International Journal of Electrical and Computer Engineering (IJECE)
Vol.8, No.6, December2018, pp. 5089~5097
ISSN: 2088-8708, DOI: 10.11591/ijece.v8i6.pp5089-5097

Post Event Investigation of Multi-stream Video Data Utilizing Hadoop Cluster

Jyoti Parsola, Durgaprasad Gangodkar, Ankush Mittal
Department of Computer Science & Engineering, Graphic Era University, India

Article Info

Article history:

Received Mar 5, 2018
Revised Jul 4, 2018
Accepted Jul 29, 2018

Keyword:

Hadoop Distributed File System
Map Reduce
Reducers

ABSTRACT

Rapid advancement in technology and in-expensive camera has raised the necessity of monitoring systems for surveillance applications. As a result data acquired from numerous cameras deployed for surveillance is tremendous. When an event is triggered then, manually investigating such a massive data is a complex task. Thus it is essential to explore an approach that, can store massive multi-stream video data as well as, process them to find useful information. To address the challenge of storing and processing multi-stream video data, we have used Hadoop, which has grown into a leading computing model for data intensive applications. In this paper we propose a novel technique for performing post event investigation on stored surveillance video data. Our algorithm stores video data in HDFS in such a way that it efficiently identifies the location of data from HDFS based on the time of occurrence of event and perform further processing. To prove efficiency of our proposed work, we have performed event detection in the video based on the time period provided by the user. In order to estimate the performance of our approach, we evaluated the storage and processing of video data by varying (i) pixel resolution of video frame (ii) size of video data (iii) number of reducers (workers) executing the task (iv) the number of nodes in the cluster. The proposed framework efficiently achieve speed up of 5.9 for large files of 1024X1024 pixel resolution video frames thus makes it appropriate for the feasible practical deployment in any applications.

Corresponding Author:

Jyoti Parsola,
Department of Computer Science & Engineering,
Graphic Era University, Dehradun, India.
Email: jyotee.negi@gmail.com

1. INTRODUCTION

Intelligent video surveillance system (VSS) has evolved as an active study area in computer vision because of its numerous real time applications for social security. It intends to detect, identify and track the object in various video frames or image sequence. The motive behind is to establish an intelligent visual surveillance system and reinstate the traditional surveillance system due to deployment of multiple cameras for continuous monitoring. When an event occurs then the capability to perform scalable and timely analytics to this extensive accumulated data is a high preference for every intelligent VSS. Therefore the major challenges faced by video surveillance system are; a) Storage of continuously increasing gigantic data, generated by the multiple surveillance cameras. b) Prompt processing of progressively rising data when an event is triggered

Processing and storing consistently growing data with conventional network storage and database system is not an easy task. Hadoop, which was originally designed by Google, has evolved into dominant processing model for such applications which are data exhaustive [1]. It's extensible, tolerant to error and splits and copy data, sends the computation where data resides. Hadoop has received so much recognition because of its easy accessibility.

The structure of Hadoop is very rigid so it is not trivial to develop and deploy the complex algorithms to the MapReduce model. Although a lot of research have been performed for video processing [2]-[6] with Hadoop, yet it has not been utilized for post event investigation. Thus the motivation of our work is given as follows:

- Handle multiple streams from various surveillance cameras.
- Storage and timely analysis of extensively accumulated massive data to identify an event of interest.
- Speed up in performance

We have used Hadoop for storage and processing single stream surveillance data on a single node cluster as discussed in [7]. In this paper, we propose a framework for post event investigation as shown in Figure 1, which stores the multi-stream data accumulated from multiple cameras deployed for video surveillance application, into the HDFS. If an event occurs then, user sends query to analyze the required data along with the time duration when the event is suspected to occur. Based on the time duration, the system identifies the location of the data residing in the DataNode in the cluster and computation is executed by Hadoop MapReduce. Our proposed approach for post event investigation of such a massive data, overcomes the need for analysis of entire data generated by the set of video cameras deployed for monitoring purpose thus reducing the computation time.

