Content Based Image Retrieval on Hadoop Framework

U.S.N. Raju, Shibin George, V. Sairam Praneeth, Ranjeet Deo,Priyanka Jain
Department of Computer Science and Engineering,
National Institute of Technology, Warangal, India
usnraju@yahoo.com, george.shibin1993@gmail.com, sairampraneeth@gmail.com,
devranjeet@gmail.com, pjain700@gmail.com

***Abstract*—This paper presents an approach for implementing Content Based Image Retrieval (CBIR) on Hadoop MapReduce framework. Although any algorithm can be used for the actual retrieval (provided it can be parallelized), in this paper, we discuss the approach of Local Tetra Patterns (LTrPs). The MapReduce implementation details presented here can be used for most CBIR techniques that involve feature-vector computations and distance measures.**

Keywords-CBIR; Hadoop; MapReduce; Local Tetra Patterns; SequenceFiles;

I. INTRODUCTION

Due to growing number of digital data, an efficient method to retrieve images from large databases is required. The two approaches to image retrieval are: *Context based* approach in which the image is indexed by humans by using keywords which requires considerable effort and, *Content based Image Retrieval (CBIR)*, more popular these days, which indexes the image by using low-level features such as color,shape, texture, faces, spatial layout, etc. that are extracted from the image and are used to represent it in database.

Section II of this paper deals with an algorithm that we use for image retrieval. The algorithm makes use of a feature descriptor *Local Tetra Patterns* first introduced in [1]. Since the algorithm for feature-vector computation and query matching all operate independently on an image, the algorithm turns out to be parallelizable to a great extent.

Section III of this paper deals with details of implementing CBIR on Hadoop-MapReduce framework. This section applies in general for all CBIR techniques that can be parallelized and involve feature-vector computation and query-matching using an appropriate distance measure.

II. CONTENT BASED IMAGE RETRIEVAL USING LOCAL TETRA PATTERNS

Local Tetra Patterns (LTrP), as a feature descriptor for CBIR, was first introduced in [1].

Given image I, the first-order derivatives along 0^0 and 90^0 directions are denoted as $I^1_{0^0}$, $I^1_{90^0}$. Let C_0 denote the center pixel in I. Consider the eight-pixel neighborhood of C_0 in Fig. 1. Then, the first-order derivatives at the center pixel are calculated by using (1).

$$\begin{aligned} I^1_{0^0}(c_0) &= I(c_4) - I(c_0) \\ I^1_{90^0}(c_0) &= I(c_2) - I(c_0) \end{aligned} \tag{1}$$

and the direction of the center pixel can be calculated by using equation (2):

$$I^1_{Dir}(c_0) = \begin{cases} 1, & I^1_{0^0}(c_0) \ge 0 \quad and \quad I^1_{90^0}(c_0) \ge 0 \\ 2, & I^1_{0^0}(c_0) \ge 0 \quad and \quad I^1_{90^0}(c_0) < 0 \\ 3, & I^1_{0^0}(c_0) < 0 \quad and \quad I^1_{90^0}(c_0) \ge 0 \\ 4, & I^1_{0^0}(c_0) < 0 \quad and \quad I^1_{90^0}(c_0) < 0 \end{cases} \tag{2}$$

From above equation, it is obvious that 1, 2, 3, or 4 are the possible direction for each center pixel and thus the image is converted into four directions. The second-order LTrP is defined in equation (3) and equation (4)as follows:

$$\begin{aligned} LTrP^2(c_0) = \{ & f_1(I^1_{Dir}(c_0), I^1_{Dir}(c_1)), f_1(I^1_{Dir}(c_0), I^1_{Dir}(c_2)), \ldots \\ & f_1(I^1_{Dir}(c_0), I^1_{Dir}(c_k))\} \quad |k = 8 \end{aligned} \tag{3}$$

where,

$$f_1(I^1_{Dir}(c_0), I^1_{Dir}(c_k)) = \begin{cases} 0, & I^1_{Dir}(c_0) = I^1_{Dir}(c_k) \\ I^1_{Dir}(c_k), & else \end{cases} \tag{4}$$

Finally,it gives 12 (4×3) binary patterns and the details of LTrP are given in [1].The magnitude pattern is also calculated for the image. Thus, total 13 patterns need to be calculated for LTrP.

