DOI: 10.2507/27th.daaam.proceedings.095

MAPREDUCE-BASED FACE DETECTION IN IMAGES

Ana Pinjuh, Linda Vickovic & David Cavar

This Publication has to be referred as: Pinjuh, A[na]; Vickovic, L[inda] & Cavar, D[avid] (2016). Mapreduce-Based Face Detection in Images, Proceedings of the 27th DAAAM International Symposium, pp.0658-0663, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-08-2, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/27th.daaam.proceedings.095

Abstract
Huge amounts of visual data have been generated daily on the Internet. Created data requires not only massive storage but also massive computational power for processing and analyzing data in efficient manner. Therefore, new distributed platforms like Hadoop are becoming more popular because of their capability to deal with huge amount of complex data in real time. MapReduce, a part of Hadoop, is frequently used programming model for writing distributed applications. This paper describes and presents results of MapReduce-based face detection process using Hadoop platform on BIOID image database.

Keywords: image processing; Hadoop; face detection.

1. Introduction

Today, in the information age computerization of data has led to the explosive increase in information generated on daily basis. Development of mobile technologies and social networks had a great influence on data explosion. Also, with the invention of digital cameras humans are given the possibility to store their every moment in picture form. Number of multimedia content is constantly growing at an increasing rate. Amount of the image data significantly increased not only due to the use of social networks but also because of the use of surveillance cameras and satellite images. Therefore, development of digital technologies made creating and storing visual data equally frequent as the text documents. Availability of such large amount of constantly growing visual data requires tools which can efficiently find and retrieve visual information on demand. The challenge is how to effectively manage all the computations and storage requirements put on by the influx of data.

Hadoop, at the moment is one of the most popular platforms for Big Data analysis and as such is useful in wide range of different applications. Some of them include: sorting, analyzing, machine learning, prediction models, risk assessment, pattern recognition and many other operations. Using Hadoop, users can write distributed applications and analyse huge data sets without understanding the underlying details of distributed computing and storage. This paper describes the role of Hadoop platform in image processing, specifically in face detection process.

The remainder of the document is organized as follows. Section 2 gives the survey of related work in the area of image processing and Hadoop. Hadoop, as one of the leading platforms is described in Section 3. Section 4 reports the process as well as results of face detection in experimental dataset. Finally, Section 5 concludes the paper and presents possible extensions.

2. Survey of the related work

Researches conducted in the field of image processing using Hadoop prove it as a valuable tool because of its ability to execute multiple parallel processes in real time. MapReduce, a part of Hadoop, is especially interesting as a large amount of problems are easily expressible as Hadoop-based MapReduce computations [1]. As a result, many researches are developing Hadoop-based MapReduce algorithms for sophisticated data analysis like is presented in [2], [3], [4], [5]. Current computer vision applications are not able to use big volumes of data because of the lack of resources for analyzing and storing large amount of data. Therefore, distributed platforms are becoming more popular as well as image processing features for those platforms. For example, design of Hadoop Image Processing Interface (HIPI) is described in [6]. HIPI is designed to enable: free libraries for image processing in MapReduce framework, easy image storing and filtering in MapReduce and to parallelise processes without any knowledge of parallelisation details. Also HIPI provides integration with OpenCV libraries which are very useful for computer vision algorithms.

Additionally, MapReduce implementations of several popular computer vision algorithms, such as clustering for image data and classifier training are described in [7]. Presented results show significant improvement compared to traditional implementations.

A distributed MapReduce algorithm for face matching based monitoring movement of individuals over large surveillance space is presented in [8]. The goal is to show benefits of using Hadoop for analyzing video data.

In [9] research have been conducted in using Hadoop for processing medical images. Authors have designed a MIFAS system for fast and efficient access to medical images using Hadoop. Furthermore, in [10] Hadoop is used to process ballistic images with a goal of recognizing samples from crime scenes when compared to image database.

Although, the data manipulation is one of the most explored Hadoop properties its performances are also under study. So in [11] different techniques for managing scheduling scheme are discussed, while in [12] shuffling phase of Hadoop's cluster have been researched.

