Distributed Face Recognition Using Hadoop

A. Thorat, V. Malhotra, S. Narvekar and A. Joshi

Dept. of Computer Engineering and IT College of Engineering, Pune
{abhishekthorat02@gmail.com, vinayak.malhotra20@gmail.com, sidnarvekar123@gmail.com, adj.comp@coep.ac.in}

Abstract. This paper introduces a scalable and flexible solution for face recognition using distributed Hadoop cluster. With the increase of use of CCTV for security purposes there is a need of efficient solution to utilize the vast data generated by them .In case of an investigation it is ideally required that the evidences are evaluated as fast as possible. Usually a week's recording of a CCTV footage is approximately 100 GB. Hence in this proposed solution we use avconv, OpenCV and Hadoop MapReduce for fast distributed video processing. We have experimented and analyzed the performance of facial recognition on different models and distributed configuration. We found Hadoop as an effective framework for video processing to recognize face.

Keywords: ***MapReduce, Face Recognition, Distributed video analysis, Parallel computing.***

1 Introduction

Over the years use of digital surveillance system has increased manifold. This increase has risen the need of fast processing of large video data. As stated by Seagate, it takes 3TB storage to store 36 days of surveillance video (quality: 1280 X 1024 at 20fps). If any incident occurs for authorities to take appropriate action quickly, then they need to gather information and evidence.

To extract meaningful information from large video data is booth storage and processing intensive. Spy agencies in India need constant scanning of man surveillance video to check for suspected terror or theft activities. Hence to achieve face recognition on large video set we need distributed environment to parallely process huge video data sets. We here in this project use Apache Hadoop, a well- known open source framework to process video for face recognition in distributed manner. The job of face recognition is run on each node using Map Reduce.

We used OpenCV, an open source tool mainly written in C, C++ to detect and recognize face. All the processing related to face recognition is done on each of the nodes present in the cluster. The output present at each node consists of a time stamp, which provides the information of when was the requested person seen in the provided video. Later, all the output files from each of the nodes is combined into a single output and returned to user. We have used avconv, a C, C++ based audio-video library to split the input video at desired rate and later uploaded them the HDFS.

2 Related Work

Earlier, many such video processors have been implemented in distributed environments. Similar transcoding system based on FFmpeg and Hadoop were proposed by Y. Cui et al. [1]. They made use of Xuggler, Java interface of FFmpeg library. A similar computing system was implemented by Yang and Shen with Hadoop and FFmpeg. [2]. X264farm is one such system that parallelizes a video encoding job in distributed environment. It divides a video into group of pictures (GOPs) and later this GOPs are distributed over the cluster with proper load balancing [3]. PCA based face recognition techniques are also very popular among the researchers due to its accuracy of classification [4]. Pereira et al.[5] have also implemented a similar with the help of MapReduce framework. They have a similar concept of the split and merge for the input video data. In their solution they go on adding computational nodes to their system. FFmpeg and Hadoop are used for a video processing in Mohohan . It uses various applications to split a video file and send for processing and later combine them after processing is done. H. Kalva et al. implemented a prototype to measure the performance of such transcoding systems [6]. a similar transcoder is also implemented using Hadoop, FFmpeg and OpenCV[7]

B. Iyer, S. Nalbalwar and R. Pawade (Eds.)
ICCASP/ICMMD-2016. Advances in Intelligent Systems Research.
Vol. 137, Pp. 420-425.

3 Background

Apache hadoop is an open source software framework that supports processing and storing of large data in a distributed environment. It is highly scalable and can scale up easily from one server to thousands of machines. Each of the nodes of the cluster has local computation and storage capabilities.

3.1 Hadoop Common

It is the assemblage of common utilities and libraries that form the base/core of the Hadoop framework and provides basic and essential services like abstraction of underlying operating system and file system.

3.2 Yarn

It provides resource management and monitoring of the cluster and job scheduling. Yarn involves setting up golbal as well as application specific resources management components.

