A scalable approach for content based image retrieval in cloud datacenter

Jianxin Liao, Di Yang, Tonghong Li, Jingyu Wang, Qi Qi & Xiaomin Zhu

Information Systems Frontiers
A Journal of Research and Innovation

ISSN 1387-3326

Inf Syst Front
DOI 10.1007/s10796-013-9467-0

Springer

Your article is protected by copyright and all rights are held exclusively by Springer Science +Business Media New York. This e-offprint is for personal use only and shall not be self-archived in electronic repositories. If you wish to self-archive your article, please use the accepted manuscript version for posting on your own website. You may further deposit the accepted manuscript version in any repository, provided it is only made publicly available 12 months after official publication or later and provided acknowledgement is given to the original source of publication and a link is inserted to the published article on Springer's website. The link must be accompanied by the following text: "The final publication is available at link.springer.com".

Author's personal copy

Inf Syst Front
DOI 10.1007/s10796-013-9467-0

A scalable approach for content based image retrieval in cloud datacenter

Jianxin Liao · Di Yang · Tonghong Li · Jingyu Wang · Qi Qi · Xiaomin Zhu

© Springer Science+Business Media New York 2013

Abstract The emergence of cloud datacenters enhances the capability of online data storage. Since massive data is stored in datacenters, it is necessary to effectively locate and access interest data in such a distributed system. However, traditional search techniques only allow users to search images over exact-match keywords through a centralized index. These techniques cannot satisfy the requirements of content based image retrieval (CBIR). In this paper, we propose a scalable image retrieval framework which can efficiently support content similarity search and semantic search in the distributed environment. Its key idea is to integrate image feature vectors into distributed hash tables (DHTs) by exploiting the property of locality sensitive hashing (LSH). Thus, images with similar content are most likely gathered into the same node without the knowledge of any global information. For searching semantically close images, the relevance feedback is adopted in our system to overcome the gap between low-level features and high-level features. We show that our approach yields high recall rate with good load balance and only requires a few number of hops.

Keywords Cloud computing · Content based image retrieval · Peer-to-peer · Locality sensitive hashing · Relevance feedback

J. Liao · D. Yang (✉) · J. Wang · Q. Qi · X. Zhu
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, P.O. Box 296, Beijing 100876, China
e-mail: yangdi.bupt@gmail.com

D. Yang
e-mail: yangdi@ebupt.com

J. Liao
e-mail: liaojx@bupt.edu.cn

J. Wang
e-mail: wangjingyu@bupt.edu.cn

Q. Qi
e-mail: qiqi8266@bupt.edu.cn

X. Zhu
e-mail: zhuxm@bupt.edu.cn

T. Li
Department of Computer Science, Technical University of Madrid, Madrid 28660, Spain
e-mail: tonghong@fi.upm.es

1 Introduction

Cloud computing enables users to flexibly access reconfigurable computing resources without the burden of managing and maintaining the resources (Peng et al. 2012). The paradigm has brought about the essential characteristics including reliable and infinite storage capacity, data access independent of locations and time, and dynamical resources provision in a multi-tenant way to avoid costly wasting (Dikaiakos et al. 2009). Due to these features, cloud computing becomes prevalent in the distributed storage and retrieval field. On the other hand, constructing a robust storage model is also a driving force for the development of cloud computing. In fact, the datacenter provides a substrate for high capacity storage models and modern Internet applications. From cloud providers' viewpoint, datacenters provide the illusion of unlimited and powerful information storage, with the purpose of offering on-demand high quality applications, e.g. Amazon's S3 and Microsoft's SkyDrive and Live Mesh. From cloud consumers' viewpoint, large datacenters can provide an online service running on the cloud, which permits fast data access.

Cloud-based services rely on the datacenters which store massive data (Demirkan and Delen 2012). Therefore, it is important to choose appropriate topology to establish the high-performance datacenters which can satisfy the

Published online: 26 October 2013

Author's personal copy

requirements of searching and analyzing large dispersive datasets. The datacenter architecture is categorized into two models: centralized and decentralized. Most commercial cloud offerings are centralized. In such a model, resources in large datacenters are centrally managed and the management nodes become bottlenecks. Moreover, since datacenters scale exponentially with the rapid growth of data volume, a single point of failure may occur due to fires, power outages, natural disasters, etc (Yang et al. 2010). To address this problem, decentralized datacenters are designed according to the peer-to-peer (P2P) paradigm, which provide better scalability and adaptability. The P2P based datacenters can be built by connecting many individual peers, without any central monitoring or coordination components. Each peer takes charge of part of data and replicas to improve the clouds' reliability. P2P techniques are very likely to be adopted in Clouds (Forestiero et al. 2010).

Nowadays, since smart phones, tablet computers and many lightweight devices have been penetrating into our lives, millions of files including images, videos and plain texts are transferred into datacenters. Figure 1 shows an application scenario of image retrieval in the P2P datacenter. Content providers can be cloud providers or cloud customers. The content providers upload their significant resources to P2P cloud datacenters. In addition, cloud customers can also upload interesting images to their Facebooks or Microblogs 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, query results are sent back to the cloud customer. However, it is challenging to locate files in such a distributed datacenter storing a large amount of files. As a case of study, we present this 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 similarity search is performed in a P2P paradigm.

Most current work about image retrieval in the P2P paradigm assumes that images are described in text by users (Gnutella 2000; Lv et al. 2002; Bawa et al. 2003). The searching of images is only based on their names and mainly relies on keywords matching. However, since it is difficult to annotate images very exactly, identifying of images in this way is inaccurate and cannot satisfy 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 (Kalnis et al. 2004). The content based image retrieval (CBIR) can find similar images through sample images instead of keywords. But most work in CBIR needs the global information in a centralized fashion, which does not scale well in the distributed situation (Lee and Guan 2004). Therefore, our objective is to design a system which can process smart queries and improve search performance in terms of precision and recall rate, without any global information.

In this paper, we present a novel CBIR system called LSH-based and Relevance Feedback for Image Retrieval (LRFIR) for the large scale P2P datacenter. LRFIR supports both content similarity search and semantic similarity search, which is different from the keyword search used in existing systems. The efficient index construction service and query processing service are proposed for LRFIR.

