Stream Processing To Solve Image Search by Similarity

Jair Lobos*, Verónica Gil-Costa*, Nora Reyes, A. Marcela Printista*
LIDIC, Universidad Nacional de San Luis
*** Conicet, CCT-San Luis**
San Luis, 5700, Argentina
gvcosta@unsl.edu.ar
and
Mauricio Marin
DIINF, Universidad de Santiago, Chile
Centro de Biotecnología y Bioingieniería, Chile
mauricio.marin@usach.cl

ABSTRACT

The classic use of Stream Processing platforms enables working with data in real time, which allows you to generate data analysis quickly attending to a decision-making process. However, you can use these platforms for other applications such as indexing and subsequent use of similarity search objects in a database. The images can be displayed on a metric space, which has features that allow rules to discard a not similar image quickly without making costly computations. This paper presents the use of a Stream Processing platform to index images generated by different users. For this, it is necessary to represent these images by vectors containing different MPGE-7 features. This paper shows a Stream Processing platform using its processing elements (PEs) in parallel to speed up the operations involved in the index construction.

Keywords: Stream Processing, Metrics Spaces, MPEG-7, Sparse Spatial Selection.

1. INTRODUCTION

Actually we are in a fully instrumented and connected world, there is a large number and variety of data resources available either software systems or hardware sensors. Every day, there are numerous industries where their process and interaction with customers generate millions of events that produce activity traces that are not exploited properly. In addition, those traces can be used to detect anomalies to predict the behavior and trends of customers, among other activities that can improve the productivity of a company or institution that generate this type of events.

The events collected from users actions form a continuous amount of data stream. Some examples can be found in: market analysis; telecom call detail records; video surveillance systems; vital signs of a patient in a medical system; intrusion records system networks; the behaviour in a system of Web 2.0, among others. In all these applications it is necessary to collect, process and analyze the data stream, and then generate results or produce some specific actions to the process. Also, users can use these technologies in other type of problems, like no structure data indexation – example, images- so we use a representation about the same image to store on metric data base. In metric data base we use similarity search. Therefore it is possible to characterize an object with certain properties and thus determine the similarity with another object. To search among these objects, there are structures called 'indexes' that allow a comparison operation in less time and a quick access to objects as a result. Stream Processing platform is possible to make a subdivision of tasks that characterize an object. On one hand get MPEG-7 descriptors like: Histogram, Scalable Color and Color Layout, and on the other hand realize index process of them. This process uses pivot structure in space search and a stream platform for images' characteristics.

This paper is structured as follows: Section 2 describes stream processing paradigm and the S4 platform. Section 3 briefly describes metric spaces concepts, MPEG-7 standard for images, pivots based index and Sparse Spatial Selection technique. Section 4 describes the propose model. Section 5 shows the results. Finally, Section 6 presents the conclusions and future work.

2. STREAM PROCESSING

In this section the main Stream Processing properties are discussed, where Stream Processing makes sense, and how it ranks within the paradigm of Big Data. S4 processing platform, used in this paper is also described.

A. Streaming Processing and Big Data

Big Data is defining 3D Data Management: Controlling Data Volume, Velocity and Variety [L01]. Big Data is used to describe the exponential growth and availability of structured and unstructured data. A more recent definition states that "Big Data represents information assets that are characterized by high volume, speed and variety to require specific technology and analysis methods for processing into value" [DGG15].

On the other hand, stream processing is used by the needs or requirements, including processing a variety of data in real time and quickly. Therefore, Big Data and Stream Processing can complement each other. Stream processing was first used to study financial problems. Today, it is used in almost all industries in which data flows are generated by human activities or automatically by sensors. Events are generated online at unpredictable times. The union of events forms a continuous flow of information that may have dynamic variations in traffic intensity. In this context, the process used to store and organize/index events in a convenient form to process them in batches can be very expensive because of the enormous volume of data and the amount of computational resources required for processing them. Even if this is feasible, it is often desirable or even

essential that the data is processed as they arrive to the system, as soon as they are detected to deliver results in real time.

Particularly, Stream Processing corresponds to a distributed computing paradigm that supports the process of gathering and also the analysis of large volumes of heterogeneous data stream to obtain ideas which allow performing actions and making decisions in real time.