Equations (5), (6) and (7) are used for calculating magnitude pattern, they are as follows:

$$M_{I^1}(c_p) = \sqrt{(I^1_{0^0}(c_p))^2 + (I^1_{90^0}(c_p))^2} \tag{5}$$

C_1	C_2	C_3
C_8	C_0	C_4
C_7	C_6	C_5

Figure 1.8-pixel neighborhood of C_0

$$LP = \sum_{p=1}^{P} 2^{(p-1)} \times f_2(M_{I^1(c_p)} - M_{I^1(c_0)})\big|_{P=8} \quad (6)$$

where,

$$f_2(u) = \begin{cases} 1, & u \geq 0 \\ 0, & u < 0 \end{cases} \quad (7)$$

*QueryMatching***:**

The feature vector for the j^{th} image in database is represented by $f_{DB_j} = (f_{DB_{j1}}, f_{DB_{j12}}, \ldots f_{DB_{j1g}})$, where j=1,2, ... $|DB|$. Query image feature vector is represented by $f_Q = (f_{Q1}, f_{Q2}, \ldots f_{Qn})$. The n best images are found by calculating distance using distance measure as shown in equation (8).

$$D\,(Q, DB) = \sum_{k=1}^{l_g} \frac{(f_{DB_{jk}} - f_{Qk})}{(f_{DB_{jk}} + f_{Qk} + 1)} \quad (8)$$

The entire algorithmcan besummarized in the following manner:

```
Input: Query image
Output: Retrieval result

 1. Load the query image and convert it into gray-scale image.
 2. Apply the first-order derivatives along horizontal and vertical
    diagonal axis.
 3. Load the query image and convert it into gray-scale image.
 4. Apply the first-order derivatives along horizontal and vertical
    and diagonal axis.
 5. Calculate the derivative patterns along different directions.
 6. Calculate the histograms of the calculated patterns.
 7. Calculate the direction for each centre pixel.
 8. Divide patterns into parts based on centre pixel direction.
 9. Calculate the tetra patterns, and separate them three binary
    patterns per each direction.
10. Construct feature vector using histograms from step 4 and 8.
11. Compare the query image with the images in the
    constructed database.
12. Retrieve the images based on the best matches.
```

III. Content Based Image Retrieval on Hadoop

A. *Loading images onto the* HDFS

All the images are loaded onto the HDFS using a simple shell script that makes use of *put* command, shown as:

hdfsdfs –put <filename>

Once all the images are loaded, we proceed to next step.

B. *Converting images to SequenceFiles*

Hadoop is suitable for large files but small in number. This is why Image retrieval applications pose a problem, as they are effectively small files large in number.

However, these small files can be consolidated to form larger files. Step *B* uses a *MapReduce* program that takes as input all the image files on the HDFS and converts them to SequenceFile format. SequenceFile is Hadoop's inbuilt mechanism to handle binary data, images for instance. SequenceFile is an ordered collection of Key-Value pairs. This step is crucial because moving around images in each Mapper results in significant I/O overhead in further steps. On the other hand, since SequenceFiles are collection of images, a single SequnceFile can be moved with less overhead, and each Mapper can handle more number of images. The outline of this step is described in Fig. 2.

C. *Constructing Feature-vector from images*

Step *B* produces a bunch of SequenceFiles, each containing collection of images. Step *C* uses a MapReduce program where each Mapper process takes as input a SequenceFile and computes feature-vector for each image in the file. The results of these computations are again consolidated to get a SequenceFile i.e. Step B produces SequenceFiles ofcollection of Key-Value pairssuch that the key is again the filename and value is the feature-vector computed from that image. The outline of this step is described in Fig. 3.

At the end of Step *C*, we have computed the feature-vectors of all the images in the database. Steps *A* to *C* need to be performed only once per dataset of images.

D. *Performing CBIR query*

Step *D* uses a MapReduce program where each mapper takes as input a SequenceFile produced in Step *C*, consisting the feature vectors of a group of images. Let us suppose that only the top *n* images have to be retrieved from the dataset. The feature-vector of the image which is to be queried is matched against that of others and the top *n*results from each mapper. Note that each mapper needs to forward only the top n queries to the reducer as the final top ten will be present in this set. Only a single reducer is needed that collects top *n* results from each mapper.

