Optimized Image Handling in Big Data Applications Using Hadoop Sequence Files

Dr. S. Hemalatha*, Dr. P. Ravi Kumar, Dr. K. V. Mohan
*Associate Professor, Department of CSE, GVP College of Engineering, Visakhapatnam, Andhra Pradesh, India
Professor & Head, Department of CSE, Andhra University College of Engineering, Visakhapatnam, Andhra Pradesh, India
Professor, Department of CSE, GVP College of Engineering, Visakhapatnam, Andhra Pradesh, India

ABSTRACT

This paper presents MapReduce as a distributed data processing model utilizing open source Hadoop framework for work huge volume of data. The expansive volume of data in the advanced world, especially multimedia data, makes new requirement for processing and storage. As an open source distributed computational framework, Hadoop takes into consideration processing a lot of images on an unbounded arrangement of computing nodes by giving fundamental foundations. We have lots and lots of small images files and need to remove duplicate files from the available data. As most binary formats—particularly those that are compressed or encrypted—cannot be split and must be read as a single linear stream of data. Using such files as input to a MapReduce job means that a single mapper will be used to process the entire file, causing a potentially large performance hit. The paper proposes splitable format such as SequenceFile and uses MD5 algorithm to improve the performance of image processing.

KEYWORDS: MapReduce, distributed data processing, Hadoop, sequence file.

INTRODUCTION

The vast measure of image data has become considerably as of late because of the development of social networking, observation cameras, and satellite images. Be that as it may, this development is not constrained to multimedia data. This enormous volume of data on the planet has made another field in data processing which is called Big Data that these days situated among main ten vital technologies [1]. Big Data alludes to the huge measures of data gathered after some time that are hard to examine and handle utilizing basic database management tools [2]. Because of the expanding measure of data that is getting to be accessible consistently and quickened development of information innovation, one can't give a reasonable meaning of the size and size of enormous data. Be that as it may, a multi-terabyte data sets (every terabyte = 1000 gigabytes) to multi-petabytes (every petabyte = 1000 terabytes) are considered as Big data. While huge data can yield to a great degree helpful information, it additionally gives new difficulties admiration to the amount of data to store, the amount this will cost, whether the data will be secure, and to what extent it must be looked after [3]. The main way to deal with take care of this issue was the utilization of multi-processor systems with high stockpiling limit. The fast development of data volumes and ongoing processing needs and the many-sided quality of programming improvement and algorithms that can adequately use all processors kept this way to deal with meet the new demands of data processing [3]. The following methodology for processing extensive volumes of data and image was distributed systems with Message Passing Interface (MPI). Alongside parallel data processing in distributed computing nodes and spread of data in every node, this methodology guaranteed a brilliant future for new data processing needs. Be that as it may, the issue that this procedure was confronted with was parallel coordination and execution of the required algorithms that totally relied on upon the system programmer and developer. Along these lines, it was not generally grasped because of the absence of specialists and professional developers [4]. Google as one of the main organizations in the field of big data, proposed the MapReduce programming model [5] which was intended to process a lot of distributed data. The primary favorable circumstances of this model are its straightforward programming structure, distributed file system, and distributed administration which is disappointment safe. The principle issue in the commonness of this model was the arrangement of computing cluster for its usage. It requires energy, cooling systems, physical space, vital hardware and software for setting it up. These prerequisites are exorbitant for some little and average size companies and enterprises [4]. This obstruction has been determined now by the prominence of distributed computing that gives customers minimal effort hardware and software in view of the asset use. Simply lease the quantity of computing nodes and the required resources when required, then run your algorithm and get the outcome. One of the understood case in this field is the creating PDF records from scanned day by day chronicle of the New York Times in 2007. For this situation 11 million photographs with a volume of around 4 terabytes were changed over to PDF just in 24 hours by utilizing 100 nodes of Amazon Cloud Computing. This errand would keep going for a long time utilizing

general systems and algorithms [6]. In this paper, we introduce the MapReduce model as the basis of the modern distributed processing, and it's open-source exertion named Hadoop, the work that has been done in this area, its merits and demerits as a framework for distributed processing, especially in image processing. In this paper, we present the MapReduce model as the premise of the current distributed processing, and it's open-source effort named Hadoop, the work that has been done around there, its benefits and bad marks as a framework for distributed processing, particularly in image processing. In this paper, it is accepted that the perusers are acquainted with cloud computing. In look, cloud computing gives online computation and computer processing for clients, without being agonized over the quantity of required computers, resources, and different contemplations. Clients pay the cost in light of the measure of resource utilization. Allude to source number [7] to take in more about this famous subject in cutting edge data innovation. A couple of frameworks can be utilized for image processing as a part of Hadoop. HIPI (Hadoop Image Processing Interface) [8] is a framework distinctly intended to empower image processing in Hadoop.

