the hash table. Then they are mapped into

resource *IDs* without loss of local sensitivity. Afterwards, these indexes including resource *IDs* are published to the target nodes through overlay routing. In this way, the indexes of similar images are clustered to the same node with high probability. Instead of flooding queries, query messages are routed to special nodes which may answer the queries.

Figure 2 shows the interactions among key components of LFFIR. A client uses LFFIR to share local objects or issue the query. LFFIR 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 other users. In addition, each node has a multi-feature fusion which generates fusion features specific to different image formats. The image feature fusion accesses images in the local database, which adopts various algorithms to fuse color, shape and texture features in order to analyze image content. 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 for a query. 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 generates the resource *IDs* and the query messages in the same way as ICS. 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 best matching results, and returns them to the query node. The user surface ranks returned results based on the Euclidean distance and presents the final results to users.

4 DHT-based CBIR approach

In this section, we describe details of the index construction service and the query processing service designed for LFFIR.

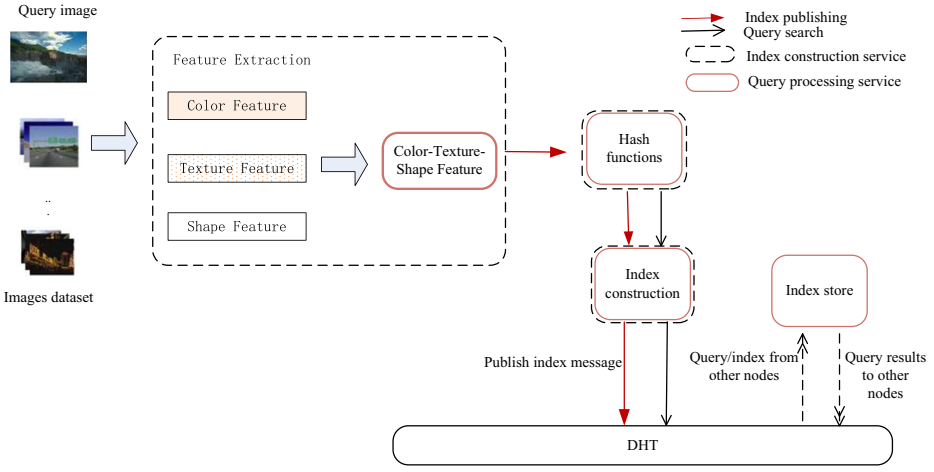


Fig. 2 Components of a LFFIR node

4.1 Features extraction and fusion

When a user shares an image, LFFIR automatically extracts the image features. It adopts MPEG-7 descriptors, which convert visual content to a 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, to support image retrieval [7]. These visual descriptors represent human visual perception as feature vectors, so that the similarity of two images in appearance can be evaluated.

In recent years, research on color features has focused more on the summarization of colors in an image, that is, the color signature construction. In LFFIR, we use Grid Color Moment (GCM). Each image is partitioned into 3×3 grids and three types of color moments are calculated for each grid. Thus, an 81-dimensional color moment is obtained for the color feature.

The efficient and robust representation of the shape plays an important role in retrieval. For shape feature, we employ an edge direction histogram. A Canny edge detector is used to get the edge image and then the edge direction histogram is computed. Each histogram is quantized into 36 bins of 10° each. An additional bin is used to count the number of pixels without edge information. Hence, a 37-dimensional edge direction histogram is used for the shape feature.

Texture features are intended to describe the granularity and repetitive patterns of surfaces within an image. In image retrieval, a popular way to form texture features is to use Gabor filters. Each image is scaled to 64×64. Gabor wavelet transformation is applied on each scaled image with five scale levels and eight orientations, resulting in 40 sub-images. For each sub-image, three moments are computed: mean, variance, and skewness. Thus, a 120-dimensional feature vector is adopted for the texture feature.

LFFIR adopts early fusion features [30] before performing retrieval. After analysis of the three types of visual features, they are fused into a 238-dimensional feature to represent each image in the database.

4.2 Publishing LSH-based index messages

When a user shares an image, LFFIR constructs and publishes index messages after image features are extracted. The remaining question is how to construct resource *ID* for the fusion feature and answer the query efficiently. LFFIR computes resource *IDs* utilizing *p*-stable LSH whose locality sensitive property is explored [15]. The intuition behind this approach is that the similar fusion features are assigned to the same resource *IDs* and their indexes likely to be stored in the same node.

4.2.1 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 [16]. Thus, a LSH family is defined as: a family $H=\{h: S \rightarrow U\}$ is called (r_1, r_2, p_1, p_2)-sensitive for any two points $q, v \in D$:

$$\text{If } dist(q,v) \leq r_1 \text{ then } \Pr_H(h(q) = h(v)) \geq p_1 \quad (1)$$

$$\text{If } dist(q,v) > r_2 \text{ then } \Pr_H(h(q) = h(v)) \leq p_2 \tag{2}$$

where D specifies the domain of points, *dist* is the distance metric in this domain and Pr is the collision probability.

