Logotype Detection in Streaming Multimedia Using Apache Storm

José Herrera, Germán Moltó
Instituto de Instrumentación para Imagen Molecular (I3M)
Centro mixto CSIC - Universitat Politècnica de València
Camino de Vera s/n, 46022, Valencia, Spain
email: jherrera@upv.es, gmolto@dsic.upv.es

Atsuhiro Takasu
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku,
Tokyo, 101-8430 Japan
email: takasu@nii.ac.jp

***Abstract*—The massive amount of multimedia information currently available through the Internet requires efficient techniques to extract knowledge from data. Indeed, processing video information using real-time Big Data techniques is currently a challenge for the existing data processing frameworks. This paper proposes and evaluates different topologies for logotype detection in streaming multimedia implemented by means of open-source libraries and supported by the *Apache Storm* distributed real-time computation system. Different data movement strategies across the topologies are defined and evaluated. An experimental analysis that involves logotype detection in an excerpt of the Formula One Suzuka Japan Grand Prix 2015 is performed. The results indicate that logotype detection in videos can greatly benefit from the distributed computing capabilities offered by Apache Storm.**

***Index Terms*—Big Data; Apache Storm; multimedia; logotypes; Bloom Filter**

I. Introduction

Video Analytics is one of the areas that is a challenge for Big Data due to the use of highly unstructured data. Indeed, the use of video has dramatically increased during the last years within our information society. As an example, the statistics published by YouTube in [7] indicates that a third of the people on the Internet has a YouTube account, watching hundreds of millions of hours on that platform and uploading more than 400 hours of video every minute [6].

Indeed, video analytics for trademark management is one of the areas that can significantly benefit from Big Data processing. The total amount of logotypes worldwide is estimated to exceed 20 millions and the number trademarks raises at a rate of seven millions per year [12] according to data of World Intellectual Property Organization (WIPO). The ability to track the appearance of a brand in a video enables to quantify its level of exposure to the media for a certain event.

In this paper, we propose a topology based on *Apache Storm* to process large amounts of multimedia data from streaming video. This topology is applied in order to extract knowledge concerning the appearance of brand logos in the multimedia streams. We aim at processing this data on-the-fly, without previously storing the data, by using a distributed computing framework. This approach has many applications in different professional fields which include, but are not limited to, advertising companies to track the fulfilment of the advertising contracts in live sports events

After this introduction, the remainder of this paper is structured as follow. First, section II describes the related work in the area. Second, section III briefly introduces the *Apache Storm* framework and its ability to process tuples in real-time, we also describe the libraries and algorithms employed to solve this challenging problem, together with the topologies implemented in order to address the case studies envisioned. Third, section IV assesses the effectiveness of the proposed implementation through case studies and highlighting some useful steps to take into account for the topology implementation. Finally, section V summarizes the main contributions of this paper and points to future work.

II. Related Work

Video processing is a computationally and data intensive task. Indeed, a second of high-definition video, in terms of amount of data, is equivalent to 2.000 pages of text, according to Anyika et al. [25]. Gandomi and Haider [16] discuss surveillance, retail and video indexing for common video analysis applications using Big Data techniques. They discuss two architectural approaches to video analytics: i) server-based architecture where multiple videos are captured and routed back to a server that carries out the video analytics and ii) edge-based architecture, where the video analytics are applied at the device that captures the video. Application fields are diverse. As an example, Gantz and Reinsel [17] propose improving military intelligence comparing correlation pattern in videos captured from drones.

Indeed pattern recognition is wide research area in which trademark detection in images has been previously addressed [21] [27]. Pattern recognition for logotype detection in video is also an area of study that aims at detecting logotypes [10], detecting and removing [28], and even detecting logotypes in low resolution video[11]. Also, new technologies are being applied to address this problem, such as using deep convolutional neural networks in the work by Iandola et al. [19], surpassing state-of-the-art accuracy on popular logo recognition datasets.

