FEATURE EXTRACTION IN HADOOP IMAGE PROCESSING INTERFACE

Ishit Vyas[1], Prof. Bhavika Gambhava[2] and Digvijaysingh Chauhan[3]

[1]M. Tech Student, Dharmsinh Desai University,
Nadiad-387001, Gujarat, India
ishit711@gmail.com

[2] Assistant Professor (Computer Engineering), Dharmsinh Desai University,
Nadiad-387001, Gujarat, India
bhavika.ce@ddu.ac.in

[3] Project Scientist, Bhaskaracharya Institute for Space Applications and Geo-Informatics,
Gandhinagar 382007, India
d*igvijaych@gmail.com*

ABSTRACT

With the ease of Internet access and higher data rate availability to citizens all over the world, data sharing has become versatile in terms of type of data. Instead of just sharing data in textual form like few years back, people now are sharing the data in forms of pictures and videos. In most widely used social media platform – Facebook, users upload 350 million pictures every day. But, the traditional image processing systems are not capable of processing such a huge amount of data due to limitations in storage and computational capabilities. Moreover, MapReduce programming model of Hadoop framework provide processing of large amount of data. But since MapReduce programming model was created with intend to process the textual data, it is quite inefficient to process the images using MapReduce programming model directly. Also, technical complexity of MapReduce is high and hence before working on the image processing tasks, one has to understand this model. This is inconvenient and time consuming process. To ease this complexity, various image processing frameworks have been introduced which abstracts the MapReduce model while focusing on image processing task. Hadoop Image Processing Interface is one of those frameworks with various features and support to OpenCV.

Key words: Hadoop, HIPI, Image Processing, Feature Extraction

1. INTRODUCTION

The use of data presented in image format in fields of satellite imaging, medical imagery, astronomical data analysis, computer vision etc. has been increased over the years. And as a result of it, requirements to process those images have also been increased. Various algorithms, tools and techniques have been developed to analyze and process those images. In last few years, the overall data stored and shared in digital form is increased so much that it is difficult for traditional standalone data processing systems to analyze and process those data and get an useful outcome from it. To overcome these issues various technologies such as distributed processing and parallel programming models have been introduced. Also, requirements to modify or develop new algorithms for processing data in distributed and parallel environments have been arisen.

1.1 BIG DATA

In field of computing, Big data refers to newfound ability to crunch a vast quantity of information, analyze it instantly, and draw sometimes astonishing conclusions from it [1]. Essentially, the data is large enough that traditional data processing systems are not capable of handling.

1.2 APACHE HADOOP

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures [2]. Even though there are other technologies exist, most of the Big Data are processed in Hadoop because it's highly fault tolerant highly scalable characteristics [3].

Hadoop project consist of four modules: Hadoop Common; which provides common utilities and support, Hadoop Distributed File System; which is a distributed file system, Hadoop YARN; which is a framework for resource management and job scheduling and most importantly Hadoop MapReduce; a system for parallel processing of large amount of data.
MapReduce is heart of Hadoop [4] ecosystem.

1.2.1 MapReduce

MapReduce is a programming model and an associated implementation for processing and generating large data sets [5]. MapReduce essentially consist of two phases: Map and Reduce, and an additional hand-off process called Shuffle and Sort. Mapper and Reducer are the interfaces implemented in order to provide the map and reduce methods.

2. BIG DATA IMAGE PROCESSING

Conventional image processing systems are not capable of processing big image data [6]. Hadoop is efficient at processing textual data, but when it comes to processing images, it becomes quite difficult since the data of the image to be processed is taken as String format. Other than this, there is a major known problem called as Small File Problem.

2.1 HADOOP IMAGE PROCESSING INTERFACE (HIPI)

HIPI is an image processing library designed to be used with the Apache Hadoop MapReduce parallel programming framework.

HIPI facilitates efficient and high-throughput image processing with MapReduce style parallel programs typically executed on a cluster. HIPI abstracts highly technical details and allows us to implement many of the image processing techniques.

To overcome above problem given in section 2.1, HIPI introduces a new data type called HIPI Image Bundle (HIB) that stores many images as one big pile so that MapReduce jobs can be performed more efficiently.

HIB (HIPI Image Bundle) is made up of two different files:

I. Data File: It contains the concatenated bundle of images

II. Index File: It contains information about offset of images in data file

HIB have similar speed to SequenceFile, but it does not have to be read serially. Also HIBs are more customizable and mutable as compared to both of the above. In addition to this, HIPI maximizes data locality by altering the data flow in MapReduce model.

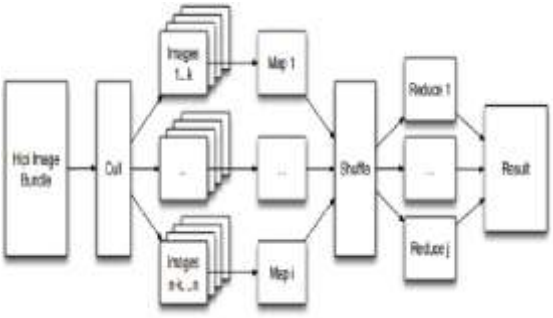

Fig: Data processing flow in HIPI [7]

In addition to HIB, which is the key component of HIPI, it also introduced a new step in processing of images. The stage is called as "culling". These steps allows filtering the images in a HIB based on various user defined conditions. A new class – culler is implemented for this operation. This will reduce the unnecessary overhead of processing irrelevant images for further stages.

