Multiple Features For Image Retrieval In Distributed Datacenter

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

***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 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 the scalable image retrieval framework which can efficiently support content similarity search in the distributed environment. Its key idea is to integrate image fusion features into distributed hash tables (DHTs) by exploiting the property of the locality sensitive hashing (LSH). Thus, the images with similar content are most likely gathered into the same node without the knowledge of any global information. To the best of our knowledge, there is less comprehensive study on large-scale CBIR with fusion features in the distributed environment.**

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

I. INTRODUCTION

Nowadays, since users can access Internet services over lightweight portable devices and desktop PCs, millions of files are transferred into Peer-to-Peer (P2P) datacenter. It can be built by connecting many individual peers, without any central monitoring or coordination component. Each peer takes charge of part of data and replicas to improve the clouds' reliability and the datacenter's resilience to correlated failures. The P2P model show good performance of handling large dispersive data, and provide better scalability and adaptability [1]. It is well known that P2P techniques are very likely to be adopted in Clouds. The content providers invest their significant resources to P2P cloud datacenters. They use a decentralized P2P paradigm. When a cloud customer issues a query request, it is sent to the datacenter which takes charge of the search processing. Afterwards, the query results are sent back to the cloud customer.

Most current works about image retrieval in the P2P paradigm assume that images are described in text [2]. Since it is difficult to annotate images very exactly, identifying images in this way is inaccurate. 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 [2].The content based image retrieval (CBIR) can find similar images through the sample images instead of keywords. But many works for CBIR need the global information in a centralized fashion, which does not scale well in the distributed situation [3]. Therefore, our objective is to design a system which processes smart queries and improves search performance, without any global information. In our previous work, we present a CBIR system called LRFIR [4] based on a single feature for the distributed environment. It only adopts the texture feature, to represent images. However, such features can hardly describe the content of images at different angles.

In this paper, we present a novel CBIR system called LSH-based and Fusion Features for Image Retrieval (LFFIR) for the large scale P2P datacenter. LFFIR supports content similarity search, which is different from the keyword search used in existing systems. LFFIR firstly extracts the color, texture and shape feature. Then, the three types of vision features are integrated into a feature, where the content similarity can be measured quantitatively. Accordingly, we employ a set of locality sensitive hashing (LSH) functions [5], which convert feature vectors of similar images to the same hash value with high probability, owing to the locality-preserving property.

Our main contributions are as follows: (1) image multiple features are integrated into DHT for implementing an efficient indexing and locating approach; (2) we introduce the fusion features to represent the image content in the P2P retrieval model. In this way, the image can be described in different aspect.

II. RELATED WORK

Since unstructured P2P has little control over network topology, we focus on the information retrieval in the decentralized structured P2P paradigm. M-Chord [6] 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 [7], 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 computing still works in a centralized fashion. In Prism [8], it stores multiple indexes for one object in many Chord peers based on the distances between the object's vector and reference vector, so that indexes of similar objects are clustered to the same peer. But reference vectors are still chosen in a centralized fashion, which is not well-suited for large data-

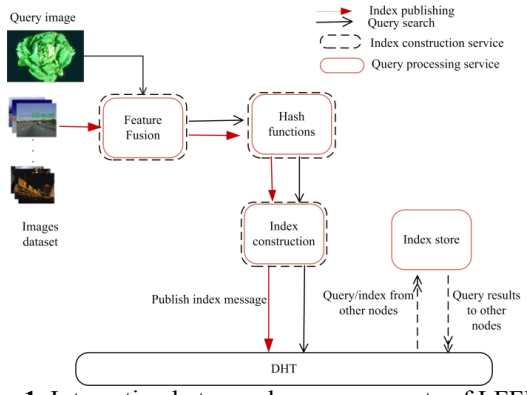

Fig. 1. Interaction between key components of LFFIR.

sets. Zhu et al. [9] generate the same index for semantically close files by LSH and Vector Space Model (VSM), with the purpose of answering queries visiting a small number of nodes. But these hash values are directly used as resource keys of Chord, which destroy the load balance. In iDISQUE [10] framework, the data on each peer is clustered, and then LSH functions only map cluster centers to Chord resource keys. The key of the cluster center represents the data within the cluster. However, the hash values of queries may not be equal to these of cluster centers.