DISTRIBUTED FRAMEWORK FOR DATA PROCESSING

Hadoop is an open source framework for processing, storage, and analysis of huge amounts of distributed and unstructured data [8]. The main origin of this processing framework returns to Internet search companies, Yahoo and Google. They needed new processing tools and models for web page indexing and searching. This framework is designed for parallel data processing at Petabyte and Exabyte scales which are distributed on the normal compute nodes, such a way that a Hadoop cluster can easily be extended horizontally. Hadoop is currently developed and expanded by The Apache Foundation. Hadoop was made by Doug Cutting and Mike Cafarella in 2005. Cutting, who was working at Yahoo at the time, named it after his child's toy elephant. It was initially created to bolster appropriation for the Nutch internet searcher venture. The Apache Hadoop structure is made out of the following modules [9]:

- Hadoop Common contains libraries and utilities needed by other Hadoop modules.
- Hadoop Distributed File System (HDFS)—a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster.
- Hadoop YARN—a resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications.
- Hadoop MapReduce—a programming model for large scale data processing.

HOW DOES HADOOP WORK?

In this framework, vast data files, for example, exchange log files, bolster reader of social networks, and other data sources are fragmented and after that distributed in the network. Sharing, securing, and recovering broad files on a Hadoop cluster is endeavored by its scattered record framework called HDFS [10].To expand the genuineness of the framework, every part of the document is distributed among numerous compute nodes. Consequently, if a hub quits working, its record can be recovered once more. There are three sorts of compute nodes in HDFS [10]. Name management node is in charge of sharing the files and putting away the location of every part. Intermittent survey of nodes and deciding their being eliminated are likewise the undertakings of Hadoop file management system. Information node that envelops every one of Hadoop part PCs contains file blocks. There is a name management node in Hadoop system for every information node set. The third sort is the secondary node that there is a duplicate of name management node information on it. Accordingly if the node quits working, the information won't be lost. Figure 1 demonstrates a layout of Hadoop file management. After information distribution in Hadoop system, analysis and processing would be completed by the MapReduce part [11]. Figure 2 demonstrates this procedure unmistakably. In the initial steps, the client sends his/her solicitation to a node which is in charge of running the solicitations (job tracker). This solicitation more often than not is a Java question dialect. Now, job tracker checks the files to see which one are required for noting the client's inquiry. By then by the help of name management node, it finds the nodes containing those parts in the cluster. After that, this requesting is sent to each node. These nodes, that we call them task trackers, perform data handling unreservedly and in parallel by running Map limit [12]. After the task trackers' works is done, the outcomes will be stored on the same node. Plainly, the widely appealing results would be close-by and divided in light of the fact that they depend on upon the data open on one node. In the wake of arranging of the transitional results, the job tracker sends the Reduce sales to these nodes. Therefore, it performs the keep going handling on the outcomes and the result of customer's sales would be saved in a last figure node. Presently, MapReduce is done, and propel preparing of the outcomes should be performed by Big Data analysts. This handling can be performed straightforwardly on the outcomes or customary strategies for data analysis can be used by trading the resulting data into a relational databases or data warehouse [13].