For generating InputSplit, HIPI introduces HibInputFormat class, which is inherited from FileInputFormat class. Then, the images are represented as objects of HipiImage abstract class associated with HipiImageHeader. In that, images are represented in different formats and then given as input to the Mapper. The rest of the data flow is same as regular MapReduce model as per mentioned earlier. HIPI also includes support for OpenCV.

3. FEATURE EXTRACTION USING HIPI

We have implemented Scale Invariant Feature Transform (SIFT) feature extraction algorithm using HIPI.

The six step approach is as per given below.

1. Create HIPI Image Bundle (HIB) from the set of images.
2. Convert all the images of HIB from FloatImage to OpenCVMat.
3. Convert the images from result of step 3 to graysclae.
4. Detect the features by using SIFT feature detector.
5. Extract the features descriptors with the use of SIFT descriptor extractor.
6. Compute the keypoints.

```
class FeatureExtraction{
        method floatImage2OpenCVMat
        method extractFeature
        method mat2json

    class FeatureExtractionMapper{

            method setup
            method map{

                if passed value is a valid image{

                    call method floatImage2OpenCVMat

                    create a new Mat object space for grayscale
                    conversion convert image to grayscale using cvtColor
                    convert the datatype of image from float to unsigned integer

                    call method extractfeature
                    get metadata of file by using method getMetaData on key
                    write descriptor along with filename in output
                    }
                }
            }

            method run{

                if valid HIB is passed in argument{

                    create an object for job
                    set input and output paths
                    set input and output format classes
                    set the Jar file
                    set mapper class
                    set map output key and value classes
                    set output key and value classes
                }
            }

        method main
        {
            call method run
        }
}
```

Algorithm: Feature Extraction using HIPI

Initially, take a large dataset of JPEG images. Convert the images to HIB using hibImport shellscript provided by HIPI. The data is converted into two files; Index and Data files and stored into HDFS. The files in HIB have the representation format FloatImage.These images of type FloatImage are converted into OpenCV compatible OpenCVMat files in order to make these images convenient to process using OpenCV library. These images are converted into grayscale as they are required in grayscale type in further steps. OpenCV provide OpenCV SIFT feature detector to detect the SIFT features. We have used this feature detector and detected the features from the grayscale images obtained in the previous step. In next step, feature descriptors are extracted with the use of SIFT descriptor extractor. In the last step, keypoints are computed and stored in JSON format. These keypoints are stored in HDFS.

The implementation of above given approach was implemented in following environment:

Processor: Intel Core-i7 Processor
RAM: 10 GB
Hadoop Environment: Hadoop 2.7.3 Pseudo Distributed Environment
Dataset: 1.1 GB INRIA Holidays[8] dataset containing 812 JPEG images

4. COMPARISON BETWEEN TRADTITIONAL MATLAB APPROACH AND HIPI

1.3 MB of image from the same dataset was used to conduct the experiment in MATLAB.
1.1 GB dataset was used as whole to conduct the experiment.

Amount of time taken using HIPI for 1.1 GB dataset = 29.96 Minutes
Average amount of time taken using HIPI for a 1.3 MB image = 2.1 Seconds
Amount of time taken using MATLAB for 1.3 MB image = 1003.96 Seconds

From the above comparison, we can observe that there is significant amount of reduction in terms of time complexity while using HIPI. We have conducted experiment on both the platforms multiple times. Amount of time taken between consecutive runs is almost similar.

5. CONCLUSION AND FUTURE WORK

From the above given approach to implement SIFT feature extraction algorithm, we can conclude that image processing in Hadoop environment using Hadoop Image Processing Interface can result into significant reduction in time complexity. Furthermore, by developing such algorithms, Hadoop can be utilized and can be made fully compliant for image processing. In future, the extracted features can be used for feature matching between images in order to use it for creating image mosaics. Moreover, other existing image processing algorithms can be modified and tested in Hadoop environment.

ACKNOWLEDGEMENT

We are thankful to Shri T. P. Singh – Director at BISAG for supporting this research and providing the infrastructure and Dr. M. B. Potdar – Project Director at BISAG for his valuable continuous guidance and support.

REFERENCES

[1] V. Mayer-Schonberger and K. Cukier, ¨ Big Data: A Revolution That Transforms How we Work, Live, and Think. Houghton Mifflin Harcourt, 2012. (Chinese translated version by Y. Sheng and T. Zhou, Zhejiang Renmin Press)

[2] https://hadoop.apache.org/

[3] Mehul Nalin Vora, "Hadoop-HBase for large-scale data," *Proceedings of 2011 International Conference on Computer Science and Network Technology*, Harbin, 2011, pp. 601-605. doi: 10.1109/ICCSNT.2011.6182030

[4] Janani.J and Kalaivani.K "Hadoop MapReduce using Cache for Big Data Processing", International Conference on Current Research in Engineering Science and Technology (ICCREST-2016) pp. 31-37

[5] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (January 2008), 107-113. DOI: https://doi.org/10.1145/1327452.1327492

[6] L. Dong *et al*., "A Hierarchical Distributed Processing Framework for Big Image Data," in *IEEE Transactions on Big Data*, vol. 2, no. 4, pp. 297-309, Dec. 1 2016. doi: 10.1109/TBDATA.2016.2613992

[7] S. Chris, L. Liu, A. Sean, and L. Jason, HIPI: A hadoop image processing interface for image-based map reduce tasks, B.S. Thesis. University of Virginia, Department of Computer Science, 2011.

[8] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search". In the proceedings of the 10th ECCV, October 2008.