Comparison of Approaches ofDistributed Satellite Image Edge Detection on Hadoop

Tapan Sharma
Research Scholar
ASET, Amity University
Noida, Uttar Pradesh, India
tapan.sharma.tech@gmail.com

Dr. Vinod Shokeen
ASET, Amity University
Noida, Uttar Pradesh
India
vshokeen@amity.edu

Dr. Sunil Mathur
MAIT,
GGS Indraprastha University,
Delhi, India

***Abstract*— In the world of bigdata, Hadoop has become the major platform for storage and processing. Due to advancement in technology in the area of remote sensing, tremendous growth in data has been witnessed. In this paper, we have conducted experiments and compared two methods in which edge detection of satellite images is performed on Hadoop. Since, edge detection is one of the prime steps in the field of image processing and is being used for object detection in the image, we have targeted this basic algorithm of image processing for our experiments. In earlier research for edge detection on Hadoop, SequenceFiles were used to store and process images. In our experiments, we have leveraged distributed processing of Hadoop by logically splitting the file on HDFS and performing edge detection in distributed manner. The experiments were performed on Amazon AWS Elastic MapReduce (EMR) cluster using different satellite images varying from 10MB-200MB. The paper describes the comparison of the two approaches.**

Keywords—Hadoop; SequenceFile;Distributed;Edge Detection, Satellite images;HDFS

I. INTRODUCTION

As remote sensing imagery is witnessing huge growth, it has become crucial to explore new and better ways of processing large amount of data. Edge detection is one of the oldest and basic component in the field of image processing. It is the main step needed for feature extraction, object detection and segmentation [1]. As there could be different mechanisms to process images on Hadoop, here we have compared two approaches. There are many algorithms for edge detection but in our experiment we have chosen Canny Edge algorithm to run on Hadoop.

Edge detection is a memory intensive job. As the size of satellite images can vary from few MBs to TBs, it becomes quite important that these algorithms should get executed in optimized manner by leveraging the platform and improving the resource utilization.

When the image size is huge it cannot be loaded into memory on a single machine. Due to this limitation, researchers and developers split the image and then process it region-wise. Since Hadoop is meant for distributed processing we have also logically split the image and process its individual regions on different machines in the cluster. Instead of vertical scale, horizontal scaling is much more efficient with respect to resource utilization in the field of bigdata. In this way as many number of large images can be processed simultaneously.

In section 2, we have discussed the research performed in the related area followed by the brief description of Hadoop framework in section 3. The subsequent sections describe the proposed methodology, experiment and conclusion based on our findings.

II. LITERATURE SURVEY

There are various gradient operators used for edge detection such as:

- Robert Operator: This is the simplest operator to detect edges.
- Sobel Operator: The operator convolves with the original image to calculate approximations of the derivatives. It uses two 3*3 masks for calculation.
- Prewitt Operator: Prewitt's mask values are different than Sobel but performs in the similar way.
- Laplace of Gaussian (LoG): LoG is one of the advanced techniques for edge detection. It is based on calculation of zero crossings in the second derivative of an image.
- Canny Edge: This algorithm is more complex and advanced than the other algorithms. At the same time it produces much better output by lowering error rate and reducing false positives.

The gradient is calculated for each image coordinate. The pixel is considered an edge if gradient value exceeds the threshold amount. The intensity of edges is relatively higher than nearby pixels. Canny Edge is considered the most complex among all but produces the best output [2]. The paper compared the qualities of all the gradient operators and found that the effect of noise is least in Canny Edge. Again, in [3], researchers showed that Canny Edge performs best in object

978-1-5090-6785-5/18/$31.00 © 2018 by IEEE

extraction. The algorithm produces very few false edges as compared to all other. Sobel operator took least time. The studies have also come up with new Mathematical morphological edge detection technique having quality similar to existing techniques [4]. In [5] also, researchers worked on comparing and providing advantages and disadvantages of these algorithms.

There have been researches performed in the field of bigdata and especially Hadoop. The study has been conducted to find the challenges being faced during image segmentation on Hadoop [6]. As the image has to be split for better utilization of resources there are certain challenges that need to be known and rectified. In [6], authors provided image decomposition and recomposition algorithms to remove fake artifacts. The experiments have been performed on content based image retrieval while indexing millions of images with the help of MapReduce [7]. As the satellite images are large in size as well as are generated they are ideal for processing on Hadoop. The paper [8] performed K-Means clustering on Hadoop using satellite images and recently [9], implemented MapReduce job for edge detection algorithms.

