SCALABLE IMAGE PROCESSING THROUGH DISTRIBUTED CLOUD ENVIRONMENT

Dr Dayananda R B[1], Sreenivas T[2]
[1]GSSIETW, Mysuru, Karnataka
[2]SJCE, Mysuru, Karnataka

Abstract

With the rapid growth of social media, the number of images being uploaded to the internet is exploding. Massive quantities of images are shared through multi-platform services such as Snapchat, Instagram, Facebook and WhatsApp. Recent studies estimate that over 1.8 billion photos are uploaded every day. However, for the most part, applications that make use of this vast data have yet to emerge. Most current image processing applications, designed for small-scale, local computation, do not scale well to web-sized problems with their large requirements for computational resources and storage. The emergence of processing frameworks such as the Hadoop-MapReduce platform addresses the problem of providing a system for computationally intensive data processing and distributed storage.

Keywords: Image Processing, Parallel and distributed processing, Mapreduce, Hadoop, HIPI.

I. INTRODUCTION

With the rapid growth of social media, the number of images being uploaded to the internet is exploding. Massive quantities of images are shared through multi-platform services such as Snapchat, Instagram, Facebook and WhatsApp. Recent studies estimate that over 1.8 billion photos are uploaded every day. However, for the most part, applications that make use of this vast data have yet to emerge. Most current image processing applications, designed for small-scale, local computation, do not scale well to web-sized problems with their large requirements for computational resources and storage. The emergence of processing frameworks such as the Hadoop-MapReduce platform addresses the problem of providing a system for computationally intensive data processing and distributed storage. However, to learn the technical complexities of developing useful applications using Hadoop requires a large investment of time and experience on the part of the developer. As such, the pool of researchers and programmers with the varied skills to develop applications that can use large sets of images has been limited. When considering operations such as face detection, image classification and other types of processing on images, there are limits on what can be done to improve performance of single computers to make them able to process information at the scale of social media. Therefore, the advantages of parallel distributed processing of a large image dataset by using the computational resources of a cloud computing environment should be considered. In addition, if computational resources can be secured easily and relatively inexpensively, then cloud computing is suitable for handling large image data sets at very low cost and increased performance. In this information world there is enormous amount of data flow between various Medias at very high rate. This data is categorized as "Big Data". Most of this generated data are images and videos.

Big data analysis requires extensible computing power and practiced data mining statistics, machine learning, and pattern recognition capabilities. It is exaggerative in image processing domain since the video and image processing algorithms become more and more

complicated, which demands even more power in computation. Some of these image processing requires even real-time processing capability. It is essential to create a domain specific cloud for image processing research in order think these challenging requirements. To fill the gap between more complicated modern architectures and highly emerging image processing algorithms for big data, our image processing with cloud project aims to produce a high-productivity and high-performance image processing research environment integrated within a cloud computing infrastructure. This proposal aims to combine cloud computing and big data analysis technology to provide image processing cloud infrastructure and big data processing engine to satisfy the needs. Thus it is a great challenge not only to store and manage the large volume of data, but also provide solution to improve performance and scalability.

II. RELATED WORK

With the rapid usage increase of online photo storage and social media on sites like Facebook, Twitter and Picasa, more image data is available than ever before, and is growing every day. Every minute 27,800 photos are uploaded to Instagram, while Facebook receives 208,300 photos over the same time frame. This alone provides a source of image data that can scale into the billions. The explosion of available images on social media has motivated image processing research and application development that can take advantage of very large image data stores. Existing case study mostly rely on classifying and clustering billions of regular images using MapReduce. It describes an image pre-processing technique for use in a sliding-window approach for object recognition. Some of the limitations of the MapReduce model when dealing with high-speed video encoding, namely its dependence on the NameNode as a single point of failure, and the difficulties inherent in generalizing the framework to suit particular issues. It proposes an alternate optimized implementation for providing cloud-based IaaS (Infrastructure as a Service) solutions. Lv et.al [1] describes using the k-means algorithm in conjunction with MapReduce and satellite/aerial photographs in order to find different elements based on their color. Zhang et.al [2] presents methods used for processing sequences of microscope images of live cells. The images are relatively small (512x512, 16-bit pixels) stored in 90 MB folders, the authors encountered difficulties regarding fitting into Hadoop DFS blocks which were solved by custom Input-Format, Input-Split and Record-Reader classes. Powell et.al [3] describes how NASA handles image processing of celestial images captured by the Mars Orbiter and rovers. Clear and concise descriptions are provided about the segmentation of giga-pixel images into tiles, how the tiles are processed and how the image processing framework handles scaling and works with the distributed processing. Wang,Yinhai and McCleary[4] discuss speeding up the analysis of tissue microarray images by substituting human expert analysis for automated processing algorithms. While the images were gigapixel-sized, the content was easily segmented and there was no need to analyze all of an image at once. The work was all done on a specially-built high performance computing platform using the Hadoop framework.

