DIRS: Distributed Image Retrieval System Based on MapReduce

Jing Zhang, Xianglong Liu, Junwu Luo, Bo Lang
State Key Laboratory of Software Development Environment, Beihang University, Beijing, P. R. China
{ katie618, xlliu, luojunwu, langbo}@ nlsde.buaa.edu.cn

Abstract

With information technology developing rapidly, variety and quantity of image data is increasing fast. How to retrieve desired images among massive images storage is getting to be an urgent problem. In this paper, we established a Distributed Image Retrieval System (DIRS), in which images are retrieved in a content-based way, and the retrieval among massive image data storage is speeded up by utilizing MapReduce distributed computing model. Moreover, fault tolerance, ability to run in a heterogeneous environment and scalability are supported in our system. Experiments are carried out to verify the improvement of performance when MapReduce model is utilized. Results have shown that image storage and image retrieval based on MapReduce outperform that in centralized way greatly when total number of images is large.

Keywords: Content-based Image Retrieval, Distributed Storage, MapReduce.

1. Introduction

Image is a kind of important information carrier. With the development of information and networking, the varieties and quantity of image are increasing fast. The number of images captured in 2006 exceeded 250 billion worldwide. By 2010, IDC is forecasting the capture of more than 500 billion images[1]. How to retrieve a desired image rapidly and accurately among massive image data storage is becoming an urgent problem.

Traditional image retrieval systems are mostly based on metadata such as keywords, tags, and/or descriptions associated with the image. These kinds of systems may produce lots of garbage in search results if metadata of images is not well filled. Also having humans manually enter keywords for images in a large database can be inefficient, expensive and may not capture every keyword that describes the image.

In the 90s of last century, Content-based image retrieval (CBIR)[2] was brought out. “Content-based” means that the search will analyze the actual contents of the image such as color, shape and texture rather than the metadata associated with the image. Both commercial companies and universities or academic institutions have shown great interest in CBIR since it was raised. There have been some recent CBIR systems released. For example, FIRE[3] developed at RWTH Aachen University, Content Based Image REtrieval System (CIRES)[4] built by The University of Texas at Austin, CORTINA[5] raised by Vision Research Lab in University of Santa Barbara, a reverse image search engine TinEye[6] released by Idée Incorporation, and BRISC[7] image retrieval system established by BRISC team in DePaul University. However, most CBIR systems concentrate on the content-based technologies such as visual features extracting technology and features matching technology. Also most of them are deployed on relational databases. When the amount of image data to be processed becomes larger and larger, to query desired images from relational database with millions of rows will meet serious efficiency problem, and probability of errors occurring is increasing. Moreover, relational database is not designed with very large scale and distribution originally, so scalability of these systems is also a great problem.

In order to retrieve desired images among massive image data efficiently and accurately, this paper established a Distributed Image Retrieval System (DIRS). DIRS applies CBIR to massive image datasets. It utilizes MapReduce[8] distributed computing model as computing layer to query image data in parallel to improve performance. Also DIRS adopts distributed database HBase[9] as the storage layer, which is column-oriented and suitable for large scale data storage. Because of the usage of MapReduce framework, the details of parallel tasks scheduling, fault tolerance, scalability are all handled transparently.

This paper begins with the desired properties and the architecture of DIRS. Section 2 describes distributed image storage further while Section 3 provides the details of parallel image retrieval. Performance testing experiments and results analysis are provided in Section 4. Section 5 gives the conclusion of the paper.

2. The Architecture of DIRS

2.1. Desired Properties

978-1-4244-9142-1/10/$26.00©2010 IEEE

In this section, we discuss the desired properties of a system to process image data at terabyte scale. The desired properties of DIRS are shown below.

1. Support parallel CBIR. To retrieve the most similar images of sample image, it is possible that a large dataset is required to be scanned thoroughly, which will bring much computing. Moreover, CBIR needs compare images' similarity in shape, color and texture and so on. Therefore, parallel retrieval should be taken to increase efficiency.