In the index construction service, LRFIR leverages an image feature extraction algorithm called multi-texton histogram (MTH) (Liu et al. 2010), which combines the texture and

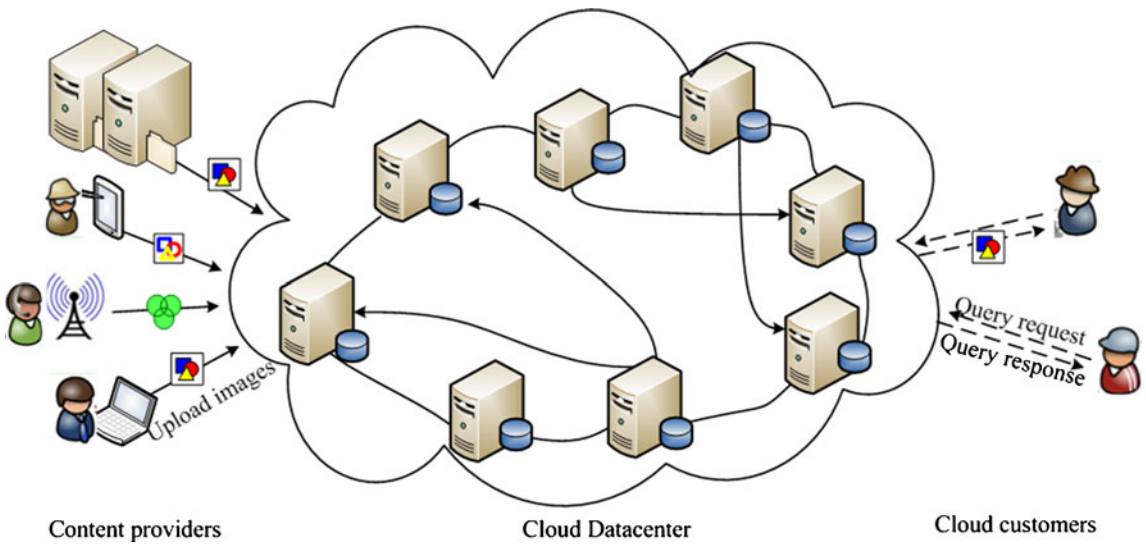

Fig. 1 Image retrieval in the P2P datacenter

Author's personal copy

color feature. Thus, each image's content can be represented as a feature vector from which the content similarity can be measured quantitatively. Accordingly, we employ a set of locality sensitive hashing (LSH) functions (Indyk and Motwani 1998; Datar et al. 2004), which convert feature vectors of similar images to the same hash value with high probability, owing to the locality-preserving property. 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 node in the DHT layer with high probability. When a query is issued, the query processing service produces a set of resource IDs from the query's feature vector by using the same LSH functions. Query messages are then forwarded to the nodes responsible for the IDs. To support semantic search, the relevance feedback technique (Zhou and Huang 2003), originated from information retrieval technique, is adopted to overcome the gap between low-level features and high-level features. It allows users to iteratively refine the result by marking a set of relevant and non-relevant images so that LRFIR can learn the semantics of the query image. Finally, we implement a prototype system based on Next Generation Service Overlay Network (NGSON) (Liao et al. 2012), in which different functional overlays can systematically be coordinated with each other. It is used to evaluate the performance of our algorithm in two image datasets, i.e., Corel 10000 (Liu et al. 2010) and the subset of Catltech 101 Object Categories (Li et al. 2007). The experiments show that our algorithm achieves high recall rate and precision with only a small number of lookup hops. Moreover, query results can be further improved by using the relevance feedback technique.

In this paper, our main contributions are as follows: (1) the image feature vector is integrated into DHT for implementing an efficient indexing and locating approach. The index construction is based on the image content represented by the feature vector instead of keywords; (2) the characteristic of LSH is exploited to place the similar images to the same node without any global information, and the query is only sent to the nodes which are more likely to answer it. In this way, the communication cost is reduced while the result accuracy is guaranteed; (3) we introduce the relevance feedback to the P2P model with the purpose of supporting semantically close image search. This approach allows users to interact with LRFIR to refine the query vector; (4) we evaluate our approach using two real-world image datasets and demonstrate that LRFIR is very effective.

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

2 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 are also applied to the image retrieval.

Searching over unstructured P2P system like Gnutella (2000) relies on flooding queries to all neighbor nodes. Lv et al. (2002) 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 (2002). 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 (2011) choose the neighbor node to forward the query, according to the probability functions 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 the multimedia retrieval based on the name 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. There are many methods employing this model, such as SETS (Bawa et al. 2003), metric space (Vlachou et al. 2012), interesting-based location solution (Sripanidkulchai et al. 2003), DISCOVIR (King et al. 2004), P2P-CBIRM (Chen et al. 2008), and SWIM (Androutsos et al. 2006). Bawa et al. (2003) propose a topic-segmented overlay which assigns nodes with similar content (topic) to the same group. But this method needs center nodes to manage topics segment and suffers from the single point of failure. Vlachou et al. (2012) propose that peers sharing similar data are linked to the same super node, while super nodes are organized as an M-Tree structure. But it still needs centralized management within the community. The decentralized interesting-based location solution loosely organizes the peers into interesting-based structure for fast content location, where each peer creates an interesting-based shortcut to another peer with interested content. But it still relies on message flooding when there is no shortcut available. 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

Author's personal copy

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 MPEG-7 descriptor similarities. However, due to the lack of global information, it is difficult for these methods to discover the new topics that do not belong to the current topic clusters, without broadcasting signature messages to the overall network.

Regarding information retrieval in the decentralized structured P2P paradigm, there are many studies in this issue, such as MCAN (Falchi et al. 2005), M-Chord (Novak and Zezula 2006), Psearch (Tang et al. 2003), Prism (Sahin et al. 2005) and iDISQUE (Zhang et al. 2010b). MCAN using CAN as the underlying structure adopts a pivot technique to map data objects to N-dimensional vectors . 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 model. 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. And LSI still works in a centralized fashion. In Prism, it stores multiple indexes for one object in many Chord peers based on the distances between the object's vector and the reference vectors, so that the indexes of similar object 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 and Hu (2007) generate the same index for semantically close files by using LSH and Vector Space Model (VSM), with the purpose of answering queries by only visiting a small number of nodes. But these hash values are directly used as resource keys, which destroy the load balance of Chord. In iDISQUE framework, the data on each peer is clustered, and then LSH functions only map cluster centers to Chord resource keys. The key of a cluster center represents the data in the cluster. However, the hash values of queries may be not equal to these of cluster centers.