Hu et al. exposed in [18] that multimedia is one of the six types of Big Data applications and it involves overcoming the semantic gap of multimedia data. Indeed, there are other works

in the literature that aim at analysing multimedia data using Big Data Techniques. The work by Weishan et al. in [30] focuses on stream processing with others purposes such as proposing a combination between batch processing and real-time processing. They state that effectiveness and efficiency requires using both off-line batch processing capabilities and on-line real-time in-memory processing. They use *Apache Storm* for glasses detection analyzing the detection results and the framework performance in [14]. Processing images for face detection is the proposal of Dong-Hyuck et al. [20], where the Storm distributed framework is employed. However, no execution results are included in the paper.

The work by Zhang et al. [29] use similar techniques for image processing but the results shown are focused on the detection process rather than on the platform employed. Finally, the books by Allen, Pathirana and Jankowski [8] together with the book by Leibiusky, Eisbruch and Simonassi [23] on Apache Storm have been fundamental to underpin the foundations of this work.

As a final note, Kesidis et al. in [22] expose open challenges in trademark retrieval. Due to the number of trademark registered it is necessary to consider methodologies based on distributed systems that can scale gracefully. There is substantial variability in trademarks including geometric, monochrome, stylized or plain variations.

III. Materials and Methods

This section describes the frameworks, libraries and methods employed to tackle the problem of brand logo detection on multimedia streaming data.

A. Apache Storm

Apache Storm [1] is an open source distributed real-time computation system that can process streams of data. A topology, or Storm application, consists of *Spouts*, *Bolts*, *tuples*, *streams* and *stream grouping* specifications for transfers. Spouts are sources of stream data that read tuples and emit them into the topology, i.e., to be processed by the bolts. Bolts process the received tuples performing different functions such as filtering, aggregation, join, etc. A tuple is a data model defined for a named list of values, and a field can be an object of any type. A Stream is a set of tuples in sequence. The grouping specification is the way that a group of objects is transferred to the next step in the processing carried out by a Storm topology. Finally, stream grouping specifies how topology sends data between two components. Spouts and bolts execute in parallel, as part of the topology, across one or more worker processes. Storm attempts to spread the load, i.e., tasks to be executed, evenly across the workers.

To understand parallelism in *Apache Storm* it is convenient to know that every Bolt may run in a different level of parallelism. These are the main components:

- *Workers*. They execute a subset of a specific topology running its own JVM. A topology runs across one or more workers. By default, every node has four workers.
- *Executors*. An executor is a thread of execution that is spawned by a worker process and run in the worker's JVM. It runs one or more tasks for the same bolt or spout.
- *Tasks*. A task performs the actual data processing. It runs in an executor's thread and shares this thread with other tasks. An executor is used by all tasks in the same Bolt or Spout in the same worker.

As a final note, *Apache Storm* can run several topologies at the same time providing a web user interface to control the running cluster. The important attributes of this framework are: simple programming model, support for multiple languages, fault-tolerant, transactional, scalable, reliable and fast.

B. Image Detection Library

An open-source image detection library is used to find the logotypes in the images extracted from the videos. There are two main open-source free vision libraries to process images. On the one hand, OpenCV (Open Computer Vision) [2] is library written in C/C++ designed for computational efficiency and strongly focused on real-time applications. It is a well documented and widely used project. More than 500 functions are implemented in OpenCV to cover areas such as robotics, security, medical imaging or factory product inspection. It also includes a general-purpose Machine Learning Library (MLL) focused on statistical pattern recognition. On the other hand, OpenIMAJ (OPEN Intelligent Multimedia Analysis in Java) [3] is a Java library and tool for scalable multimedia content analysis and image indexing. It includes broad state-of-the-art computer vision techniques. The distribution is made using a modular set of *jar* files under a BSD-style license. The design and implementation keeps all the components modular to maximise code maintainability.

Both libraries employ similar algorithms to detect images. With all these similarities, we selected OpenIMAJ to implement our code that use the Scale-Invariant Feature Transform (SIFT) [24] to extract some interest pixels from images and describe them. To find analogue pixels, we use Random Sample Consensus (RANSAC) [15] to fit a geometric model called an Affine Transform to the initial set of matches. After the pattern detection process, we obtain a number of matches or similarities. Closeness can be compared with a threshold. If the number is greater, we have a frame that contains the logotype. Notice that the goodness of the systems greatly depends on the goodness of the library.

