An Improved Content Based Image Retrieval System On Apache Spark

Yixiao Duan[1], Linhua Jiang[1,2,*], Xiao Lin[1] and Xiaodong Chen[2]
[1] Shanghai Key Lab of Modern Optical Systems, University of Shanghai for Science and Technology, Shanghai 200093, P.R.China
[2]Information Science and Technology Research, Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai 201210, P.R.China
[*]Corresponding author

***Abstract*—Traditional CBIR (Content Based Image Retrieval) system on parallel platform spends too much time facing large-scale images. In this paper, we present an improved CBIR system based on Apache Spark, a fast and general large-scale data processing engine. At first, we do preliminary processing to build system environment and extract feature vectors. Then, in order to solve the problem of huge numbers of small files, we construct image database by using Apache Avro to combine these small files and improve structure of the common Avro file and so making it more suitable for our system. Finally, at the matching stage, we make the Avro file to be persisted in memory and reused conveniently. Multiple experimental results show our approach performs well in terms of both speed and accuracy rate.**

Keywords-image retrieval; CBIR system; apache spark; apache avro

I. Introduction

With the booming of Internet and social media, huge image data are generated, while it becomes a serious problem that how we can efficiently retrieve similar image data by a search engine. Traditional searching method is TBIR (Text-based Image Retrieval) [1], which describes an image with text like author and size, etc. Advanced method is CBIR (Content-based Image Retrieval) [2], which is more objective compared to TBIR, it extracts color, texture, and shape properties as feature vectors to describe an image. The advantage of this approach is reducing human intervention and making use of the high performance of computer processors. Although CBIR has major advantages over TBIR, it is still time-consuming facing the mass image data. Now distributed processing system is available for mass data storage and processing. Apache Hadoop [3] and Apache Spark [4], as stable and mature platforms, can effectively reduce time cost when process huge data. However, the file system of these methods still has some defects. Therefore, we propose an improved CBIR system to reduce the time and resources cost. This paper is organized as follows: Section II presents related background; Section III we propose the improved CBIR system based on Spark; Section IV shows the result and comparison; Section V makes a conclusion.

II. Related Background

A. CBIR

CBIR is superior to the traditional image retrieve method, using feature vectors to retrieve similar images. The feature vector is of extracting information of color, faces, texture, shape, etc. from an image. CBIR can be sorted into three layers [5]:

- Retrieving a picture through feature vectors.
- Retrieving by recognizing object classes and spatial topological relation base on low-level features of images.
- Retrieving through abstract attribute such as behavior semantic, emotion semantic and scene semantic.

As the limitation of computer vision and image recognition, conventional CBIR can't support image retrieval based on semantic which is so called semantic gaps. So far we still use feature vectors to search similar images [6]. And this approach can be described as two steps:

- Feature Extraction. The extraction of image characteristics and expression is the foundation of the content-based image retrieval technology. Color, texture, shape and spatial relations are commonly used image characteristics [7].
- Feature Matching. Feature extracted from the image characteristics can be composed of a vector, then define a distance or similarity measure to calculate the similarity degree and set the threshold to determine whether two images are similar.

B. Spark

Apache Spark is an over speed, ease of use and complex analysis framework of large data processing. Compared with Hadoop and Storm technology, Spark has obvious advantages [8]. Spark provides us with a comprehensive, unified framework for managing a variety of different properties (such as text data, charts, data) data sets and the data source (batch data or real-time streaming data) of large data processing requirements. It keeps intermediate results in the memory so that it is no longer needed to read and write on HDFS (Hadoop Distributed File System) [9], therefore it can do better performance on data mining and machine learning which need iterative calculation.

Resilient Distributed Dataset (RDD) is the most basic abstraction of Spark [10]. As the core service, it provides a way to operate distributed data sets just like operating local collection. Operations on RDD are divided into two types: transformation and action. Transformation will generate a new RDD, and built dependencies between old and new one. Transformation operation is Lazy mode. All operations will not execute until meet an action operation. Then action will return a value instead of RDD or save RDD on filesystem. Once meet action operation, a DAG (Directed Acyclic Graph) which is formed through dependencies of RDD will be delivered to DAGScheduler, meanwhile a Job will be submitted. DAGScheduler walks through the complete chain of linked dependencies [11]. According to the types, the Job is divided into different stages. Then DAGScheduler produces a series of task in each stage and encapsulate them into the TaskSet, which will be sent to TaskScheduler later. Finally, TaskScheduler submits the TaskSet to cluster to execute and then report results.