Considering the relevance feedback, originated from the well-known information retrieval, Lee and Guan (2004) use Gaussian-shaped radial-basis function network (RBFN) as a feedback model. And peers with the similar image are gathered into the same community. Meanwhile, images in each community are needed to input RBFN for determining whether images are relevant to the query. This approach is less practical in decentralized networks, since every community must keep the same RBFN model which can be improved in each feedback iteration. In MURK (Zhang et al. 2010a), relevance feedback is also used to improve the efficiency, where peers are organized as the Kd-tree structure that does not scale well when data dimensions are high.

3 System framework

In this section, we present an overview of LRFIR 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 traversing 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 (Sahin et al. 2005). Hence we propose a scalable scheme that distributes the indexes of content similar images to the same node and returns approximate answers by only visiting a small number of nodes. Besides, relevance feedback technique is also adopted to support semantic search.

In the underlying DHT layer of LRFIR, the participating nodes are organized into a structured P2P network, Chord (Stoica et al. 2001), without loss of generality, which also natively offers node join and leave mechanisms. Therefore, LRFIR can support efficient routing, due to the DHT layer. The index publishing and query routing are automatically accomplished by Chord. For each image, LRFIR constructs a few index messages, each of which contains the resource ID, the image feature vector and the IP address of the data owner. Since only feature vectors whose dimensionalities are not very high are added to the indexes, each index will not cause significant storage overhead to the nodes. Given a query message, it is only forwarded to a few particular nodes which are likely to store indexes of similar images. To achieve the objective of efficiently searching similar images for the query, the index construction and query processing should satisfy the following requirements: (1) the index of content similar images should be stored at the same peer; (2) the semantic search should be supported with the help of users.

To satisfy these requirements, LRFIR constructs indexes based on the image feature vectors and further improves the results by using the relevance feedback. To meet the first requirement, a set of LSH functions are employed to produce the image indexes. At the time of the system start, these hash functions are generated and used in all nodes. Due to the locality sensitive property of LSH, it is more likely that the feature vectors of content similar images have the same hash values. Then these hash values denoted by the integer vectors are mapped to resource IDs. Therefore, if two images are similar, they may share the same resource ID. On the other hand, considering load balance, these indexes are distributed as evenly as possible. To satisfy the second requirement and improve the retrieval performance, LRFIR employs the relevance feedback by leveraging the information retrieval

Author's personal copy

technique, which can narrow the gap between low-level concepts and high-level features. In addition, the human can interact with LRFIR to help refine the query. LRFIR can learn the semantics of query images through iterative feedback and query refinement.

The interactions among key components of LRFIR are illustrated by Fig. 2. LRFIR is located between the user 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 extractor which consists of a set of feature extractors specific to different image formats. The image feature extractor accesses each image in the local database, which adopts MTH algorithm to analyze the image's texture feature and compute its feature vector. For the feature vector, a set of LSH functions are adopted to compute the hash values and determine the number of indexes for each image. For each hash value, the index construction generates the resource ID and publishes the index message to the DHT layer. Once receiving an index message from the DHT layer, the node inserts it in the index storage according to the resource ID. In addition, the image indexes in the 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 message is routed through the underlying DHT layer using the resource ID as the destination. Once receiving the query message, the destination node invokes the local search method which checks the local index storage according to the resource ID and then returns the top T indexes. After the user surface of the query node merges all the results obtained from the participating nodes, the final results are shown to the user. Furthermore, the query refinement method is used for relevance feedback. The query refinement re-computes the query vector according to the relevant images the user chooses. And the new query message is generated and routed through the DHT layer again. The relevance feedback iteration stops when the user ends it.

4 DHT-based CBIR approach

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

4.1 Image features extraction

When a node wants to share an image, the feature vectors of the image are automatically extracted. LRFIR adopts Motion Picture Experts Group-7 (MPEG-7) descriptors, which represent visual contents with feature values. 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 (Datta et al. 2008). These visual descriptors represent human visual perception as feature vectors to evaluate the similarity of two

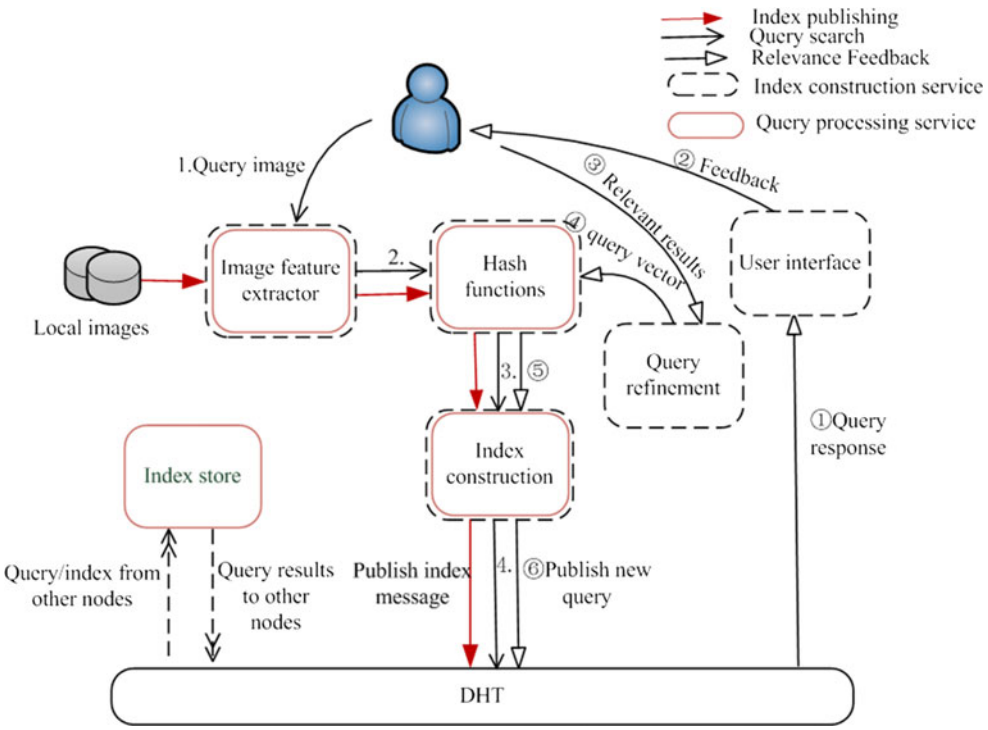

Fig. 2 Interactions between key components of LRFIR

Author's personal copy

images in appearance. The index construction is based on the visual feature space.