Researchers have also proposed new algorithms for clustering of images. Xia et. al proposed MapReduce based Parallel Kaufmann (MPK) algorithm targeting the same problem [10]. In [11], experiments were conducted to compare performance and scalability of edge detection algorithms. In these experiments, the images were stored in SequenceFile. SequenceFile is one of the types of file that stores key and value in binary form [12]. The disadvantage of this approach is experiment couldn't leverage the distributed processing capability of Hadoop as the whole image got loaded and processed on single machine. In our experiment we have processed and the image in distributed manner. Next few sections would describe it in more details.

III. BIGDATA AND MAPREDUCE

Over the period of recent years, Hadoop has become de-facto platform for processing of bigdata. The distributed storage and processing capability helps in processing of large amount of data efficiently and faster than traditional way. Hadoop has two major components viz. Hadoop Distributed File Storage (HDFS) andMapReduce (MR) [13].

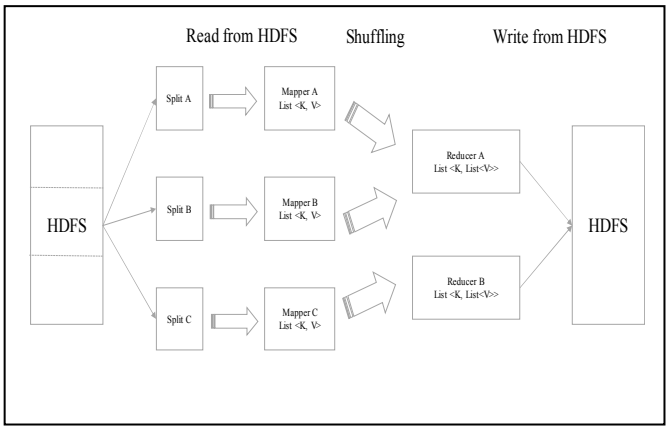

Fig.1Basic MapReduce architecture over HDFS

Hadoop ecosystem supports different file formats such as text files, sequence files, Avro, Parquet, RC, ORC etc. but doesn't have any reader for reading images. Therefore images need to be stored in binary format in SequenceFiles or the developer has to implement customized Input format to read the image.

The Hadoop cluster constitutes of two types of nodes: Name node and Data node. Data nodes are the nodes where file data gets stored while Name node stores the information or the metadata of those files. The file with larger than block size gets split and is stored on HDFS in parts. The block size is generally 64MB that means the file bigger than 64MB would be split into multiple chunks. The information about these chunks is maintained by name node.

MapReduce is the distributed processing engine which reads the portions of the file on respective data nodes where the file chunks are residing and process them on individual data nodes. In this way, the different regions of the same file can be processed simultaneously leveraging resources of multiple data nodes.

Fig.1 shows the MapReduce architecture. There are many phases in a MapReduce job. Initially, mapper tasks are launched on respective data nodes where the file split is stored. The mapper reads the file in the form of key and value pair. The data type of key and value is based on the type of file reader. After processing is done on the pair, map task outputs again a pair of key and value. During the phase of shuffling and sorting, the values belonging to same key are being received by same reducer. Reducer can again perform the logic on the pair of key and list of values corresponding to that particular key. The output of reducer gets stored on HDFS.

IV. METHODOLOGY

As mentioned in section 2, earlier research stored satellite images in the form of SequenceFile which means they did not take advantage of distributed processing. Although the sequence file gets distributed and simultaneously got processed on multiple nodes but the image got processed on single machine only. In our experiment, we have implemented our own InputReader using which we logically split the images into different regions and those regions got processed in different map tasks. Fig. 2 and Fig.3 clearly depicts the difference in the two approaches.