The remaining paper is arranged in the following way, section II discusses the relevant study performed by various researchers. Section III discusses the architecture of post event investigation with Hadoop and analyses the algorithm proposed for multi-stream video data processing and storage with Hadoop section IV which shows results and performance analysis and section V discusses the conclusion.

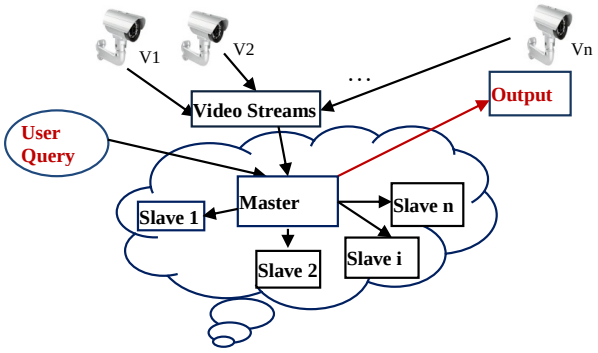


Figure 1. Framework for post event investigation with hadoop

2. RELATED WORK

It has been shown in [1] that Hadoop Mapreduce is appropriate for processing text data which require same computation to be performed in the entire massive data residing in the HDFS. Therefore initially MapReduce was used for the problems like searching, sorting large data, large scale indexing, graph computation [8] matrix computation [9]. Some researchers have tried to use it for image processing and large scale query processing and query search as well [10], [11]. In [12] parallel execution of scattered database is performed. A colored image is converted into grayscale image and parallely features are drawn out. High resolution images are processed and features are removed with Hadoop MapReduce [13]. Hadoop MapReduce framework is also utilized for application like image retrieval based on the content [14]. An image refinement method with MapReduce is discussed in [15]. It needs images to be streamed only once compared to other file system which needs each time entire image or part of image streamed after applying filter. [16] Proposed a distributed image processing system named SEIP, which is built on Hadoop, and employs extensible in node architecture to support various kinds of image processing algorithms on distributed platforms with GPU accelerators. [17] Have used hadoop for clustering categorical data.

Few research investigators have used video data processing [18] (video transcoding) with Hadoop MapReduce framework as discussed by [2]. [3] Performs parallel video analysis and processing whereas video playing, sharing and storage [4] with Hadoop cluster. Hadoop has been used for large video management [5] and for Object detection and classification [6]. The work implemented in [19] is the distributed visual enhancement using histogram equalization algorithm on image database from surveillance cameras. The experiment is conducted in pseudo distributed mode under Hadoop MapReduce architecture.

Hence based on the above literature survey it is evident that Hadoop MapReduce has not been utilized for post event investigation applications and moreover the problem of multiple streams storage and processing of surveillance data is still a challenge.

3. ARCHITECTURE FOR HADOOP AND ANALYSIS OF MULTI STREAM VIDEO DATA USING HADOOP

We have designed our event investigation system based on a MapReduce framework for data storage and data processing. Hadoop is an open source software framework licensed by an apache, it is used for distributed processing [20], [21], [26], [27] and distributed depository of extensive dataset across group of nodes build on low priced computers. Traits intrinsic to Hadoop are data partitioning and parallel computation of large datasets. Its storage and computational capabilities scale with the addition of hosts to a Hadoop cluster, and can reach volume of sizes in the petabytes on clusters with thousands of hosts. It comprises of two principal elements as discussed in [7]. First is Hadoop Distributed File System (HDFS) used for distributed file system, second is MapReduce which is the execution engine or data processing framework as shown in Figure 2.

The analysis of the multi-stream video data using Hadoop is done in three different phases

- Storing the multi-stream video data to the HDFS
- Processing multi-stream video data with mapreduce
- Accumulating all the results and displaying the result.

In VSS, data accumulated from various cameras deployed for monitoring purpose is massive and continuously keep on increasing. Question is to store such an extremely large data. Moreover the issue becomes more complex when an event is triggered and the data is to be processed to extract the useful information regarding any event. Traditional method used for extracting useful information is to check the entire database, which is computationally expensive. There should be some measure where the user can search the particular data based on the time of occurrence of event, instead of searching the entire database.