3. Hadoop platform

Hadoop is an open source, batch data processing system for enormous amounts of data. This platform consists of two primary components: a reliable shared storage (Hadoop Distributed File System) and analysis system (MapReduce). However, these are not the only components of Hadoop. The Hadoop ecosystem provides a set of applications, libraries and systems for developing scalable Big Data applications. These numerous tools are based around Hadoop core (HDFS and MapReduce) for specific client's needs. For example, Mahout is a scalable machine learning and data mining library. Pig is high level data flow language while Hive provides ad hoc querying. Hbase is non - relational, distributed database written in Java. Zookeeper is a high performance coordination service for maintenance. Therefore, all these tools are in use for specific assignments and they represent upgrade on primary components of Hadoop.

3.1. Hadoop architecture

Machines controlled by Hadoop are organized in clusters, where a cluster is a group of connected computing nodes that work and act like a single system using local area network. One node, called *master*, receives request for particular job from client and it controls the job execution. All other nodes in cluster, responsible for computation are called *slaves*. The simple master – slave architecture of Hadoop cluster is shown in

Fig.**1**. First, it is important to notice that both, *master* and *slave* consist of two layers [13]: MapReduce and HDFS.

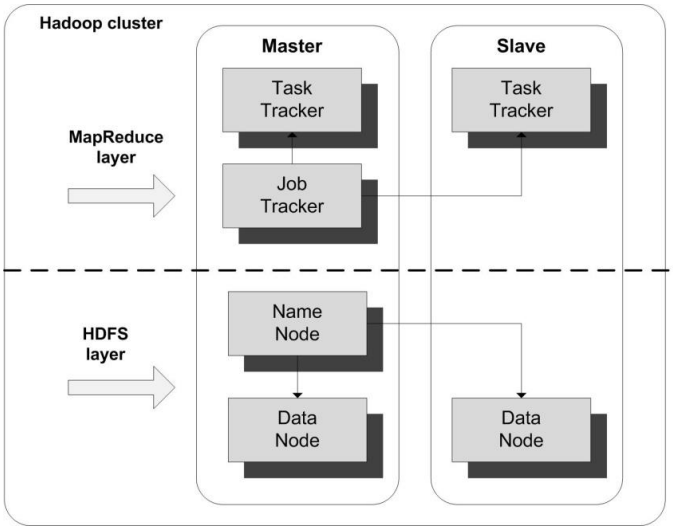


Fig.1. Hadoop architecture

MapReduce layer is responsible for job execution, so it contains two types of nodes a *jobtracker* located on *master* and a number of *tasktrackers* located on *slaves*. The *jobtracker* coordinates all the jobs run on the system by scheduling tasks to run on *tasktrackers*. *Tasktrackers* run tasks and send progress reports to the *jobtracker*, which keeps record of overall progress of each job. On the other hand, a HDFS layer is responsible for data storage. It also includes two types of nodes: a *namenode* (*master*) and a number of *datanodes* (*slaves*). The *namenode* keeps record of every file and block in the file system in memory. Without the *namenode* file system cannot be used. *Datanodes* are responsible for storing and retrieving blocks when they are told to by client or *namenode* and they report back to *namenode* periodically with list of blocks that they are storing. *Master* node keeps record of available *slave* nodes and lets *slave* node know what the next task is and where required data are located.

3.2. *MapReduce*

MapReduce, one of the core components of Hadoop, is linearly scalable programming model and execution environment for data processing. It runs on large clusters of commodity machines [13] and works by dividing the execution processing into two phases: map phase and reduce phase.

In map phase, the input dataset splits into chunks which are processed by the *mappers* in a parallel way. Each chunk can be processed by only one *mapper*. Once the data is processed by *mappers* the output of this phase are {*key*, *value*} pairs which are then forwarded to *reducers*. Hadoop sorts the outputs of the *mappers* according to the *key* values, which are then the input to be processed by the *reducers*. Each *reducer* gets the list of values related to a given *key* and the output of the reduce phase is saved into the distributed file system. Reduce phase cannot start until the map phase is completely finished.

Hadoop job not only consists of parallel map and reduce tasks but also of a shuffle and sort phase. Users write code for map and reduce phase while the Hadoop MapReduce framework automatically handles the shuffling, sorting and parallelizing the processes over multiple nodes [12]. MapReduce job execution is shown in

Fig.**2**.