978-1-5090-6785-5/18/$31.00 © 2018 by IEEE

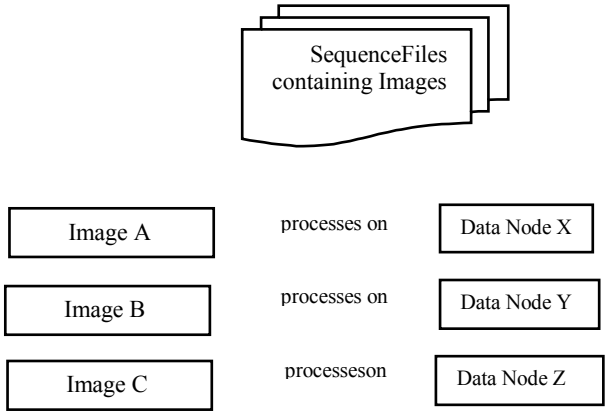

Fig. 2First approach: images stored in Sequence Files

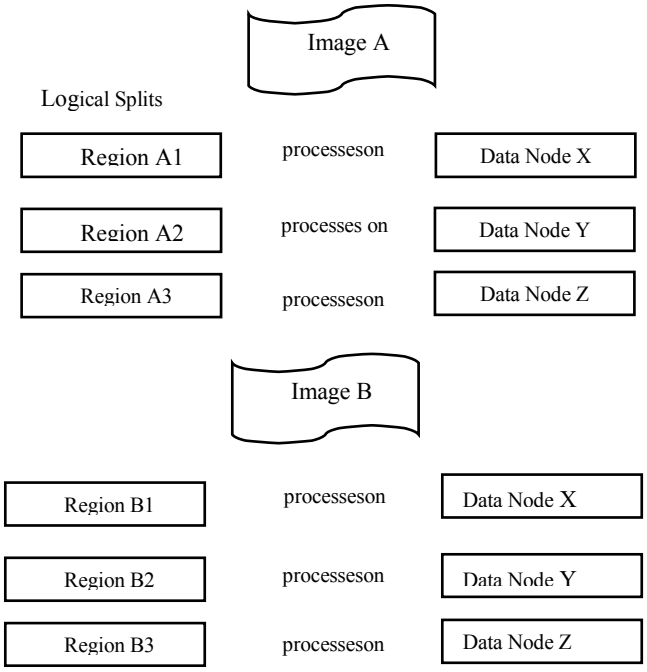

Fig.3. Second approach: Distributed processing of image

For our experiments, we have implemented ImageInputFormat class which provides the custom reader for image files. The images are read using FileImageInputStream and then regions are created. These regions are considered as records and each such region is processed in different map tasks. Fig. 4 lists down the classes being implemented to read the regions of the image. After the edge detection algorithm is executed on respective regions, all these regions are again merged into a single image inside the reducer task. The image is read in the form of ByteWritable object.

***a. class* ImageInputFormat**
This is the custom ImageInputFormat class as hadoop does not provide native support for reading images
***function recordReader* createRecordReader**
return new ImageRecordReader();
***function boolean* isSplitable()**
return true**;**

***b. class* ImageRecordReader**
This is the custom record reader to read image files

***function void* initialize** (*InputSplit* genericSplit, *TaskAttemptContext* context)
read file as *FileImageInputStream*stream
convert file stream into image
***function boolean* nextKeyValue**()
key = new **Text**(imagefilename)
value = getPartOfEntireImage()

Each ge. Each image i 28 MB, otherwis increase the mem
Fig 5 ocessing of the in

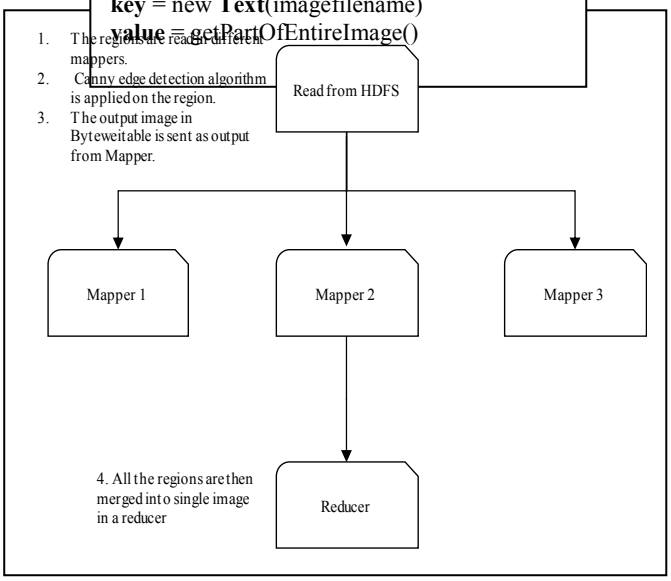

Fig. 5 Experiment workflow

978-1-5090-6785-5/18/$31.00 © 2018 by IEEE

V. EXPERIMENTS

A. Data Set

For our experiments we have used satellite images from MODIS, LandSat, Google Earth Engine and Sentinel sensors. We performed our experiments on five images of size varying from 75 MB to 200 MB while the number of pixels vary from few million to 380 million approximately. Since our objective was to observer the time taken during different approaches we didn't work on large number of images. The image size and the number of pixels are provided in table I.

TABLEI.IMAGE DATASET

Image	Size	Number of pixels
Image A	14 MB	3.3 million
Image B	133 MB	93.1 million
Image C	26 MB	23 million
Image D	193 MB	378 million
Image E	102 MB	179 million

B. Resources

We conducted our experiments on AWS EMR cluster [14]. It had 1 name node and four data nodes. Each node had 16 GB RAM and 100 GB hard disk. The job ran on Apache Hadoop 2.7.2 with block size of 64 MB.

C. Results

We performed two set of experiments with two different approaches described in fig. 2 and fig.3 respectively.First approach has already been implemented in earlier researches. Our findings with respect to disadvantages of first approach were on the similar lines as discussed in previous studies. Table II contains performance numbers for first approach while table IIIshows the numbers for second approach.