Therefore for this purpose Hadoop HDFS is used. Data in HDFS is processed in batches. Therefore streams are buffered into local memory and then data is transferred into the HDFS. Moreover Hadoop was originally designed for text processing thus, there is no support in Hadoop for processing video data. We extract frames from video stream and store them as *Sequence file* in the HDFS. *Sequence files* are Hadoop particular archive file layouts very much like to tar and zip. It brings together the file set with a key and a value combination where key is the file name and value is the content of file. The generated sequence file is mostly half the size of the original data and hence takes limited memory area in HDFS converting it storage efficient. These files can be separated and processed in parallel. For video analytic applications like motion detection, rather than comparing every alternate frame it is sufficient to process every alternate fifth frame [22].

3.1. Data Storage

Rather than storing all the video frames we store every alternate fifth frame which further reduces the storage space in HDFS. We use a novel technique for storing the sequence files (frames) using Algorithm 1. Every video camera is identified with a unique identifier like V1, V2,….,Vn while storing the sequence file we generate the name of sequence file by concatenating camera identifier, date, time and frame number, eg. V1_1_07_2016_10_12_22_1 where V1 is the name of the camera or stream, 1_07_2016 is the date (Day_Month_Year format), 10_12_22 is the time (Hour_Minutes_Seconds format) and 1 is the frame number. This approach facilitates appropriate identification of DataNode where the frame has been stored. Thus overcoming the time required to search entire data accumulated so far. This data in the HDFS is separated into blocks (default size of block is 64 Mb) and further stored in various nodes of the cluster. Each block is replicated with 3 copies in the machines of the cluster.

Algorithm 1: Data storage in HDFS

Input: multiple video streams s_n
Output: Sequence file
Step 1: For every video frame (V_s) of s_n
Store V_s with the name as s_n+ date + time+ Frame
number;
Vs +5;
Step 2: Convert stored V_s to sequence file
Step 3: Copy Sequence file to the HDFS

3.2. Data Processing

The Hadoop MapReduce work flow as shown in Figure 3, user enters the query with the time and sends it to the master. In the master Jobtracker divides the job to various tasks and sends it to the Tasktracker residing in the slave nodes. The tasks are executed by the map and reduce function respectively. The process of DataNode identification and data processing algorithm is further discussed in Algorithm 2 Results is accumulated by the TaskTracker and final output is generated. To prove the efficiency of our proposed work we are finding motion in the video data on the basis of the time of occurrence of event as provided by the user. Moving object detection based on motion segmentation is itself a challenge in VSS.

A lot of research have been performed for motion segmentation and has been broadly classified into Background subtraction and temporal differencing. In background subtraction motion [23] is detected by finding the difference between the present frame and the reference background whereas in case of temporal difference [24] motion is determined by calculating the pixel wise difference between the present frame and the earlier frame. The motion detection algorithm as proposed in [25], is used in our system for finding moving objects. One of the efficient methods of frame differencing is block matching, for identifying moving objects. In block matching as shown in Figure 7 the current frame C_k is divided into blocks and similarly previous frame P_k is also divided into blocks and the blocks of P_k are searched into C_k so if a block of P_k is found in different pixel location inC_k it implies that motion is produced.

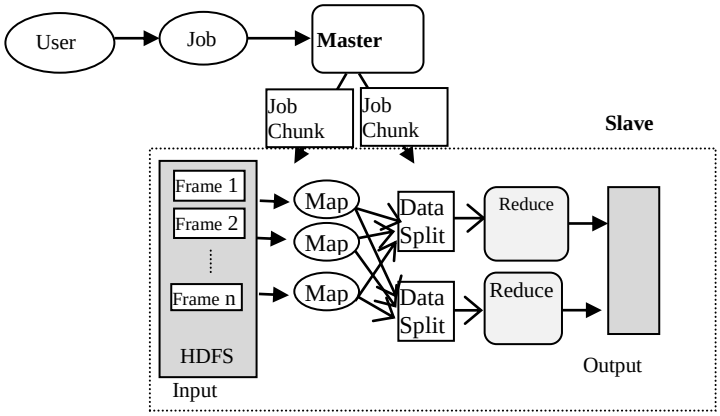


Figure 3. Map reduce job execution flow for motion detection

Algorithm 2: Identification of DataNode & data Processing

Input: Data, time and duration of an event
Output: Resultant motion vector.

