Kalpa Publications in Computing
Volume 2, 2017, Pages 188–195
ICRISET2017. International Conference on Research and Innovations in Science, Engineering &Technology. Selected Papers in Computing

Image Processing in Hadoop Distributed Environment

Mosab Shaheen[1*] and Dr. Madhukar B. Potdar[2†]
[1]Post Graduate School, Gujarat Technological University,
Gandhinagar, Gujarat, India
[2] Bhaskaracharya Institute for Space Applications and Geo-Informatics (BISAG),
Gandhinagar, Gujarat, India
[1] mosab.shaheen@gmail.com, [2]mbpotdar11@gmail.com

Abstract

Satellite images are a growing resource of information and have many applications. In this research, the multispectral satellite images have been subjected to unsupervised classification based on K-Means clustering using Hadoop Framework, which is designed for big data processing, along with Hadoop Image Processing Interface (HIPI). We developed support for the GeoTIFF format, which is usually used for satellite images, and we will show that our methodology enhances the performance.

1 Introduction

Satellite images are becoming more popular and available [9]. This makes traditional platforms unable to handle such big amounts of data [3]. Hence, it comes other frameworks that can work under data and computing intensive tasks like Apache Hadoop framework [12]. Hadoop has shown greater performance when dealing with large size files rather than large number of small files [3] [4] [5] [6] because of the Input Output (IO) time needed for reading and writing these files and due to the extra memory used by NameNode to store metadata about the blocks. HIPI framework solved this issue for images by bundling the images into two files, index and data.

Satellite images can contain many bands like red, green, blue, near infra-red, mid infra-red, etc. For this purpose, one image standard called GeoTIFF has been commonly used; it also allows georeferencing information to be embedded within the TIFF file format. To be able to use the satellite images in HIPI library, we need to add the support for this standard.

Image Processing refers to the operations of pre-processing, enhancement, transformation, filtering, classification, etc. [14]. These can be used for detecting objects like landmines [10],

[*] First Author
[†] Second Author and Guide

classifying [8] [9] & clustering areas, creating maps, discovering pollution areas, monitoring the climate change, and so forth. Classification, supervised or unsupervised, is the most used operation for analyzing remotely-sensed images [8]. The proposed system will segment satellite GeoTIFF files using K-means algorithm on Hadoop and HIPI. The segmented images can be used to predict the future changes in desertification areas, green cover, urban growth and many more.

2 Related Work

2.1 Hadoop as Efficient Framework for Big Data Processing

Bajcsy et al. [2] gave a comparison between some parallel and distributed systems including Hadoop, Terasort, Teragen and Java Remote Method Invocation (RMI). The experiment shows that Hadoop does not give good performance for large number of small size files; on the contrary, when large size files are used, Hadoop outperforms other frameworks.

Yan et al. [3] built an engine based on Hadoop framework using OpenCV library for image processing; also they emphasized that the speed-up is greater for big size files. Moreover, Li et al. [6] showed that the performance of Hadoop on large number of small size files is less than on small number of large size files

2.2 HIPI Outperforms Other Frameworks Regarding Small Size Files Issue of Hadoop

To solve the large number of small size files issue, Sweeney et al. [4] implemented the HIPI library. HIPI creates an image bundle, which is a collection of images grouped in one file. HIPI Image Bundle (HIB) consists of two files the data file and the index file. While Hadoop Archive (HAR) files can be used as archives of files, they may give slower performance due to the technique used for accessing the files inside. Sequence files gives better performance than the standard Hadoop applications. However, sequence files must be read serially and they take considerable time to be generated. On the contrary, HIPI is not restricted to serial reading and it has similar speed to sequence files.

Sozyki et al. [5] showed another framework for image processing called MapReduce Image Processing framework (MIPr). They made a comparison between MIPr, HIPI, and OpenIMAG. It was shown that HIPI gave the best result regarding the time to perform the task.

Li et al. [6] showed that, HAR cannot be changed whenever created and the name cannot contain spaces; and sequence files are serially read. Whereas CombineFileInputFormat is an abstract class and needs implementation. The authors created HMPI library based on HIPI and compared it to HAR and standard Hadoop. The result indicated that HMPI gave the best performance.

2.3 HIPI Lacks Support for GeoTIFF

HIPI only supports JPEG, PNG, and PPM formats [13] so to process satellite images we need to support the GeoTIFF standard.

3 Problem Statement

Apache Hadoop cannot work effectively on large number of small files; rather it works fine on large size files. For this purpose, HIPI library is created; and it shows better performance than MIPr, OpenIMAG and other Hadoop structures. However, HIPI does not support GeoTIFF, which is usually used for satellite images, so we added this support to process these images.