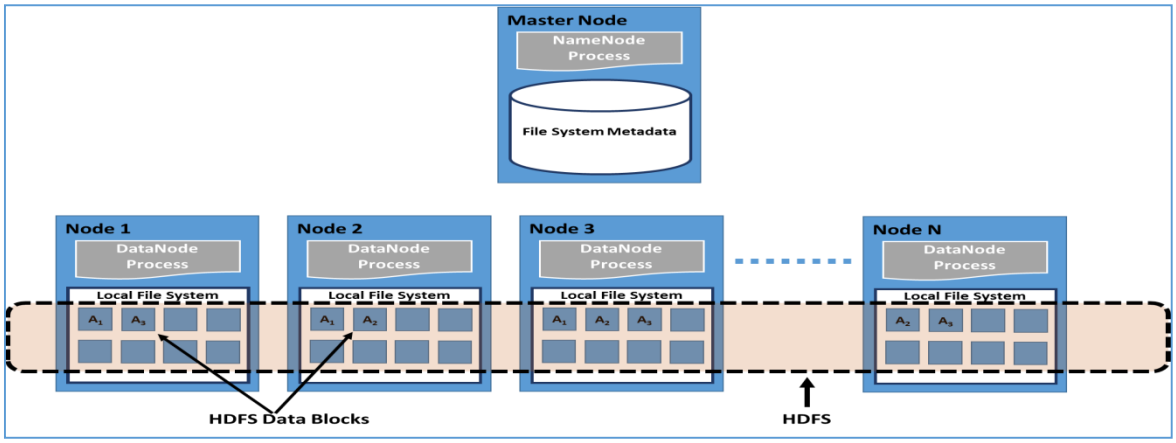

Figure 1 HDFS File System

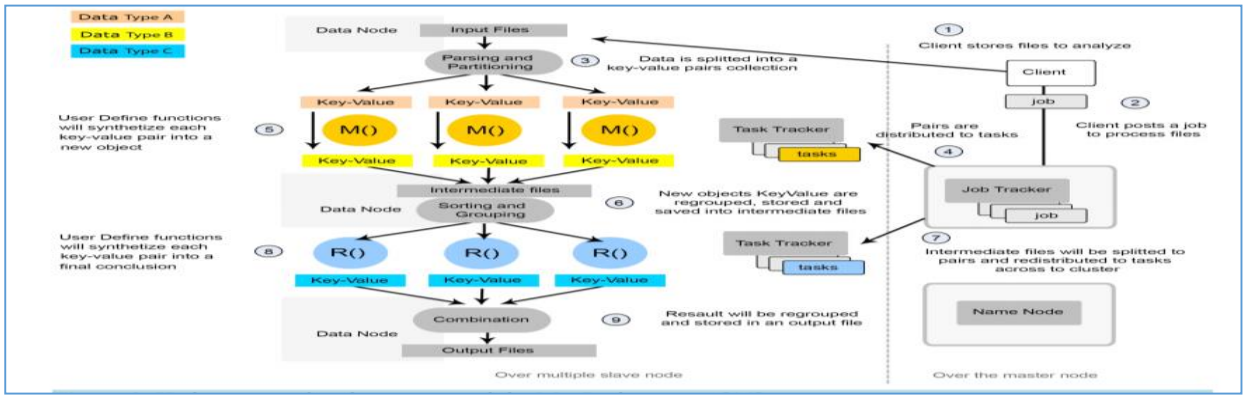

Figure 2 Hadoop operational structure and the MapReduce Steps

PROBLEM STATEMENT

Image files can be entirely vast, and bigger document sorts mean more disk utilization and slower download. Compression is a term used to depict methods for cutting the extent of the document. Compression schemes can by lossy or lossless. Another clarification behind the various report sorts is that images contrast in the amount of hues they contain. Lossless and lossy pressures are terms that depict whether or not, in the compression of a report, each and every unique data can be recovered when the record is uncompressed. With lossless compression, every single bit of data that was initially in the record stays after the archive is uncompressed.

We have parts and bunches of little images files and need to expel duplicate files from the accessible data. As most binary formats—especially those that are compressed or encrypted—can't be split and should be perused as a single linear stream of data. Utilizing such files as contribution to a MapReduce job implies that a single mapper will be utilized to prepare the whole document, bringing on a conceivably expansive performance hit. The regular image formats are as taking after:

- PDF - Portable Document Format
- JPG - Joint Photographic Expert Group
- JPEG - Joint Photographic Expert Group
- TIFF- Tag Index File Format
- GIF - Graphic Intercharge Format
- BMP - Bitmap Image