Step 1: User sends the query in the form of date and time to the master.
Step 2: Master sends the computation to the JobTrackerand NameNode identifies data location
Step 3: JobTracker splits the job into the TaskTracker
Step 4: The computation is executed further by map and reduces in the data residing in the data node and result is sent back to the TaskTracker.
Step 5: JobTracker accumulates the result from TaskTracker and forwards it to the master

Many techniques have been proposed by various scholars for performing matching computation. Sum of Absolute Difference (SAD) is used in [25] for measuring the difference between the two video frame block, as it is highly efficient. The lower value of SAD means the higher similarity between the two blocks. It is calculated using eq.1.

$$SAD(x_1, x_2) = \sum_{a=0}^{b=N-1} \sum_{a=0}^{b=N-1} |I_p(a, b, t_1) - I_c\,(a + x_1, j + x_2, t_2)| \quad (1)$$

where (x_1, x_2) ={-z ≤ x_1,x_2 ≤ z}

Motion vector (mv) = (x_1, x_2)) |min sad (x_1, x_2)) (2)

In (1) & (2) I_p(,) and I_c(,) displays the intensity of pixels in the earlier and present frame subsequently. SAD(x_1, x_2) is the total value of absolute difference at the pixel location x_1, x_2 [-z, z] is the search area in the search window. mv indicates the motion vector at smallest rate of SAD computed in frames distant with time t_1 and t_2. Dimension of motion block is chosen for 16 x 16 and search window size is of 32 x32 pixels.

The job of computing motion vector in the scope between [-31, 32] into frames of a video is computationally high priced, as a result search is started when, motion block is on the same position of the reference frame i.e. block and search window are coinciding in middle. If there is no change then value of SAD is zero and if a block includes movement then block presents the maximum value of absolute difference SAD_o is computed with the ex.(3):

$$SAD_0 = \sum_{a=0}^{b=N-1} \sum_{a=0}^{b=N-1} |I_r(a,b,t_1) - I_c(a,b,t_2) \quad (3)$$

where a , b indicates position of pixel in earlier (reference) and present frame.
Adding to this a threshold (*th*) is enforced to SADo to decrease processing time. It helps to determine whether to initialize the search or not on the basis of ex.(4).

$$\text{Search Decision} = \begin{cases} 1 & \text{If } SAD_o < th \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

For each block mild *th* can be fixed as part of 256 X 15=3840 where, 256 is 16 X 16 block pixel value. Part value lies within the range of(0.4 ,0 .1).

The resultant is a set of motion vector. Accumulating all the results for every frame of a video final motion is plotted. In order to find out the motion detection we use SAD to detect the motion in the video frames as aforementioned. Map function reads two frames as an input and splits each frame into 32 by 32 pixel size blocks and each block is assigned a key and value containing the 32 by 32 block and this output is called as intermediate data. Each key value pair is passed to the reduce function in such a manner that the values containing the same key is passed to the same reducer. The task of motion detection is performed by the reduce function and it finds the moving object based on motion segmentation.

3.3. Result Accumulation

Finally all the results computed by various reducers for all the blocks of the video frame are accumulated a final output is obtained displaying the moving object on the corresponding video frames. Algorithm 3 shows the data accumulation process. This approach can be used to detect the event in multiple streams where the possible time of occurrence of the event is provided by the user. It can be observed from the above explanation that our proposed approach can achieve the following

- Efficient Storage of multi-stream video data accumulated from numerous cameras deployed for surveillance into the HDFS.
- Extract data based on the time of occurrence of event provided by the user.
- Analytics of the massive data with MapReduce in short time.