Stream Processing appears as a result of the rigorous data management which is increasingly demanding because of the information generated by business and scientific applications, which are fully linked to the technological progress. It is also related to the advancement in hardware and software databases; the management of large amount of data in distributed systems; the use of techniques such as signal processing, statistics, data mining and optimization theory.

Stream Processing's aims are processing data in real time and in a fully integrated way, providing information and outcomes for consumers and/or end users. Also it aims to integrate new information to support decision making in the medium and long term.

The high volume of events flows coming from different data sources which makes it impossible to store this information, such as model-based on data warehouse where all the data is stored and then to make the appropriate processing and analysis off-line. Stream Processing needs to fulfill certain performance requirements in terms of latency and throughput. Specifically, processing must keep up with the rate of intake data, while it provides a high level of quality of analysis of results as fast as possible. Additionally, the application components and infrastructure must be fault-tolerant.

B. S4 - Simple Scalable Streaming System

S4 acronym for "Simple Scalable Streaming System" is a system of general purpose, distributed and scalable which allows applications to process data flows continuously without restrictions [NRNK10][S414]. S4 is inspired by MapReduce [AGT14][DG04], designed in the context of data mining and machine learning of Yahoo! Labs for on-line advertising systems.

In S4 each event is described as a pair (key, attribute). In the system, processing elements (PEs) are the basic units and messages are exchanged among them. The PEs can send messages or post results. PEs are allocated in the so-called processing nodes (PNs) servers. The PNs are responsible for receiving incoming events, routing the events to the corresponding PEs and dispatching events through the communication layer. The events are distributed using a hash function over the key of the events. Furthermore, the communication layer uses Zookeeper [HKJR10] which provides management and automatic replacement clusters if a node fails.

To run an application with S4, we need to deploy an Adapter application. Adapters are S4 applications that can convert external stream into stream of S4 events. Figure 1. shows a simple Tweet language count for Twitter. In this example, input events contain a language descriptor for a tweet from Twitter. The Adapter gathers tweets from twitter and filters only the language descriptor. Then, the Adapter sends an event to PE1. PE1 listens to Tweet events with all possible keys. For each possible key, PE1 emits a new event of type TweetLang. PE2-ns listen to TweetLang events emitted with the key lang. For example, PE1 emits an event with key lang="es". PE2 receives all the type of events with TweetLang keyed lang="es". If there is a corresponding PE to the emitted key, the PE is called and the counter of language is incremented. Otherwise a new PE is instantiated and linked to the new key. Whenever a PE increments its counter, it sends the update count to the PE called PEm and this shows the results.

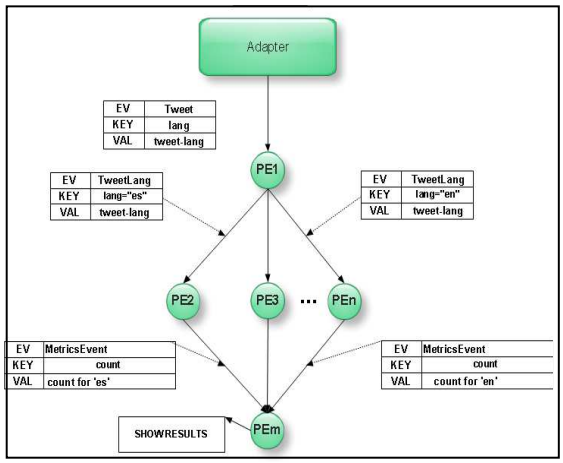

Figure 1. Application design with S4.

3. METRIC SPACES

The *similarity search* problem can be formally defined in the model of metric spaces, which provides an independent conceptual framework of the application domain. A metric space ***(X, d)*** is formed by a universe of valid objects X and a distance function $d: X \times X \rightarrow \mathbf{R}^+$ among its elements. A finite subset $U \subset X$, with $|U| = n$, is the elements set or database, where queries are performed. Distance function d will measure the "no-similarity" among the elements while smaller is the value of d, the elements will be more similar to each other. The functions of distances must fulfill the following properties which assure their consistent definition: *Strict positiveness:* $d(x,y) > 0$ if x distinct of y; *Symmetry*: $d(x,y) = d(y,x)$; *Non-Reflexivity*: $d(x,x) = 0$. However, if d is a metric, it must also satisfy the *Triangle inequality* property where: $d(x,y) \leq d(x,z) + d(z,y)$, it is the only property that avoids comparisons in the search: if the distances are known $d(x,y)$ *and* $d(x,z)$, result of $|d(x,y) - d(x,z)|$ it is a lower bound for the distance $d(z,y)$, and in some cases knowing this, the calculation of the current distance can be avoided.