3.3 HDFS

HDFS is abbreviation for Hadoop Distributed File System. This file system is used to store data on the cluster in a fault tolerant and low cost fashion. HDFS gives streaming data access to application running on it. HDFS works on master-slave model, where master-Name node consists of all the meta data and manage file system operations. The data Nodes store the data in form of blocks. Every file in HDFS is split to fit the size of the block and stored. The mapping of these data blocks are all stored in NameNode in form of meta data. We can set the replication factor (configuring hdfs-sites.xml) which decides the number of copies of the data to be stored in HDFS

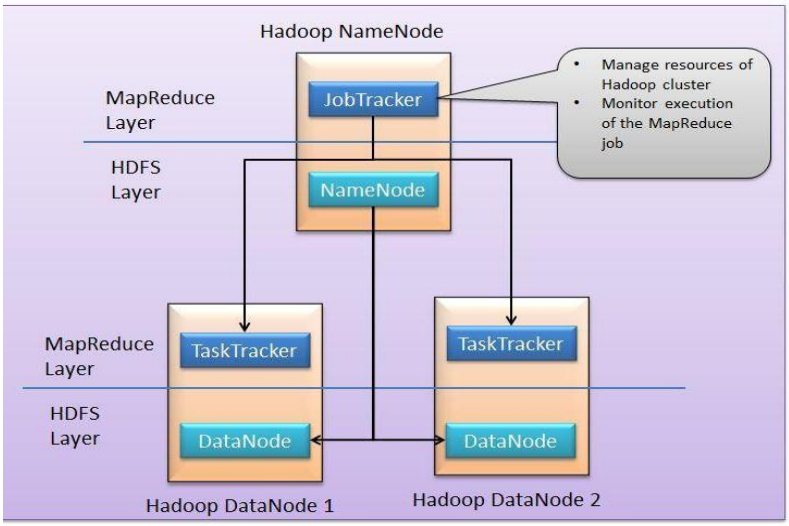

Fig.1. MapReduce, NameNode/ DataNode and JobTracker/ TaskTracker

3.4 MapReduce

MapReduce is a software framework that allows parallel processing of data over the cluster. As the name suggests, it has two main task, Map and Reduce Task.

Map: Here the already split input data is sent to the respective nodes present in the cluster for processing. Each data set is represented as a 'key, value' pair.
Reduce: Reduce task is optional. It takes the output of various map tasks and combines them in a single output.

3.5 NameNode

NameNode is the nervous system of the Hadoop File System. It contains the information of all the files and folders present in the HDFS. It stores this information in a tree data structure and tracks where across the cluster the required data is kept.

3.6 DataNode

A DataNode is the one, where data is stored in Hadoop File System. A file system may have more than one DataNode. While configuring Apache Hadoop, we can decide the number of replications we want the file system to keep. Depending on that number, copies of the files on the HDFS are stored on various nodes.

3.7 Job Tracker [8]

The JobTracker is the service within Hadoop that farms out MapReduce tasks to specific nodes in the cluster, ideally the nodes that have the data, or at least are in the same rack.

3.8 Task Tracker [9]

Task Trackers job is to accept the tasks from the Job Tracker. Tasks such as Map, Reduce and Shuffle.

4 Framework

The solution presented in this paper takes a reference image, input video and a threshold value as input and returns a text file with timestamp of matching instances of the reference image.

4.1 Input video splitting and distribution

The input video is directly passed to avconv, which splits video into various parts of a fixed size. The fixed split size should to be less than the data block size of the HDFS. Here in our case we took 100MB as the split size for 128MB of block size of Hadoop v2.6. The split videos are uploaded on the HDFS from the master node. The reference image is also uploaded on the HDFS along with the video parts. We are also uploading a text file that contains the start time of the last split of the video, which is required further for reduce operation.

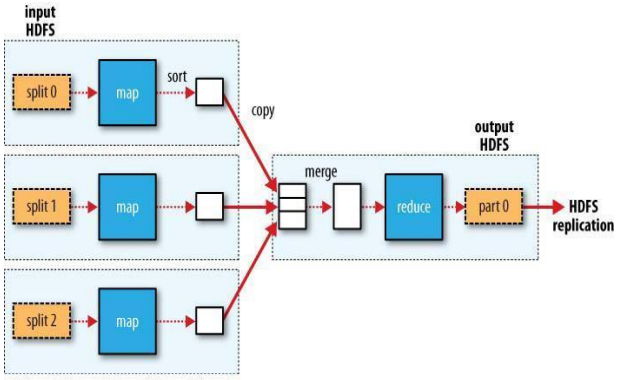

Fig.2. Overview

Hadoop framework is mainly used to process large text files, there is no inbuilt object for videos as input and output. To use videos over Hadoop cluster, we created a new video object, which extends the Hadoop writable class. In video object we used a byte array to read-write and send the video files over the Hadoop cluster. To validate the input specification, a new video class- Input Format that extends File Input Format was implemented and its methods is Split table was overridden to always return false, so that our own logic will be used to split the video. Record Reader is to generate next key as filename of the video split.