C. Avro

Apache Avro is a data serialization system, designed to support the application of large quantities of data exchange [12]. It supports binary serialization forms and can be convenient to quickly process large amounts of data. Avro depends on Scheme to implement the data structure definition. Scheme can be regard as a Java class, it defines structure of each instance, what attributes are involved in. When Avro data is stored in a file, its schema is stored with it, so that files may be processed later by any program. Avro schemas are defined with JSON. Eight primitive types and six complete types are supported and user can create complex data structures.

III. System Design

A. Preprocessing

Apache Spark is a general large-scale data processing engine, so we still need a distributed file system. As HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. And Spark can access diverse data sources including HDFS [13]. So we choose it for storing image data.

1) Hadoop cluster setup

a) The software should be unpacked on all machines in the cluster to intall a Hadoop cluster. One machine should be designated as Namenode, which as the role of master. The rest of the machines are DataNode, called slaves.

b) After software installation, we should configure environment. All before, JAVA_HOME should be correctly defined on each node.

c) In order to avoidentering password, we setup passphreseless ssh. Last step is modifying core-site.xml, mapred-site.xml and hdfs-site.xml on both master and slaves node.

2) Spark cluster setup

a) Instal scala and spark on all nodes. Modify spark-env.sh, add path of scala and spark, then modify slaves file to add all hostname of slave nodes.

b) On the basis of the Hadoop cluster starts successfully, input start-all.sh to start Spark cluster. Enter http://Master:8080 to check Spark web-ui.

B. Extracting Feature Vectors

The core algorithm of this step is partial feature detection algorithm. SIFT (Scale-invariant Feature Transform) algorithm extracts the feature of images by calculating interest points and descriptions of the scale and orientation and it has been proved quite effective [14]. Inspired by SFIT descriptor, SURF (Speeded Up Robust Feature) is based on integral image, using determination of Hessian matrix to describe extreme point. With different size of box filter and the integral image for convolution and it's convenience for parallel operation. In general, the SURF operator is several times faster than SIFT operator and has better robustness in the case of massive pictures [15].

TABLE I. Comparison Between Sift and Surf

Method	**Time**	**Scale**	**Rotation**	**Blur**	**Affine**
SIFT	Common	Best	Best	Common	Good
SURF	Best	Common	Common	Best	Good

Our system adapted SURF to extract feature vectors. Specific steps are as follows:

- Build the Hessian matrix.
- Build scale space.
- Precisely locate interest points.
- Select main direction of interest points.
- Generate character descriptions.

C. Constructing Image Atabase

According to SURF algorithm, every image will generate any number of interest points. We define the interest point as a class and its members include coordinates, scale, orientation, laplacian operator, descriptors and index [16]. We define another class which includes filename and collection of interest points as domains. Therefore, the second class can represent an image. Then this class can be serialized and stored as a file [17]. However, in the actual situation, this file is too small and it causes serial issues. At first, in HDFS, any file, directory or block are re-presented as an object and stored in namenode's memory. Every object occupies 150 bytes. It will beyond the storage capacity of memory if there exist a large number of files. Second, to access these small files, it is necessary to constantly transfer from one datanode to another one so that seriously affect the performance. Third, system needs to generate lots of task to deal with small files that means starting and releasing task will spend a lot of time.