In LRFIR, texture descriptors are used to extract the visual feature of an image. The texture describes the granularity and repetitive patterns of surfaces within an image (Datta et al. 2008). A novel and efficient method called MTH (Liu et al. 2010) is adopted to represent the texture feature. MTH explores the spatial correlation between neighboring colors and the one between neighboring texture orientations. Then it takes advantage of the histogram and co-occurrence matrix to improve the texture features. Compared with MTH, other approaches such as machine learning techniques (Datta et al. 2008; Liu et al. 2010) generally train some examples to learn a classifier which should be kept consistency in all the nodes. So these approaches are impractical in the distributed environment.

For a color image, its texture orientation is firstly detected, which can be used to estimate the shape of the textured images. Sobel operator is applied as the gradient operator, which returns two gradient images, i.e., the image along horizontal and the one along vertical. Sobel operator is used to the red, green and blue channel. And then texture orientation image θ (x, y) is obtained.

Secondly, in the RGB color space, R, G and B are respectively quantized into 64 colors, for simplification. And the quantized image is denoted by C (x, y).

Thirdly, four different texton templates are designed to detect texture in C (x, y), and each of them is a 2×2 grid. In the color image, the grid is moved from left-to-right and top-to-bottom with 2 pixels in one step. If a texon is found, the original pixel values in the grid are kept unchanged. Otherwise, they become zero. In the end, we obtain a texton image, T (x, y). Finally, the image features presented by MTH descriptor is defined as:

$$H(T(P_1)) = N\left\{\theta(P_1) = v_1 \wedge \theta(P_2) = v_2 \middle| |P_1 - P_2| = D\right\} where \theta(P_1) = \theta(P_2) = v_1 = v_2 \quad (1)$$

$$H(\theta(P_1)) = \overline{N}\left\{T(P_1) = w_1 \wedge T(P_2) = w_2 \middle| |P_1 - P_2| = D\right\} where\, T(P_1) = T(P_2) = w_1 = w_2 \quad (2)$$

Where $P_1=(x_1, y_1)$ and $P_2=(x_2, y_2)$, P_1 and P_2 specify two neighboring pixels whose distance is D. The texton image T (x, y) at P_1 and P_2 is denoted as $T(P_1)=w_1$ and $T(P_2)=w_2$, respectively. The angle at P_1 and P_2 is denoted as $\theta(P_1)=v_1$ and $\theta(P_2)=v_2$ in the texture orientation image θ (x, y), respectively. N denotes the co-occurring number of v_1 equal to v_2, and $\overline{N}$ denotes the co-occurring number of w_1 equal to w_2. $H(T(P_1))$ represents the spatial correlation between neighboring texture orientations by using the color information. $H(\theta(P_1))$ represents the spatial correlation between neighboring colors by using the texture orientation information. So the image's feature vector f_v is defined as:

$$f_v = H(T(P_1)) \odot H(\theta(P_1)) \quad (3)$$

where $\odot$ means the join operation.

Therefore, the similarity between image a and b is defined by

$$S_I(x,y) = \|f_v(a) - f_v(b)\|^{-1} \quad (4)$$

where $\|f_v\|$ denotes the Euclidean 2-norm of f_v.

4.2 LSH based index construction

After image features are extracted, the question is how to construct resource IDs for images and answer the query efficiently. To improve the search efficiency, the content-based similar images measured in Euclidean space should share the same resource ID. The resource identifier space of Chord is one-dimension, while the dimension of feature vectors may be very high. To overcome the problem, the locality sensitive property of p-stable LSH is exploited (Haghani et al. 2009). $m \times k$ hash functions are generated, which map the feature vector to m integer vectors. And each of the mapping is denoted as: $R^d \rightarrow Z^k$. Next, each integer vector is mapped to one resource ID, denoted as: $Z^k \rightarrow N$, without destroying the locality sensitive property of LSH. For each image, after the resource ID is constructed, an index message is sent to the node responsible for the ID through the DHT layer.

4.3 $R^d \rightarrow Z^k$

In this process, LSH is the core of the mapping. The key idea of LSH is that the close feature vectors with small Euclidean distance are hashed into the same value with high probability (Indyk and Motwani 1998). That is, the collision probabilities for the feature vectors close to each other are much higher than those far apart. 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, $v \in S$:

$$\text{If } dist(q,v) \le r_1 \text{ then } \Pr_H(h(q) = h(v)) \ge p_1 \quad (5)$$

$$if\ dist(q,v) > r_2 \text{ then } \Pr_H(h(q) = h(v)) \le p_2 \quad (6)$$

Author's personal copy

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

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 functions are built to increase the collision probability.

In this paper, we employ the family functions of p-stable LSH (Datar et al. 2004), 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 defined as the 2-stable distribution. The hash function $h_{a,b}$ is defined as follow:

$$h_{a,b}(v) = (a \cdot v + b)/W \tag{7}$$

Where a is a d-dimensional vector whose elements are chosen independently from the *2*-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 to an integer.

In particular, the gap between the "high" probability p_1 and the "low" probability p_2 is amplified through constructing m hash tables $G=\{g_1,...,g_m\}$, where m is randomly chosen. Each hash table is defined as k independent hash buckets $g(v)=(h_1(v),\ldots,h_k(v))$, and each table $G=\{g: R^d \rightarrow Z^k\}$ maps a d-dimensional vector to a k-dimensional integer vector, i.e., the hash value. In this way, if the number of hash tables is large enough, close feature vectors have a greater chance to have the same hash value at least in one hash table g_i, where $i=1,\ldots,m$.

4.3.1 $Z^k \rightarrow N$

As a result, an integer vector Z^k is obtained from one hash table $g_i(v)$, where $i=1,\ldots,m$. In the next step, the k-dimension space is transformed to the one-dimension space, i.e., $Z^k \rightarrow N$, without destroying the locality sensitive property. On the other side, the load, defined as the number of indexes on a node, should be kept balanced as much as possible. To construct a resource ID i.e., resID, the mapping function $\mathfrak{I}(v)$ is defined as:

$$resID_j = \mathfrak{I}\left(\sum_{i=1}^{k} h_i(v) \cdot d_i\right), \ where \ \ h \in g_j \ and \ j = 1, \ldots, m. \tag{8}$$

d_i is a randomly chosen integer. $\mathfrak{I}$ function is denoted as the consistent hash function SHA-1.