Algorithm 3: Data Accumulation
Input: Motion Vector for every frame
Output: Moving Object
Step1: Motion vectors are obtained for every set of video frames.
Step 2: Results are accumulated.
Step 3: Motion vectors are plotted according to the user query

4. RESULTS & PERFORMANCE EVALUATION

We have analyzed performance of our proposed framework in the following manner;

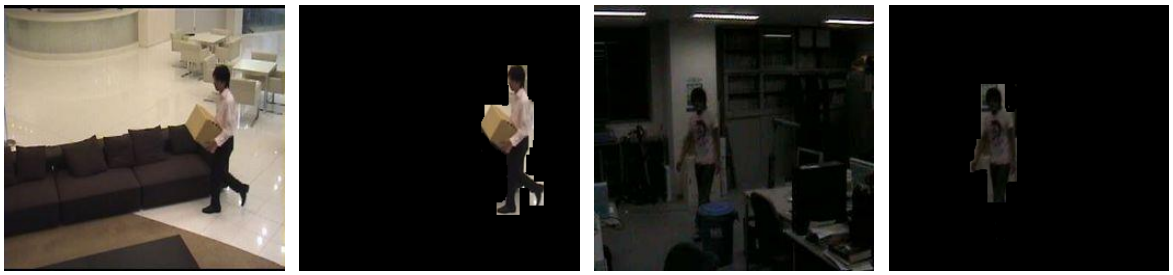
- By measuring the computation time for varying

1) number of reducers (workers)
2) number of nodes building cluster

- Computational efficiency for higher pixel resolution video frames by varying the size of video frame.

(a) (b) (c) (d)

Figure 4. *Shadow* [28] (a) original frame (b) segmented moving object and *Baseline* [28] (c) Original frame (d) Segmented moving object



(a) (b) (c) (d)

Figure 5. *Intermittent Object Motion* [28] (a) original frame (b) segmented moving object and *Ground Truth* [29] (c) Original frame (d) Segmented moving object

(a) (b) (c) (d)

Figure 6. *CAVIAR_Meet_WalkTogether1* [30] (a) Original frame (b) Segmented moving object and *CAVIAR_ walk3* [30] (c) Original frame (d) Segmented moving object

The proposed work is implemented on Intel core i5 3.10 GHz with 4 GB of memory on Ubuntu 12.04 as an operating system Hadoop version is 1.2.1. We have used 5 and 10 nodes cluster for performance evaluation. We have evaluated performance by varying file size and data size of a cluster and its affect on the computation time. Detection of motion is driven on video frames (grayscale) with pixel size 256 X 256, 512 X 512 and 1024 X 1024. Colored video frame are converted to grayscale before processing. Calculated motion detection on

various video sequences is displayed on figure 4 to 6. JavaCV which is wrapper for OpenCV library [31] is used to plot motion vector. Figure 4 to 6 shows the original frame and the corresponding segmented frames where motion is identified. Experiment is computed on standard dataset accessible openly Change Detection Benchmark [28], Laboratory for Image & Media Understanding (LIMU) [29], Context Aware

Vision Using Image-Based Active Recognition (CAVIAR) [30]. For analyzing MapReduce performance for motion detection algorithm, Hadoop MapReduce Cluster is established. The cluster consists of 9 salve nodes and 1 master node.

Our proposed framework efficiently reduces the storage space in HDFS and the results of data size reduction are shown in Table 1. First column of the table represents the original data size, second column is the data size which is reduced when only alternate fifth frame is stored the data reduction produced is about 80 -85 % and third column displays the data size reduction achieved by the compression due to the sequence file generation and the compression is about 80%. The result clearly shows the efficiency of our approach in terms of storage.

Table 1. Storage space reduction in HDFS.

Original Data Size	Reduced Data size	Sequence File compressed data
500 MB	100 MB	20 MB
1 GB	204.8 MB	40.96 MB
1.5 GB	307.5 MB	61.6 MB
2 GB	409.6 MB	81.92 MB
2.5 GB	511 MB	103.2 MB

4.1. Performance Evaluation on Multi Node Cluster

1) Analysing task execution time with varying number of nodes in the cluster.