III. SYSTEM FRAMEWORK

In this section, we present an overview of LFFIR framework. In the underlying DHT layer of LFFIR, the participating nodes are organized into the structured P2P network, Chord [11], without loss of generality. LFFIR can support efficient routing, due to the DHT layer. The indexes publishing and queries routing are automatically accomplished by Chord, which also natively offers node join and leave mechanisms.

For each image, LFFIR 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 the feature vectors whose dimensionalities are not very high are added to the indexes, each index will not cause significant storage overhead to the nodes.

The interactions among key components of LFFIR are illustrated by Figure 1. 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 others. In addition, each node also has the component of feature fusion which extracts of a set of features specific to different image formats. It accesses images in the local database, adopts various algorithms to compute image feature fused by color, shape and texture, and analyze the content of images. 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 resource IDs. In addition, the image indexes in local database are refreshed after a period of time to ensure the validity of resources.

The query processing service adopts the similar way to generate the resource *ID* and the query message. Then the query 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 *K* 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.

IV. DHT-BASED CBIR APPROACH

A. Fusion Feature Extraction

When a node wants to share an image, it automatically extracts feature vectors of the image. LFFIR adopts Motion Picture Experts Group-7 (MPEG-7) descriptors, which convert visual contents to measurable feature space.

For color, we use Grid Color Moment (GCM). Each image is partitioned into 3 × 3 grids and three types of color moments are extracted for each grid. For shape, 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. For texture, we adopt Gabor feature. Each image is scaled to 64 × 64. Gabor wavelet transformation is applied on the scaled image with 5 scale levels and 8 orientations, which results in 40 subimages.

LFFIR which relies on early fusion extract features before performing retrieval. After analysis of the three types of visual features, they are fused into a 238-dimensional feature represent each image in our image database.

B. LSH based Index Construction

After image features are extracted, the question is how to construct resource identifier for images and answer the query efficiently.

1) $R^d \rightarrow Z^k$.

In this work, we employ the family functions of *p*-stable LSH [12], which exists for $p \in (0,2]$. Since Euclidean distance is supposed to be the most widely-used distance metric, the Gaussian distribution working for the Euclidean distance is used as the 2-stable distribution. The hash function $h_{a,b}$ is defined as follow:

$$h_{a,b}(v) = \lfloor (a \cdot v + b) / W \rfloor \quad (1)$$

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 one integer.

In particular, to increase the collision probability, *m* hash tables $G=\{g_1,...,g_m\}$ are constructed, where *m* is randomly chosen. Each hash table is defined as *k* independent hash buckets $g(v) = (h_1(v),...,h_k(v))$, that means $G=\{g: R^d \rightarrow Z^k\}$ and returns a *k*-dimensional integer vector, i.e., the hash value. In this way, if two close feature vectors is hashed by more

hash tables, they may have the same hash value at least in one hash table g_i, where i=1,...,m.

2) $Z^k \rightarrow N$.

As a result, an integer vector Z^k is obtained from one hash table g_i, where i=1,...,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. On the other side, the load, defined as the number of indexes on the node, should be kept balanced as much as possible. To construct a resource *ID*, i.e., *fusID*, the mapping function $\Im(v)$ is defined as:

$fusID_j = \Im(\sum_{j=1}^{k} h_j(v) \cdot d_j)$, where $h \in g_i$ and i=1,...,m. (2)

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

3) Index Construction Service.

The purpose of this service is to generate the same *fusIDs* for the similar images with high probability, and then publish the indexes to special nodes through the DHT layer. Thus, the indexes of the similar images are randomly distributed across the DHT. However, we propose the Index Construction Service (ICS) which adopts p-stable LSH to preserve image similarity and distributes the indexes to the Chord as evenly as possible.

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* φ_m = {$fusID_1$, $fusID_2$,…,$fusID_m$}, where $fusID_i = \Im(f_v)$, through m p-stable LSH tables.