2. Distributed storage for massive image data. As the image data to be stored is at terabyte scale, distributed storage strategy should be adopted. What's more, distributed storage is a strong support for parallel retrieval.

3. Fault tolerance. The system will be deployed on the distributed clusters. Actually any of the nodes in cluster may fail when participating the cooperative computing. Thus, fault tolerance mechanism is required to deal with the failure and eliminate the bad effect made to computing result, to guarantee a computing task end correctly.

4. Ability to run in a heterogeneous environment. It is impossible to get homogeneous performance across each compute node, even if each node runs on identical hardware or on an identical virtual machine. Hardware failures or software failures may not cause complete node failure, but result in degraded performance. In one computing task, a "straggler" machine who takes long time to complete a task will lead to lengthen total time. A system designed to run in a heterogeneous environment must take appropriate measures to prevent this from occurring.

5. Support scalability. Scalability is an advantage of distributed cluster. But applications deployed on it are influenced or even damaged when adding or removing a node. In that case the scalability turns out to be a failure. Nodes expanding should be seamlessly.

2.2. Desired Properties

For the desired properties, DIRS mainly makes use of Hadoop[10] distributed computing framework. Hadoop is an open source distributed computing framework, which consists of two layers: (i) a data storage layer or the Hadoop Distributed File System (HDFS)[11], which is similar to Google File System(GFS)[12], and (ii) a data processing layer or the MapReduce Framework. And Bigtable-like[13] distributed database HBase is a subproject of Hadoop and is built on top of Hadoop.

DIRS is deployed on the Hadoop distributed computing environment to processes massive image data, and possesses all the desired properties proposed above.

DIRS adopts methods provided by Lucene Image REtrieval（LIRE）[14] to extract visual features of images for storage and match features between different images for retrieval. LIRE is a light weight open source Java library for content based image retrieval. It is intended for developers and researchers who want to integrate CBIR in their applications. It implements several visual features extracting algorithms based on MPEG-7, such as ScalableColor, ColorLayout, EdgeHistogram and so on.

For parallel image retrieval, DIRS retrieves image among massive image data by utilizing MapReduce distributed computing model. MapReduce distributed computing model abstracts the massive data processing to two functions, map and reduce. MapReduce framework divides the input files into N splits, each split is passed to a map task and N Map tasks are processed parallelly by each node in the cluster without communicating with each other. The output of map tasks is repartitioned across all nodes of the cluster. Finally, a set of Reduce tasks are executed parallelly by each node on the partition it receives. The mechanism of MapReduce framework enables high parallel computing and the reliability of data.

DIRS takes HBase as the database to store massive image data. HBase is a distributed database and suitable for random, real time read/write access to big data. It is built upon Hadoop and greatly integrated with Hadoop, so HBase can be data source and sink of MapReduce job seamlessly, which is a strong support for parallel image retrieval.

Hadoop can make applications deployed on a large cluster of commodity machines. Moreover, Hadoop gives the solutions for fault tolerance, running in a heterogeneous environment and scalability. By detecting failed tasks and assigning these tasks to other nodes in cluster to re-execute, fault tolerance is achieved. The ability to run in heterogeneous environment is achieved by backup tasks. The scalability of Hadoop relies on the scalability of HDFS, in which data is spread on the new-added node seamlessly.