To solve this problem, we import Avro to combine small files. Compared with other methods like SequeceFile [18], Avro can automatically deal with filed additions and removal, forward and backward compatibility, branched versioning, and re-naming, all largely without any awareness by an application. Besides, it has a good compression ratio. To use Avro, we should first define a schema, the schema of interest point is as follow:

```
    {
      "namespace": "example.avro",
      "type": "record",
      "name": "SURFInterestPoint",
      "fields": [
        {  "name": "mX",
          "type": "float"  },
        {  "name": "mY",
          "type": "float"  },
            {  "name": "mScale",
          "type": "float"  },
            {  "name": "mOrientation",
          "type": "float"  },
            {  "name": "mLaplacian",
          "type": "int"    },
            {  "name": "mDescriptor",
          "type":
[{"type":"array","items":"float"},"null"]   },
            {  "name": "mDx",
          "type": "float"  },
            {  "name": "mDy",
          "type": "float"  },
           {  "name": "mClusterIndex",
          "type": "int"    } ]
    }
```

FIGURE I. DESING OF AVRO SCHEME

According to the schema, we can generate a class which represents an interest point. The same procedure can be applied to the class which represents an image. We can combine any number of image classes and serialize them to store as an Avro file. Compared to the one class, one file used before, Avro file collects small files together into a big one. In order to make the Avro file according with our CBIR system, we optimize structure of Avro file.

An Avro file consists of the header and data block. Data block can store serialized objects and a sync marker. Meanwhile, Avro supplies an API to set sync maker when storing objects. It forces the end of current block, emitting a synchronization marker. Therefore, we can store just an object in a block, and random read the object just like operating an array via using sync marker from an Avro file.

In our system, we rename image files by the serial number from 1 to n, convert image files to classes by running the SURF algorithm to extract feature vectors. Then we append every 100 classes into an Avro file. Finally, all avro files will be uploaded to HDFS.

D. Matching

Upload an image to the specified path on HDFS, use SURF to generate a corresponding class. Call API of Spark to read all Avro files and create distributed datasets. Each element of RDD is an Avro file. We can use a method to get position by the sync marker and get the class which can represent an image. Then match the collection of interest points from the class with the interest points from the new uploaded image. If the number of matched interest points beyond threshold, we can say two images are match and save the name of original image.

IV. EXPERIMENT AND RESULT

The Spark cluster includes a master node, three data slave nodes, is running at standalone mode. Distributions of all nodes are as follows:

TABLE II. IP ADDRESS AND ROLE OF NODES IN SPARK CLUSTER

Server Name	IP address	Role
Mater	192.168.1.112	Master
Slave	192.168.1.113	Worker
Slave	192.168.1.114	Worker
Slave	192.168.1.115	Worker

TABLE III. HARDWARE INFORMATION OF NODES IN CLUSTER

Server Name	CPU	Memory
Master	Xeon E5-2609 v3	32G
Slave	Core i7-4790	8G
Slave	Core i7-4790	8G
Slave	Core i7-4790	8G

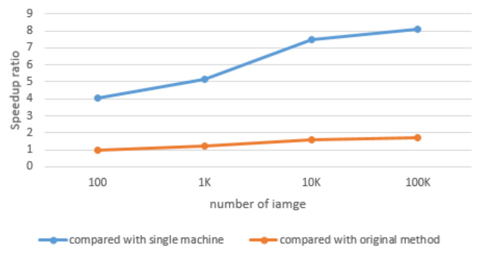

FIGURE II. SPEEDUP RATIO

Spark provides a Web UI to check information of cluster and monitor the execution result. Default address is http://Master:8080. About this system, building image database belongs to the preparatory work. Therefore we ignore the time spent on building Spark environment and image database, our analysis is focused on speed of retrieving similar images. In order to contrast, we test three kinds of CBIR system:

Improved method on single machine, original method on Spark and improved method on Spark. The capacity of image database is 100, 1K, 10K, 100K, respectively.

The results of the test and comparison analysis are presented in Table IV and Figure II.

TABLE IV. EXECUTION TIMES ON THE CLUSTER

Size	Single Machine(time in seconds)	Original Method On Spark(time in seconds)	Improved Method On Spark(time in seconds)
100	2.341	0.545	0.578
1K	18.294	4.312	3.554
10K	152.714	32.145	20.327
100K	1273.231	270.343	157.216

Table IV shows Spark cluster has obvious strength than single machine, although it still spend more time due to small files. And our improved method on Spark significantly improved the matching speed. When database is small, our method does not show the superiority. Once the database is stored with larger amounts of data, matching speed achieves rapid improvement.