$Pr_H(h(q)=h(v))$ indicates the collision probability, i.e., the probability of mapping point v and q into the same buckets. If $r_1<r_2$ and $p_1>p_2$, these functions have the property that close feature vectors are more likely to be mapped to the same hash value than those far apart. In practice, several hash tables are built to increase the collision probability.

In this work, we employ the family functions of p-stable LSH [6], which exists for p. Since the Euclidean distance is supposed to be the most widely-used distance metric, the Gaussian distribution working for the Euclidean distance is the 2-stable distribution. The hash function $h_{a,b}$ is defined as follows:

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

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 $h_{a,b}(v): R^d \rightarrow Z$ 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., $G=\{g: R^d \rightarrow Z^k\}$. In this way, if two close feature vectors is hashed by more hash tables, they may obtain the same hash value at least in one hash table g_i, where $i=1,\ldots,m$. 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 $i=1,\ldots,m$. 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., $Z^k \rightarrow N$, without destroying the local sensitivity. 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., *fusID*, the mapping function $\Im(v)$ is defined as:

$$fusID_j = \Im\left(\sum_{i=1}^{k} h_i(v) \cdot d_i\right), \text{where } h \in g_j \text{ and } j = 1, \ldots, m. \tag{4}$$

d_i 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 $\{fusID_1,\ldots, fusID_m\}$. Therefore, given $\varphi_m=\{fusID_1,\ldots, fusID_m\}$, we can map a point v to m Chord keys $fusID_1(v),\ldots, fusID_m(v)$.

Obviously, as discussed in the previous section, similar vectors should have the same *fusID* after this mapping, without destroying local sensitivity. Given two similar vectors v_1 and v_2, we would have $g_i(v_1)=g_i(v_2)$, where $i=1,\ldots,m$. If $t(v)=\sum_{i=1}^{k} h_i(v)\cdot d_i=g_i(v)\cdot[d_1,d_2,\ldots,d_k]^T$. Then we have: $|t(v_1)-t(v_2)|=(g_i(v_1)-g_i(v_2))\cdot[d_1,d_2,\ldots,d_k]^T$ In that way, we have $t(v_1)=t(v_2)$, i.e., if two similar vectors have the same hash value in a hash table, they will have the same *fusID*. Note that function does not destroy the locality sensitive property of LSH.

Algorithm 1 *Index construction*

```
1. Input: v as the image fusion feature
2. g_i, each of which contains k hash functions h_j
3. Generate k random integer numbers, each of which is d_j
4. for each g_i , i=1 to m
5. for each h_j, j=1 to k
6. fusID[i]+=h_j(v)×d_j
7. end for
8. end for
9. for i=1 to m
10. publish the index < fusID[i],v,nodeID > into the DHT layer
11.end for
```

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.

4.2.2 Index construction service

In this section, we describe the detail process of the index construction service (ICS) in LFFIR. The purpose of the service is to cluster the indexes of similar fusion features to the same node with high probability. ICS adopts *p*-stable LSH to preserve fusion feature similarity and distributes the indexes to the Chord as evenly as possible. That is different from the traditional location approach [22], where DHTs access an image through the hash key of a single feature or key words.

ICS generates the index messages based on the same *fusIDs* of the fusion features, and then publishes 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 fusion feature f_v. After that, $m\times k$ hash functions are generated, which map the feature vector to m integer vectors, e.g., $G=\{g: R^d\rightarrow Z^k\}$. Next, each integer vector is mapped to one resource *ID*, e.g. $Z^k\rightarrow N$, without destroying the local sensitivity of LSH. And then ICS maps f_v into m index replicas $\varphi_m=\{fusID_1, fusID_2,\ldots,fusID_m\}$, where $fusID_i=\Im(f_v)$.

After the *fusIDs* φ_m of an image is obtained, ICS constructs indexes in the form of $<fusID_i, f_v, nodeID>$ where $i=1\ldots m$, *nodeID* is the address of the object owner. ICS routes each index message to the node responsible for *fusID* through overly routing, as shown in the right part of Fig. 3. 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 *fusID* in index storage are gathered into the same list to facilitate the location of local indexes. The whole procedure is shown in Algorithm 1.

The number of index replicas 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 above discussed, 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 similar to the query.