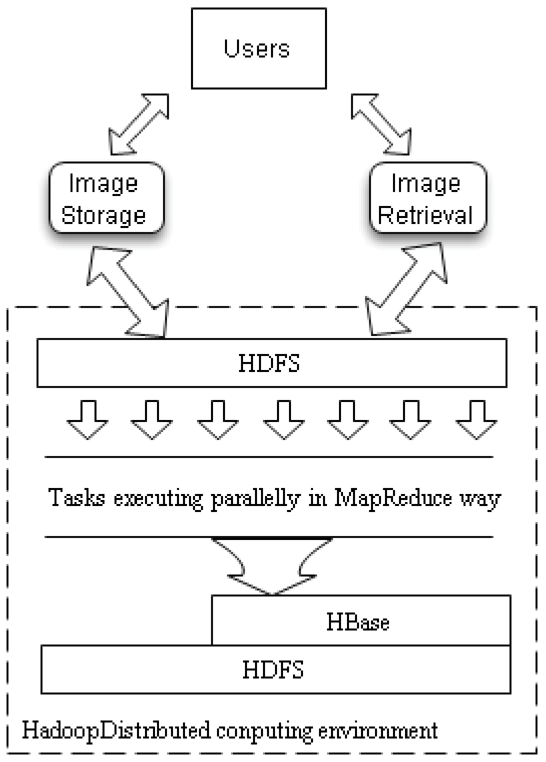
Figure 1 shows the architecture of DIRS. It provides a uniform interface to receive users' request. After processing, results are delivered to users. How images are stored and retrieved is transparent to users. This is a kind of loose coupling design to make system maintenance and extensibility easier.

Core of the DIRS is the image processing module deployed on Hadoop cluster, which follows the MapReduce paradigm. Read input data from HDFS, process after the map and reduce functions, and then write output results to HDFS or HBase.

3. Distributed Storage for Images and Features

3.1. Content-based Visual Features Extracting

The image features used in the DIRS system are extracted by using LIRE library. The following 9 image



Users
Image Storage
Image Retrieval
HDFS
Tasks executing parallelly in MapReduce way
HBase
HDFS
HadoopDistributed computing environment

Figure1. Architecture of DIRS

features are included in LIRE: Color Histograms in RGB and HSV color space, MPEG-7 descriptors scalable color, color layout and edge histogram, Tamura texture features coarseness, contrast and directionality, Color and Edge Directivity Descriptor (CEDD) , Fuzzy Color and Texture Histogram (FCTH), Auto Color Correlation, Scale-invariant feature transform (SIFT)[14].

The interfaces of feature extraction methods mentioned above are similar by taking image file as input and feature vectors as output, but varying in the processing and algorithm (see[14]). Most of these methods extract mainly for color or texture features, some integrate the two visual characteristics, such as CEDD and FCTH. The fact remains that it is more feasible and effective to combine features to describe the image content than to use single one.

3.2. The Design of HBase Table Structure

In DIRS, massive image data is stored in HBase.

Data is logically organized as table, row, family and column in HBase. HBase stores data rows in labeled tables. A data row has a sortable row key and an arbitrary number of columns. The table is stored sparsely, so that rows in the same table can have widely varying numbers of columns.

A column name has the form "<family>:<label>" where <family> and <label> can be arbitrary byte arrays. Sets of <familiy>s, or “column families” are fixed since table is created. Adjusting the sets of families is done by performing administrative operations on the table. However, new <label>s can be added to <familiy>s without pre-announcing it [9].

As shown in Table 1, row key of our HBase table is assigned to image’s ID, and families are “files” and “features”. Label “source” and “thumbnail” are added under family “file”, representing for source image file and thumbnail respectively. Under family “features”, label “descriptorScalableColor”, ”descriptorColorLayout”, ”descriptorEdgeHistogram”, ”featureTAMURA”, ”featureCEDD”, ”featureFCTH”, ”featureAutoColorCorrelogram”, ”featureColorHistogram” are added, representing features extracting by using LIRE library. In HBase, only a single row at a time may be locked by default, and row writes are always atomic, so our design enables the whole information of each image to be stored in each row, easy to read and write.

Table 1. Design of table structure for image storage

ID	file		features		
	file: source	file:	features: feature CEDD	features:	features: feature FCTH
……	……	…...	......	......	......