The extensibility and robustness of the framework is evaluated by analysing the multi stream video data on various nodes of the cluster. Experiment is executed with different number of nodes to be able to understand speed up. Parallel speed up $S_p$is measured as given by e.q. (5)

$$S_p = \frac{T_1}{Tn} \quad (5)$$

where T_1 is the total execution time calculated in one node cluster and T_n is the total execution time calculated in n node cluster were n > 1. value of S_p shows the number of times parallel execution is faster than running the same MapReduce algorithm on the single node cluster. If it is greater than 1, it entails that there is at least some gain from doing the work in parallel. Execution time for video frames of pixel resolution 256 x 256, 512 X 512 and 1024 X 1024 in sequential (a simple java program) and MapReduce cluster of various nodes and computed speed up is shown in Figure 7. The processing time is the total time to calculate motion detection in the required data size and we have searched 100 MB data in the HDFS as well as in sequential and further performed motion detection in the respective data.

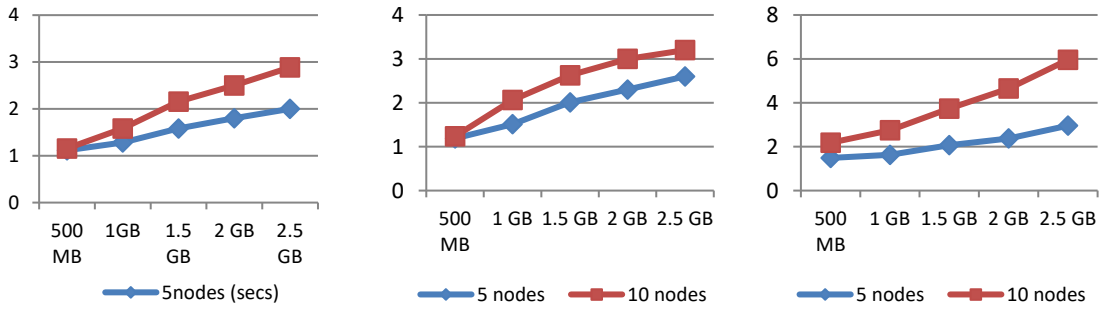


Figure 7. Speed up for motion detection algorithm of a) 256 X 256 pixel resolution video frame with different number of nodes in a MapReduce cluster b) 512 X 512 pixel resolution video frame with different number of nodes in a MapReduce cluster c) 1024 X 1024 pixel resolution video frame with different number of nodes in a MapReduce cluster.

4.2. Analysing Task Execution Time by Varying Number of Reducers (Workers) Performing the Job

We have also analyzed the performance of motion detection algorithm by varying the number of reducers (workers). Figure 8 shows the outcome of different number of reducer for various volumes of data and various pixel size video frames. We also tested execution time by varying map tasks but results were not

remarkable. Execution time for smaller data volume is almost similar but for larger data volume reduction in processing time is achieved considerably. The table clearly shows that it is not necessary that

- For low resolution video frame 250- 300 reducers on an average provides good results.
- For high resolution video frame 500- 700 reducers on an average provides good results.

Thus this gives prior information to set the number of reducers for computation as finding the number of reducers providing efficient result is a tedious task.