C. Probabilistic structures operation

In order to prevent counting duplicate logotypes, a mechanism to discover if detection was made previously is required. One way to perform this is using the Java Sets available in the Java Development Kit (JDK) to store the previously processed frames. However, this procedure is not ideal when using a Storm topology due to constraints in the size of the set and the speed required to perform such detection. Instead, a Bloom filter [9] (BF) is a space-efficient probabilistic data structure. It is based on the use of hash functions to define data codification

in a bit array. It is employed to test whether an element belongs to a set. Elements can be introduced but not removed. It is important to point out that false positives are possible, but false negatives are not. There exist some variations from the original Bloom Filter. One of the most interesting, for the purpose of this paper, is the Counter Bloom Filter [13] (CBF) which allows to delete information previously inserted in the Bloom filter.

In order to understand the benefits of the Bloom filter compared to the existing implementations of sets available in the JDK, Figure 1 shows the total execution time, in milliseconds, for different operations, comparing the CBF with respect to the aforementioned set implementations from the JDK.

It can be noticed in figure that CBF does not achieve the best time performance. However, CBF excels at memory consumption. If we compare the memory required by the data structures, as shown in Table I, then we observe a significant memory reduction by the CBF compared to other implementations.

Items	Hash Set	Tree Set	Linked Hash Set	Bloom Filter
1,000	70	70	70	1
10,000	711	711	711	19
100,000	7,215	7,215	7,215	195
1,000,000	73,133	73,133	73,133	1,954

TABLE I
STRUCTURE SIZE TO STORE DATA (IN KB)

Therefore, we adopted CBF as the underlying data structure to detect duplicate logotypes during the video processing.

D. Proposed Topologies

To provide a code implementation independent from the image processing library used, we developed a wrapper with a well defined interface which isolates the topology steps from the library used. This way, only four methods are implemented: *load_frame()*, *load_logotype()*, *process()* and *get_total_points()*.

Concerning testing purposes, we developed three different code implementations. Topologies #1 and #2 are full *Apache Storm* topologies, while #3 code is a sequential execution that is used for comparison purposes. These are the implementations:

- Topology #1. **Transfer references in the tuples**. This first topology proposal is composed by only one spout process and several bolts, as explained in section III-E. In this case, the spout reads a video file, separates it into individual images and stores them into a central node that will act as a repository of images. After saving the images, it sends image references to the next bolt creating a stream of images with additional information (video origin and frame number) to be processed. The first goal in this proposal is to reduce the data exchange between the topology elements. Since only the image reference is transferred, bolts that use image bitmaps need to access this centralized node to retrieve the images. This way, data exchanges between topology nodes are clearly reduced at the expense of increasing the load of reads in the node with the repository of images.
- Topology #2. **Transfer images in the tuples**. To improve processing speed, we created a topology using the same directed acyclic graph but, in this case, the data transferred between topology elements includes transferring the images content between executors by inserting the image serialization bits into the tuples. Under these conditions, the Spout splits the video file, serializes the images using Java serialization and sends a tuple with the image serialization bits and the same additional information as in the previous topology. An image processing bolt, deserializes it, processes the image and serializes it again. Data exchange is increased as well as the processing time dedicated to serializing and desearializing the image data. However, no access to the image repository node is required in order to retrieve the images.
- Code #3. **Sequential Execution**. In order to compare the distributed Storm-based executions with a sequential execution, we developed a code that executes on a single processor (with multiple cores) and uses threads to obtain and process the video data.

E. Topology composition

Both topologies, #1 and #2, have the same directed acyclic graph, spout and bolts. We designed a five mandatory step topology adding two optional bolts to control topology operation.

1) **VideoSpout** (Compulsory Spout). This step creates a stream of data. In our case, this stream of tuples is a source of frames (images) or references that compose a video. In Topology #1 the stream only refers to the frames. In Topology #2, we create a stream of objects called *image_enhanced* that contain the serialized image bits and frame number. For testing proposal these streams are created from a video with MPEG format (ISO/IEC JTC1/SC29 WG11).
2) **PreProBolt** (Compulsory Bolt). The pre-processing bolt is the first processing step in our topology. For example, image segmentation, black/white colour or border detection are processing options to improve the algorithm used. The key goal is to adapt and adjust the input frame to obtain the best results. Content analysis algorithms work better using images under certain rules. In this work, only a size reduction was made since the image size clearly dominates the total execution time. The final result of this bolt is an image properly adapted to the detection algorithm used.
3) **CartesianBolt** (Compulsory Bolt). In sport broadcasts, logotypes appear on different positions and with different designs. Corporate image manuals have available trademark images with alternative horizontal and vertical orientations, with different shape and necessary adjustments of colour, shape or lighting. Alternative designs can appear on a variety of places to increase the scope