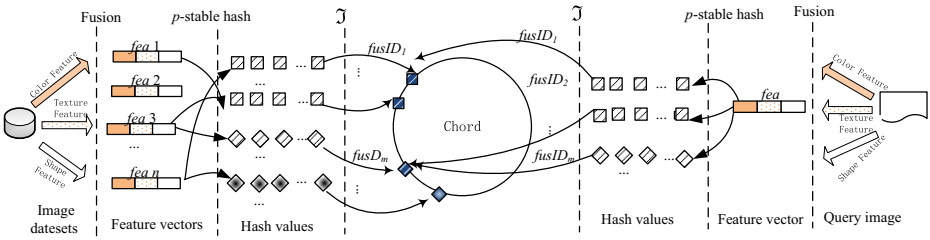


Fig. 3 Publishing the indexes and the query image

Algorithm 2 *Query processing*

```
1. Input: f_q as the query image feature vector
2. g_i, each of which contains k hash functions h_j
3. k integer numbers, each of which is d_j
4. for each g_i , i=1 to m
5. for each h_j, j=1 to k
6. fusID[i]+=h_j(f_q) × d_j
7. end for
8. end for
9. for i=1 to m
10. publish the query message < fusID[i], f_q ,nodeID>
11. end for
12. Merge the results that satisfy the query from all replies
13. Establish connection to the data owner
```

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, e.g., the same *fusID*. That can lead to fewer clusters and accordingly each cluster has more indexes. Once a *fusID* is located, more relevant images can be obtained when it is in fact similar to the query. That can also cause high load of nodes in charge of the *fusID*, 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. Besides, when a query is issued, LFFIR first checks the caches to make sure that there are not suitable answers. Otherwise, the ICS is invoked to publish query messages to Chord.

4.3 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.

4.3.1 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 searches images which have similar color, texture and shape. It is analogous to the process of index constructing. Query images are converted to a set of *fusIDs*. And then the query messages are published to the special nodes.

As above described, the fusion feature vector f_q is transformed into a set of resource $IDs=\{fusID_1, fusID_2,\ldots, fusID_m\}$, where $\text{fusID}_i=\Im(f_q)$ 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 *fusID* in the form of $<fusID_i, f_q, nodeID>$where $i=1\ldots m$, 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 *fusIDs*. Therefore, the similar search of LFFIR is probabilistic and relies on the number of indexes to achieve highly accurate query results.

Once a query message reaches its destination through the DHT layer, the node in charge of *fusIDs* may probably contain the colliding *ID*. It checks the local index storage to find if there exists the same *fusID* as it receives. Afterwards, the node that receives the request computes the similarity between the query feature and the image 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. The whole procedure is shown in Algorithm 2.

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 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 the final 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.

5 Performance evaluation

5.1 Datasets and system settings

In our experiments, two image datasets are used: Corel 10000 and the subset of Caltech 101 Object Categories. The Corel dataset commonly used contains 10,000 images of various contents, such as flowers, food, wave, pills, sunset, beach, car, horses, fish and door, etc. It contains 100 categories and each category contains 100 images in JPEG format. For the

second dataset, 3700 images are randomly chosen from Caltech 101 Object Categories. We choose 75 objects of sunflower, dollar, headphone and faces, etc. And each object contains 50 images. In each category, we randomly choose 20 images, so 2000 queries are drawn from the Corel and 1500 queries from Caltech 101 Object Categories, respectively. Image feature vectors are extracted as described in Section 4.1.

Queries are initiated at randomly chosen nodes, after all the nodes join LFFIR. 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. Two metric is used: Recall and Precision. Recall rate is defined as the percentage of relevant retrieved images among all the relevant images in the dataset. Precision is defined as the percentage of relevant images among the retrieved images. In order to compare with LRFIR [22], we have implemented the method and parameters which are set as described in [22]. Our aim is to evaluate the performance of the distributed indexes of a signal feature and multiple features. To fairly compare against this method, we ignore the processing of relevance feedback in LRFIR. We report here the best results which have achieved a fair balance as well.

We compare our work with the image retrieval framework based on the M-Chord. We select 20 reference points for M-Chord, and generate 20 corresponding clusters. In the experiments, we only visit the top 10 most promising clusters, according to the distance between the query and the reference points. c is set to 10,000.

5.2 Query accuracy

We use two metrics to verify the query accuracy of our system. As shown in Figs. 4 and 5, 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 Fig. 4 and from 5 to 50 for Fig. 5. The y-axis denotes the average recall and precision measured on the top ranked images for both datasets. m and k represents the number of hash tables and buckets, respectively. As both datasets show, 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 while decreasing the number of buckets k, for both datasets.

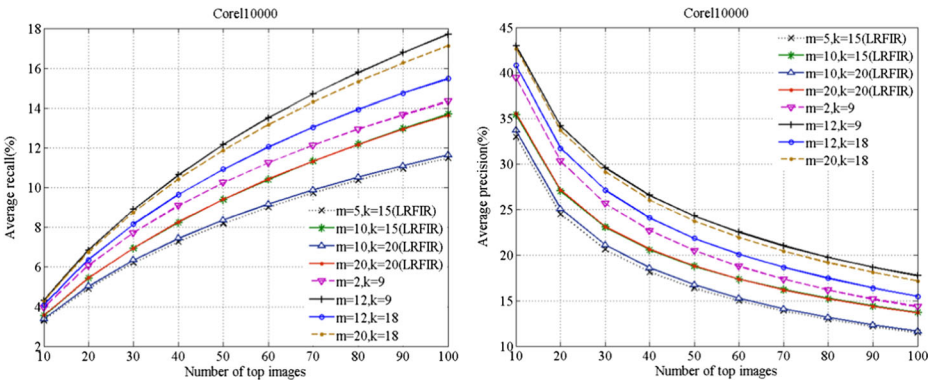


Fig. 4 Accuracy evaluation with Corel. **a** Average recall. **b** Average precision