There are different types of queries that may involve similarity searches, but as an element $q \in U$ as query, the most commons are:

- *Range queries*: recover all elements from database U that they are at a distance as r of query q. That is to say, $\{u \in U / d(q,u) \leq r\}$.
- *K-nearest neighbour queries*: retrieve k elements from U nearest to q. That is to say, any collection $A \subset U$, that fulfills that $|A| = k$ y $\forall v \in A, v \in U - A, d(q,u) \leq d(q,v)$. The query for the *nearest element*, or the nearest neighbour, could be seen as the particular case when $k = 1$.

Overall the calculation of distance function can be very expensive and therefore, in large databases, it is not feasible to conduct a thorough search to a query. Thus, the general solution is to pre-process data base U in order to construct an index, which allows to save distance calculations during searches. There are basically two general approaches to constructing indexes: based on compact partitions and based on pivots. In the indexes based on compact partitions a set of objects called centers are selected and the remaining objects are divided among those centers by proximity, thereby determining the zones associated with each center; searches are discarded by the triangle inequality entire regions of space using information areas. In pivots indexes a set of objects called pivot are selected, the distances for all the database objects to these pivots are calculated and stored; during the search, it is calculated the distance of query q to the pivots and the objects are discarded using the triangle inequality [ZADB06] [S05] [CNBYM01]. This paper is focused on pivot-based methods.

Mpeg-7

MPEG-7 is a standard representation of audiovisual information that allows multimedia content description. This standard arises from the need to describe audiovisual content due to the increasing amount of information. Within the scope of this standard it is important to describe the major aspects of the contents such as audio, voice, video, graphics and 3D models. In MPEG-7 [SBMCA05] there are descriptors representing a syntactically and semantically defined characteristic. In the case of images, we have some descriptors such as:

Scalable Color: is a descriptor of a histogram in an HSV colour space into a container 256 using a Haar transformation to compress.

Color Layout: this descriptor represents the spatial distribution of the images in a compact form. The image is divided into discrete blocks of 8 x 8 and the representative colours in the YCbCr space are extracted. The descriptor is obtained by applying the discrete cosine transform on each block using coefficients. The descriptor produces a representation (64-Y, 64-Cb, Cr-64) of the image.

Histogram: represents the distribution of five types of edges in the image. It is divided into 4 x 4 sub-images and computes. A 16-block produces an image descriptor of 80 bins.

Pivots Index

As mentioned, an index corresponds to a data structure built to facilitate searches in a set U of elements. In particular, an index based on pivots selects a set $P \subset X$ of elements called pivots and stores the distance to each pivot $p_i \in P$ to every element $u \in U$ in any appropriate data structure. These distances define an equivalence relation over U. So, if one has a set $P = \{p_1, p_2, ..., p_m\} \subset X$ of pivots, of size $m = |P|$, distances $d(p_1,u), d(p_2,u), ..., d(p_m,u)$, $\forall u \in U$ are calculated, which makes up the index properly.

Different pivot-based methods, such as: Approximating and Eliminating Search Algorithm (AESA), Burkhard-Keller Tree (BKT), Fixed-Queries Tree (FQT), Fixed-Height FQT (FHQT), Vantage Point Tree (VPT), Linear AESA (LAESA) and Sparse Spatial Selection (SSS), they differ from each other mainly for the method they choose the P set [BNC03] and the structure wherein the distances are stored [CNBYM01]. Most of them are static, because they need to know beforehand the elements of the database to build the index. However, the proposed index in [Bris06], *Sparse Spatial Selection* (SSS), admits to being created as elements of the database arrive, which is the setting of interest for this paper.

