Analysis and Comparative Study of Hadoop Image Processing Frameworks

Ishit Vyas[1], Digvijaysingh Chauhan[2], Dr. M B Potdar[3] and Prof. Bhavika Gambhava[4]

[1]M.Tech Student, Dharmsinh Desai University,Nadiad-387001, Gujarat, India
[2] Project Scientist, Bhaskaracharya Institute for Space Applications and Geo-Informatics, Gandhinagar 382007, India
[3] Project Director, Bhaskaracharya Institute for Space Applications and Geo-Informatics, Gandhinagar 382007, India
[4] Assistant Professor (Computer Engineering), Dharmsinh Desai University,Nadiad-387001, Gujarat, India

ABSTRACT

With the advancement in networking and storage technologies over the past few years, sharing of data over the Internet has been increased rapidly. Data types of the shared data has also become versatile. Data are shared in forms in text, image, videos etc. With this increase of data, there has been arisen requirement to process those data in order to find useful outcome out of that. In order to do so, Big Data technologies have been developed. Hadoop is one of the most popular frameworks developed in Big Data technologies. But since Hadoop was primarily developed to support textual data analysis, it is quite difficult to perform analysis on other formats such as images and videos. To support image and video processing various other frameworks and libraries have been developed. This paper demonstrates analysis and comparison between two Hadoop image processing frameworks; HIPI and MIPr.

Key words: *Hadoop, HIPI, Image Processing, MIPr*

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.

Data flow in MapReduce is as follows:

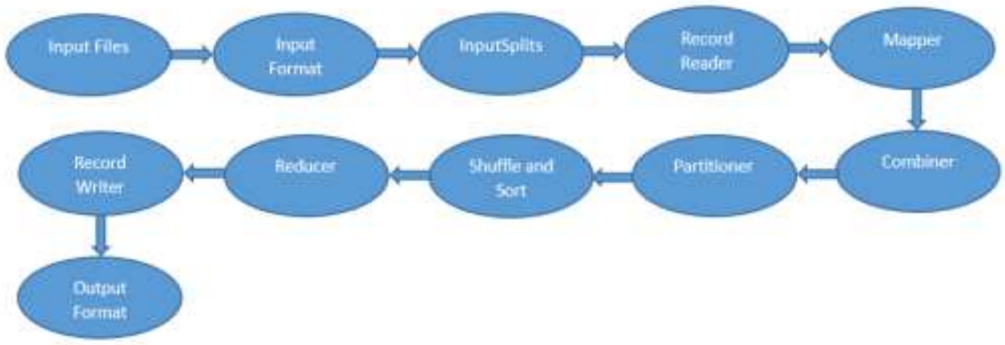

Fig -1: MapReduce Data flow

Table -1: MapReduce data flow stages

Input files	Input files for MapReduce task are stored in HDFS.
InputFormat	InputFormat defines how input files are split and read.
InputSplits	InputSplits created by InputFormat represents the part of data processed by one mapper.
RecordReader	RecordReader converts data into key/value pair.
Mapper	Mapper takes input from RecordReader and convert it into intermediate key/value pair.
Combiner	It performs local aggregation and minimize transfer between mapper and reducer.
Partitioner	Partitioner is used when there is more than one reducer used. It performs partitioning of combiner output.
Shuffling and Sorting	The function of Shuffling is to transfer output of mapper to reducer, while Sorting perform merging and sorting of map output according to keys.
Reducer	The intermediate key/value pair generated by mapper is given as input and runs reducer function.
RecordWriter	It writes key/value pair from the output of reducer to output files.
OutputFormat	The format in which the output key/value pairs obtained from output file is determined by OutputFormat.

2. Big Data Image Processing

As per defined above, 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 Problem while handling small files in Hadoop

The small file is one which is much smaller than the block size of HDFS. MapReduce tasks run more efficiently when the input is one large file as opposed to many small files. If the file is very small and there are a lot of them, then each map task processes very little input, and there are a lot more map tasks, each of which imposes extra bookkeeping overhead [7]. Hence, there is significant reduction in performance. Reading many number of small files is computationally expensive for HDFS. This occurs because each file, directory and block in HDFS occupies 150 bytes in namenode [8]. To overcome the given problem, Hadoop provides two file formats: Hadoop Archive (HAR) and Sequence Files. But, HAR is for archival purpose and is actually perform slower reading than in standard way. [9]. While, Sequence files are time consuming to create and it should be read serially.

2.2 Hadoop Image Processing Interface

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.

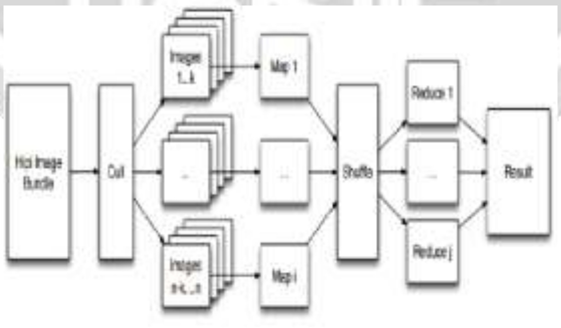

Fig - 2: Data processing flow in HIPI [9]

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.