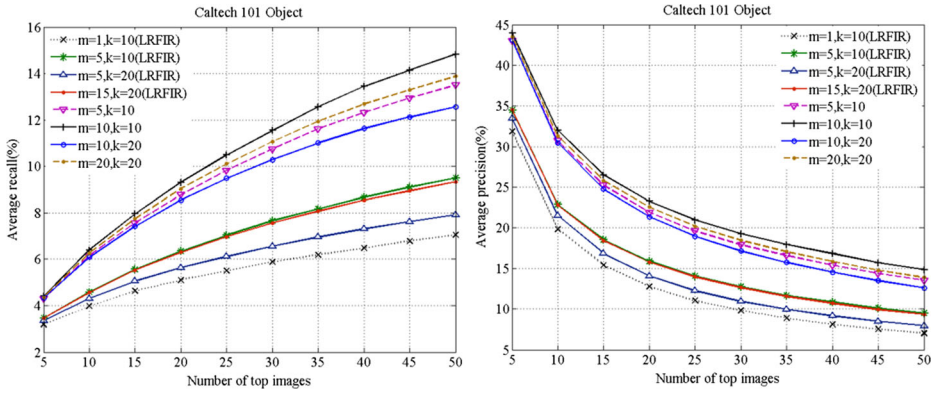


Fig. 5 Accuracy evaluation with Caltech101 Object. **a** Average recall. **b** Average precision

The reason is that when increasing m, the collision probability of content-based similar images grows. On the other hand, decreasing k can lead to fewer clusters, accordingly more images are gathered into a cluster. Once a cluster is located, many relevant results are searched. However, for Fig. 4a, b, both m=12, k=9 and m=20, k=18 achieve the best recall rate and precision. They almost have the same value. A similar observation can be made in Fig. 5a, b, where both m=10, k=10 and m=20, k=20 also achieve the best accuracy. We can choose m=12, k=9 for the Corel and m=10, k=10 for the Caltech101 Object, since the computational overhead is reduced with the small number of hash functions.

Figures 4 and 5 also show comparing results versus the number of hash tables for two datasets. We see a big increase in terms of accuracy for both datasets. For example, under the Corel, the average recall of LFFIR (m=12, k=9) achieves about 6 % improvement over LRFIR (m=10, k=15) on the top 100 images in Fig. 4a. With respect to the average precision, LFFIR represents about 10 % increase on the top 10 images in comparison with LRFIR, as shown in Fig. 4b. For the Caltech101 Object, we can also obtain the improvement of query accuracy. The LFFIR(m=10, k=10) outperforms the LRFIR(m=5, k=10) by 6 % on the top 50 images in Fig. 5a and enjoys 9 % improvement on the top 5 images in Fig. 5b. Even in the worst cases