After the *fusIDs* φ_m of an image is obtained, ICS constructs indexes in the form of <$fusID_i$, f_v, *IP*> where i=1…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 *fusID*. Once a node receives the index message from the DHT layer, ICS inserts this message in the index storage. The indexes with the same *fusID* in the index storage are gathered into the same list to facilitate the location of local indexes.

C. Distributed Query Processing

1) Query processing

When the node issues a query, QPS is invoked. It converts the query image to a set of *fusIDs*, and then sends the query messages to the special nodes. The query processing is similar to the index construction. As above described, the feature vector f_q is transformed into a set of resource *IDs*= {$fusID_1$, $fusID_2$,…, $fusID_m$}, where $fusID_i = \Im(f_q)$ 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 in the form of <$fusID_i$, f_q, *IP*> where i=1…m, and *IP* is the *IP* 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*. 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 *fusID* as it receives. To reduce the network transmission

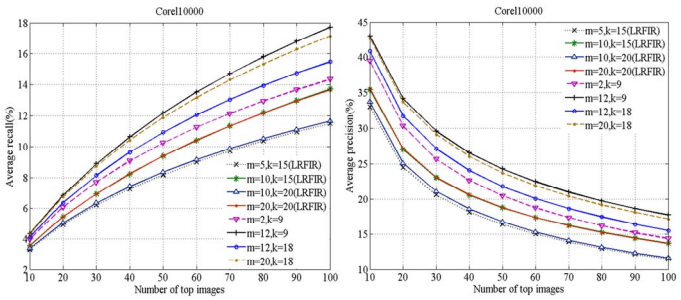

Fig. 2. Performance evaluation (a)Average recall. (b)Average precision

cost, it only returns the top K indexes sorted in terms of the Euclidean distance from the query. After the query node receives all the results, it merges them before showing them to users. The results of each query feature are obtained by sorting returned indexes according to the Euclidean distance, and the top K ones are chosen to form a result list. Then connections are established between the query node and data owners to transmit the final results to the query node. The top K most similar images are shown to users.

V. EXPERIMENTAL RESULTS

A. Dataset and System Settings

In our experiments, the image dataset Corel 10000 is used. It contains 10000 images of various contents, such as flowers, food, pills and door, etc. Unless otherwise noted, the default value is W=2.0 for p-stable hash function. The default network size N is 1000.

B. Query accuracy

The corresponding query accuracy is evaluated with the different number of hash functions and top images, as shown in Figure 2. We use two metrics and compare LFFIR with LRFIR [4]. The x-axis represents the number of top ranked images, varying from 10 to 100. The y-axis denotes the average recall and the average precision measured on the top ranked images. m and k represents the number of hash tables and buckets, respectively. The average recall and precision rapidly rise by increasing the number of hash tables m while decreasing the number of buckets k.

The reason is that the collision probability of content-based similar images is increased by increasing m. However, for Figure 2(a), both m=12, k=9 and m=20, k=18 of LSH achieve the best recall rate and they almost have the same value. That is because 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 Figure 2(b), where m=12, k=9 and m=20, k=18 also achieve best precision. We can choose m=12, k=9, since the computational overhead is reduced with the small number of hash function.

As revealed in these figures, LFFIR substantially outperforms LRFIR. For example, 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 Figure 2(a). With re-

spect to the average precision, LFFIR represents about 10% increase on the top 10 images in comparison with LRFIR, as shown in Figure 2(b).

C. Load Balancing

In this section, the effectiveness of the load balancing is investigated. Figure 3 shows the index distributions for different cases. The *x*-axis shows the percentage of nodes, where the number of nodes varies from 10 to 5000, and the corresponding values are the percentage of indexes assigned to nodes. Note LFFIR shows much less skew on load distribution as the number of nodes increases. That means the load balance is almost achieved. This is because when the number of node increases, the interval between nodes becomes smaller and more nodes are assigned to store these indexes. Therefore, there are not many indexes in each node. In addition, as k is large enough, the number of images in the cluster is reduced. In the above experiment, we reasonably choose k=9. Moreover, SHA-1 distributes the indexes to a large set of possible intervals of Chord. Compared to LRFIR, the curves of LFFIR appear to be much steadier, especially when n=1000 and n=5000. This indicates that the load balance of LFFIR is better than that of LRFIR.