3.3. Image Storage Based on MapReduce

Storage is the base of retrieval. What DIRS deals with is massive images, and together with their visual features. If processed in centralized way, the more images to store, the more time to be consumed. Therefore, parallel processing is necessary. DIRS adopts MapReduce computing model to extract the visual features of images and then write the features and image files all into HBase parallelly.

The implementation method of distributed image storage is shown in Figure 2.

A MapReduce job is constructed to store images parallely. Before the job starts, upload the raw images to HDFS.

- Input: HDFS directory containing all raw image files.
- Map: Extract visual features, generate thumbnails for each image, and then write “ID, image, thumbnail, all features” as a row into HBase. ID is the unique key to identify each image. If one image is not properly parsed by LIRE and unable to be stored, emit its ID, called ineffective IDs, as the output of map tasks.
- Shuffle and Sort: All <k, v> pairs will exchange among nodes to send all values with

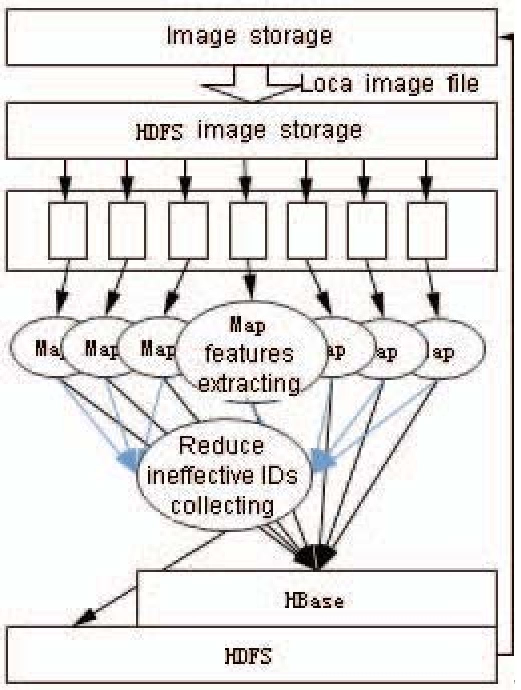


Figure 2. **Flows of image storage**

same key to a single reducer.

- Reduce: Usually there are just a few ineffective IDs, so just one reducer is used. It will take all inefficient IDs as input, collect them together, and output to HDFS.
- Output: All ineffective IDs.

Read those inefficient IDs out, return to callers to inform that those IDs are not properly stored.

4. Parallel Image Retrieval

4.1. Content-Based Image Features Matching

In the feature matching process, CBIR system calculates the similarity between the sample image and the target images, and then returns those images matching the sample image most closely. Distance and angle are two similarity measures used widely. Distance measure is based on the distance of two feature vector, such as Euclidean distance and Mahalanobi distance. Angle measure is usually defined as the cosine of two feature vector.

LIRE mainly used Euclidean distance and Tanimoto coefficient as similarity measures, the definitions of them are as follows.

$$Euclidean(A_i, A_j) = \sqrt{\sum_{m=2}^{r} (A_{im} - A_{jm})^2} \qquad (1)$$

Tanimoto coefficient：

$$T(A_i, A_j) = \frac{\sum_{m=1}^{r} A_{im} A_{jm}}{\sum_{m=1}^{r} A_{im} A_{im} + \sum_{m=1}^{r} A_{jm} A_{jm} - \sum_{m=1}^{r} A_{im} A_{jm}} \qquad (2)$$

Where A_i and A_j are the feature vector of two images, *r* is the dimension of vector.

4.2. Image Retrieval in Parallel

Because the target database contains large datasets, it will spend long time on whole-table search (See experiments of centralized retrieval in section 5). Therefore MapReduce computing model is taken to execute map tasks and reduce tasks parallelly to reduce running time and increase efficiency.

The implementation method of parallel retrieval is shown in Figure 3.

MapReduce job is built to retrieve images in parallel.