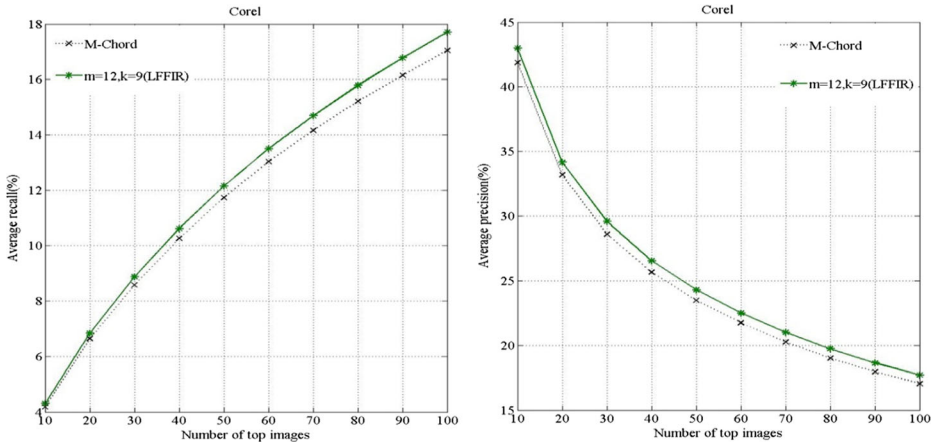


Fig. 6 Corel aquracy under comparison with M-Chord

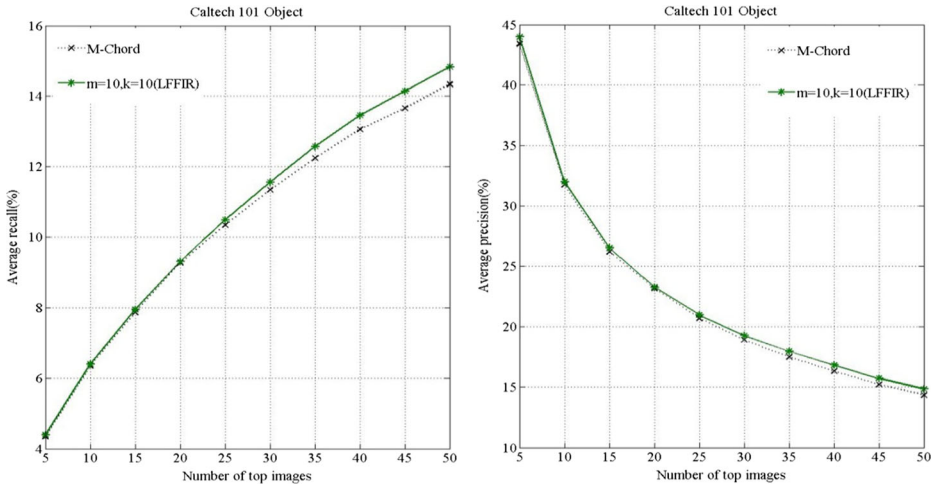


Fig. 7 Caltech aquracy under comparison with M-Chord

where m=10, k=20 for LFFIR and m=1, k=10 for LRFIR, the recall of the former 5 % higher than that of the latter on the top 50 images in Fig. 5a. In terms of precision in Fig. 5b, LRFIR is 11 % lower than LFFIR on the top 5 images.

Figures 6 and 7 show the comparison of LFFIR and M-Chord in terms of the query accuracy. They have almost the same recall and precision rates in two datasets.

5.3 Load balancing

In this section, Figs. 8 and 9 show the effectiveness of load balancing with various arrangements. We define the node load as the number of index messages stored on a node. In Fig. 8, the x-axis shows the percentage of nodes, where the number of nodes varies from 10 to 5000, and the corresponding values represent the percentage of indexes assigned to these nodes. Note both datasets show much less skew on load distribution as the number of nodes increases. That indicates 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.

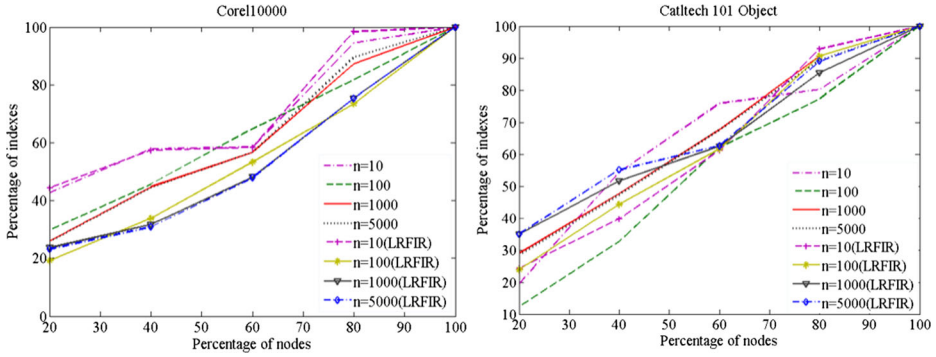


Fig. 8 Effect of load on the index distribution with different nodes. **a** Corel **b** Caltech 101 Object

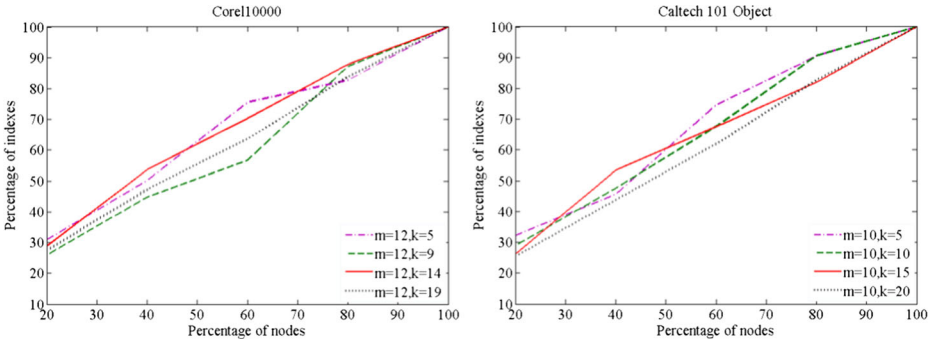


Fig. 9 Effect of load on the index distribution. **a** Corel **b** Caltech 101 Object

In Fig. 8, when the number of node ranges from 1000 to 5000, the load is more balanced than other cases. However, for the Corel, when n is 10, the load of LFFIR is skewed and 40 % of nodes stores around 60 % 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 LFFIR for both datasets appear to be much steadier, especially when n= 1000 and n=5000. It indicates that the load balancing of LFFIR is better than that of LRFIR.

Figure 9a 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. Taking the Corel for example, the line of m=12, k=9 is much smooth than others. In the above experiment, we reasonably choose k=9 for Corel, k=10 for Caltech101 Object, respectively. 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. Similar conclusion can be drawn in the Caltech 101 Objects dataset, as shown in Fig. 9b. As previously mentioned, the number of buckets has a larger impact on the network load.