D. Network Hops

The effect of network hops is depicted, as shown in Figure 4. 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,

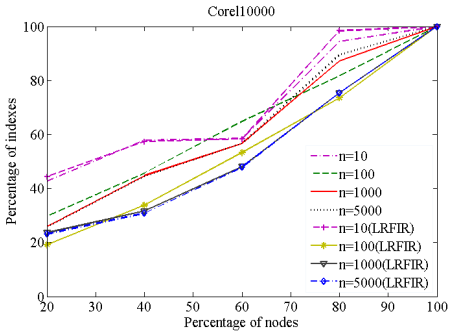

Fig. 3. Effect of load on the index distribution.

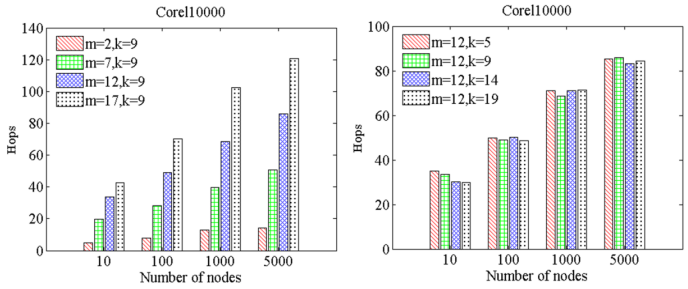

Fig. 4. Network hops (a) (b)

when m varies from 2 to 17 for Figure 4(a) and k varies from 5 to 19 for Figure 4(b). As shown, the number of lookup hops mainly depends on the number of tables and are independent on the number of buckets. As we expect, the lookup hops increase when the number of tables increases in Figure 4(a). For a large number of nodes, n=1000 and n=5000, the number of hops only has a slight increase. In Figure 4(b), especially for n=100 and n=1000, the number of hops almost remains unchanged while 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 can be chosen to guarantee the best query accuracy while not incurring too many network hops.

VI. CONCLUSIONS

We propose an effective framework to support CBIR in the distributed cloud datacenter. LFFIR is implemented on the DHT layer which provides efficient routing mechanisms.

ACKNOWLEDGMENT

This work was jointly supported by: (1) the National Basic Research Program of China (No. 2013CB329102); (2) National Natural Science Foundation of China (No. 61372120 , 61271019, 61101119, 61121001, 61072057, 60902051); (3) PCSIRT (No. IRT1049); (4) the Key (Keygrant) Project of Chinese Ministry of Education.(No. MCM20130310); (5) Beijing Higher Education Young Elite Teacher Project (No. YETP0473); (6) the Specialized Research Fund for the Doctoral Program of Higher Education (No. 20100005110008).

REFERENCES

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

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

[3] I. Lee, and L. Guan, "Semi-automated relevance feedback for distributed content based image retrieval," ICME, 2004.

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

[5] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," SoCG, 2004.

[6] D. Novak, and P. Zezula, "M-Chord: a scalable distributed similarity search structure," INFOSCALE, 2006.

[7] C. Tang, Z. Xu, and S. Dwarkadas, "Peer-to-peer information retrieval using self-organizing semantic overlay networks," SIGCOMM, 2003.

[8] O.D. Sahin, A. Gulbeden, F. Emekci, D. Agrawal, and A.E. Abbadi, "PRISM: Indexing multi-dimensional data in p2p networks using reference vectors," MM, 2005.

[9] Y. Zhu, and Y. Hu, "Efficient semantic search on DHT overlays," J. Parallel Distrib. Comput., 67(5), 604-616, 2007.

[10] X. Zhang, L. Shou, K. Tan, and G. Chen, "iDISQUE: Tuning High-Dimensional Similarity Queries in DHT Networks," DASFAA, 2010.

[11] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *SIGCOMM*, 2001.

[12] P. Haghani, S. Michel, and K. Aberer, "Distributed similarity search in high dimensions using locality sensitive hashing," EDBT, 2009.