PROPOSED APPROACH

We have lots and lots of small images files and need to remove duplicate files from the available data. As most binary formats—particularly those that are compressed or encrypted—cannot be split and must be read as a single linear stream of data. Using such files as input to a MapReduce job means that a single mapper will be used to process the entire file, causing a potentially large performance hit. In such a situation, it is preferable to either use a splitable format such as SequenceFile, or, if you cannot avoid receiving the file in the other format, do a preprocessing step that converts it into a splitable format. To implement the proposed approach we use MD5 algorithm

SEQUENCE FILE

Sequence File is a flat file consisting of binary key/value pairs. It is extensively used in MapReduce as input/output formats. It is also worth noting that, internally, the temporary outputs of maps are stored using SequenceFile, and figure3 demonstrates the sequence file approach.

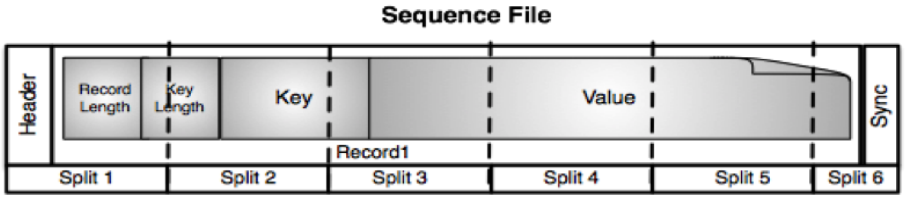

Figure 3

USING HADOOP SEQUENCE FILES

So what would it be a good idea for us to do keeping in mind the end goal to manage tremendous measure of images? Use hadoop sequence files! Those are map files that characteristically can be perused by map reduce applications – there is an info organize particularly for sequence files – and are splitable by map reduce, so we can have one enormous file that will be the contribution of numerous map tasks. By utilizing those sequence files we are giving hadoop a chance to utilize its points of interest. It can split the work into pieces so the processing is parallel, however the lumps are sufficiently huge that the procedure stays productive. Figure 4 showing the working periods of map reduce in processing the images files.

Since the sequence file are map file the wanted configuration will be that the key will be content and hold the HDFS filename and the quality will be BytesWritable and will contain the image substance of the file. Comment – for this present illustration's purpose it is far better to store the processed binary file, rather than the entire bytes of the file. Be that as it may, I'm keen on demonstrating to read and write binaries so we will adhere to the general BytesWritable. That would all say all is extremely well, however how to produce a sequence file from the images files? Since there are bunches of them it might demonstrate not a simple task by any means. There are some ways that I had considered.

On the off chance that conceivable the most ideal path is to produce the sequence file when the images are procured. That way the system is constantly prepared for image handling without the need to do some preprocessing to produce the sequence file. Since sequence files can be appended to this can be connected likewise if the images are not being retrieved at the same time. We can do this by utilizing class org.apache.hadoop.io.SequenceFile.Writer.

Putting away every one of the images as a tar file and utilizing the tool written by Stuart Sierra to change over it to a sequence file. Utilizing a map reduce job to gather all the image files and store them as a sequence file. The map input will be the HDFS path of those images and the map output will the sequence file (no reduce undertaking for this situation). Before starting diving into code a quick question glides in the psyche. In case I'm trying to load every one of the images bytes into the memory with a specific end goal to keep in touch with them to a sequence file, won't I would experience the ill effects of the considerable number of issues identified with a tremendous measure of small files that I had said above? Furthermore if the images are in the memory won't it be best to do the image preparing – discovering image duplicates – at this stage as opposed to making a sequence file and at exactly that point attempt to discover the duplicates? The answer has two unique perspectives:

- On the off chance that the image processing incorporate more than simply image copy finder and there are more than a single map reduce job that need to peruse every one of the images, then you might want it to be as a sequence file with a specific end goal to perform better
- The pre process of changing over the images into a sequence file should be possible for every image autonomously on the off chance that we are not having every one of the images yet. The copy finder in any case, is a process that needs every one of the images keeping in mind the end goal to be processed. Decoupling the job of making the sequence file from the job of finding the copies can be utilized as a part of a few cases to plan the framework in a way that the sequence file creation will be summoned on the images even before every one of the images are present.