4 Methodology

The input files follow the GeoTIFF standard. GeoTIFF allows storing georeferencing information inside a TIFF file and supports many transformations, which map the spatial location to the pixels of an image. GeoTIFF file is basically a TIFF file, so many images can be included inside the file. This allows storing a pyramid representation of a map

There are two phases in this work: the first one is for bundling the images, and the second one is for extracting and processing them.

4.1 Phase 1: Generating HIPI Image Bundle (HIB)

Figure 1 illustrates this phase; it groups the GeoTiff satellite images into one HIB bundle using the HIPI library and following MapReduce Model. The image dataset will be read using the BundlerInputFormat class. This class is responsible for creating the splits and for providing the record reader. The number of MapTasks usually equal to the number of splits. As we are going to read large number of images we do not want to create a lot of splits, and hence MapTasks, so we are going to use the CombineFileSplit class which can combine many files into one split. The record reader will read the records (files) from the CombineFileSplit and send it to the Mappers. Pairs of < key, value> will be generated, containing the text array of the file paths, and will be passed to the Mappers.

Each Mapper will get the images' paths and detect the file type of them. If it is a valid TIFF file then the TIFFImageUtil class, which uses the GeoTools and Metadata-Extractor libraries, extracts the image header information and stores it in ImageHeader class. If this operation is successful, a temporary Hadoop Image Bundle (HIB) is created. This operation will be repeated for the other files in the text array and will be added to the temporary HIB generated previously.

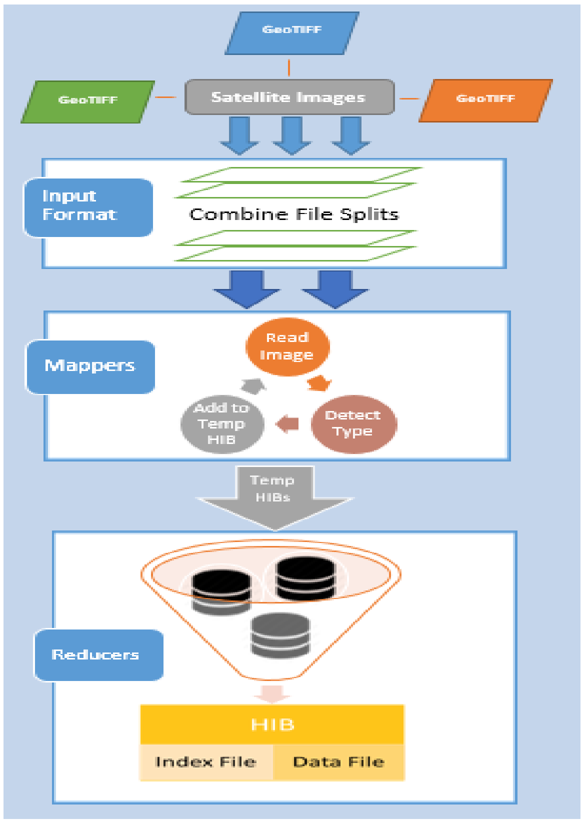

Figure 1: Phase 1: Generating HIPI Image Bundle (HIB)

The result of Mappers is <key, value> pairs containing the temporary HIBs. These pairs will be shuffled, sorted and provided to the Reducers. The reducers will create a final new HIB, which combines all the temporary HIBs, so that all the images now are stored in one HIB bundle. Figure 2 shows pseudocode of phase 1.

```
Program Phase1

Input:
      GeoTIFF Images in directory Dir
Output:
      HIB (data and index files)

Provide Dir as input path to BundlerInputFormat

BundlerInputFormat:
    Get CombineFileSplits of files in Dir
    Set Splits array to returned CombineFileSplits
    Forward Splits to BundlerRecordReader

BundlerRecordReader:
      Construct a new TextArrayWritable object
      Get FilePaths from Splits
      Set TextArrayWritable object to FilePaths
      Forward TextArrayWritable object to BundlerMapper

BundlerMapper:
      Create tempHIB
      For each filePath in TextArrayWritable object Do
            Get ImageHeader object using TIFFImageUtil
            If ImageHeader is valid Then
                  Add the file to tempHIB:
                  - Store length, type, and file bytes in Data File
                  - Store offset of the data in Index File
            End If
      Forward tempHIBs to BundlerReducer

BundlerReducer:
      Create HIB
      For each tempHIB in TempHIBs Do
            Add tempHIB to HIB
            Delete tempHIB Data File
            Delete tempHIB Index File
      End For
```

Figure 2: Phase 1: pseudocode corresponding to flow chart depicted in Figure 1

4.2 Phase 2: Clustering Using HIPI and K-means

Now the resulting HIB is split into CombineFileSplit objects. The TIFFImageUtil class will take the data, obtained from the splits, and decode it into ImageHeader and TIFFFloatImage objects. The TIFFFloatImage class contains pixels of an image, longitude and latitude information, and information about the image like width, height, and number of bands. After that, the TIFFFloatImage object will be processed by a MapTask which gets the pixels from the object, apply the k-means algorithm on them, and generate segments representing different covers or areas in the image. The cover types can be water, soil, streets, forests, building, etc.