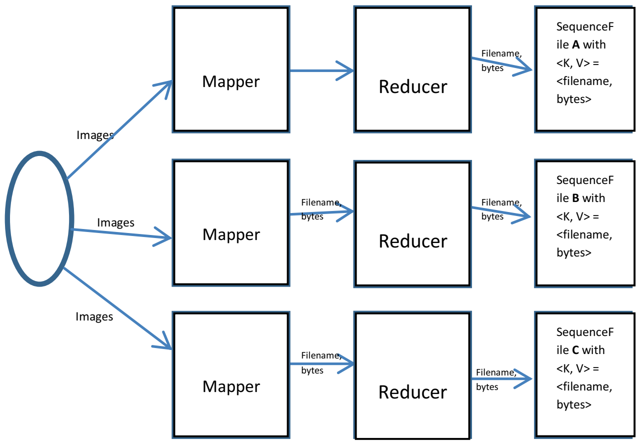

Images
Images
Images
Mapper
Mapper
Mapper
Reducer
Reducer
Reducer
Filename, bytes
Filename, bytes
Filename, bytes
Filename, bytes
Filename, bytes
SequenceFile **A** with <K, V> = <filename, bytes>
SequenceFile **B** with <K, V> = <filename, bytes>
SequenceFile **C** with <K, V> = <filename, bytes>

Figure 2.Outline of Step B

If there are m mappers, the reducer receives $n \times m$key-value pairs containing the filename and its distance measure from the queried image. The $n \times m$pairs are again sorted, and finally, the global top n results are displayed. The outline of this step is described in Fig. 4.

IV. CONCLUSION

For the purpose of demonstration, an HPC cluster (with 9 nodes). The master node has a HP Prolaint DL380P Gen.8 processor (6 cores), 64GB RAM and 3×600GB HDD, running Cent-OS 6.5. Each of the slave nodes has 16GB RAM and 2×300GB HDD. The tests were performed on the following datasets:

- Corel-1K data-set [2] is used as a standard for image retrieval applications. The dataset has *1,000* images across 10 categories (each category having 100 images).
- Corel-10K data-set [3] has *9,902* images distributed across 100 categories (each category having around 100 images).
- A set of *13,473* images collected from ImageNet database [4]. The categories chosen were the same as those in Corel 1K.

Fig. 5 shows the sample execution screenshots for an image of category *Food*(from Corel 1k). Note that out of 10 images retrieved, one is of another category (*Building*). The results (in Table 1) show that the process can be parallelized and the execution times can be lowered in image retrieval applications on Hadoop.

The structure of the implementation is such that any feature-extraction algorithm can be easily applied to the data-set with some simple modifications in Step C. We used the Local Tetra Patterns (Section II) with feature vector length of *767*for textural features.

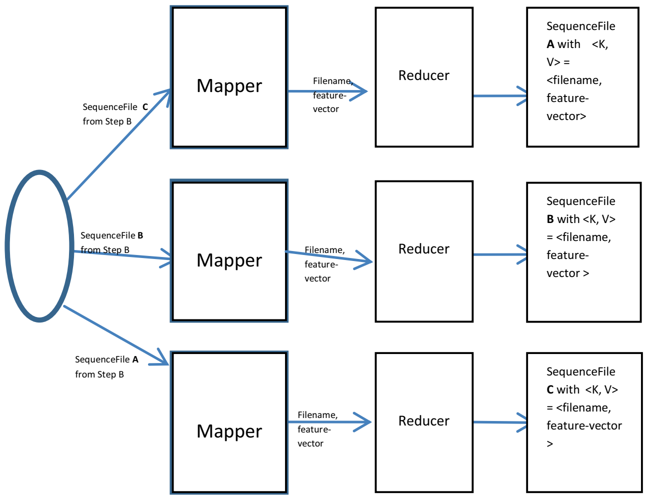

SequenceFile C from Step B
SequenceFile B from Step B
SequenceFile A from Step B
Mapper
Mapper
Mapper
Filename, feature-vector
Filename, feature-vector
Filename, feature-vector
Reducer
Reducer
Reducer
SequenceFile A with <K, V> = <filename, feature-vector>
SequenceFile B with <K, V> = <filename, feature-vector >
SequenceFile C with <K, V> = <filename, feature-vector >