Sparse Spatial Selection

Sparse Spatial Selection (SSS) [BFPR06] is the technique that dynamically selects a set of well distributed pivots around the metric space. This index is based on the idea that, if the pivots are scattered "spatially" in space, they will be able to discard more objects during the search. To achieve this, when an object is inserted into the database, it is selected as a new pivot if it is "quite far" from the already selected pivots, which is a desirable property of a set of pivots [BNC03]. It is considered that an element is "quite far" from another pivot if it is greater or equal distance $M \times \alpha$ where M is the maximum distance between any two objects in X and α a parameter whose optimal values are near 0.4 (i.e. The object is selected as a pivot if there is a little less than half of the maximum distance from all other pivots). Figure 2 shows an example of a set of objects on the plane $\mathbf{R}^2$ with Euclidean distance and how the pivots are distributed in space.

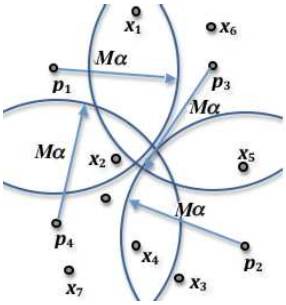

Figure 2. Example distribution pivots with SSS.

The α parameter affects the number of pivots that are selected. And for the M parameter you must have in advance all the items in the collection of objects to estimate its value.

A great advantage of this technique, apart from the dynamics, it should not be fixed in advance the number of pivots, because the number of pivots adapts to the "complexity" or dimension of metric space. It has been empirically shown that the number of pivots that SSS obtained is very similar to the optimum number obtained with other strategies.

4. S4 MODELING

This section presents the design of an application that receives objects, images, processes and index them into a system based on pivot index. The application is designed to take advantage of the characteristics of the S4 platform and performs image processing in real time. To do this, each image that enters the system should be transformed into a vector of descriptors, which can be handled easily by the indexing algorithm. Figure 3 shows the application schema.

The application Adapter initially reads the images and sends them to a PE (Init), where they will be routed to a process of reducing the image (Est0 and Est1). That is, if you enter an image of 512x512 pixels, the image becomes 8x8 pixels. In this paper two reducers images were used so that the utilization does not exceed 50%.

The new reduced image is sent to the PEs to be responsible to obtain a characteristic of MPEG-7 using the open-source library LIRE [L13]. Each PE works independently. In particular, in this paper we use the features of MPGE-7 called histogram (Hist), Scalable Color (SC) and Color Layout (CL). At the same time, the process Color Layout can be parallelized by three processes which are responsible for obtaining red, green and blue images colour filters (Y, Cb and Cr). The process that calculates the Histogram generates a vector of 80 descriptors. Scalable Color process generates a descriptor vector of 32, and the process that computes the Color Layout generates a vector of 64 descriptors. Therefore, each vector representing an image having a size of 176 descriptors.

Each PEs that is responsible for obtaining MPGE-7 features sends the result of the descriptor vector to a PE (Join) in charge of joining all the features in a single vector.

The descriptor vector representing the image is sent to a PE responsible for determining whether the vector corresponding to a pivot. To this objective, the PE previously called DLi (i = 0..3), calculates the distance between the existing pivots for image collection and the new image vector generated. This distance calculation process is parallelized so that any PE participating in the operation exceeds 50% utilization. Finally, the PE called PIV applies the function SSS described in the previous section to determine if the new vector image corresponds to a pivot.

To parallelize the construction of the index, we use an approach based on partitioning columns or pivots. In other words, a PE Index will be responsible for managing a sub-set of pivots. To do this, there is initially a single PE Index (Ind0) responsible for building the pivot index. When the PE Index exceeds 50% utilization, it creates a second PE Index, This new PE manages another sub-set of pivots. This process is repeated as many times as necessary to avoid saturation of the PE Index.

When Pivot PE receives a vector image, this PE sends all the vector images to the PE Index. If a vector is not selected as a pivot for the PE Pivot, then this vector is attached as a row in each PE Index. Note that this involves calculating the distance between the vector and all the pivots of the sub-table pertaining to each PE Index.

If the vector is selected as a pivot, this vector is also sent to all PEs Index to be annexed as row, because a pivot table is also part of the collection of images that can be a candidate in response to a query. Figure 4 left shows this situation.

However, as shown in Figure 4 on the right side, the work to be performed is more computationally expensive for the PE responsible for appending Index as pivot (as a column). Because PE must not only compute distances between vectors of the sub-images and new pivot table, PE must also compute the distance between the new pivot to all pivots existing in the sub-table. That is to say, as a last resort PE should append a column and a row of distances.

