International Journal of Advanced Engineering & Innovative Technology,
Volume 1, Issue 1, April-2014, 19- 23 ISSN: 2348-7208

Recognization of Satellite Images of Large Scale Data Based On Map-Reduce Framework

Vidya Dhondiba Jadhav, Harshada Jayant Nazirkar, Sneha Manik Idekar

Dept. of Information Technology, JSPM's BSIOTR (W), Wagholi, Pune, Maharastra.

Guide – Prof. P.A.Bandgar, Head, Dept. of Information Technology

For correspondence: vidyadjadhav@gmail.com;harshada.nazirkar@gmail.com; smidekar@gmail.com

Abstract:

Today in the world of cloud and grid computing integration of data from heterogeneous databases is inevitable. This will become complex when size of the database is very large. M-R is a new framework specifically designed for processing huge datasets on distributed sources. Apache's Hadoop is an implementation of M-R. Currently Hadoop has been applied successfully for file based datasets.

This project proposes to utilize the parallel and distributed processing capability of Hadoop M-R for handling Images on large datasets. The presented methodology of land-cover recognition provides a scalable solution for automatic satellite imagery analysis, especially when GIS data is not readily available, or surface change may occur due to catastrophic events such as flooding, hurricane, and snow storm, etc. Here, we are using algorithms such as Image Differentiation, Image Duplication, Zoom-In, Gray-Scale.

Index Terms: Map-Reduce (M-R), HDFS(Hadoop Distributed File System) ,HIPI(Hadoop Image Processing Interface)

I. INTRODUCTION

Apache Hadoop is an open-source software framework for storage and large scale processing of data-sets on clusters. Hadoop cluster is a set of commodity machines networked together in one location. Data storage and processing all occur within this cluster of machines. Different users can submit computing "jobs" to Hadoop from individual clients, which can be their own desktop machines in remote locations from the Hadoop cluster. A small Hadoop cluster includes a single master and multiple worker nodes. The master node consists of a Job Tracker, Task Tracker, NameNode and Data Node shown in Fig 1.

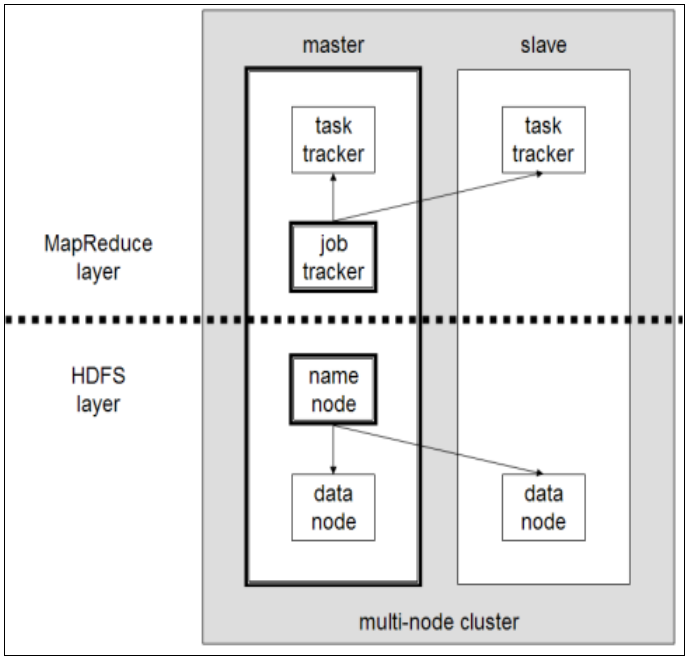


Fig 1. A multi-node Hadoop cluster.

A slave or worker node acts as both a Data Node and Task Tracker, though it is possible to have data-only worker nodes and compute-only worker nodes. In a larger cluster, the HDFS is managed through a dedicated Name Node server to host the file system index, and a secondary Name Node

that can generate snapshots of the name node's memory structures, thus preventing file-system corruption and reducing loss of data. Similarly, a standalone Job Tracker server can manage job scheduling. In clusters where the Hadoop Map Reduce engine is deployed against an alternate file system, the Name Node, secondary Name Node and Data Node architecture of HDFS is replaced by the file-system-specific equivalent.