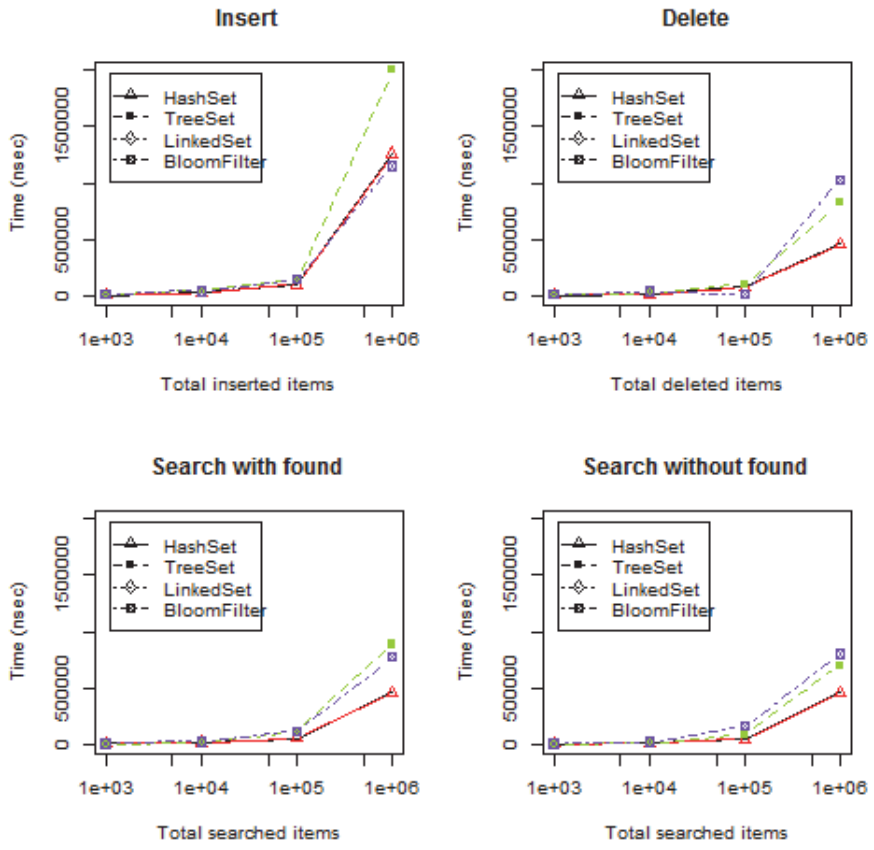


Fig. 1. Comparing Counter Bloom Filter Execution with others Set implementations in Java (all time expressed in nanoseconds).

of advertising. For our purpose, this may hinder the ability to recognize the logotypes. To overcome this problem, we take a sample of each of the different designs. The sampling number is only conditioned to the abilities of the recognition algorithm. Making in this bolt a cartesian product of input tuples (source image) and logotypes samples, we create an output tuple stream of pair image-logotype. This significantly increases the number of recognitions at the expense of multiple logo detections.

4) **FindImageBolt** (Compulsory Bolt). In this part of the topology sequence, pairs image-logotype are delivered to this bolt to be processed. After getting both images, the content analysis algorithms are applied. Then, after the use of SIFT and RANSAC algorithms provided in OpenIMAJ library, it is obtained a degree of similarity that is compared with a global threshold parameter. The major computational complexity of this step is due to the recognition library.
5) **BrandOccCountBolt** (Compulsory Bolt). As a result, this bolt writes a line for each recognized trademark. Listing 1 shows the usual output that includes a timestamp, the file processed, the frame number and the logotype detected. Sometimes, specially in topology executions that involve heavy executions, some pairs of image-logotype are processed multiple times. This is because of losing the tuple delivery acknowledge due to very large processing times (timeouts in the topology) in the FindImageBolt step together with overloads in the hardware running the Storm workers. As a result, two positive detections could be received in this bolt. To solve this issue, we implemented a window scale control using a Counter Bloom Filter that avoids writing several times the same frame. Also, in case of delayed old frames, it filters out them using the same procedure.