Figure 10 shows load balancing in comparison to M-Chord. The M-Chord line growths steadily comparing with LFFIR(m=12, k=9) for Corel and LFFIR (m=10, k=10) for Caltech 101, which implies the load of M-Chord is more balancing than LFFIR. The reason is that M-Chord assigns a resource *ID* to an image, while some images share an index in LFFIR. Thus, M-Chord can allocate indexes to more Chord nodes.

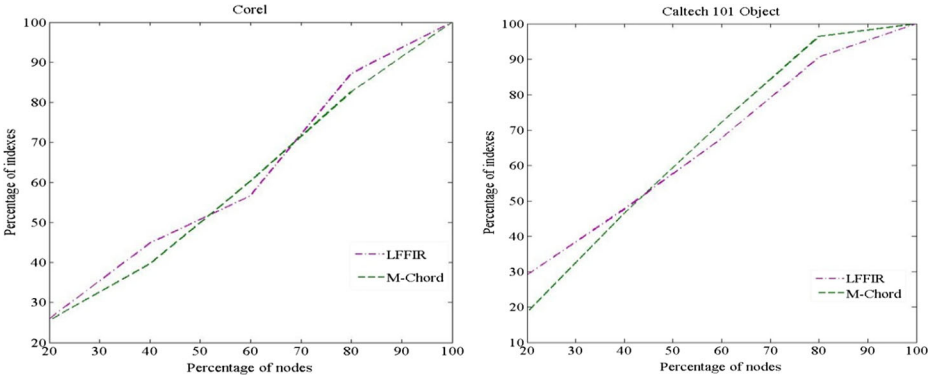


Fig. 10 Load under comparison with M-Chord **a** Corel **b** Caltech 101

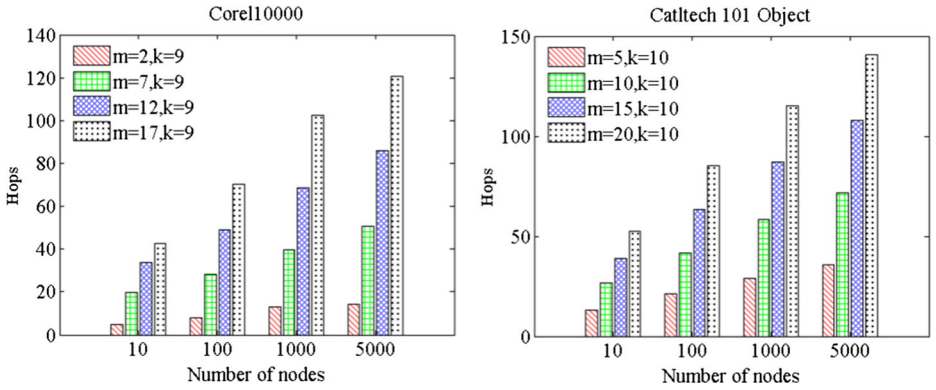


Fig. 11 Network hops vs. hash tables No. **a** Corel **b** Caltech 101 Object

5.4 Network hops

The effect of network hops is depicted, as shown in Figs. 11 and 12. 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 2 to 17 for Fig. 11a and from 5 to 20 for Fig. 11b. 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 for both datasets. In Fig. 11a, however, for a large number of nodes, $n=1000$ and $n=5000$, the number of hops only has a slight increase. The similar conclusions can be drawn in Fig. 11b.

As we expect, the number of lookup hops is independent on the number of buckets In Fig. 12a, b, especially for $n=100$ and $n=1000$, the number of hops remains almost the same when the number of buckets varies.

We can see that the lookup process of LFFIR does not need many network hops. So $m=12$, $k=9$ for Corel and $m=10$, $k=10$ for Caltech101 Object, can be chosen respectively to guarantee the best query accuracy while not incurring too many network hops.