II. PROPOSED SYSTEM

In a Hadoop cluster, data is distributed to all the nodes of the cluster as it is being loaded in. The HDFS will split large data files into chunks which are managed by different nodes in the cluster. In addition to this each chunk is replicated across several machines, so that a single machine failure does not result in any data being unavailable. An active monitoring system then re-replicates the data in response to system failures which can result in partial storage. Even though the file chunks are replicated and distributed across several machines, they form a single namespace, so their contents are universally accessible.

IV. SYSTEM ARCHITECTURE

The system architecture includes the following- 1.Large no. of images stored in file system.2.This Bundle of images is fed to hadoop distributed file system.3.On HDFS , we execute set of operations like duplicate image removal , zoom in and find differences among Images, using M-R Programs.4.The Result is then uploaded in web server, and shown to user through web application.

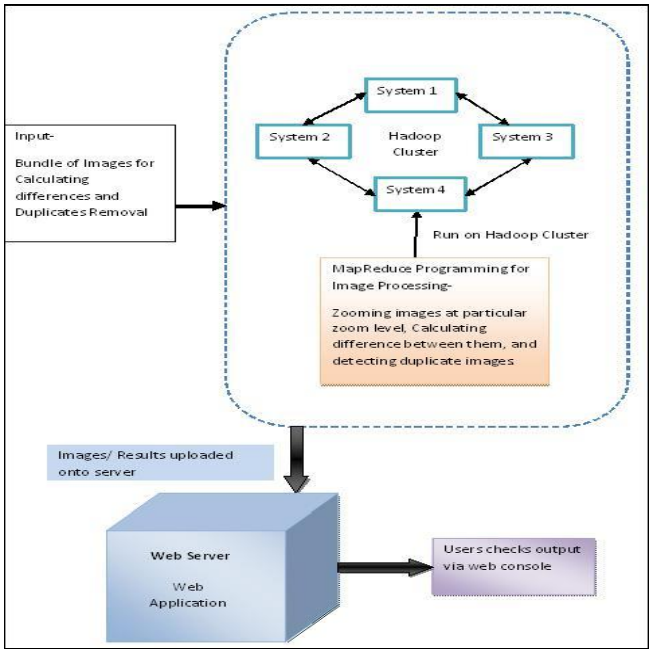


Fig.2. System architecture

V. ALGORITHMS USED

A. Zoom in Algorithm

This algorithm takes the original image, creates four image tiles out of it, that means splits the original image into four pieces, re-draws each of the part on each of the four new images. Size of each of the new image is equal to the original size. For eg. Original image = 100*100 size, Split it into four parts of 50*50 size each next we re-draw these parts into 100*100 size images. Hence, we get four 100*100size images from original image of 100*100size.

B. Difference Algorithm

Here we divide the images into small chunks. We compare the respective chunks of image one and image two.

Comparison process:

1. Compare the intensity of the chunks
2. Compare the colour codes of chunks

if chunks are different then mark the chunks with a red box.

Again repeat the comparison process for all the chunks. And draw a new image with the red boxes marked i.e. showing the differences. Upload the difference image on tomcat server.

C. Duplication Algorithm

In this algorithm, we divide the images into small chunks. We compare the respective chunks of images. We have a sequence file with all the files of a binary data and it is the actual job that will filter & find the duplicates.

D. Grayscale Algorithm

A grayscale image is simply one in which the only colours are shades of gray. The reason for differentiating such images from any other sort of colour image is that less information needs to be provided for each pixel. In fact a `gray' color is one in which the red, green and blue components all have equal intensity in RGB space, and so it is only necessary to specify a single intensity value for each pixel, as opposed to the three intensities needed to specify each pixel in a full colour image. Often, the grayscale intensity is stored as an 8-bit integer giving 256 possible different shades of gray from black to white.

VI. TECHNOLOGY USED

- M-R Framework

Map Reduce is also a data processing model. Its greatest advantage is the easy scaling of data processing over multiple computing nodes. Under the Map Reduce model, the data processing primitives are called *mappers* and *reducers.* Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once you write an application in the Map Reduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to the Map Reduce model.