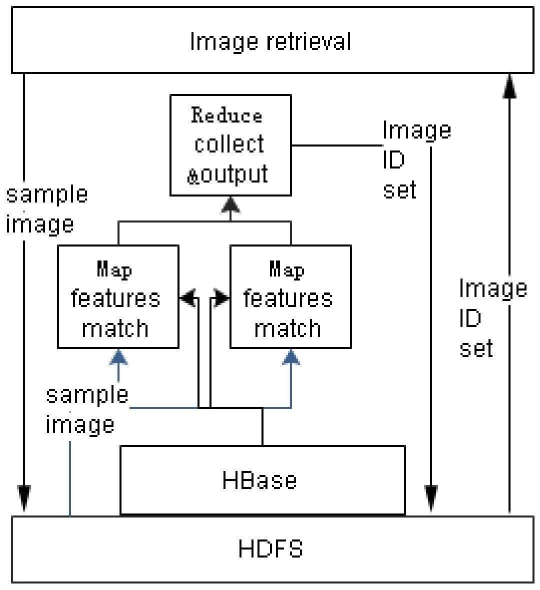


Figure 3. **Flows of parallel image retrieval**

Before MapReduce job is started, the sample image should be added to Distributed Cache to enable every map task to access it.

- Input: Target HBase table.
- Map: Read sample image from Distributed Cache, extract its visual features, and then compare the feature vectors with feature vectors of target images from HBase, finally score the target images. <score, ID> pairs are emitted as the output of map tasks.
- Shuffle and Sort: The entire <score, ID> pairs are sorted by score and repartitioned, then go to one reducer.
- Reduce: Collected all <score, ID> pairs and write image IDs of the top N <score, ID> pairs to HDFS.
- Output: The IDs of those images most similar to the sample image.

5. Experiments and Evaluation

5.1. Cluster Configuration

The Hadoop cluster in which DIRS is deployed on consists of 9 nodes. Configuration of each node and the role it plays in the cluster is shown in Table 2. Here, M stands for Master in which NameNode, JobTracker and HMaster are running; while S stands for Slave in which DataNode, TaskTracker and RegionServer are running. ZK stands for ZooKeeper.

Table 2. **Configuration of Cluster**

Nodes	Memory	CPU	Disk	Role		
	(G)	(G)	(G)	M	S	ZK
Node0	4	2*3	500	Y		Y
Node1	4	2*3	500		Y	Y
Node2	4	2*3	500		Y	Y
Node3	3	2*2.9	285		Y	
Node4	4	2*3	500		Y	
Node5	3	2*2.9	450		Y	
Node6	4	2*2.9	284		Y	Y
Node7	3	2*2.9	285		Y	Y
Node8	4	2*3	500		Y	

5.2. Testing Results and Their Analysis

5.2.1. Performance Testing on Image Storage. Performance testing on distributed image storage based on MapReduce is compared with that in centralized way. Experiments are carried out to store 100, 500, 1000, 2000, 5000, 10000, 20000 images[15] in both ways respectively.

Figure 4 shows the total time consumed to store *x* images where horizontal axis stands for total number of images *x*, and vertical axis stands for total time consumed. When number of images is small (100<=*x*<=500), time consumed to store images based on MapReduce model and in centralized way is close. Because MapReduce framework should initialize the job, and exchange data among clusters after map phase, so time to process images takes smaller percentage in total time. The advantage of MapReduce in processing large data is not obvious when quantity of data is small. When the number of images is getting lager (500<=*x*<=5000), time consumed to store images based on MapReduce model is less and less than that in centralized way. That is because image processing time takes more percentage in the total time with the quantity of images increasing. When quantity of images is even larger (5000<=*x*<=20000), time gap between MapReduce adopted and centralized way is obvious. Image storage in centralized way tends to grow exponentially. But Image storage based on MapReduce tends to grow slowly.