Figure 3.Outline of Step C

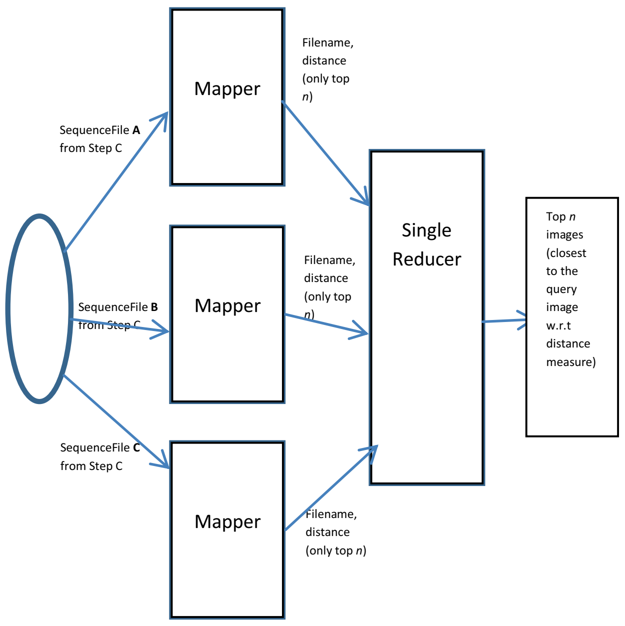

SequenceFile A from Step C
SequenceFile B from Step C
SequenceFile C from Step C
Mapper
Mapper
Mapper
Filename, distance (only top n)
Filename, distance (only top n)
Filename, distance (only top n)
Single Reducer
Top n images (closest to the query image w.r.t distance measure)

Figure 4. Outline of Step D

Some optimizations like support for incremental additions to the image data-set still remain to be implemented and can be useful more real-life applications where data is added not in bulk but in incremental fashion. Also, distributed applications like *HBase*(NoSQL database for Hadoop)can be used to store feature-vectors in Step C and for efficient query / retrieval in Step D. Further tests on larger data-sets are being carried out. Future works will be aimed at reducing the response / time to suit the needs of web-based applications as well.

TABLE I. EXECUTION TIMES ON THE CLUSTER

Dataset Name	**MapReduce Step B (time in seconds)**	**MapReduce Step C (time in seconds)**	**MapReduce Step D (time in seconds)**
Corel 1K	20	18	14 - 16
Corel 10K	23	32	15 - 18
ImageNet	40	195	14 - 22

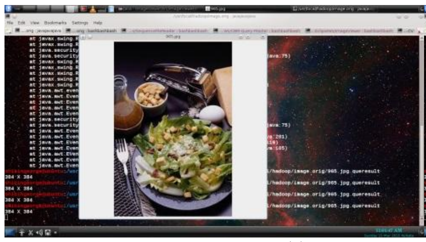
REFERENCES

[1] Subramanyam Murala, R.P.Maheshwari and R Balasubramanian "Local Tetra Pattens: A New Feature Descriptor for Content-Based Image Retrieval", IEEE Trans. on Image Processing, Vol. 21, No.5, May 2012, pp. 2874 - 2886.

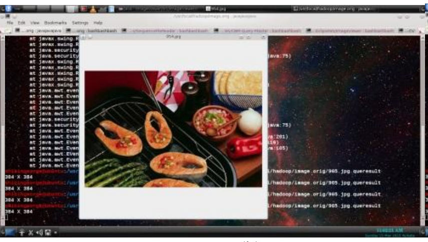
[2] Jia Li, James Z. Wang, ``Automatic linguistic indexing of pictures by a statistical modeling approach," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 9, 2003, pp. 1075-1088.