Hadoop Distributed File System (HDFS) is designed to store data with high bandwidth across the cluster nodes and Hadoop job is design to read that data. Usually, file sizes in HDFS range from gigabytes to terabytes in size. They are broken into block-sized chunks, which are stored as independent units and distributed to hundreds of nodes in a single cluster. Default size of the block in HDFS is 64 MB. To insure fault tolerance and availability each block is replicated to a number of physically separate machines [13]. By default replication is set to three.

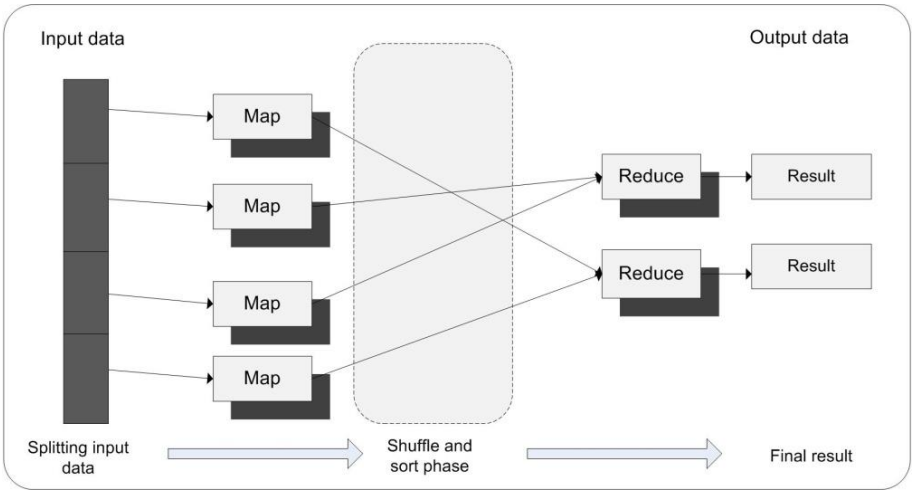


Fig.2. MapReduce job execution

4. Face detection experiment

The main goal of the experiment is to determine whether human faces appear in input image and its position. Dataset used for this experiment is composed of 1000 images (384 x 288 pixel, grayscale) of different persons downloaded from BIOID database [14]. Experiment is run on a single node Hadoop cluster with use of Hadoop version 2.6.4 and Hadoop Image Processing Interface [6]. HIPI is necessary feature for preserving the whole image as an input to mapper. By default HDFS is creating parts or splits the input data which are then replicated and stored in different computer nodes so without HIPI successful face detection would be difficult. For face detection process is used Haar-cascade classifier.

Face detection process in this experiment is conducted in several steps. First of all, Hadoop Image Processing Interface is storing data as a collection of images called HIPI Image Bundle which are then sent as an input to mapper. Input values are distributed among the available map tasks. The MAPPER class has a MAP method that is called once for each input. Map method is executing the following initial steps: loading OpenCV libraries to enable image processing, converting

HIPI Image Bundle for compatible OpenCV format and creating classifier for content-based image analyzing. Afterwards, face detection process is executing and the result is emitted to REDUCER in {*key*, *value*} form where key is metadata from image and value is coordinates of detected faces in images.

The REDUCER class has a REDUCE method that is called once for each unique key. Reduce method calculates number of detected faces and emits final result to HDFS. See Fig.3. where step-by-step Hadoop face detection process is presented, while its logic for a concrete implementation is given in Algorithm 1.