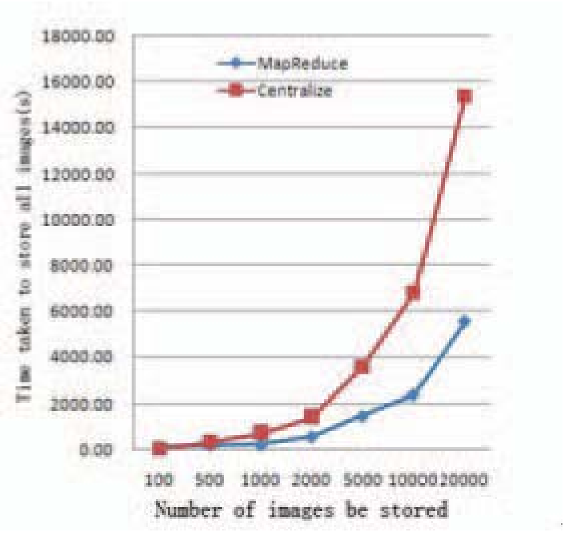


Figure 4. **Comparison of total time consumed to store images in two ways**

Figure 5 is the comparison diagram of average time consumed to store one image based on MapReduce model and in centralized way, in which horizontal axis stands for total amount of images *x*, vertical axis stands for average time consumed, red curve stands for image storage in centralized way and blue curve stands for image storage based on MapReduce model. It is obvious in this diagram that to store one image in centralized way consumes less time when image quantity is small. Thus centralized way has an advantage over MapReduce model at the beginning. But the red curve is ascending while the blue curve is descending with the total quantity increasing. When the total image quantity increases to one point (two curves intersect when *x* is about 400), time consumed to store one image based on MapReduce model and in centralized way is almost equal. After this point, the gap between two curves is enlarging, implying that time consumed when MapReduce model is adopted is much less than in centralized way. This verifies the conclusion that performance of data processing based on MapReduce model is significant when image data is large.

5.2.2. Performance Testing on Image Retrieval. Performance testing of parallel image retrieval is also compared with that in centralized way. We made experiments on retrieving 100 images which are most similar to the sample image in content among different scale of HBase tables.

The comparison diagram of time consumed to retrieve images in parallel and in centralized way among HBase tables containing *y* rows is shown in Figure 6. When scale of table is small (800<*y*<10000), image retrieval in centralized way will take less time. Image retrieval only needs to extract the features of sample image and compare with those features in HBase. Therefore when scale of HBase table is small, there's not so much computing. When scale of table is

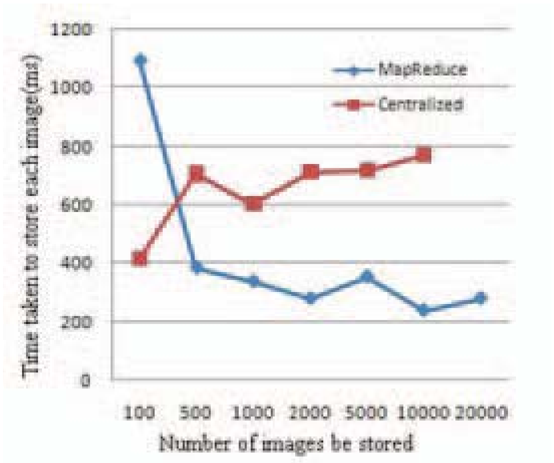


Figure 5. **Comparison of average time consumed to store one image in two ways**

larger (10000<*y*<20000), retrieval in parallel and in centralized way will have similar processing capability. When the scale of table is even larger (*y*>20000), time consumed to retrieve images in parallel tends to grow slowly while image retrieval in centralized way tends to grow exponentially. Time gap between them is getting lager. Therefore, retrieving images from large scale table based on MapReduce will increase efficiency greatly.