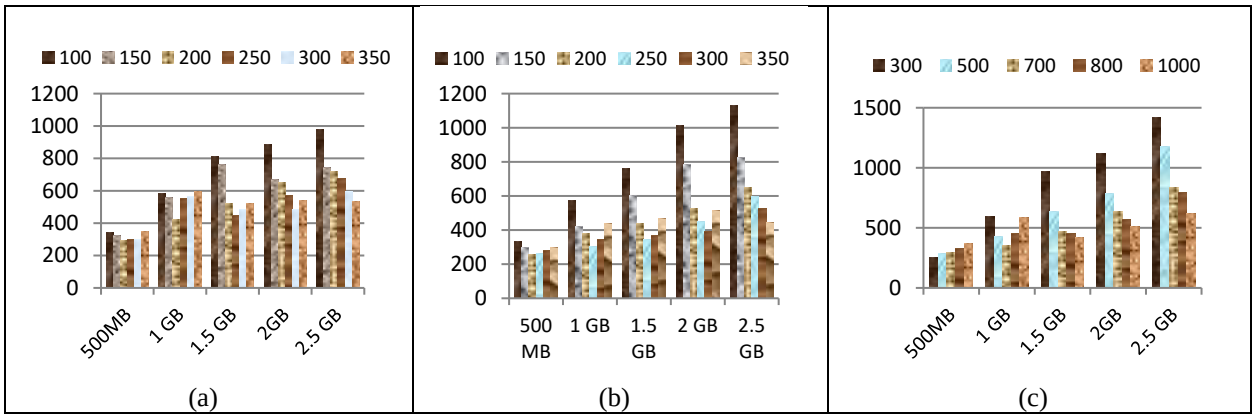


Figure 8. Motion detection computation time for (a) 256 X 256 pixels (in seconds) size video frame of various data size with varying number of reducers, (b) 512 X 512 (in seconds) pixel size video of various data size with varying number of reducers, (c) 1024 X 1024 (in seconds) pixel size video frame of various data size with varying number of reducers

5. CONCLUSION

We have proposed and implemented an efficient approach for performing post event investigation on massive volume of surveillance data which is one of the challenges of Video Surveillance system. We have used Hadoop HDFS for distributed storage and Hadoop MapReduce for parallel and distributed processing of massive accumulated multi-stream video data. We have proposed an algorithm for efficient storing video data in the HDFS. Hence when an event is triggered we automatically extract data based on the time of occurrence of event and process it further to find useful information. To prove the competence of our proposed approach in handling and processing extremely huge data, we have implemented motion detection algorithm in Hadoop cluster.

Hadoop cluster consists of maximum of 10 nodes. Our experiment result precisely indicates that the computing period is shortened, when pixel resolution of video frame is increased. We also analyzed the performance by measuring the computation time for varying number of reducers (workers). Network latency also affects the execution time in a cluster. To solve this issue execution time can be further improved. Moreover through the increment in number of nodes of a cluster, computation time can be cut down more. Our framework is robust and can cope with varying number of nodes in the cluster as well as increasing data volume. Hadoop performs excellent for application which need similar task to be performed in distinct data sizes; hence application requiring different jobs to be performed in various data sets in aligned manner is not possible with Hadoop MapReduce.

REFERENCES

[1] J. Dean and S. Ghemawat, "Map Reduce: Simplified Data Processing on Large Cluster", *ACM Commun*., vol. 51, no. 1, pp. 107–113, 2008.
[2] R. Pereira, K. Breitman, "A Cloud Based Architecture for Improving Video Compression Time Efficiency: The Split &Merge Approach", *In 3rd IEEE Int. Conf. on Cloud Computing (CLOUD)*, pp. 482 – 489, 2010.
[3] H. Tan and L. Chen, "An approach for fast and parallel video processing on Apache Hadoop clusters", *IEEE Int. conf. on Multimedia and Expo (ICME)*, vol. 1, no. 6, 2014.
[4] C. Liu, *et al.*, "A distributed video share system based on Hadoop", *in IEEE 3rd Int. Conf. on Cloud Computing and Intelligence Systems (CCIS)*, pp. 587-590, November, 2014.
[5] X. Liu, *et al.* "A Distributed Video Management Cloud Platform Using Hadoop", *IEEE Access*, vol. 3, no. 1, pp. 2637-2643, 2015.