The result of Mappers is <key, value> pairs containing the clustered pixels. They will be forwarded to reducers. The reducers in turn will save the clustered pixels in RGB format in the HDFS. These reducers are useful to minimize the time and overhead on the Mappers to do the output operations in HDFS. Pseudocode of phase 2 is provided in Figure 3 while figure 4 shows the whole procedure of clustering.

```
Program Phase2

Input:
       HIB bundle
Output:
       Segmented Image Files

Provide HIB as input to ProcessInputFormat
ProcessInputFormat:
    Get CombineFileSplits of files in HIB
    Set Splits array to returned CombineFileSplits
    Forward Splits to ProcessRecordReader

ProcessRecordReader:
      Get Paths from Splits
      For each path in Paths Do
            Extract ImageHeader object using TIFFImageUtil
            Extract TIFFFloatImage object using TIFFImageUtil
            Forward ImageHeader and TIFFFloatImage object to
                    ProcessMapper
      End For

ProcessMapper:
      Get Pixels array from TIFFFloatImage
      Apply K-means on the Pixels:
          Initialize and Set Points list
          Initialize and Set Clusters list
          Initialize and Set Clusters's Centroids
          Set Boolean finish to false
          While (not finish) Do
               Assign LastCentroids to Cluster's Centroids
               For each point from Points Do
                   For each centroid from Cluster's Centroids Do
                         Set distance to distance (point, centroid)
                         If (distance decreased) Then
                               Update selected cluster of point
                         End IF
                   End For
                   Add point to selected cluster
              End For
              Update Cluster's Centroids
              If (distance (LastCentroids, Cluster's Centroids) ==0)
               Then
                  Set finish to true
              End If
          End While
      Set Segments to Clusters Points
      Forward Segments to ProcessReducer
```

```
ProcessReducer:
      Create WritableRaster object using Segments
      Create BufferedImage object using WritableRaster
      Save BufferedImage in RGB image file.
```

Figure 3: Phase 2: pseudocode corresponding to flow chart depicted in Figure 4

5 Results

We applied clustering on the BISAG Dataset. The dataset is a collection of GeoTIFF images each one around 2 MB in size. Clustering was done using the previous methodology, which is compared to clustering without using HIPI. The experiment is applied repeatedly, each time with different number of images to see the changes in performance when the data size grows. The work is done by creating a Virtual Machine (VM) holding Ubuntu OS and Hadoop Standalone Installation with 4 MapTasks and 1 ReduceTask. We dedicated 3 CPUs and 8GB of RAM for the VM, whereas the hosting OS is a workstation that has 8CPUs and 16GB RAM. Table 1 shows the time taken, to execute the task, in seconds:

Number of Images	With HIPI	Without HIPI	Diffe-rence
100 (200mb)	586	604	18
200 (400mb)	1143	1171	28
300 (600mb)	1812	1863	51
400 (800mb)	2304	2347	43
500 (1gb)	3102	3140	38

Table 1: Results of clustering

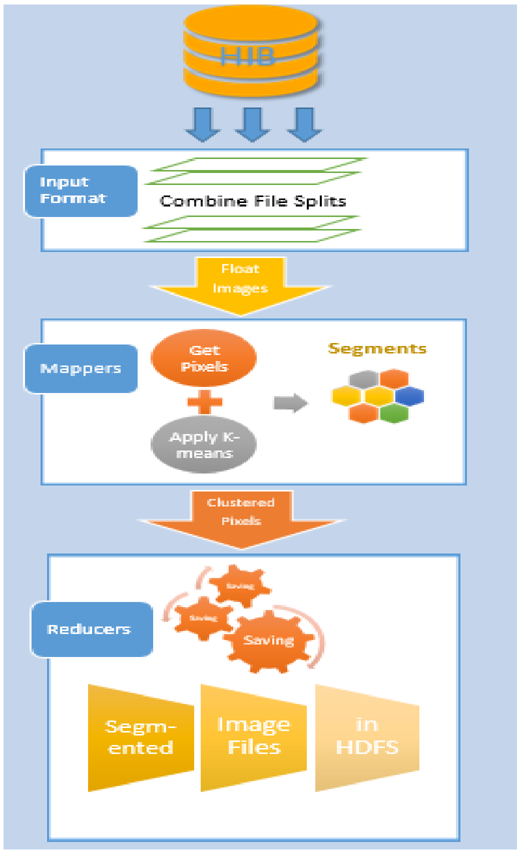

Figure 4: Phase 2: Clustering Using HIPI and K-means

The experiment shows that HIPI gives better performance than the clustering without using it. There is no big difference in this experiment between both ways because of the task nature, which is computing intensive rather than data intensive, also because we gave small resources, i.e. RAM and CPU, small number of Map & Reduce Tasks due to the standalone installation of Hadoop which is very essential factor to increase the speed, and small size of dataset. However, it is clear that there is a difference especially for average data load per node, which is here around 300 images of 600 MB in size. Besides that, in the future we are going to run the test on multinode Hadoop cluster and reduce these limitations to get more realistic results. One sample output of a segmented image is shown in Figure 5.