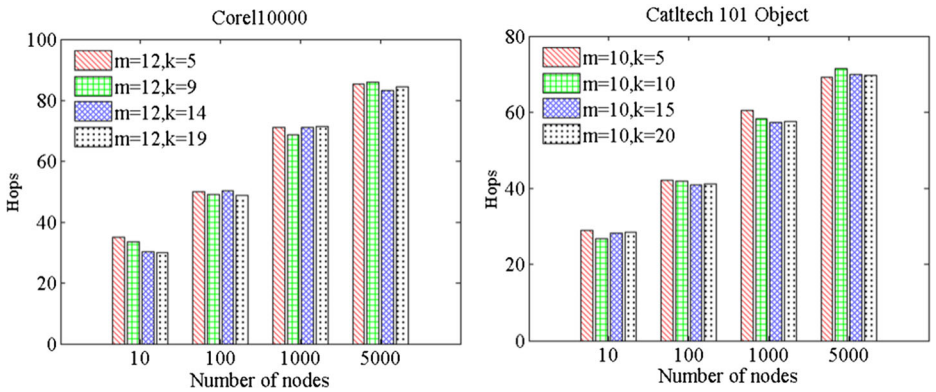


Fig. 12 Network hops vs. buckets No. **a** Corel **b** Caltech 101 Object

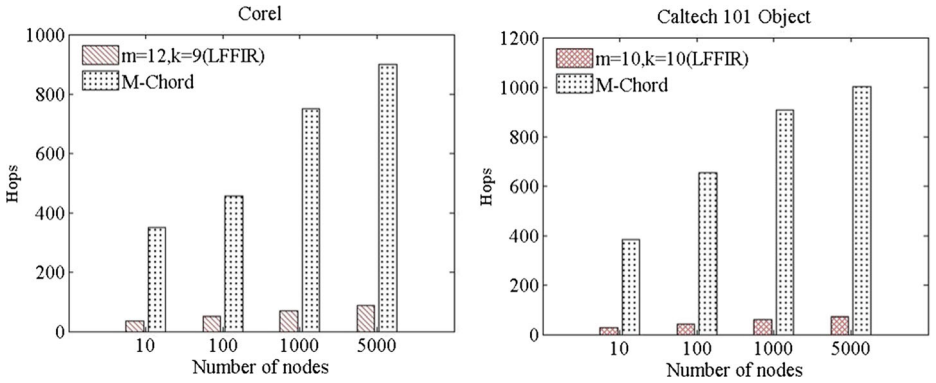


Fig. 13 Lookup hops under comparison with M-Chord

Figure 13 shows the comparison of lookup hops. LFFIR responses the query with small number hops in comparison with M-Chord. The reason is that LFFIR sends the query to the most promising nodes. For example, $m=12$, 12 query messages are sent, and up to 12 nodes visited for the query. A query can be answered in a small number of lookup hops, e.g., 50–60. However, M-Chord transfers the kNN query to range queries in each cluster. And the large range means to visit a lot of response node's neighbors. Although visiting the direct successor needs just only one hop, visiting the predecessor many hops which increases the query cost.

6 Conclusions

We propose an multi-feature fusion framework to support CBIR in the distributed cloud datacenter which stores a huge number of resources. LFFIR is implemented on the DHT which provides efficient routing mechanisms. The ICS constructs the distributed indexes based on fusion features and the property of p-stable hash functions. Therefore, the indexes of content similar images are probabilistically clustered into the same node. The QPS processes the query image and publishes the query messages through overlay routing. The experiments show that our approach achieves high accuracy with well balancing. Comparing with the image framework based on M-Chord, it only needs a small numbers of network hops to response the query.

As future work, we plan to investigate the following issues. Firstly, some sensitive information is being centralized into the cloud, so we may search over encrypted cloud data. Secondly, image quality may not be very high, which can be improved by using the image pretreatment technique before processing the query.

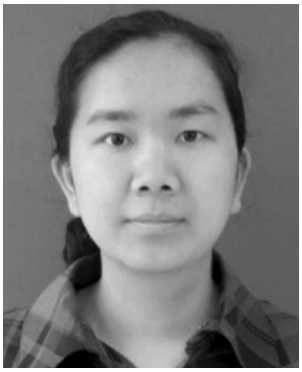
Acknowledgments This work was jointly supported by: (1) the National Basic Research Program of China (No. 2013CB329102); (2) National Natural Science Foundation of China (No. 61471063, 61421061, 61372120,61271019, 61101119, 61121001); (3) the Key(Keygrant) Project of Chinese Ministry of Education.(No. MCM20130310); (4) Beijing Municipal Natural Science Foundation (No. 4152039); (5) Beijing Higher Education Young Elite Teacher Project (No. YETP0473); (6)Spanish Research Council (No: TIN2013-46883); (7)Regional Government of Madrid (No: S2013/ICE-2894) cofunded by FSE & FEDER.