- Pseudo-code for map and reduce functions

```
map(String filename, String document)
{
        List<String> T = tokenize(document);
         for each token in T
{
 emit ((String)token, (Integer) 1);
          }
 }
reduce(String token, List<Integer> values)
{
        Integer sum = 0;
        Understanding MapReduce 13
          for each value in values
        {
        sum = sum + value;
        }    emit ((String) token, (Integer) sum);
}
```

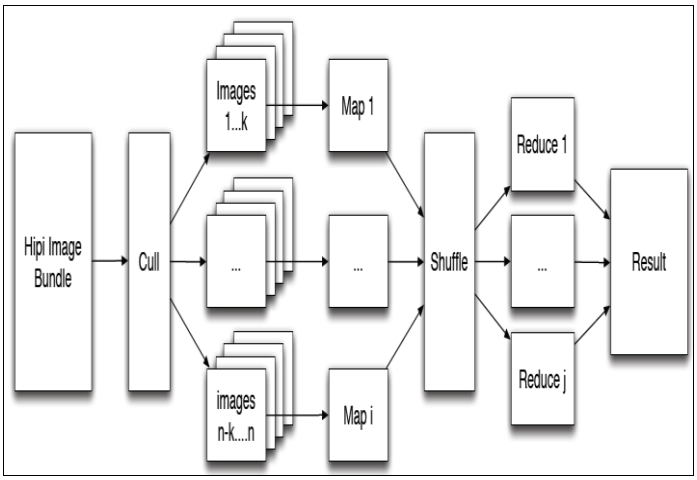
- *HDFS (Hadoop Distributed File System)*

The Hadoop Distributed File System (HDFS) is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. In a large cluster, thousands of servers both host directly attached storage and execute user application tasks. By distributing storage and computation across many servers, the resource can grow with demand while remaining economical at every size. We describe the architecture of HDFS and report on experience using HDFS to manage 40 petabytes of enterprise data at Yahoo! A high-end machine with four I/O

channels each having a throughput of 100 MB/sec will require three hours to *read* a 4 TB data set! With Hadoop, this same data set will be divided into smaller (typically 64 MB) blocks that are spread among many machines in the cluster via the Hadoop Distributed File System (HDFS). With a modest degree of replication, the cluster machines can read the data set in parallel and provide a much higher throughput. And such a cluster of commodity machines turns out to be cheaper than one high-end server!

- *The HIPI Framework*

HIPI is a library for Hadoop's Map Reduce framework that provides an API for performing image processing tasks in a distributed computing environment. Here is an overview of our system:



HIPI was created to empower researchers and present them with a capable tool that would enable research involving image processing and vision to be performed extremely easily. We used HIPI because the following points.

1. Provide an open, extendible library for image processing and computer vision applications in a Map Reduce framework.

2. Store images efficiently for use in Map Reduce applications

3. Allow for simple filtering of a set of images

4. Present users with an intuitive interface for image-based operations and hide the details of the Map Reduce framework

5. HIPI will set up applications so that they are highly parallelized and balanced so that users do not have to worry about such details.

VII. CONCLUSION

This work has demonstrated the feasibility of massively parallel semantic classification of satellite imagery. The complex details of Hadoop's powerful Map Reduce framework and bring to the forefront what users care about most: images. Our system has been created with the intent to operate on large sets of images. We give the user a simple way to filter image sets and control the types of images being used in their Map Reduce tasks. In Map Reduce programming to apply set of operation like duplication, zoom-in, difference removal, Grayscale conversion on particular satellite image.

VIII. ACKNOWLEDGEMENT

We take this opportunity to thank our project guide and Head of the Department Prof. P.A.Bandgar. for their valuable guidance and mentoring throughout this project. Additionally, we must give great thanks to Miss. Sneha. and Mr. Bhagat. for guidance and support.

REFERENCES

[1] T. White, Hadoop: The Definitive Guide, 2nd ed. O'Reilly Media / Yahoo Press, 2010.

[2] APACHE, 2010. Hadoop mapreduce framework. http://hadoop.apache.org/mapreduce/.

[3] J. Dean and S. Ghemawat, Mapreduce: Simplified data processing on large clusters, Communications of the ACM, vol. 51, no. 1, pp. 107-113, Jan. 2008.

[4] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang, Mars: A mapreduce framework on graphics processors, in Proc. of the 17th International Conference on Parallel Architectures and Compilation Techniques, New York, NY, USA: ACM, 2008, pp. 260-269.

[5] CONNER, J. 2009. Customizing input file formats for image processing in hadoop. Arizona State University. Online at: http://hpc. asu. edu/node/97.