Obviously, as discussed in the previous section, similar vectors should have the same resID after this mapping, without destroying the locality sensitive property. 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:

$$\begin{aligned} |t(v_1)-t(v_2)| &= g_1(v) \cdot [d_1, d_2, \ldots, d_k]^T - g_2(v) \cdot [d_1, d_2, \ldots, d_k]^T \\ &= (g_1(v)-g_2(v)) \cdot [d_1, d_2, \ldots, d_k]^T. \end{aligned}$$

In this way, we have $t(v1)=t(v2)$, i.e., if two similar vectors have the same hash value in a hash table, they will have the same resID. Note that $\mathfrak{I}$ function does not destroy the locality sensitive property of LSH.

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

4.3.2 Index construction service

The purpose of this service is to generate the same resIDs for the similar images with high probability, and then publish the indexes to particular nodes through the DHT layer. That is different from the traditional location approach, where DHTs access an image through the hash key of the image name annotated by the human. Thus, the indexes of the similar images are randomly distributed across the DHT. As a result, it is difficult to guarantee the search accuracy. In this paper, we propose the Index Construction Service (ICS) which adopts p-stable LSH to preserve the locality sensitive property and distributes the indexes to the Chord as evenly as possible.

For each image in the local database, the image feature extractor is firstly invoked to extract its visual feature f_v, and then ICS maps f_v into m resource IDs $\varphi_m=\{resID_1, resID_2, \ldots, resID_m\}$, where $resID_i=\mathfrak{I}(f_v)$, through m p-stable LSH tables.

After the resource IDs φ_m of an image is obtained, ICS constructs indexes in the form of $<resIDi, f_v, IP>$ where $i=1\ldots m$, IP is the IP address of the object owner. For each index, ICS sends an index message through the underlying Chord network, which forwards the message to the node responsible for the resID, as shown in Fig. 3. Once a node receives the index message from the DHT layer, ICS inserts this message in the index storage. The indexes with the same resID in the index storage are gathered into the same list to facilitate the localization of local indexes.

The number of hash tables, m, is a system parameter. For ICS, it also represents the number of indexes for an image and has an impact on the query efficiency and communication cost. As above discussed, more hash tables can provide better chance of finding the images that are similar to the query. However, more hash tables means more index messages to be published and requires more storage. So we should make a tradeoff between m and the query efficiency.

The number of buckets in each hash table, k, is another system parameter. It impacts not only the query efficiency but also the load of nodes. Fewer k means that more images are clustered to the same hash value, i.e., the same resID. This can lead to fewer clusters and accordingly each cluster has more images. Once a resID is located, more relevant images can be obtained. On the other side, the node in charge of the resID stores more indexes, if k is too small.

All the images of the local database in the node are reevaluated periodically, e.g., once a week or a month. If an

Author's personal copy

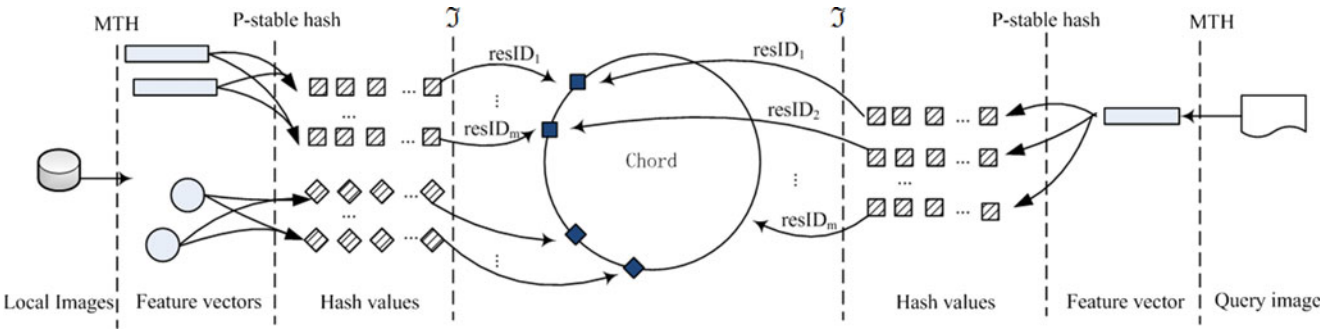

Fig. 3 Publishing the indexes and the query image

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 these two versions is less than the threshold, the indexes remain unchanged. Otherwise, the ICS is invoked to reconstruct the indexes.

4.4 Relevance feedback query processing

In this section, the query processing service (QPS) is discussed, supposing that all the image indexes are published into the DHT layer. When a node issues a query, QPS is invoked and the content-based similar images are retrieved. The objective of this service is to answer a query effectively. Search effectiveness is measured by the quality of search result, i.e., recall rate and precision. Furthermore, once the query iteration is completed, the user interacts with LRFIR to help improve the search result through the relevance feedback technique. After several iterations, the search accuracy can be significantly improved.

4.4.1 Query processing

When the node issues a query, QPS is invoked. It converts the query image to a set of resIDs, and then sends the query messages to the particular nodes. The query processing is similar to the index construction. As above described, the feature vector f_q of the query image is firstly extracted by the image feature extractor. Then f_q is transformed into a set of resource IDs $\varphi_m = \{resID_1, resID_2, \ldots, resID_m\}$, where $resID_i = \mathcal{J}(f_q)$, by $m \times k$ p-stable LSH function. Note that the set of hash functions used in QPS are the same as these in ICS. Afterwards, the node sends the query messages in the form of $<resID_i, fq, IP>$ where $i = 1 \ldots m$, and IP is the IP address of the query node. However, the query message is only forwarded to the nodes responsible for the resIDs. Therefore, if the images satisfy the requirement of the query, they are more likely to be retrieved due to the same resIDs. In this way, the query cost is controlled and the search efficiency is guaranteed.

Once receiving a query message from the DHT layer, the node checks the local index storage to find if there exists the same resID as it receives. To reduce the network transmission cost, it only returns the top T indexes sorted in terms of the distance from the query, defined in (4).

Similar to ICS, the number of query messages also depends on the number of hash tables m. if m increases, more query messages need to be published, which causes high query cost. But increasing m 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.

After the query node receives all the results, it merges the results before showing them to the user. The merging processing contains two steps. First, it eliminates the duplicates. If two indexes have the same f_v, we consider that they might represent the same image and one of them will be randomly selected. Second, all the results are sorted in terms of the distance defined in (4), and the top T indexes are chosen. Then connections are established between the query node and data owners, and images can be transmitted to the query node. Finally, the top T most similar images are showed to the user.

4.4.2 Relevance feedback

Relevance feedback technique, originated from the information retrieval, is employed in LRFIR to narrow the gap between low-level features and high-level ones. The features extracted from an image are low-level features, such as color, shape and texture, while high-level features are human perception of images, i.e., semantics. However, low-level features cannot fully represent the high-level semantic concepts. Moreover, there is no direct link between these two level features (Datta et al. 2008).