2.3 MapReduce Image Processing (MIPr) framework

MIPr is MapReduce Image Processing framework. MIPr facilitates image processing in large size. MIPr includes the implementation of image processing algorithm in MapReduce manner. MIPr is developed with the aim of making image processing simpler and easier in Hadoop.

MIPr's architecture is three layered [10]. The first one is core components layer which provides image representation in various forms in Hadoop format such that that can be used as a value in MapReduce programs. It also provides I/O tools. Second layer includes serial image processing function and MapReduce drivers. Third layer consists of MapReduce image processing algorithms.

MIPr currently support image representations based on JAVA 2D and OpenIMAJ. In MIPr, their customized InputFormat and OutputFormat classes are designed. Unlike HIPI and Open IMAJ, which combines many small files into large files, MIPr does not necessarily require it. To solve the problem of handling many small files,

MIPr uses CombineFileInputFormat, which combines many small files into one large split. As a result of this, less number of Mappers are required. For processing images, current implementation of MIPr includes Java Image Processing API.

3. Comparison: HIPI and MIPr

We have compared HIPI and MIPr both, theoretically and practically in order to analyze efficiency of both.

3.1 Practical Comparison

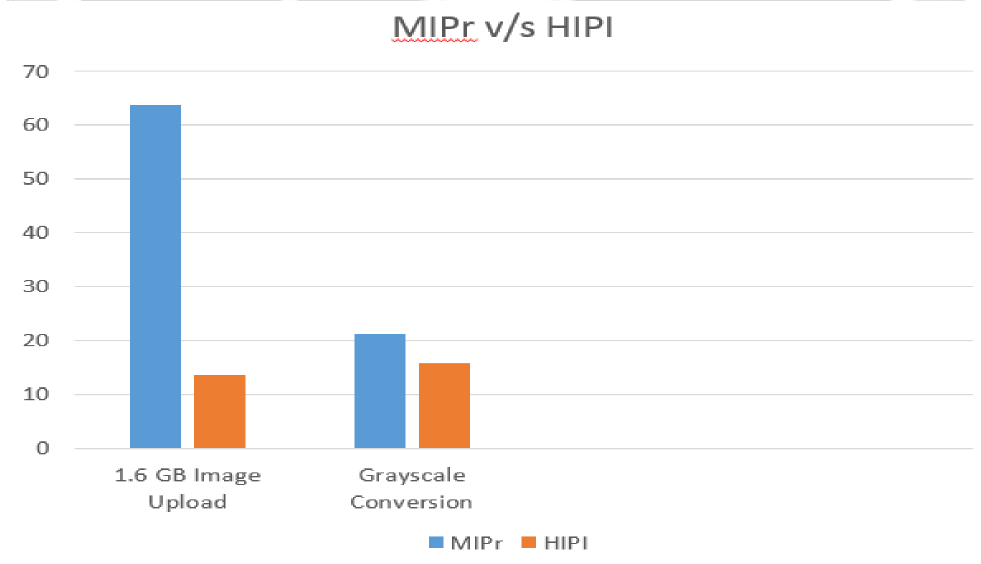

Fig- 3: Practical Comparison: HIPI and MIPr

Two different experiments; 1.6 GB image upload and GrayScale Conversion were conducted in order to analyze performance of HIPI and MIPr. The experiments were conducted on Intel core-i7-4790 with 16 GB of RAM. The results of the experiments shows that in both of the experiments, HIPI outperforms MIPr. Moreover, it was observed that job with MIPr fails while running the experiments with larger size of data.

3.2 Theoretical Comparison

Parameter	**HIPI**	**MIPr**
Input Format	HibInputFormat	BufferedImageInputFormat, CombineImageInputFormat
Output Format	BinaryOutputFormat	BufferedImageOutputFormat
Image Representation Format	ByteImage, FloatImage, RasterImage, RawImage	BufferedImage, Fimage, MBFImage
Solution to small file problem	By combining several small files in HIB	By use of CombineFileInputFormat
Support to OpenCV	Yes	Yes
Types of Images	JPEG, PNG, PPM	Unspecified (JPEG)
Writable Implementation	OpenCVMatWritable	BufferedImageWritable
RecordReader Implementation	HIbRecordReader	BufferedImageRecordReader
RecordWriter Implementation	BinaryRecordWriter	BufferedImageRecordWriter
Culling	Yes	No

4. Conclusion

After studying both the frameworks thoroughly, implementing the experiments and conducting the comparison between HIPI and MIPr, we can conclude that HIPI outperforms MIPr in every aspect. Hence, to perform image processing in Hadoop environment, HIPI should be preferred.

5. Acknowledgement

We are thankful to Shri T. P. Singh, Director, BISAG for providing infrastructure and encouragement to carry out this project at BISAG.

6. 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] http://blog.cloudera.com/blog/2009/02/the-small-files-problem/

[8] Vaibhav Gopal Korat, Kumar Swamy Pamu, " Reduction of Data at Namenode in HDFS using harballing T echnique" International Journal of Advanced Research in Computer Engineering & Technology,Volume 1, Issue 4, June 2012

[9] 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.

[10] A. Sozykin and T. Epanchintsev, "MIPr - a framework for distributed image processing using Hadoop," *2015 9th International Conference on Application of Information and Communication Technologies (AICT)*, Rostov on Don, 2015, pp. 35-39. doi: 10.1109/ICAICT.2015.7338511