Hossein Kardan Moghaddam proposed MapReduce as a distributed data processing model using open source Hadoop framework for manipulating large volume of data. Muneto Yamamoto et al. [5] suggested methods of parallel image database processing with mapreduce and Hadoop streaming.

III. PARALLEL APPROACHES FOR DATA PROCESSING

A. MAP REDUCE

Map reduce is a framework for distributed parallel processing of large image database [8]..Map reduce model is having many different variation with different technology and framework .Google ,Apache Hadoop ,HIPI, Microsoft SCOPE, Apache Pig, and Apache Hive all these have their own customized map reduce implementation.

1) Hadoop Mapreduce: System Architecture

Hadoop is an open source, distributed, scalable java based implementation which follows Google's MapReduce concept [9.] Hadoop is framework which is having its own distributed file storage system which is Hadoop Distributed File System (HDFS) and its own computational paradigm known as Map reduce.

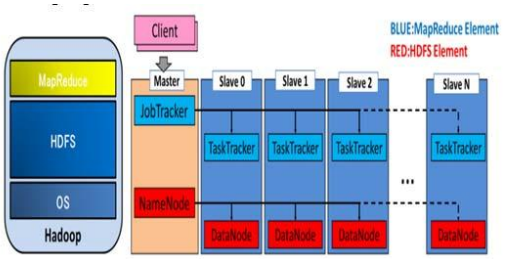

Fig.1 Hadoop MapReduce Paradigm [8]

While processing Data through Hadoop Input and output is always given through HDFS Mapreduce is having two main elements namely JobTracker and TaskTracker and Two functions namely Map and Reduce. HDFS is having 2 main elements namely Name node and Data node

A) JobTracker manage resources of distributed system and manage job scheduling [8].
B) TaskTracker accepts task and returns the results after executing tasks received by JobTracker.
C) Name node is a master server that manages the namespace and access to files by client's Name and executes file operations, such as opening, closing, and renaming files and directories. It also determines the mapping of blocks to Data Nodes [8].
D) DataNodes manage the storage that is attached to the nodes on which they run and perform block creation, deletion, and replication[8] .It is a place where execution of task take place.
E) Secondary NameNode is a helper to the primary NameNode responsible for supporting periodic checkpoints of the HDFS metadata[8].It is especially useful in case of primary name node failure
F) Map task take multiple input key- value pairs <k,v> and generate multiple <k',v'> intermediate pairs.

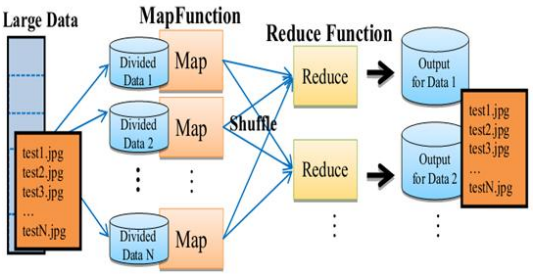

Fig 2 working of Map and reduce phase [8]

G) Reduce phase take list of input<k', list v'> and give final summarized output.
H) After Map task shuffle task is performed on intermediate values to efficiently aggregate different pairs and to save network bandwidth.

2) HADOOP IMAGE PROCESSING INTERFACE (HIPI)

HIPI is an image processing library designed to be used with the Apache Hadoop Mapreduce parallel programming framework [5]. HIPI facilitates efficient and high-throughput image processing with MapReduce style parallel programs typically executed on a cluster . It is flexible enough to withstand continual changes and improvements within Hadoop's Mapreduce system. The goal of HIPI is to create a tool that will make development of large-scale image processing and vision projects extremely accessible .

Primary objective of HIPI are as below:

1) Provide an open source framework over Hadoop MapReduce for developing large-scale image applications [5].

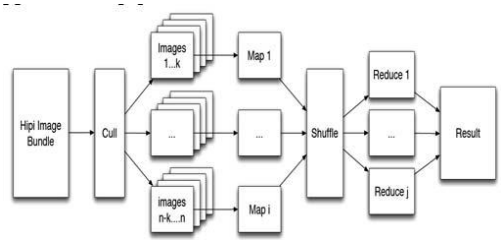

Fig 3 Organization of Mapreduce in HIPI

2) Provide the ability to flexibly store images in various Hadoop file formats .
3) Provide interoperability between various image processing libraries [5].
4) Store images efficiently for use in MapReduce applications and simple filtering of a set of images .
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 .