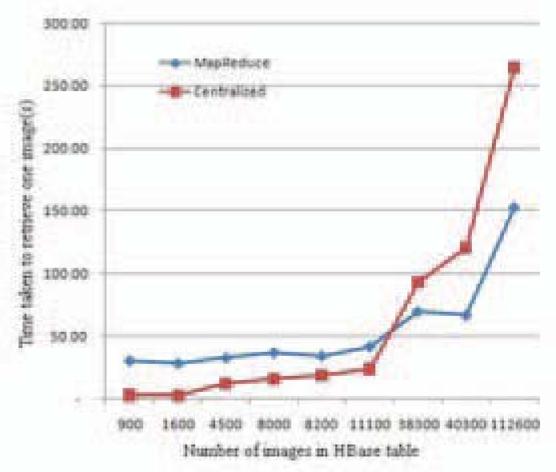


Figure 6. **Comparison of time consumed to retrieve image in two ways**

6. Conclusion

This paper provides a method to apply CBIR to massive image datasets and its implementation system DIRS. DIRS is deployed on Hadoop distributed computing environment, stores images and their features in column-oriented database HBase, and utilizes MapReduce computing model to improve the performance of image retrieval among massive image data. The desired properties such as fault tolerance, ability to run in a heterogeneous environment and scalability are all achieved in DIRS. Moreover, compared with centralized way, image storage and image retrieval based on MapReduce distributed computing model are more efficient when target image data is large. The time gap between storing images to HBase based on MapReduce model and in centralized way is big when total image number is large, same as retrieving images in two ways, implying images processing based on MapReduce model outperforms centralized way greatly. Therefore, the advantage of MapReduce model in processing large data is reflected.

Acknowledgement

This work was supported by a National major project "Advanced Unstructured Data Repository" (Grant No.2010ZX01042-002-001-00) and the Foundation of the State Key Laboratory of Software Development Environment (Grant No.SKLSDE-2009ZX-06).

References

[1]John F. Gantz. The Expanding Digital Universe. *IDC*, 2007.
[2]Hirata K., Kato T.. Query by Visual Example – Content-Based Image Retrieval[C]. *EDBT*, 1992.
[3]Thomas Deselaers, Daniel Keysers, Hermann Ney. FIRE - Flexible Image Retrieval Engine: ImageCLEF 2004 Evaluation[C]. *CLEF 2004*, LNCS 3491, pp. 688–698, 2005.
[4]Qasim Iqbal, J. K. Aggarwal. Feature Integration, Multi-image Queries and Relevance Feedback in Image Retrieval[C]. *VISUAL*, 2003.
[5]Elisa Drelie Gelasca, Pratim Ghosh, Emily Moxley, Joriz De Guzman, JieJun Xu, Zhiqiang Bi, Steffen Gauglitz, Amir M. Rahimi, B. S. Manjunath. CORTINA: Searching a 10 Million + Images Database[C]. *VLDB*, 2007.
[6]http://www.tineye.com/.
[7]Michael O. Lam, Tim Disney, Daniela S. Raicu, Jacob Furst, and David S. Channin. BRISC: An Open Source Pulmonary Nodule Image Retrieval Framework.Journal of Digital Imaging[J]. Vol 20, Supplement 1: 63-71, 2007.
[8]Jeffrey Dean, Sanjay Ghemawat. Map Reduce: Simplified Data Processing on Large Cluster[C]. *OSDI*, 2004.
[9]http://hbase.apache.org/.
[10]http://hadoop.apache.org/core/.
[11]http://hadoop.apache.org/hdfs/.
[12]Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System[C]. *SOSP*,2003.
[13]Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data[C]. *OSDI*, 2006.
[14]Mathias Lux, Savvas A. Chatzichristofis. LIRe: Lucene Image Retrieval - An Extensible Java CBIR Library[C]. *ACM, 2008*.
[15]J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database. *IEEE Computer Vision and Pattern Recognition (CVPR), 2009*.