Listing 1. Output unordered sample

```
1447806008922 japan_gp_15_split.mpg 00000053 pirelli
1447806011650 japan_gp_15_split.mpg 00000006 amd
1447806012235 japan_gp_15_split.mpg 00000054 pirelli
1447806013423 japan_gp_15_split.mpg 00000042 amd
```

6) **InstantCountBolt** (Optional Bolt). As an auxiliary process to describe the way the topology is working, this Bolt summarizes the video frames that are processed in the topology and the number of total positive logo recognitions during the last 10 seconds. It is not a required bolt but it helps us to understand the execution progress of the topology.

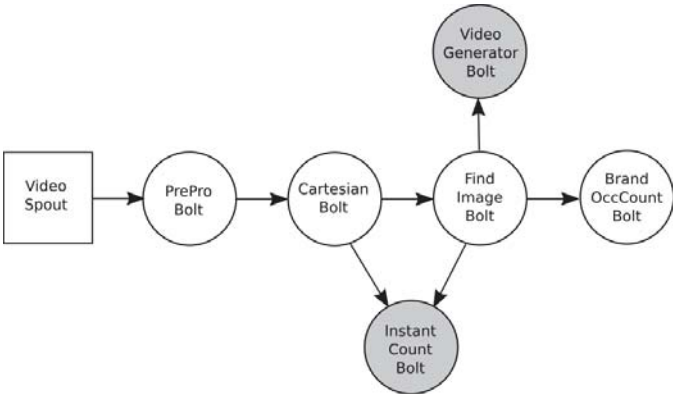


Fig. 2. Acyclic graph of implemented topologies #1 and #2.

7) **VideoGeneratorBolt** (Optional Bolt). As in the previous case, this optional bolt is used to analyze the work made by the topology. Using the input from previous bolts, it creates a motion picture. With data processing results, existing brand and image processed, it joins all pictures. After that, we analyse the sequence to understand the process made and to debug problems and results from the *FindImageBolt* phase.

IV. Configuration, Testbed and Measuring Experiments

To evaluate the effectiveness of the system designed, we selected an excerpt of a pre-recorded video from the Formula One Suzuka Japan Grand Prix 2015. A total of 500 frames were selected from a bigger stream in which the frame size is 1280x720 pixels. Using a speed of 50 frames per second, the slice has a duration of 10 seconds.

We tested black and white image transformations on a sample dataset. However, no significant differences were obtained in the ability to recognize the logotypes. Therefore, all images were created, stored, transformed, serialized or deserialized used True Colour (24 bits) colour depth.

Images are scaled down for some tests maintaining the aspect ratio. In particular, 1.280x720 images are scaled down up to 500x281 pixels. In the figures, we only indicate the width values for the images, in pixels.

A. Experiment 1

For our first experiment, the Storm cluster is composed by 4 Virtual Machines (VMs) with 2 vCPUS (emulating the Intel Xeon E312xx processor), 15 GB of RAM and 30 GB of hard disk. All nodes are running Apache Storm except one that also runs Zookeeper [5] and a master node daemon (so-called internally Nimbus). The Apache Storm version used is 0.9.5 both for the development and execution environments, Zookeeper version 3.4.6 and Zeromq version 2.1.7.

1) Execution Results: The first test evaluated the data processing capabilities of the topologies. Figure 3(a) shows the time required for the execution of the three topologies varying the frame size, using a 500 frames stream (a ten seconds video). The worst topology was T. Data (Topology #1, see section III-D), where only the references to the images are transferred between bolts. Sequential Execution #3 and Topology #2 (Transfer Image) have similar execution times. However, the sequential execution ran in one node with 2 cores and Topology #2 ran in the Apache Storm cluster (four nodes each with 2 cores).