Relevance feedback intends to model the high-level image semantics through iterative feedback and query refinement. Humans are engaged in this alternative search process and help LRFIR to learn the semantics of query images. When one search iteration is completed, the user picks up relevant images and non-relevant images to update the previous query vector. In LRFIR, we choose Rocchio's formula (Zhou and

Author's personal copy

Huang 2003) shown in (9), where the image's semantics is captured by a set of weights. The latest query vector Q can be refined by assigning higher weights to the relevant terms and lower weights to the non-relevant terms. It provides a query point movement approach, which moves the query point towards positive samples and away from negative ones in the vector space.

$$Q' = \alpha Q + \beta \left(\frac{1}{N_{R'}} \sum_{i \in D'_R} D_i \right) - \gamma \left(\frac{1}{N_{N'}} \sum_{i \in D'_N} D_i \right) \quad (9)$$

Where $\alpha+\beta+\gamma=1$, α, β and γ are weights for the original query, relevant terms and non-relevant terms, respectively. D'_R and D'_N are denoted as the feature vector of relevant images and non-relevant images, respectively. N'_R and N'_N represent the number of images in D'_R and D'_N, respectively.

In LRFIR, relevance feedback is incorporated to further improve the retrieval performance. To simplify the procedure, non-relevant images are ignored. In each iteration, after the user marks a set of relevant images that are semantically close to the query, the query refinement is executed according to the feature vectors of relevant images. Based on the user's feedback, Eq. 9 is used to modify Q. After the new query vector Q' is generated, the QPS is invoked again. QPS produces the query messages for Q' as above described. Then the new query message is sent to the DHT layer.

In addition, although the resIDs of the new query Q' may be equal to these of the previous query Q, query messages containing the same resIDs are still sent. And the query processing is the same as above described. After the query vector is refined, the image feature vectors close to the new query vectors are retrieved. They are semantically closer to the query image.

In practice, several iterations of feedback are needed to improve the search accuracy. However, more iterations would bring about more network hops. In Chord, searching from the source node to the destination node needs $O(\log n)$ network hops in a network of n nodes. So in our relevance feedback search, the lookup requires $rO(\log n)$ hops in r iterations. If r is large enough, the accuracy of the query can be improved while the network cost is also increased. In contrast, if r is too small, the network cost is reduced while the accuracy might also be decreased. So we have to make a tradeoff between the network cost and the search accuracy.

The final results chosen by the user can be cached in local database for future query reuse. ICS can also construct indexes for these result images and publish them to Chord. When a query is issued, LRFIR first checks the caches. If there are no suitable answers in the caches, the QPS is invoked to publish query messages to Chord. On the other hand, the feedback process can be terminated by the user, when the results are not improved. In this way, the expensive cost of query processing can be saved.

5 Experiments and numerical results

We have implemented the proposed system and algorithms using Java 1.6. The simulation runs on a 2.83GHz Intel Core CPU with 2GB RAMs.

5.1 Datasets and system settings configuration

In our experiments, two image datasets are used: Corel 10000 and the subset of Catltech 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, 1,500 images are chosen from Catltech 101 Object Categories. We choose 30 objects of sunflower, dollar, headphone and faces, etc. And each object contains 50 images. In each category we randomly choose 5 images, so 500 queries are drawn from the Corel10000 and 150 queries from Catltech 101 Object Categories, respectively. For both datasets, image feature vectors are extracted using MTH, as described in Section 4.1.

Queries are initiated at randomly chosen peers, after all the peers join LRFIR. The reported results are the average values

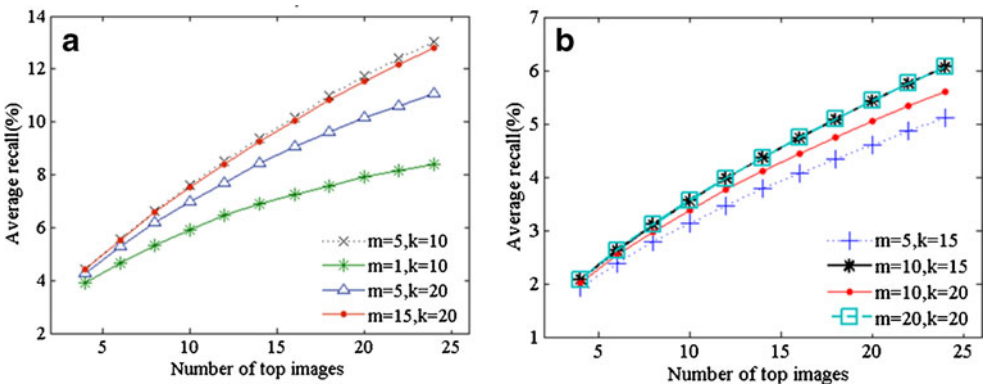

Fig. 4 Recall rates. **a** Catltech 101 Object. **b** Corel10000

Author's personal copy

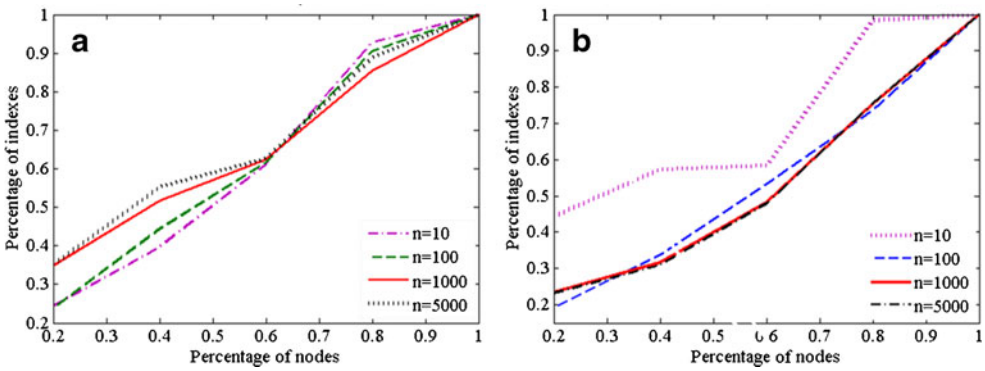

Fig. 5 Effect of load on the index distribution. **a** Catltech 101 Object. **b** Corel10000

over all the queries. In the experiments, we test with different values of system parameters. Unless otherwise noted, the default values are $W=2.0$ for p-stable hash function and $\alpha=0.8$, $\beta=0.2$,$\gamma=0.0$ for relevance feedback. The default network size is $n=1{,}000$. Besides, for the image retrieval, it is important to define suitable metrics for the performance evaluation. Two metrics are used: Recall rate and Precision (Liu et al. 2010). Recall rate is defined as the percentage of retrieved relevant images among all the relevant images in the dataset. Precision is defined as the percentage of relevant images among the retrieved images.