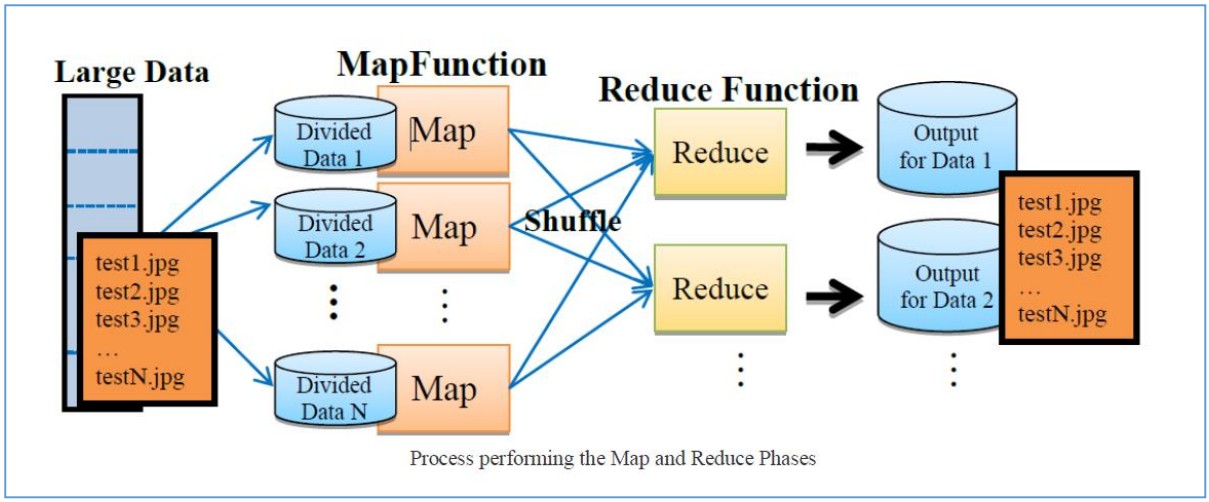

Figure 4

PRE PROCESSING JOB – GENERATING THE SEQUENCE FILE

The goal is a sequence file which has the filename as the key and the BytesWritable as the value. The input is a file contains the entire image file as HDFS filenames. For example: hdfs://localhost:8022/customer/elevy/smallArchiveImages/WonderWoman.jpg, so our map/lessen job will consolidate only a map that will read one file on the double and make it to a sequence file by using SequenceFileOutputFormat. It uses a FileSystem object as a piece of solicitation to open the HDFS file and FSDataInputStream with a particular deciding objective to peruse from it. The byte array is being created to the association as a bytewritable. Since the output configuration of the job is SequenceFileOutputFormat class, the output of the map is being made to a sequence file. This is appeared in the BinaryFilesTo HadoopSequenceFile.java which actualizes this pre method job.

BINARY FILE

A binary file is computer - readable however not comprehensible. Every single executable project are stored in binary files, as are most numeric data files. Interestingly, text files are stored in a structure (usually ASCII) that is comprehensible. The probability to process binary files with hadoop, shows with a case from the universe of images. The image duplicates pioneer manages the issue of different modestly little files as a contribution for a hadoop job and exhibits to scrutinize binary data in a map/reduce job. In this way it displays how to process binary files with hadoop. As a general rule when overview map/reduce algorithms it's about text. From the 'Word Count' surely understood case to various other hadoop jobs, one can assume that the entire hadoop framework is expected to process text data. Things being what they are, while it may be legitimate in a manner of speaking, hadoop is formed in like manner to manage colossal measures of data which is not text.

Hadoop is getting it done reading from huge data file while disseminating the processing on a few specialists. It uses optimizations to have the processing unit give requirement for its nearby data as the contribution for processing. What will happen if we will nourish the procedure with gigantic measure of moderately small files like images?

Evidently it will have a huge amount of overhead. The HDFS is keeping moderately big amount of overhead for every file to have the ability to reinforce highlights like splitting a file over numerous machines and backup every file with a few duplicates on different machines. With respect to the processing time, It will be influenced too. As per usual every data file is being send to it own map undertaking, thusly, tremendous amount of files means an immense amount of map tasks. Having heaps of map tasks can fundamentally direct the application by the overhead of every undertaking. This can be fairly calmed by reusing the undertaking's JVM and by using the MultiFileInputSplit, for more than 1 file in a split.