Table II includes the results when the frame size for Topology #2 is changed. It evidences that for a 500 pixels frame size it only processes 1 fps. If the input stream is HQ 50 fps, we need to increase the size of actual test infrastructure (4 nodes) until 50, assuming that the problem scales linearly.

	500	600	700	800	900	1,000	1,100
Speed	1	0.83	0.71	0.63	0.56	0.50	0.45

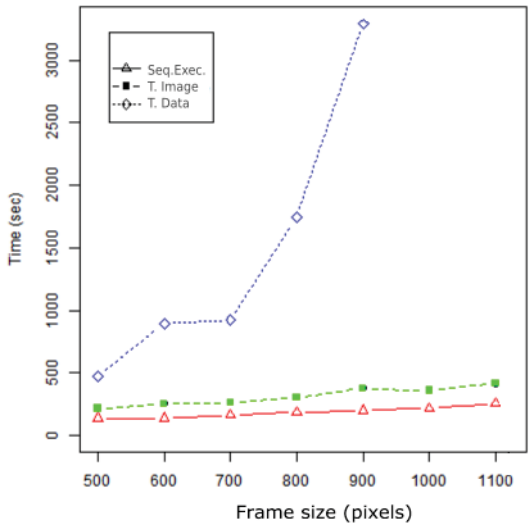
TABLE II
Number of frames per second (fps) processed by the Topology #2 for different frame sizes (in pixels).

B. Experiment 2

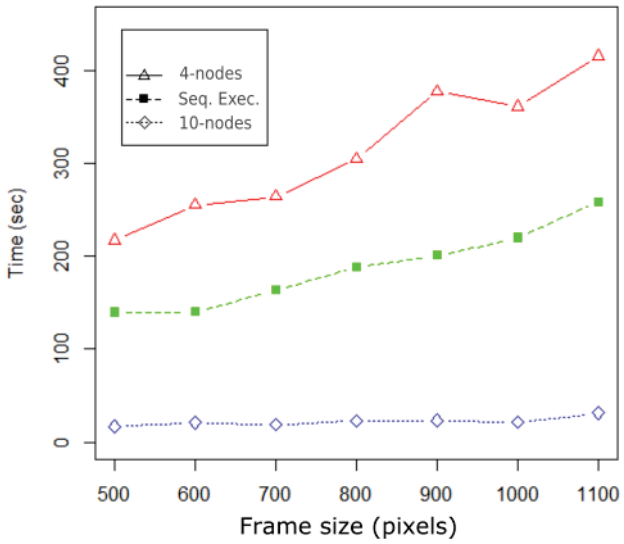
The second experiment is designed to evaluate the scalability of the topology when the number of nodes is increased. A total of 10 nodes were configured in order to evaluate and compare time with previous experiments. The nodes have the same computing performance as those used in Experiment 1. The test set used has the very same frame size and total number of pixels as the one previously used. We limited the measurements to the Topology #2.

1) Execution Results: In Figure 3(b), we compare the execution of Topology #2 with the sequential execution (Sequential Execution Code #3) for different frame sizes and runtime with different number of nodes (4 and 10 nodes) for the Apache Storm cluster. We evaluated the execution time to process the dataset using a limited image range from 500 pixels to 1.100 pixels. No memory problems or limitations were detected in the 10 nodes execution.

As expected, the best results are obtained in a 10-nodes infrastructure. This indicates that the topology scales accurately. Also we observe that transfering data in tuple makes the topology less sensitive to changes in the frame size.



(a) Execution time of the different topologies.



(b) Comparing #2 topology execution speed using ten nodes, four nodes and a Sequential Execution.

Fig. 3. Execution results with different frame sizes across several topologies and configurations.

C. Discussion

It is important to identify the limiting factors that have a direct impact on the execution time of the topologies proposed.