5.2 Recall rates

The corresponding recall rates are evaluated with different number of hash functions and top images, as shown in Fig. 4. The x-axis represents the number of top images, varying from 4 to 24 for both datasets. The y-axis denotes the recall rates measured under different number of top images. m and k respectively represents the number of hash tables and buckets. The recall rates increase as the number of top images increases for both datasets. But not many images are returned, because only the very similar images are needed to return. This facilitates convenience for people in browsing the results for the feedback. In LRFIR, 12 top images are defined as the default returned number.

The improvement in terms of recall rates is achieved by increasing the number of hash tables m or decreasing the number of buckets k, for both datasets. For Fig. 4b, $m=10$, $k=15$ and $m=20$, $k=20$ of LSH achieves the best recall rate with almost the same value. The reason is that the collision probability for content similar images is increased with the increase of m. On the other hand, decreasing k can lead to fewer clusters, accordingly more images are gathered into one cluster. Once a cluster is searched, many relevant results are returned. A similar observation can be made for Fig. 4a, where $m=5$, $k=10$ and $m=15$, $k=20$ achieves the best recall. We can choose $m=10$, $k=15$ for the Corel and $m=5$, $k=10$ for the Catltech 101 Object, since the computational overhead is reduced with the decrease of m. In the following experiments, unless specified otherwise, we choose $k=15$ for the Corel10000, and $k=10$ for the Catltech 101 Object.

5.3 Load balancing

In this section, the effectiveness of load balancing is investigated in two datasets. Figure 5 shows the index distributions for different cases. The x-axis values are the percentage of nodes, whose IDs are along Chord ring from small to large. And the number of nodes n varies from 10 to 5,000. The y-axis values are the percentage of indexes assigned to these nodes. Note that in both datasets, curves show much less skew as the number of nodes increases. That means that the load is more balanced with the increase of the number of nodes. This is because when the number of node increases, the interval between node IDs becomes smaller and more nodes are

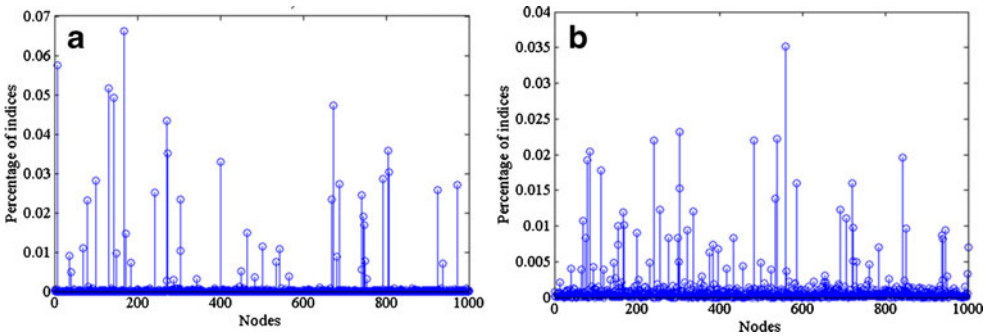

Fig. 6 The index distribution per node. **a** Catltech 101 Object. **b** Corel10000

Author's personal copy

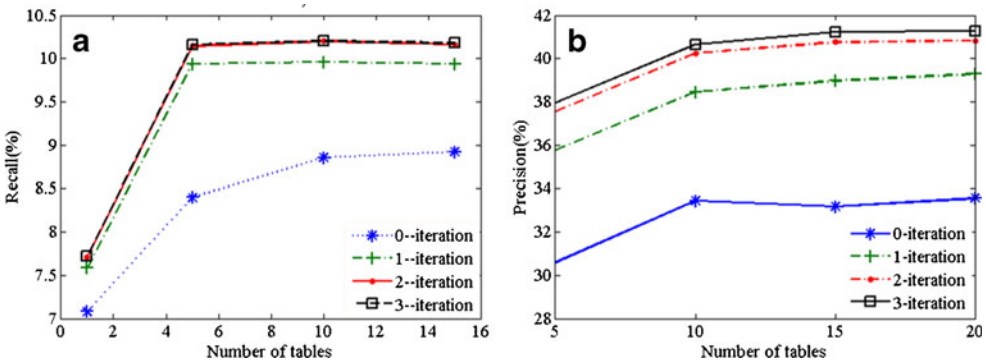

Fig. 7 (**a**) Recall vs. Feedback No. and (**b**) precision vs. Feedback No

assigned to store these indexes. Therefore, there are fewer indexes in each node. In Fig. 5, when the number of node is 1,000 and 5,000, the load is more balanced than other cases for both datasets. However, for the Corel10000, when n is 10, the load 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 node IDs becomes large. Therefore, some nodes store much more indexes than others.

When the number of nodes is 1,000, the percentage of indexes kept in each node is shown in Fig. 6. Obviously, some nodes store more indexes than others. That is the reason why the percentage of indexes in Fig. 5 does not grow very steadily as the percentage of nodes increases. The maximum percentage values for both datasets are 7% and 3.5%, respectively. Besides, the percentage of indexes kept in many nodes is very low or close to zero.

5.4 Relevance feedback results

The effect of relevance feedback on the search performance is measured, as shown in Fig. 7. The horizontal axis represents the number of hash tables, m, which varies from 1 to 15 in Fig. 7a and from 5 to 20 in Fig.7b, respectively. The vertical axis is the average recall rate and the average precisions measured for both datasets. In this experiment, 100 nodes are randomly selected to issue queries. And each node needs to issue all the queries. For both datasets, the recall rate and the precision grows slowly as m increases, regardless of the feedback iteration. The recall rate grows slightly when m is greater than 5 for the Catltech 101 Object. And the precision increases slightly as m is greater than 10 for the Corel. This is because when m is large enough, the images with their feature vectors similar to that of the query have already been gathered into the same cluster. As a result, simply increasing m cannot improve the recall and the precision.