[6] Yaseen, *et al.,* "Cloud-based scalable object detection and classification in video streams", *Future Generation Computer Systems*, vol. 80, pp. 286-298, 2018.
[7] J. Parsola, *et al*., "Efficient Storage and Processing of Video Data for Moving Object Detection using Hadoop MapReduce", *in Int. Conf. on Signal, Networks, Computing and Systems (ICNCS-2016),* JNU, New Delhi, India, 2016.
[8] J. Cohen, "Graph twiddling in a MapReduce world", *Computing in Science & Engineering*, vol. 11, no. 4, pp. 29-41, 2009.
[9] J. Myung, S.G Lee, "Exploiting inter-operation parallelism for matrix chain multiplication using MapReduce", *J. on Super Computing.* vol. 66, no.1, pp. 594-609, 2013.
[10] H. Wang, *et al*., "Efficient query processing framework for big data warehouse: an almost join-free approach", *Frontiers of Computer Science*, vol. 9, no. 12, pp. 224-236, 2015.
[11] J. Ahn *et al*. "SigMR: MapReduce based SPARQL query processing by signature encoding and multiway join", *J. on Super Computing*,vol. 71, no.10, pp. 3695-3725, 2015.
[12] M. Yamamoto and K. Kaneko, "Parallel image database processing with mapreduce and performance evaluation in pseudo distributed mode", *Int. J. on Electronic Commerce Studies*, vol. 3, no. 2, pp.211-228.
[13] H.D Zhu, *et al*., "Parallel Image Texture Feature Extraction under Hadoop Cloud Platform", *Intelligent Computing Theory. Springer Int. Publishing*. 459-465.
[14] W. Premchaiswadi, *et al*.," Improving performance of content-based image retrieval schemes using Hadoop MapReduce", *IEEE Int. conf. on High Performance Computing and Simulation (HPCS)*, pp. 615-620, 2013.
[15] T.D. Gamage, *et al*.," Image filtering with MapReduce in pseudo-distribution mode", *IEEE conf. on Moratuwa Engineering Research Conference (MERCon)*. 160-164, 2015.
[16] T. Liu, *et al*., "SEIP: System for Efficient Image Processing on Distributed Platform", *Journal of Computer Science and Technology*, vol. 30, no. 6, pp. 1215-1232, 2015.
[17] G. Bathla, *et al*., "A Novel Approach for clustering Big Data based on MapReduce", *Int. J. of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 3, pp. 1711-1719, 2018.
[18] A. Anjum, *et al*., "Video stream analysis in clouds: An object detection and classification framework for high performance video analytics", *IEEE Transactions on Cloud Computing*, 2016.
[19] A. Azli, "Distributed visual enhancement on surveillance video with Hadoop Mapreduce and performance evaluation in pseudo distributed mode," *Australian J. of Basic and Applied Sciences, vol.* 8, no.9, pp.38,2014
[20] P.S.G Aruna Sri and M. Anusha, "Big data survey", *Indonesian Journal of Electrical Engineering and Informatics*, vol. 4, no. 1, 2018.
[21] S.A. Thanekar *et al*., "A Study on MapReduce: Challenges and Trends", *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 4, no.1, 2016.
[22] D. Gangodkar *et al*. "Robust segmentation of moving vehicles under complex outdoor conditions", *IEEE Transactions on Intell. Transp. Sys.*, vol. 13, no. 4, pp. 1738-1752, 2012.
[23] W.E.L Grimson and C. Stauffer, "Adaptive background mixture models for real-time tracking", *In IEEE Conf. Comput Vision and Pattern Recognition*, vol. 1, pp. 22–29, 1999.
[24] Z. Yu and Y. Chen, "A real-time motion detection algorithm for traffic monitoring systems based on consecutive temporal difference", *In7th Conf of Asian Control Conference (ACC)*, pp. 1594-1599, 2009.
[25] D.Gangodkar *et al*. "Segmentation of moving objects in visible and thermal videos", *Int. Conf of Computer Communication and Informatics (ICCCI)*, pp. 1-5, 2012.
[26] White. T.: Hadoop: The Definitive Guide. Yahoo Press (2010)
[27] Holmes, A.: Hadoop in practice. Manning Publications Co.(2012)
[28] Change Detection Benchmark. Available: http://wordpress-jodoin.dmi.usherb.ca/dataset2014/
[29] Laboratory for Image & Media Understanding (LIMU). Available: http://limu.ait.kyushu-u.ac.jp/dataset/en/
[30] Context Aware Vision Using Image-Based Active Recognition (CAVIAR). Available http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/
[31] Open Source Computer Vision (OpenCV) [Online]. Available: http://opencv.willowgarage.com/wiki/