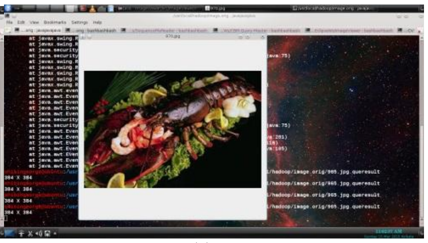
[3] James Z. Wang, Jia Li, Gio Wiederhold, ``SIMPLIcity: Semantics-sensitive Integrated Matching for Picture LIbraries," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol 23, no.9, 2001, pp. 947-963.

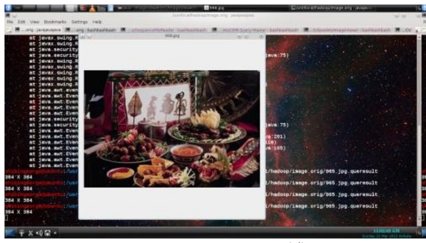
[4] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei "ImageNet Large Scale Visual Recognition Challenge", 2014.

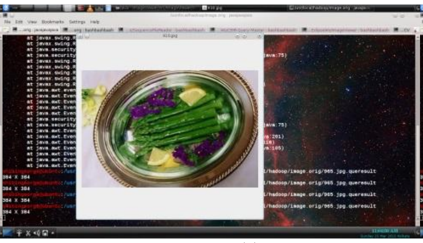

```
at javax.swing.R
at javax.swing.R
at java.security
at java.security
at javax.swing.R
at javax.swing.R
at javax.swing.R
at javax.swing.R
at javax.swing.R
at java.awt.even
at java.awt.Even
at java.awt.Even
at java.awt.Even
at java.awt.Even
at java.security
at java.security
at java.awt.Even
at java.awt.Even
at java.awt.Even
at java.awt.Even
at java.awt.Even
at java.awt.Even
at java.awt.Even
384 X 384
384 X 384
384 X 384
384 X 384

java:75)
java:75)
a:201)
110)
a:105)
/hadoop/image.orig/965.jpg.queresult
/hadoop/image.orig/965.jpg.queresult
/hadoop/image.orig/965.jpg.queresult
/hadoop/image.orig/965.jpg.queresult
```

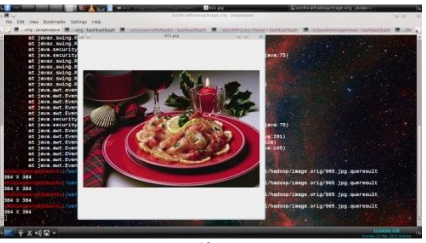
(a)

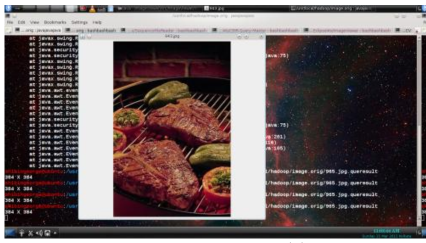

(b)

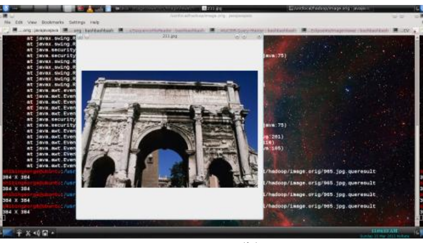

(c)

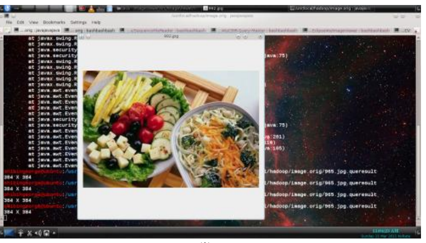

(d)


(e)


(f)


(g)


(h)


(i)

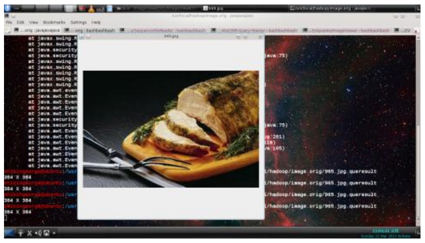

(j)

Figure 5. Screenshots of the query results: (a) is the query image, hence the first one, images (b) to (j) are the other images retrieved.As can be seen, (h) is an image of category *Buildings*and hence was wrongly retrieved. All the other images were correctly retrieved.