Regarding the effect of relevance feedback, we take the Corel for an example. Catltech 101 Object has the similar conclusion. The retrieval rates can be improved by increasing the number of feedback iteration. "0-iteration" curve represents the average precision without any relevance feedback. "1-iteration" curve represents the average precision when one feedback iteration is applied to improve "0-iteration" result. "2-iteration" curve shows the average precision when another feedback iteration is applied to refine "1-iteration" result. And so on. Obviously, we can see the improvement of "3-iteration" in terms of precision compared with "0-iteration" for the Corel. However, in Catltech 101 Object, "3-iteration" has almost similar recall rate as "2-iteration". In the Corel, "3-iteration" is a little better than "2-iteration" in terms of precision. In short, we can see that the required number of feedback iteration is small.

The effect of relevance feedback on the number of lookup hops is depicted in Fig. 8. The number of lookup hops is one

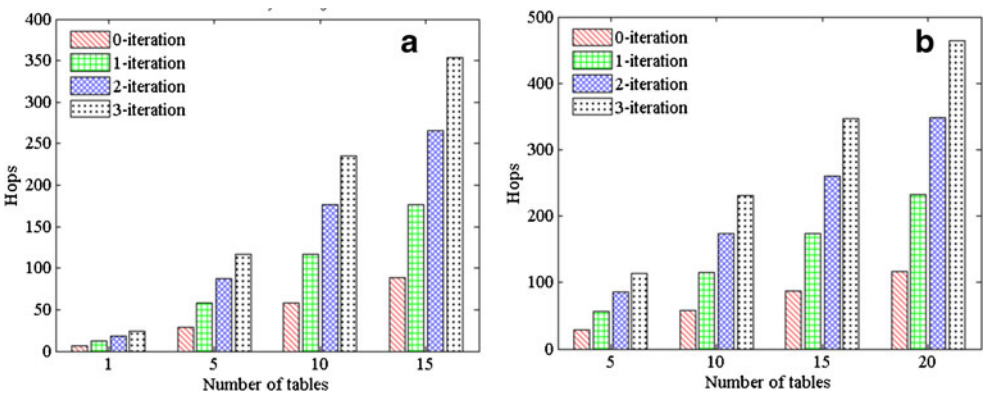

Fig. 8 Lookup hops vs. Iteration No. **a** Catltech 101 Object. **b** Corel10000

Author's personal copy

of the most critical performance parameters in the distributed environment. As shown in Fig. 8, the number of lookup hops mainly depends on the number of hash tables and the number of feedback iterations for both datasets. As we expect, the number of lookup hops increase when the number of hash tables increases. In addition, it increases fast as the number of feedback iteration increases. We can choose small m to reduce the number of hops in Chord as previously mentioned, i.e., $m=5$ for the Catltech 101 Object and $m=10$ for the Corel. Note that the number of lookup hops multiplies as the number of feedback iteration increases. For example, for the Corel dataset, when m is 10, the number of lookup hops for "0-iteration" is 50, that for "1-iteration" is about 100, that for "2-iteration" is almost 150, and that for "3-iteration" is close to 200. So we have to make a tradeoff between the search accuracy and the number of lookup hops. As shown in Fig. 7a "3-iteration" does not improve recall rate greatly compare with "2-iteration", but it causes more hops than "2-iteration". Therefore we can set the number of feedback iteration as 2. Due to the same reason, we choose "2-iteration" for the Catltech 101 Object.

We can see that $m=5$ and "2-iteration" achieves the best recall rate for the Catltech 101 Object. For the Corel dataset, $m=10$ and "2-iteration" achieves the best precision.

6 Conclusions and future work

We propose an effective framework to support CBIR in the distributed cloud datacenter. LRFIR supports both content-based similarity search and semantic search. The ICS constructs indexes based on 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. The QPS not only publishes the query message, but also employs the relevance feedback to refine the initially query vector. Therefore, the gap between low-level features and high-level features are overcome. The experiments show that our approach achieves high recall rate and good load balance, and it only needs a small number of network hops.

As for 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. Thirdly, we also plan to compare the performance of LRFIR against other existing systems.

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. 61372120,61271019, 61101119, 61121001, 61072057, 60902051); (3) PCSIRT (No. IRT1049); (4) MICINN (No. TIN2010-19077); (5) CAM (No.S2009TIC-1692).

References

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.

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.

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.

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.

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.

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.

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

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.

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.

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.

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

Gnutella. (2000). Gnutella website. http://www.Gnutella.com

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.

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.

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.

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.

Lee I., & Guan L. (2004). Semi-automated relevance feedback for distributed content based image retrieval. *The 2004 I.E. International Conference on Multimedia and Expo (ICME'04)*, Taipei, Taiwan, 1871-1874.

Li, F., Fergus, R., & Perona, P. (2007). Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. *Computer Vision and Image Understanding, 106*(1), 59–70.

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.

Author's personal copy

Liu, G., Zhang, L., Hon, Y., Li, Z., & Yang, J. (2010). Image retrieval based on multi-texton histogram. *Pattern Recognition, 43*(7), 2380–2389.

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.

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.

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.

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.

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.

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.

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.

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.

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.

Zhang L., Wang Z., & Feng D. (2010a). Efficient high-dimensional retrieval in structured P2P networks. *The 2010 I.E. International Conference on Multimedia and Expo (ICME'10)*, Singapore, 1439-1444.

Zhang X., Shou L., Tan K., & Chen G. (2010b). 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.

Zhou, X. S., & Huang, T. S. (2003). Relevance feed-back in image retrieval: A comprehensive review. *Multi-media Systems, 8*(6), 536–544.

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

Jianxin Liao is presently a professor of Beijing University of Posts and Telecommunications. He obtained his Ph.D. degree at University of Electronics Science and Technology of China in 1996. 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.

Di Yang is currently working toward the Ph.D. degree in computer science and technology at Beijing University of Posts and Telecommunications. Her research interests include network architecture, overlay network, multimedia communication, next-generation networks and information retrieval.

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

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

Qi Qi is an assistant professor in Beijing University of Posts and Telecommunications, China. She obtained her Ph.D. degree from Beijing University of Posts and Telecommunications in 2010. Now her research interests include performance evaluation for service network and future Internet, IP Multimedia Subsystem, ubiquitous services, QoS, and multimedia communication.

Xiaomin Zhu is an associate professor in Beijing University of Posts and Telecommunications. He obtained his Ph.D. degree from Beijing University of Posts and Telecommunications in 2001. Now his major is Telecommunications and Information Systems. His research interests span the area of intelligent networks and next-generation networks with a focus on 3G core network and protocol conversion.