Working of HIPI in MapReduce environment is as follow:

1) Input to the HIPI program is given in the form of HIPI Image Bundle (HIB).HIB is collection of images in variety of file format which is stored as a single file on the HDFS .
2) HIB is given to culling phase, which is new in HIPI. Main goal of culling step is

to filter the images in a HIB based on a variety of user-defined conditions like spatial resolution or criteria related to the image metadata. This functionality is achieved through the CullMapper class .

3) Images survive from cull step are given to map function to generate intermediate key value pairs .
4) Mapping output is shuffled to minimize network bandwidth usage and pre aggregate key value pairs .
5) Reduce phase will generate summarized data in the form of one key with multiple values pair .

The Hadoop Image Processing Framework is intended to provide users with an accessible, easy-to-use tool for developing large-scale image processing applications.

The main goals of the Hadoop Image Processing Framework are:

- Provide an open source framework over Hadoop MapReduce for developing large-scale image applications.
- Provide the ability to flexibly store images in various Hadoop file formats.
- Present users with an intuitive application programming interface for image-based operations which is highly parallelized and balanced, but which hides the technical details of Hadoop MapReduce.
- Allow interoperability between various image processing libraries.

Downloading and storing image data

Step 1: Input a URL List.

Initially users input a file containing URLs of images to download. The input list should be a text file with one image URL per line. The list can be generated by hand, extracted from a database or a provided by a search. The framework provides an extendable ripper module for extracting URLs from Flickr and Google image searches and from SQL databases. In addition to the list the user selects the type of image bundle to be generated (e.g. HAR, sequence or map). System divides the URLs for download across the available processing nodes for maximum efficiency and parallelism. The URL list is split into several map tasks of equal size across the nodes. Each node map task generates several image bundles appropriate to the selected input list, containing all of the image URLs to download. In the reduce phase, the Reducer will merge these image bundles into a large image bundle.

Step 2: Split URLs across nodes.

From the input file containing the list of image URLs and the type of file to be generated, the task of downloading images across the all the nodes in the cluster is equally distributed. The nodes are efficiently managed so that no memory overflow can occur even for terabytes of images downloaded in a single map task. This allows maximum downloading parallelization. Image URLs are distributed among all available processing nodes, and each map task begins downloading its respective image set.

Step 3: Download image data from URLs.

For every URL retrieved in the map task, a connection is established according to the appropriate transfer protocol (e.g. FTP, HTTP,HTTPS, etc.). Once connected, the file type is checked. Valid images are assigned to InputStreams associated with the connection. From these InputStreams, new HImage objects are generated and the images to the image bundle are added. The HImage class holds the image data and provides an interface for the user's manipulation of image and image header data. The HImage class also provides interoperability between various image data types (e.g. BufferedImage, Mat, etc.).

Step 4: Store images in an image bundle.

Once an HImage object is received, it can be added to the image bundle simply by passing the HImage object to the append Image method. Each map task generates a number of image bundles depending on the image list. In the reduce phase, all of these image bundles are merged into one large bundle.

Processing image bundle using MapReduce.

Hadoop MapReduce program handles input and output data very efficiently, but their native data exchange formats are not convenient for representing or manipulating image data. For instance, distributing images across map nodes require the translation of images into strings, then later decoding these image strings into specified formats in order to access pixel information. This is both inefficient and inconvenient .To overcome this problem, images should be represented in as many different formats as possible, increasing flexibility. The framework focuses on bringing familiar data types directly to user. As

distribution is important in MapReduce, images should be processed in the same machine where the bundle block resides. In a generic MapReduce system, the user is responsible for creating InputFormat and RecordReader classes to specify the MapReduce job and distribute the input among nodes.

The functionality of the framework's Processor module is described below:

Step 1: Devise the algorithm.

We assume that the user writes an algorithm which extends the provided Generic Algorithm class. This class is passed as an argument to the processor module. The framework starts a MapReduce job with the algorithm as an input. The Generic Algorithm holds an HImage variable, this allows user to write an algorithm on a single image data structure, which the framework then iterates over the entire image bundle. In addition to the algorithm, the user should provide the image bundle file that needs to be processed. Depending on the specifics of the image bundle organization and contents, the bundle is divided across nodes as individual map tasks. Each map task will apply the processing algorithm to each local image and append them to the output image bundle. In the reduce phase, the Reducer merges these image bundles into a large image bundle.

Step 2: Split image bundle across nodes.

The input image bundle is stored as blocks in the HDFS. In order to obtain maximum throughput, the framework establishes each map task to run in the same block where it resides, using custom input format and record reader classes. This allows maximum parallelization without the problem of transferring data across nodes. Each image bundle now applies different map tasks to the image data for which it is responsible.

Step 3: Process individual image.