6 Conclusion

Hadoop is a powerful framework to process big amounts of satellite image data. In addition, using HIPI library with Hadoop environment can improve the performance and make the work more efficient, because Hadoop cannot work efficiently with large number of small files. After supporting GeoTIFF we are able to process the satellite images and the results show that HIPI contributes in enhancing the performance of Hadoop. In the future, we are going to create a multimode Hadoop cluster and run the experiment using bigger dataset and more resources.

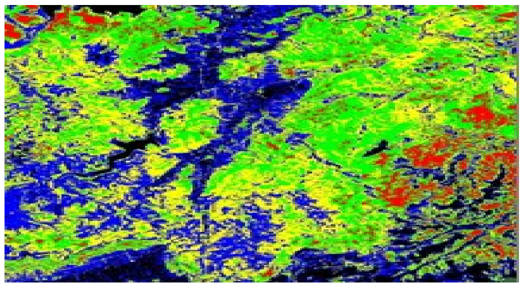
Figure 5: Sample output of a segmented image

7 Acknowledgment

We are thankful to Shri T. P. Singh - Director at BISAG for supporting this research, to Mr. Bhadreshsinh Gohil for encouragement, and to Mr. Miren Karamta - System Manager, HPC system at BISAG for providing required infrastructure.

References

[1] R. M. Esteves, R. Pais, and C. Rong, "K-means clustering in the cloud -- a Mahout test" (WAINA), IEEE Workshops of International Conference on Advanced Information Networking and Applications. Biopolis, 2011, pp. 514-519.

[2] P. Bajcsy, A. Vandecreme, J. Amelot, P. Nguyen, J. Chalfoun and M. Brady, "Terabyte-sized image computations on Hadoop cluster platforms" IEEE International Conference on Big Data. 2013, Silicon Valley, CA, 2013, pp. 729-737.

[3] Yuzhong Yan, and Lei Huang, "Large-scale image processing research cloud" IARIA The Fifth International Conference on Cloud Computing, GRIDs, and Virtualization. May 25 - 29, 2014 - Venice, Italy.

[4] Chris Sweeney, Liu Liu, Sean Arietta, and Jason Lawrence. "HIPI: a Hadoop image processing interface for image-based mapreduce tasks" University of Virginia Undergraduate Thesis. 2011.

[5] Andrey Sozykin, and Timofei Epanchintsev. "MIPr – a framework for distributed image processing using Hadoop" IEEE 9th International Conference on Application of Information and Communication Technologies (AICT), 2015

[6] Jia Li, Kunhui Lin, and Jingjin Wang V. Vazhayil, "Map-Reduce based framework for instrument detection in large-scale surgical videos" IEEE International Conference on Control Communication & Computing India (ICCC). 2015, Trivandrum, 2015, pp. 606-611.

[8] I. Chebbi, W. Boulila, and I. R. Farah, "Improvement of satellite image classification: approach based on Hadoop/Map Reduce" IEEE 2nd International Conference on Advanced Technologies for Signal and Image Processing - ATSIP. 2016 March 21-24, 2016, Monastir, Tunisia

[9] Noel C. F. Codella, Gang Hua, Apostol Natsev, and John R. Smith, "Towards large scale land-cover recognition of satellite images" IEEE 8th International Conference on Information, Communications and Signal Processing (ICICS), 2011

[10] Sahar El-Kazzaz and Ahmed El-Mahdy, "A Hadoop-based framework for large-scale landmine detection using ubiquitous big satellite imaging data" IEEE 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. 2015.

[11] Sridhar Vemula, and Christopher Crick. "Hadoop image processing framework" IEEE International Congress on Big Data. 2015.

"Design of the mass multimedia files storage architecture based on Hadoop" IEEE 8th International Conference on Computer Science & Education (ICCSE). April 26-28, 2013. Colombo, Sri Lanka.

[7] B. C. Sunny, Ramesh R, A. Varghese and

[12] The Apache Software Foundation, "What is Apache Hadoop?", 2016, hadoop.apache.org (accessed on Dec 5, 2016).

[13] University of Virginia, "HIPI Hadoop Image Processing Interface", 2016, hipi.cs.virginia.edu (accessed on Dec 5, 2016).

[14] Paul M. Mather. In "Computer processing of remotely-sensed images: an introduction, 3rd edition"; John Wiley & Sons, Inc. December 2005.