References

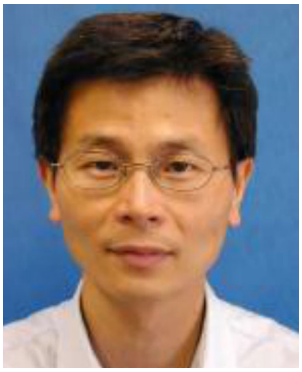
1. Androutsos P, Androutsos D, Venetsanopoulos AN (2006) A distributed fault-tolerant MPEG-7 retrieval scheme based on small world theory. IEEE Trans Multimed 8(2):278–288
2. Batko M, Falchi F, Lucchese C et al (2010) Building a web-scale image similarity search system[J]. Multimed Tools Appl 47(3):599–629
3. 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
4. Chen J, Hu C, Su C (2008) Scalable retrieval and mining with optimal peer-to-peer configuration. IEEE Trans Multimed 10(2):209–220
5. 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
6. Datar M, Immorlica N, Indyk P, Mirrokni VS (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
7. Datta R, Joshi D, Li J, Wang JZ (2008) Image retrieval: ideas, influences, and trends of the new age. ACM Comput Surv 40(2), Article 5
8. Dikaiakos MD, Katsaros D, Mehra P, Pallis G, Vakali A (2009) Cloud computing: distributed internet computing for IT and scientific research. IEEE Internet Comput 13(5):10–13
9. Eisenhardt M, Muller W, Henrich A, Blank D, Allali SE (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
10. 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
11. Forestiero A, Leonardi E, Mastroianni C, Meo M (2010) Self-chord: a bio-inspired P2P framework for self-organizing distributed systems. IEEE/ACM Trans Netw 18(5):1651–1664
12. Gaeta R, Sereno M (2011) Generalized probabilistic flooding in unstructured peer-to-peer networks. IEEE Trans Parallel Distrib Syst 22(12):2055–2062
13. Gnutella (2000) Gnutella website. http://www.Gnutella.com
14. Guo C, Lu G, Li D et al (2009) BCube: a high performance, server-centric network architecture for modular data centers[J]. ACM SIGCOMM Comput Commun Rev 39(4):63–74
15. 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
16. 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
17. Jagadish HV, Ooi BC, Vu QH (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
18. Kalnis P, Ng WS, Ooi BC, Tan K (2004) Answering similarity queries in peer-to-peer networks. Inf Syst 31(1):57–72
19. King I, Ng CH, Sia KC (2004) Distributed content-based visual information retrieval system on peer-to-peer networks. ACM Trans Inf Syst 22(3):477–501
20. Li F, Fergus R, Perona P (2007) Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. Comput Vis Image Underst 106(1):59–70
21. 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 Commun Mag 50(1):90–97
22. Liao J, Yang D, Li T, Wang J, Qi Q, Zhu X (2014) A scalable approach for content based image retrieval in cloud datacenter. Inf Syst Front 16(1):129–141
23. Liu G, Zhang L, Hon Y, Li Z, Yang J (2010) Image retrieval based on multi-texton histogram. Pattern Recogn 43(7):2380–2389
24. 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
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. 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
27. Peng C, Ksim 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
28. 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
29. Sahin OD, Gulbeden A, Emekci F, Agrawal D, Abbadi AE (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
30. Snoek C,Worring M, Smeulders AWM (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
31. Sripanidkulchai K, Maggs BM, 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
32. Stoica I, Morris R, Karger D, Kaashoek MF, 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
33. 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
34. Tian X, Jiao L, Liu X, Zhang X (2014) Feature integration of EODH and Color-SIFT: application to image retrieval based on codebook. Signal Process Image Commun 29(4):530–545
35. Urdaneta G, Pierre G, Steen MV (2011) A survey of DHT security techniques[J]. ACM Comput Surv (CSUR) 43(2):8
36. Vlachou A, Doulkeridis C, Kotidis Y (2012) Metric-based similarity search in unstructured peer-to-peer systems. Trans Large Scale Data Knowl Centered Syst 5:28–48
37. Wang X, Zhang B, Yang H (2014) Content-based image retrieval by integrating color and texture features. Multimed Tools Appl 68(3):545–569
38. Yang Z, Zhao BY, 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
39. 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
40. Zhu Y, Hu Y (2007) Efficient semantic search on DHT overlays. J Parallel Distrib Comput 67(5):604–616

Jianxin Liao was born in 1965, obtained his PhD degree at University of Electronics Science and Technology of China in 1996. He is presently a professor of Beijing University of Posts and Telecommunications. He has published hundreds of papers in different journals and conferences. His research interests are mobile intelligent network, broadband intelligent network and 3G core networks. He is the Specially-invited Professor of the "Yangtse River Scholar Award Program" by the China Ministry of Education in 2009.



Di Yang received the B.S. degree in computer science and technology from the Liaoning Normal University, in 2008 and the M.S. degree in computer software and theory from Liaoning Technical University, in 2011. She is currently pursuing the Ph.D. degree in computer science and technology at Beijing University of Posts and Telecommunications. Her research interest includes peer-to-peer network, information retrieval, image processing and software defined network.



Tonghong Li obtained his Ph.D. degree from Beijing University of Posts and Telecommunications in 1999. He is currently an assistant professor with the department of computer science, Technical University of Madrid, Spain. His main research interests include resource management, distributed system, middleware, wireless networks, and sensor networks.

Qi Qi was born in 1982, obtained his Ph.D. degree from Beijing University of Posts and Telecommunications in 2010. Now she is an assistant professor in Beijing University of Posts and Telecommunications. Her research interests include SIP protocol, communications software, Next Generation Network, Ubiquitous services, and multimedia communication.

Jingyu Wang was born in 1978, obtained his Ph.D. degree from Beijing University of Posts and Telecommunications in 2008. Now he is an associate professor in Beijing University of Posts and Telecommunications, China. His research interests span broad aspects of performance evaluation for Internet and overlay network, traffic engineering, image/video coding, multimedia communication over wireless network.

Haifeng Sun is a Ph.D. student from Beijing University of Posts and Telecommunications. His main research interests include semantic system, information retrieval and data mining.