V. CONCLUSIONS

In this paper, an improved CBIR system based on Spark is proposed to reduce the computational cost by decreasing the frequent read and write operations on HDFS and starting of large numbers of tasks. Using SURF algorithm to extract features and then converting an image to a java class which can be serialized and stored in a file [19]. However, huge amount of small files lead to plenty of time consuming. So we make use of Avro to combine these files, furthermore we modify the Avro file’s structure, making it can be random read. All improved Avro files compose image database and can be created to RDD. An Avro file is an element of RDD and it can be persisted in memory across operations which allow future actions to be much faster. Then our system can random access a data which represents an image from Avro file's block area and match it with input data.

Our improved system spends a bit more time on constructing database at the pre-processing stage, while it doesn’t affect the real-time retrieval performance and speed of a CBIR system. According to our experiments, the results show that the efficiency of proposed system is obvious better for huge amount of image data.

ACKNOWLEDGMENT

The research was partly supported by the program for Professor of Special Appointment(Easter Scholar) at Shanghai Institutions of Higher Learning, USST incubation project(15HJPY-MS02), National Natural Science Foundation of China(61502220; U1304616) and Shanghai Municipal Natural Science Foundation(15ZR1428600).

REFERENCES

[1] James Z Wang, Jia Li, Gio wiederhold, “SIMPLIcity: Semantics-sensitive Integrated Matching for Picture Libraries.” IEEE Trans. On Pattern Analysis and Machine Intelligence, vol 23, no.9, 2001, pp. 947-963.

[2] J. R. Bach, et al., The Virage image serch engine: An open framework for image management, Storage and Retrieval for Still Image and Videl Databases Iv, val. 2670, pp. 76-87, 1996.

[3] Hadoop project, http://hadoop.apache.org/

[4] Spark project, http://spark.apache.org/

[5] J.S.Hare, S. Samangooei, and P.H. Lewis, “Practical scalable image analysis and indexing using Hadoop,” Multimedia Tools and Applications, val.71, no.3, pp.1215-1248, 2014.

[6] D. Yin and D. Liu, “Contect-based image retrial based on Hadoop,” Mathematical Problems in Engineering, val. 2013, Article ID 684615, 7 pages, 2013.

[7] Jia Li, James Z. Wang, “Automatic Lingusitic Indexing of Pictures By A Statistical Modeling Approach” IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 25, no. 9, 2003, pp .1075-1088.

[8] H. Karau, Learning Spark: Lighting-Fast Big Data Analysis, O’Reilly Media, Sebastopol, Calif, USA, 2015.

[9] J. Venner, “Pro Hadoop,” Springer Berlin, 2009. pp. 177-205.

[10] L. Zhou, N. Xiong, L. Shu, A. Vasilakos, and s. Yeo, “Context aware middlewre for multimedia services in hererogeneous networks,” IEEE intelligent System, vol. 25, no. 2, pp. 40-47, 2010.

[11] L.Zhou, N.Xiong, L.Shu, A. Vasilakos, and S. –S. Yeo, “Context-aware middleware for multimedia services in heterogenrous networks,” IEEE Intelligent System, vol. 25, no. 2, pp. 40-47, 2010.

[12] Avro project, http://avro.apache.org/

[13] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” Communications of the ACM, vol. 51, no.1, pp. 107-113, 2008.

[14] H. Freeman. “Computer Processing of Line-Drawing Images,” ACM Computing Surveys, Vol. 6, No. 1, pp. 57-97, 1974.

[15] G. Qiu, “Color Image Indexing using BTC,” IEEE Trans. Image Processing. 12(1), 93-101, 2003.

[16] D. Zhang and G. Lu, “Review Of Shape Representation and Description, “ Pattern Recognition, Vol. 37, No. 1, pp. 1-19, 2004.

[17] C.W. Tsai, C.-F.Lai, H.-C. Chao, and A. V. Vasilakos,”Big data analytics:a surve,” Journal of Big Data, vol.2, no. 1, 2015.

[18] S. Fong, R. Wong, and A. Vasilakos, “Accelerated pso swarm search feature selection for data stream mining big data,” IEEE Transactions on Services Computing, 2015.

[19] A. V. Vasilakos, Z. Li, G. Simon, and W. You, “Information centric network: research challenges and opportunities,” Journal of Network and Computer Applications, vol. 52, pp.1-10, 2015.