TABLEII.PERFORMANCE NUMBER OF FIRST APPROACH (PREVIOUS STUDIES)

Image	Time taken
Image A	59 sec
Image B	Could not run
Image C	2m 3 sec
Image D	Could not run
Image E	Could not run

TABLE III. PERFORMANCE NUMBER OF SECOND APPROACH

Image	Time taken
Image A	1 m 53 sec
Image B	*2 m 26 sec*
Image C	3m 4 sec
Image D	*4m 51 sec*
Image E	3m 39 sec

We observe that first approach could not run for images that have larger pixels as the Canny Edge Detection algorithm is memory intensive. Since, in case of sequence file whole image is read in a single map task, huge memory is required to process edge detection. Every map task runs inside a single java virtual machine thus, due to limited container memory the images couldn't get processed.

In case of proposed approach the images were split and each region was processed in different map task therefore we were able to complete the experiments. Here, map tasks were running in different JVM and on a different machines therefore the large images also got processed.

While the second approach processes the large sized image but in some cases there could be the possibility that while merging of regions the edges may get distorted.

VI. CONCLUSION & FUTURE SCOPE

As discussed in previous sections, in earlier studies the images were stored in Sequence Files or a custom image reader needs to be implemented therefore for our experiments we considered these two approaches. Our experiments clearly indicates that in case satellite images are stored in Sequence Files the image processing algorithms can work only in case of small sized images. For larger size, a custom image split reader is implemented. Also, Sequence File become a mandatory step before the actually process can begin.

We observe that the memory requirement increases for edge detection as the number of pixels increase. In the distributed approach where the image is split across multiple nodes, the edge detection process was able to complete for larger images. The proposed approach is scalable as per the experimental values.

For future scope we can think of algorithms to decompose the image in mapper and recompose it back in a reducer such that the edges don't get distorted and quality is maintained.

REFERENCES

[1] S. V.-S. M, J. J. Camarero, J. M. Olano, N. Martín-Hernández, M. Peña-Gallardo, . M. Tomás-Burguera, H. Voorhees and T. Poggio, "Detecting textons and texture boundaries in natural images," in International Conference on Computer Vision, 1987

[2] M. Juneja and P. S. Sandhu, "Performance evaluation of edge detection techniques for images in spatial domain," International Journal of Computer Theory and Engineering, vol. 1, no. 5, 2009.

[3] S. K. Katiyar and P. V. Arun, "Comparative analysis of common edge detection techniques in context of object extraction," IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, vol. 50, no. 11, 2014.

[4] B. Kaur and A. Garg, "Mathematical morphological edge detection for remote sensing images," in International Conference on Electronics Computer Technology, 2011.

[5] R. Maini and H. Aggarwal, "Study and comparison of various image edge detection techniques," International Journal of Image Processing, vol. 3, no. 1, pp. 1-12, 2009, vol. 3, no. 1, pp. 1-12, 2009.

[6] J. Xing, R. Sieber and M. Kalacska, "The challenges of image segmentation in big remotely sensed imagery data," Annals of GIS, Taylor & Francis, 2014.

[7] D. Moise and D. Shestakov, "indexing and Searching 100M Images with Map Reduce," in Proceedings of the 3rd ACM conference on International conference on multimedia retrieval, New York, 2013.

[8] Z. Lv, Y. Hu, Z. Haidong, J. Wu, B. Li and H. Zhao, "Parallel K-Means Clustering of Remote Sensing Images Based on MapReduce," in 2010 International Conference on Web Information Systems and Mining, WISM 2010, 2010.

978-1-5090-6785-5/18/$31.00 © 2018 by IEEE

[9] R. Yadav and D. M. C. Padma, "Processing of Large Satellite Images using Hadoop Distributed Technology and Mapreduce : A Case of Edge Detection," International Journal on Recent and Innovation Trends in Computing and Communication, vol. 3, no. 5, 2015.

[10] Huiyu Xia a,b,□, Hassan A. Karimi b, Lingkui Menga , "Parallel implementation of Kaufman's initialization for clustering large remote sensing images on clouds", Computers, Environment and Urban Systems, 2014.

[11] T. Sharma, V.Shokeen, S. Mathur, "Performance Comparison Of Edge Detection Algorithms For Satellite Images Using Bigdata Platform Spark", 2016

[12] Sequence Files, https://wiki.apache.org/hadoop/SequenceFile

[13] Apache hadoop, http://hadoop.apache.org/

[14] AWS EMR, https://aws.amazon.com/emr/

978-1-5090-6785-5/18/$31.00 © 2018 by IEEE