4.2 Mapper

The name of the video splits on the HDFS are used as key and video split itself is used as a value. The mapper maps video split data block to nodes based on internal hadoop mapping logic.. Now at each node, the reference image and the video is collected from the HDFS and stored in tmp directory. The video in the tmp directory is again passed to avconv. Over here it splits video into images at 1 fps (frames per second) i.e. now for every second of the video there is an image (1 fps serves as ideal video split parameter as it is impossible for a person to go out of video frame within 1 sec). The frames obtained are stored in a directory structure and these images (frames) are passed on to OpenCV for further processing. OpenCV is used to detect faces and returns x-y coordinates. The returned x-y coordinate of the detected faces are used to crop multiple faces from a particular image. At this time we created a CSV file that stores path to the cropped images and an id which is used to represent the time-stamp for that cropped image. These cropped images are now again passed to the OpenCV using that CSV file. Local Binary Pattern face recognition algorithm of OpenCV is used for comparing the reference image provided with the cropped images obtained from the previous operation. It returns a particular confidence value depending on the percentage of the match. Based on the returned confidence value it is decided whether an image matches or not. Threshold value specified by user is used as an upper bound for the confidence value(0.0 is perfect match - ideal condition). Time-stamp of images with confidence below threshold value are written as output.

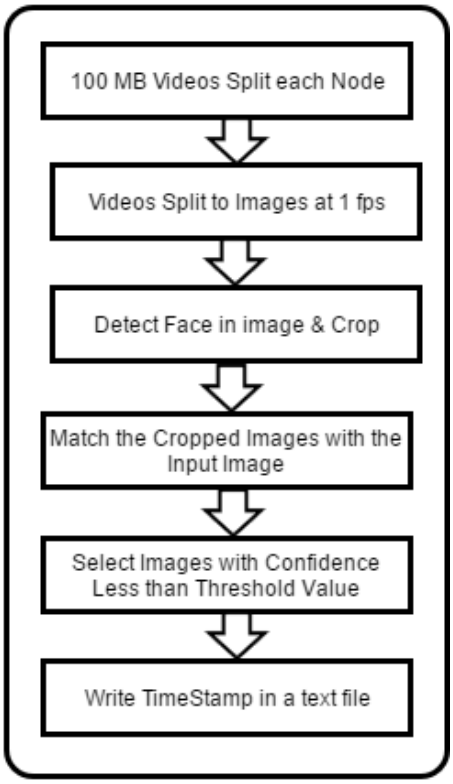

Fig.3. Processing done on each node

4.2 Reducer

The job of the Reducer in our application is quite simple. It collects the outputs present on each of the node and combines them in a single output file. The final output contains the timestamps obtained from each of each node in a single file which is present at the master node and presented to the user.

5 Experimentation

5.1 Performance

The face recognition system developed was run on different configuration of Hadoop cluster and also on a

standalone PC and their performance was evaluated. All the nodes of cluster were Intel Core i5-2400 @ 3.10GHz with 4 cores. OpenCV was installed with Intel TBB enabled on every nodes. Yarn was configured to allow two container per node.

Table 1. Performance analysis based on cluster configuration

Cluster Configuration	Time(hh:mm:ss)
4 nodes cluster	00:10:57
3 nodes cluster	00:17:05
2 nodes cluster	00:23:17
1 node cluster	00:47:52
standalone unit(no hadoop)	00:40:24

The performance was evaluated on a video of duration 2:30:00 and size 940 MB. The video was split into 8 parts each of size 120MB and uploaded on the HDFS. Now in the mapping stage each video split is loaded on the nodes and processed. For performance estimation we have excluded time of video splitting and uploading.

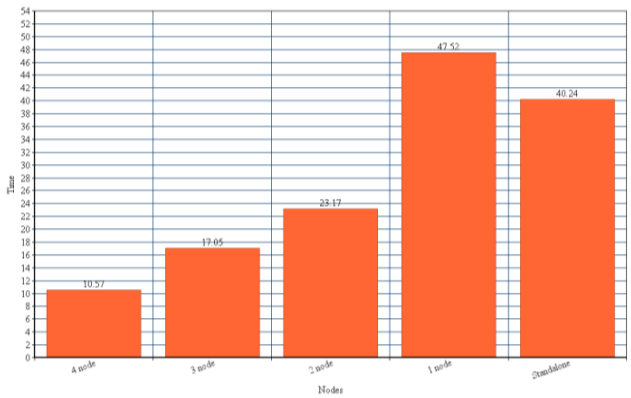

Fig.4. Increased performance using Hadoop for face recognition.