1) **Video frame size**. The most limiting factor is the size of the problem. High resolution is required to find the largest possible number of trademarks. If the image size is too small, no logotypes can be detected since the resolution is not high enough to distinguish these logos. However, bigger videos increase the processing time.
2) **Total nodes**. A reduced number of nodes in the Storm cluster increase the execution time. To achieve a real-time logo identification procedure, a suitable number of nodes are required to process frames existing in every time period. As identified in Table II, 1.4 s. are required to process a 700 pixels frame.
3) **Pre-processing and post-processing works**. Preprocessing is a big limiting factor. Converting the frame to black/white or modify frame size before recognition limits the system speed. At the same time, post-processing data mining techniques, made after leach logo search, limits process output throughput. For that, we need a new bolt to ensure that logotypes are counted only one time for every frame.
4) **Transfer data limitations**. Other related factor is transferring data between bolts. Tuples were transferred between nodes to perform video processing. In one of our topologies only the image references were transferred. The pointers to real image node/position was moved between nodes. In the second topology, streams are composed by images inside an instance of a class prepared for this target. In all cases tuples are serialized and deserialized for each internode transfer. As a result, the second option reveals to be a better and faster approach.
5) **Total logotypes to search**. Tests and observation tell us that one image brand not only appears in one logotype. There exists different colors, shapes and sizes for each one. Perhaps, we want to identify two different logos in only one broadcast. To solve this difficulty, we create a group of logotypes associated to a brand that is detected at the same time but in different bolts.
6) **Total frames pending to be processed**. The last factor identified is the total number of frames pending to be processed. This shows us the capacity to accept tuples and wait until they can be processed. This value is driven for the variable *max_spout_pending* in a topology. A very small value can easily starve our topology and a sufficiently large value can overload the topology with a huge number of processing tuples to the extent of causing failures and replays.

Regarding system speed, transferring images in tuples is always faster that using a centralized image repository. Apache Storm lacks a distributed file system, as opposed to other frameworks such as Apache Hadoop with its HDFS (Hadoop Distributed File System). Also, Storm is not optimized for parallel file access. Indeed, topologies must be designed to prevent access to remote file systems. Apache Storm was an useful tool to process video streams.

It is important to point out that the total number of nodes have to be increased to achieve a real-time execution of the topology. Using a non realistic linear estimation, around 200 nodes are estimated to be required to process a 500 pixels width frame and looking for two logotypes to obtain throughput comparable to near real-time processing. Taking into account this factor, using a faster library with improved

algorithms may reduce the number of nodes required in cluster. State-of-the-art proposals, such as the use of deep learning algorithms, as Najafabadi et al. suggest in [26], can overcome the limitations and reduce the total execution time.

V. Conclusions and Future Works

This paper used the Apache Storm distributed data processing framework together with open-source image recognition libraries in order to identify logotypes in streaming multimedia information. We proposed two approaches of a seven step topology for multimedia streaming processing in which video information is adjusted to better match the requirements of the processing algorithms. Different implementations have been performed to evaluate different strategies concerning data movement across the components of the topology.

Several experiments have been carried out pointing out that: i) the usage of the Counter Bloom Filter significantly reduces the memory consumption for the process of identifying whether a logotype had already been detected; ii) image data should be transferred between bolts for better efficiency, as opposed to letting bolts retrieve the images before processing and iii) real-time processing of streaming video information cannot be achieved with modest size clusters.

The experimental results prove that the proposed topology is able to benefit from the distributed processing capabilities in the Apache Storm framework reducing the processing time and moving forward to real-time video processing.

Future works involve focusing on using alternative image recognition libraries and increase the total number of nodes used for job processing by using public Clouds in order to attempt achieving real-time video processing for logotype identification. Other streaming processing frameworks for Big Data such as *Apache Spark* [4] will be evaluated.

Acknowledgments

This research was supported by International Internship Program in National Institute of Informatics (NII) in Japan. GM and JH would like to thank the Spanish "Ministerio de Economía y Competitividad" for the project TIN2016-79951-R. This work was partly supported by Council for Science, Technology and Innovation, Cross-ministerial Strategic Innovation Promotion Program (SIP), Infrastructure Maintenance, Renovation, and Management. (funding agency: NEDO)

References

[1] Apache Storm Webpage. http://storm.apache.org/, 2015. Accessed: 2015-11-16.

[2] OpenCV Webpage. http://opencv.org/, 2015. Accessed: 2015-11-19.

[3] OpenIMAJ Webpage. http://www.openimaj.org/, 2015. Accessed: 2015-11-19.

[4] Apache Spark Webpage. http://spark.apache.org/, 2016. Accessed: 2016-10-14.