MD5 Message Digest Algorithm

MD5 algorithm was produced by Professor Ronald L. Rivest in 1991. According to RFC 1321, "MD5 message-digest algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input … The MD5 algorithm is expected for digital signature applications, where a large document must be "compressed" in a protected manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA." Figure 5 demonstrates the MD5 Algorithm Structure.

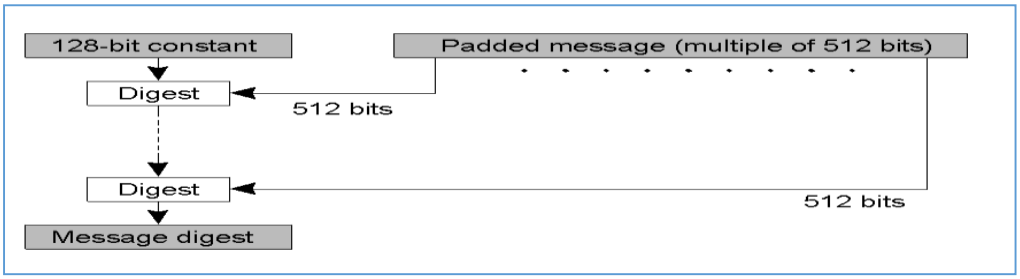

Figure MD5 Algorithm structure.

Implementation Steps

Adding padding bits

The input message is "padded" (expanded) so that its length (in bits) equivalents to 448 mod 512. Padding is constantly performed, regardless of the fact that the length of the message is as of now 448 mod 512. Padding is executed as takes after: a solitary "1" bit is attached to the message, and afterward "0" bits are annexed so that the length in bits of the padded message gets to be harmonious to 448 mod 512. No less than one bit and at most 512 bits are added.

Adding length

A 64-bit representation of the length of the message is added to the result of step1. On the off chance that the length of the message is more noteworthy than 2^64, just the low-arrange 64 bits will be utilized. The resulting message (subsequent to padding with bits and with b) has a length that is a definite multiple of 512 bits. The information message will have a length that is a precise multiple of 16 (32-bit) words.

Input MD buffer

A four-word buffer (A, B, C, D) is utilized to compute the message digest. Each of A, B, C, D is a 32-bit register. These registers are introduced to the accompanying qualities in hexadecimal, low-arrange bytes first):

Word		A:		01		23		45	67
Word		B:		89		ab		cd	ef
Word		C:		fe		dc		ba	98
Word		D:		76		54		32	10

Develop message in 16-word blocks

Four functions will be characterized such that every function takes a contribution of three 32-bit words and creates a 32-bit word output.

F (X, Y, Z) = XY or not (X) Z
G (X, Y, Z) = XZ or Y not (Z)
H (X, Y, Z) = X xor Y xor Z
I (X, Y, Z) = Y xor (X or not (Z))

Performance

Algorithm	**Key size/hash size(bits)**	**Extrapolated Speed (Kbytes/sec.)**	**PRB Optimized (Kbytes/sec.)**
TEA	128	700	-
DES	56	350	7746
Triple-DES	112	120	2842
IDEA	128	700	4469
RSA	512	7	-

SHA	160	750	25162
MD5	128	1740	62425

WORKS CARRIED OUT

Hadoop which is under the grant of Apache with the sponsorship of this foundation is accessible to researchers as an open source framework. For using MapReduce models, despite Hadoop, we can moreover use Twister [14] and Phoenix [15] or other MapReduce style frameworks [16]. These two frameworks, that are both open source use of MapReduce model, are planned for specific purposes. Twister for iterative figuring's and Phoenix for multi-processor frameworks with shared memory can be fitting decisions for Hadoop. Nevertheless, the highlight in this paper is on Hadoop, because of its high omnipresence and its inclination of being multi-purpose.