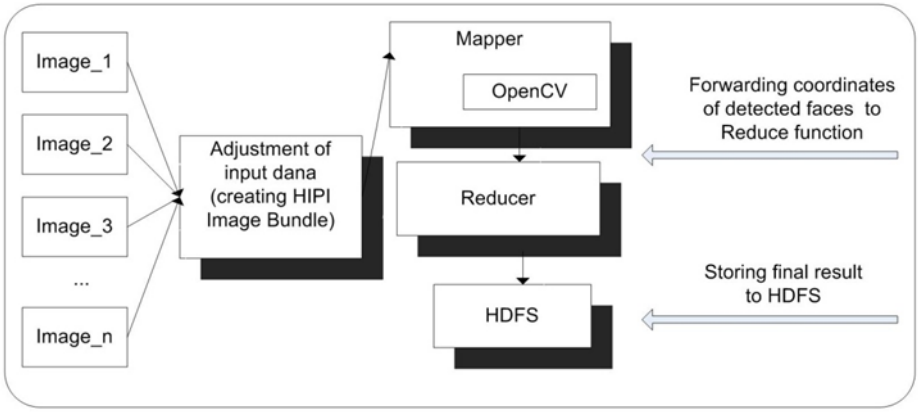


Fig.3 Hadoop face detection process

```
1:  class MAPPER
2:          method MAP (metadata meta, image i)
3:          m ← convertToCompatibleFormat (image i)
4:          cascade_data ← load()
5:          c ← initClassifier(cascade cascade_data)
6:          detections ← detectFace(mat m, classifier c)
7:          for all detection d ∈ detections do
8:                  key ← getSource(medatada meta)
9:                  value ← formulateValue(detection d)
10:                 EMIT(source key, frame value)
1:  class REDUCER
2:          method REDUCE(source key, frames[r1, r2, r3, ... ])
3:          collection ← initCollection()
4:          for all frame f ∈ frames do
5:                  appendToCollection(frame f, frame_collection collection)
6:          EMIT(Hash key, frame_collection collection)
```

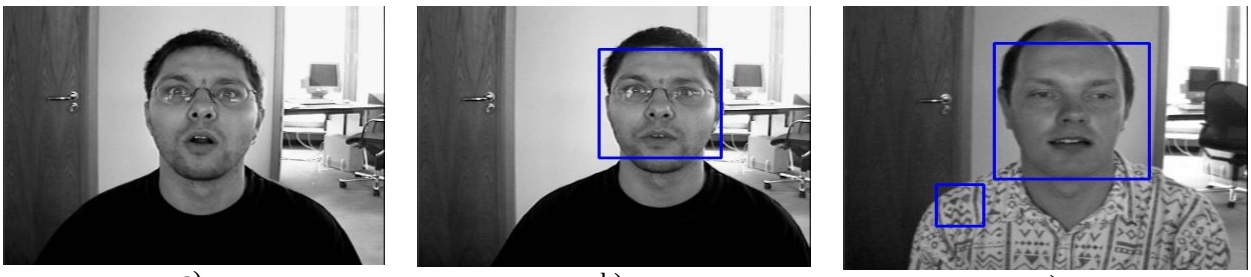
Algorithm 1 MapReduce algorithm for face detection in images

Table **1**. shows the results of MapReduce-based face detection. From total 1000 faces in all images 976 faces are successfully detected. There is a 124 false positive results as well as the 24 false negative which gives overall accuracy of 86.83%. The total execution time for this process was 3 minutes 39 seconds. Results show that MapReduce-based face detection gives high accuracy of detection on great amount of images in real time. Therefore, it can be useful tool to solve different real-time problems which require face detection and face recognition.

Observed Items	**Values**
Faces in images	1000
Faces successfully detected	976
False positive detection	124
False negative detection	24
Time elapsed	3m 39s
Accuracy	86.83%

Table 1. Hadoop face detection experiment results

Some of the sample images used for face detection from BIOID database are presented in Fig. **4**. One of the original images from BIOID database before face detection process is presented in Fig.4. a), while Fig.4. b) shows detected face on the same image afterwards. Fig. 4. c) presents sample of image with false positive detection.



a) b) c)

Fig. 4. Sample images from BIOID database: a) original image, b) image after face detection process, c) image with false positive detection

5. Conclusion

Face detection is normal human psychological process by which humans easily locate faces in visual scene. Although, recognising content of image and especially face detection is natural process for humans, doing the same task for computers is much more complex process. Therefore it is actual and interesting research area because of the all spheres of real life where face detection can be put to use like in security systems and surveillance cameras.

Efficient usage of huge amount of visual data currently generated requires distributed storage and processing. One of the most popular software frameworks for distributed storage and distributed processing is Hadoop. It provides a simple MapReduce programming model, a distributed file system and job management as well as the cluster management.

Results presented in this paper show that MapReduce-based face detection is promising tool in terms of accuracy of detection as well as the time spent on execution. However, by increasing the number of nodes in Hadoop cluster the performances would improve linearly which can be studied in future work. Also, future research should consider implementing more pose variations as well as the greater amount of images. Additionally, different elements can partially cover the faces like glasses or beards which can affect performance of face detection. Therefore, different classifiers can be used and their performances can be compared.