The processing algorithm devised by the user and provided as input to the Processing Module is applied to every HImage retrieved in the map task. The HImage provides its image data in the data format (e.g. Java BufferedImage, OpenCV Mat, etc.) requested by the user and used by the processing algorithm. Once the image data type is retrieved, processing takes place. After processing, the preserved image header data from the original image is appended to the processed image. The processed image is appended to the temporary bundle generated by the map task.

Step 4: Store processed images in an image bundle.

Every map task generates an image bundle upon completion of its processing. Once the map phase is completed there are many bundles scattered across the computing cluster. In the reduce phase, all of these temporary image bundles are merged into a single large file which contains all the processed images.

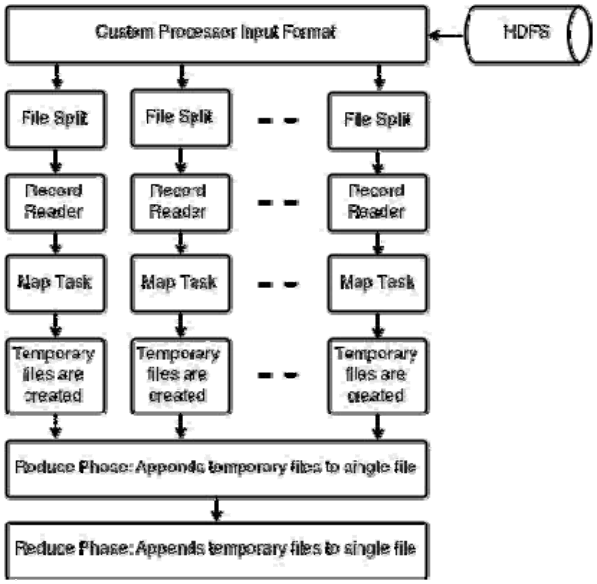

Fig. 2. Single node running the Downloader Module (handled by the framework and transparent to the user

CONCLUSION

Map reduce parallel programming model provide high scalability, reliability, fault tolerance in distributed environment. It provides sequential execution of map and reduce task. In this paper we discussed Hadoop and HIPI's map reduce implementation especially for image processing and computer graphics. The proposal aims to combine cloud computing and big data analysis technology to provide image processing cloud infrastructure and big data processing engine to satisfy the needs of not only limited for storing and managing the large volume of data, but also provide solution to improve performance and scalability.

REFERENCES

[1] Zhong, J. Wu, B. Li, and H. Zhao,"Parallel k-means clustering of remote sensing images based on mapreduce," in Proceedings of the 2010 International

Conference

[2] C. Zhang, H. De Sterck, A. Aboulnaga, H. Djambazian, and R. Sladek, “Case study of scientific data processing on a cloud using hadoop,” in High Performance Computing Systems and Applications, ser. Lecture Notes in Computer Science, D. Mewhort, N. Cann, G. Slater, and T. Naughton, Eds. Springer Berlin Heidelberg, 2010, vol. 5976, pp.400–415.

[3] M. Powell, R. Rossi, and K. Shams, “A scalable image processing framework for gigapixel mars and other celestial body images,” in Aerospace Conference, 2010 IEEE, March 2010, pp. 1–11.

[4] Y. Wang, D. McCleary, C.-W. Wang, P. Kelly, J. James,D. Fennell, and P. Hamilton, “Ultra-fast processing of gigapixel tissue microarray images using high performance computing,” Cellular Oncology, vol. 34, no. 5, pp. 495–507, 2011. [Online]. Available: http://dx.doi.org/10.1007/s13402-011-0046-4.

[5] Yamamoto, Muneto, and Kunihiko Kaneko, "Parallel image database processing with MapReduce and performance evaluation, in pseudo distributed mode." International Journal of Electronic Commerce Studies 3, no. 2 (2013): 211-228.

[6] Ryu, Chungmo, Daecheol Lee, Minwook Jang, Cheolgi Kim, and Euiseong Seo, "Extensible video processing framework in apache hadoop." ,In Cloud Computing Technology and Science (CloudCom), 2013 ,IEEE 5th International Conference on, vol. 2, pp. 305-310. IEEE,2013.

[7] Tan, Hanlin, and Lidong Chen,"An approach for fast and parallel video processing on Apache Hadoop clusters." In Multimedia and Expo (ICME), 2014 IEEE International Conference on, pp. 1-6. IEEE, 2014.

[8] Banaei, Seyyed Mojtaba, and Hossein Kardan Mogha ddam, "Hadoop and Its Role in Modern Image Processing.", Open Journal of Marine Science 4, no. 04 (2014): 239.

[9] Zhang, Bingjing, Judy Qiu, Stefan Lee, and David Crandall, "Large-Scale Image Classification using High Performance Clustering."

[10] Giachetta, Roberto, "A framework for processing large scale geospatial and remote sensing data in MapReduce environment." ,Computers & Graphics (2015).