5.2 Accuracy Analysis

In our solution Local Binary Pattern face recognition algorithm of OpenCV is used. To increase flexibility of the system we have taken threshold of our predict function from user. To measure the accuracy of the face recognition algorithm at different threshold value we took a sample video and ran 4 different input value. The outputs were compared with the manual calculation.

Accuracy% = Actual Instance of face X 100 / Total face recognized

A video of 00:02:46 duration was analysed for test purpose. 7 instances of the reference image were found manually and recorded. Those instances were compared with output of code..

5.3 Fail Percentage

Similarly we calculated the missed face recognition instances for each threshold.
Fail% = Missed face recognition X 100 / Actual Instance of face

Table 2. Fail percentage in face recognition instance for each threshold

Threshold(Opencv)	Accurate recognition %	Fail Percentage
70	46.66	-
60	87.5	-
55	-	14
50	-	57

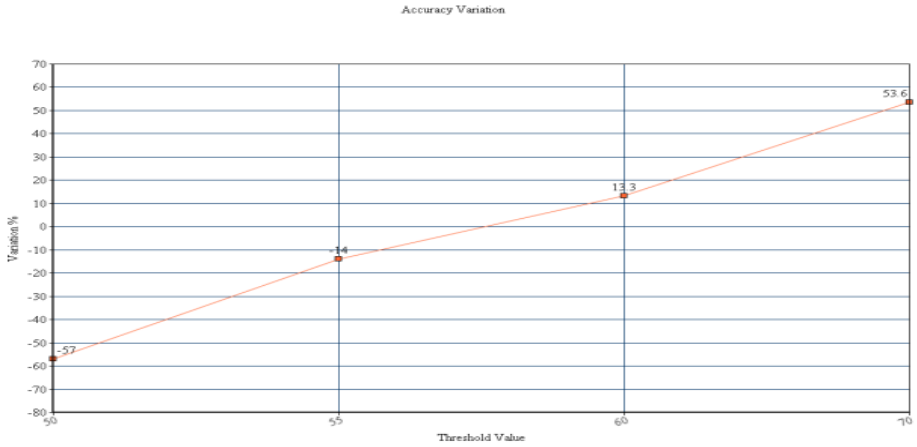

Fig .5. Accuracy

6 Conclusion

In this paper, we have introduced an efficient and scalable method for distributed face recognition using hadoop *MapReduce* programming model. We show how video should be split before distributed execution and examine the extent of overhead for Hadoop computation. Hadoop was designed mainly to operate with textual data, but we have shown how it can be applied to video data as well by modifying abstract classes. The cluster size can be easily scaled and therefore it is very suitable for elastic cloud infrastructure deployments. These features make Hadoop a worthy choice for distributing multimedia content analysis tasks with rather small overhead penalty.

References

[1] Y. Cui, M . Kim, S. Han, and H. Lee, "Towards efficient design and implementation of a hadoop-based distributed video transcoding system in cloud computing environment," International Journal of Multimedia and Ubiquitous Engineering, vol. 8, no. 2, Mar. 2013.

[2] F. Yang and Q.-W. Shen, "Distributed video transcoding on hadoop," Computer Systems & Applications, vol. 11, p. 020, 2011.

[3] R. Pereira, M. Azambuja, K. Breitman, and M. Endler, "An architecture for distributed high performance video processing in the cloud," in Cloud Computing (CLOUD), IEEE 3rd International Conference on, Jul. 2010, pp. 482 –489.

[4] M. Patil, B. Iyer and R. Arya, "Performance Evaluation of PCA and ICA Algorithm for Facial Expression Recognition Application", Proceedings of Fifth International Conference on Soft Computing for Problem Solving, vol.436, 965-976 (2016).

[5] H. Kalva, A. Garcia, and B. Furht, "A study of transcoding on cloud environments for video content delivery," in Proceedings of the 2010 ACM multimedia workshop on Mobile cloud media computing, ser. MCMC '10. New York, NY, USA: ACM, 2010, pp. 13–18.

[6] C.-H. Chen. :[Online]. Available at: http://www.gwms.com.tw/TRENDadoopinTaiwan 2012/1002 download / C3.pdf.

[7] R. Wilson. x264farm. A distributed video encoder. [Online]. Available at: http://omion.dyndns.org/ x264farm/ x264 farm.html