[5] Apache Zookeeper Webpage. https://zookeeper.apache.org/, 2016. Accessed: 2016-11-10.

[6] VIDCON 2015 Haul: Trends, Strategic Insights, Critical Data, and Tactical Advice. http://tubularinsights.com/vidcon-2015-strategic-insights-tactical-advice/, 2016. Accessed: 2016-11-12.

[7] YouTube Statistics. https://www.youtube.com/yt/press/statistics.html, 2016. Accessed: 2016-11-12.

[8] S.T. Allen, P. Pathirana, and M. Jankowski. *Storm Applied: Strategies for Real-time Event Processing*. Manning Publications Company, 2015.

[9] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[10] Julián Ramos Cózar, Nicolas Guil, José María González-Linares, Emilio L Zapata, and E Izquierdo. Logotype detection to support semantic-based video annotation. *Signal Processing: Image Communication*, 22(7):669–679, 2007.

[11] Julián Ramos Cózar, Pablo Nieto, José María González-Linares, Nicolas Guil, and Y Hernández-Heredia. Detection of logos in low quality videos. In *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, pages 630–635. IEEE, 2011.

[12] WIPO Economics and Statistics Series. World intellectual property indicators 2016. *WIPO publication*, (941E), 2016.

[13] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z Broder. Summary cache: A scalable wide-area web cache sharing protocol. In *ACM SIGCOMM Computer Communication Review*, volume 28, pages 254–265. ACM, 1998.

[14] Alberto Fernandez, Ruben Casado, and Ruben Usamentiaga. A real-time big data architecture for glasses detection using computer vision techniques. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pages 591–596. IEEE, 2015.

[15] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[16] Amir Gandomi and Murtaza Haider. Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, 35(2):137–144, 2015.

[17] John Gantz and David Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*, 2007:1–16, 2012.

[18] Han Hu, Yonggang Wen, Tat-Seng Chua, and Xuelong Li. Toward scalable systems for big data analytics: A technology tutorial. *IEEE Access*, 2:652–687, 2014.

[19] Forrest N Iandola, Anting Shen, Peter Gao, and Kurt Keutzer. Deeplogo: Hitting logo recognition with the deep neural network hammer. *arXiv preprint arXiv:1510.02131*, 2015.

[20] Dong-Hyuck Im, Cheol-Hye Cho, and IlGu Jung. Detecting a large number of objects in real-time using apache storm. In *Information and Communication Technology Convergence (ICTC), 2014 International Conference on*, pages 836–838. IEEE, 2014.

[21] Anil K Jain and Aditya Vailaya. Image retrieval using color and shape. *Pattern recognition*, 29(8):1233–1244, 1996.

[22] Anastasios Kesidis and Dimosthenis Karatzas. Logo and trademark recognition. In *Handbook of Document Image Processing and Recognition*, pages 591–646. Springer, 2014.

[23] J. Leibiusky, G. Eisbruch, and D. Simonassi. *Getting Started with Storm*. Oreilly and Associate Series. O'Reilly Media, 2012.

[24] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[25] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela H Byers. Big data: The next frontier for innovation, competition, and productivity. 2011.

[26] Maryam M Najafabadi, Flavio Villanustre, Taghi M Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1):1, 2015.

[27] Hsiao-Lin Peng and Shu-Yuan Chen. Trademark shape recognition using closed contours. *Pattern Recognition Letters*, 18(8):791–803, 1997.

[28] Wei-Qi Yan, Jun Wang, and Mohan S Kankanhalli. Automatic video logo detection and removal. *Multimedia Systems*, 10(5):379–391, 2005.

[29] Weishan Zhang, Pengcheng Duan, Qinghua Lu, and Xin Liu. A realtime framework for video object detection with storm. In *Ubiquitous Intelligence and Computing, 2014 IEEE 11th Intl Conf on and IEEE 11th Intl Conf on and Autonomic and Trusted Computing, and IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UTC-ATC-ScalCom)*, pages 732–737. IEEE, 2014.

[30] Weishan Zhang, Liang Xu, Pengcheng Duan, Wenjuan Gong, Qinghua Lu, and Su Yang. A video cloud platform combing online and offline cloud computing technologies. *Personal and Ubiquitous Computing*, 19(7):1099–1110, 2015.