However, it is shown that Hadoop can be successfully used in computer vision algorithms for real time processing because of its performances.

6. References

[1] Garcia, C. (2013). Demystifying MapReduce. *Procedia Computer Science*, Vol. 20., Complex Adaptive Systems Publication 3 (2013.), pp. 484-489.

[2] Shelly & Raghava, N.S. (2011). Iris recognition on Hadoop: A biometrics system implementation on cloud computing. *2011 IEEE International Conference on Cloud Computing Computing and Intelligence Systems*, (September 2011.), pp. 482-485., ISSN 2376-5933

[3] Almeer, M. (2012.). Cloud Hadoop map reduce for remote sensing image analysis, *International Journal of Emerging Technology and Advanced Engineering*, Vol. 2, No.4, 2012, ISSN 2250-2459

[4] Pavlech, M. (2011). Framework for development of distributed evolutionary algorithms based on mapreduce. *Annals of DAAAM for 2011 &Proceedings of the 22nd International DAAAM Symposium*, Vienna, Austria (2011) Vol.22, No.1, ISSN 1726-9679

[5] Leokhin, Y.; Myagkov, A. & Panfilov, P. (2016). GoMapReduce Parallel Computing Model Implementation on a Cluster of Plan9 Virtual Machines, *Proceedings of the 26th DAAAM International Symposium*, pp. 0656-0662, B.Katalinic (ed.), Published by DAAAM International, ISBN 978-3-902734-07-5, Vienna, Austria (2016), ISSN 1726-9679, DOI: 10.2507/26th.daaam.proceedings.089

[6] Sweeney, C.; Liu, L.; Arietta, S. & Lawrence, J. (2011). HIPI: A Hadoop image processing interface for image-based MapReduce tasks, B.S. Thesis, Department of Computer Science, University of Virginia, USA

[7] White, B; Yeh, T.; Lin, J. &Davis, L. (2010) Web-scale computer vision using MapReduce for multimedia data mining, *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, New York, USA, ISBN 978-1-4503-0220-3, ACM, NY, DOI 10.1145/1814245.1814254

[8] Mishra, R.; Kumar, P; Chaudhury, S & S, I. (2013) Monitoring a large surveillance space through distributed face matching, *Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, 18-21 December 2013, Jodhpur, ISBN 978-1-4799-1586-6, pp.1-5, IEEE, DOI: 10.1109/NCVPRIPG.2013.6776185

[9] Yang, C.; Chen, L; Chou, W & Wang, K. (2010). Implementation of a medical image file accessing system on cloud computing, *IEEE 13th International Conference on Computational Science and Engineering (CSE)*, 11-13 December 2010, Hong Kong, ISBN 978-1-4244-9591-7, pp. 321-326, IEEE, DOI: 10.1109/CSE.2010.48

[10] Kocakulak, H & Temizel, T. (2011). A Hadoop solution for ballistic image analysis and recognition, *International Conference on High Performance Computing and Simulation (HPCS)*, 4-8 July 2011, Istanbul, ISBN 978-1-61284-380-3, pp. 836-842, IEEE, DOI: 10.1109/HPCSim.2011.5999917

[11] Xie, J.; Meng, F; Wang, H.; Pan, H.; Cheng, J.& Qin, X. (2013). Research on scheduling scheme for Hadoop clusters, *International Conference on Computational Science*, Procedia Computer Science, Vol. 18, pp. 2468-2471, Elsevier B.V., DOI: 10.1016/j.procs.2013.05.423

[12] Xie, J.; Tian, Y.; Yin, S.; Zhang, J.; Ruan, X. & Qin, X. (2013). Adaptive preshuffling in Hadoop clusters, *International Conference on Computational Science*, Procedia Computer Science, Vol. 18, pp. 2458-2467, Elsevier B.V., DOI: 10.1016/j.procs.2013.05.422

[13] White, T; (2012.). *Hadoop: The definitive guide, third edition*, OReilly Media, INC., ISBN 978-1-449-31152-0, CA, USA

[14] https://facedetection.com/datasets/ (2016). Face Detection & Recognition Homepage, Accessed on: 2016-04-04