PROPOSED FRAMEWORK

HDFS is likewise utilized for storing images. In HDFS storage, data is put away in the arrangement of text. Data is separated into little pieces (called blocks) and these chunks are disseminated all through the Hadoop cluster. Hadoop gives us the facility to read/write binary files. Accordingly, anything which can be changed over into bytes can be put away into HDFS. This gives the adaptability that is alluring for huge data preparing. To begin with in the wake of social occasion videos from CCTV's, and store it in HDFS database. The accompanying strides are executed to give answers for the above said issue articulation:

1. Write map reduce algorithm to concentrate video sequence level data that is available just in the main chunk of vast video file and store it as content file in HDFS.
2. Write custom recorder to read data upto end of specific Group of pictures in video file utilizing past content output file.
3. Implement LIBMPEG instrument in custom RecordReader to concentrate Image frames.
4. Convert Image binary data into a class that executes writtable like Text class.
5. Convert pictures into hadoop sequence file
6. Write a MapReduce program that read binary Image from sequence file into Byte Array in map function.
7. Convert image in Byte exhibit to IplImage (Image Object in Opencv)
8. Apply your classifier on this picture
9. Draw rectangle on identified face
10. Bundle recognized pictures into single hadoop sequence file
11. Store the sequence file to HDFS.
12. Extract Detected Images from sequence file.

CONCLUSION

The large volume of visual data recently and their prerequisite for proficient and compelling processing invigorate the usage of distributed image processing frameworks in image processing area. So that up to the coming years, various algorithms which have been exhibited in the field of image processing and example acknowledgment should consider the necessities for large scale image processing with a particular deciding objective to be welcomed by the outside world. This paper gives a survey of distributed processing procedures and the programming models. Moreover a couple works are considered which have been done of late using Hadoop open source system. Hadoop and its processing model are as of late surrounded and like some other new developments may have its own issues, for instance, nonappearance of nature of the lion's offer of IT society with it, nonattendance of enough ace forces, and undesirable flaws and issues on account of its peculiarity. In any case, this processing style that uses MapReduce model and distributed image will be among the most significant mechanical assemblies for image processing and example acknowledgment in the coming years as a result of its consistency with distributed computing structures.

REFERENCES

[1] Gartner Reserch Top 10 Strategic Technologies for 2012, http://www.gartner.com/it/page.jsp?id=1826214
[2] Bakshi, K., Considerations for Big Data: Architecture and Approach and Aerospace Conference-Big Sky, MT, 3-10 March 2012.
[3] Miller, K.W and Michael,Big Data: New Opportunities and New Challenges. Journal of IEEE Computer Society, 46, 22-24.
[4] B., Tom, White, Y., Jimmy, L. and Davis, L.S, Web-Scale Computer Vision Using MapReduce for Multimedia Data Mining. Proceedings of the 10th International Workshop on Multimedia Data Mining, Washington DC, 25-28 July 2010, 1-10.

[5] Ghemawat.S.and Dean.J, MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM, 51, 107-114.
[6] The New York Times Blog (2007) ,http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/
[7] Kalagiakos, P. (2011),Cloud Computing Learning. Application of Information and Communication Technologies (AICT). 5th International Conference, 12-14 October 2011.
[8] C., Liu, L., Arietta, S., Lawrence, J Sweeney, HIPI: A Hadoop Image Processing Interface for Image-based MapReduce Tasks. B.s. thesis, University of Virginia (2011)
[9] Hadoop: http://hadoop.apache.org/
[10] http://en.wikipedia.org/wiki/Apache_Hadoop
[11] HDFS, http://hadoop.apache.org/hdfs/
[12] MapReduce, http://en.wikipedia.org/wiki/MapReduce
[13] Bhandarkar, M. (2010) ,MapReduce Programming with Apache Hadoop. Parallel &Distributed Processing (IPDPS) IEEE, 19-23 April 2010.
[14] Kelly, J, Big Data: Hadoop, Business Analytics and Beyond. Wikibon Whitepaper, 27 August 2012. http://wikibon.org/wiki/v/Big_Data:_Hadoop,_Business_Analytics_and_Beyond
[15] Twister, http://www.iterativemapreduce.org/
[16] The Phoenix System for MapReduce Programming, http://mapreduce.stanford.edu/
[17] Qura Question and Answer Website, http://www.quora.com/What-are-some-promising-open-source-alternatives-to-Hadoop-MapReduce-for-map-reduce