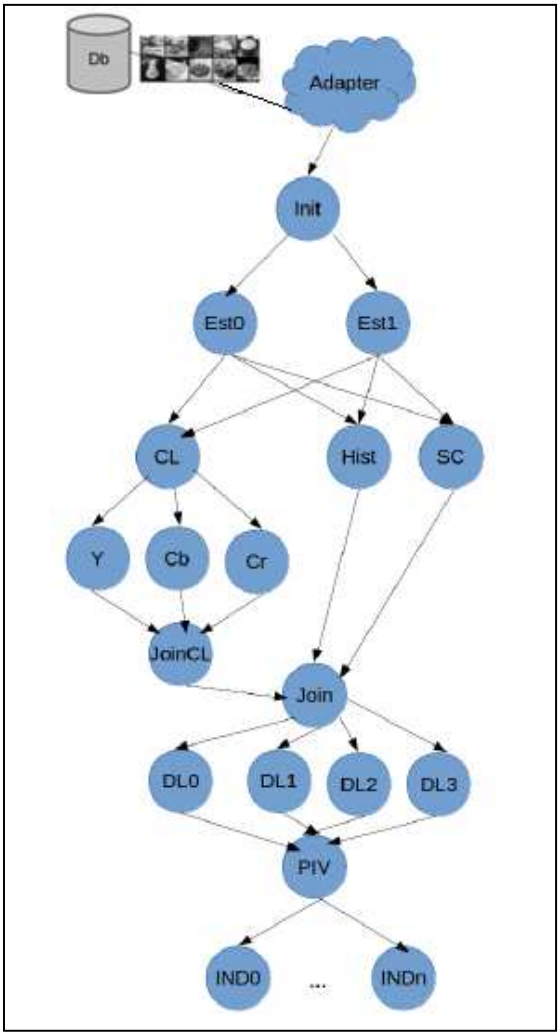

Figure 3. Application schema.

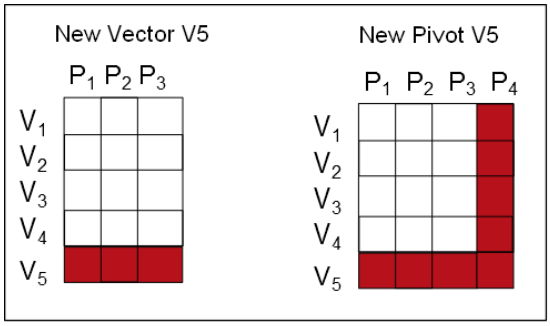

Figure 4: Selecting a new pivot application computational cost.

5. RESULTS

The results obtained are presented. In particular, we focus on service time and PEs utilization. Utilization is measured by taking the service time divided by the total runtime of the application. In other words this metric indicates the workload of each PE and allows detecting potential bottlenecks. In the case of PE Index also we evaluate the throughput (number of vectors indexed per unit time), and the average time to index a new vector.

The tests were performed on a cluster with 16 CPUs with 64-bit Intel Core Quad Q9550 2.83GHz processor and 4GB RAM DDR3 1333 Mhz each node.
We worked with a set of images [Food] which are hosted locally on each cluster node. This set contains 101 photographs of food categories, a total of 101,000 images (5 Gb of space). The total execution time required to index all the images is 479 minutes (8 hours).

Vector construction evaluation

Figure 5 shows the service time consumed by each PE that participates in the process of constructing the vector for each image received by the system.

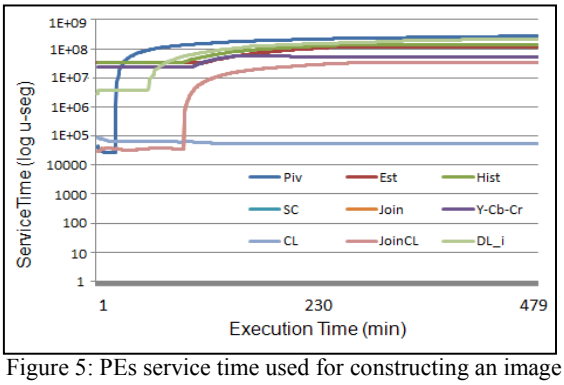

Figure 5: PEs service time used for constructing an image vector.

The x-*axis* shows the runtime in minutes required to process all images. The y-*axis* shows the time of service of each PE in logarithmic scale and in microseconds. In the case of partitioned PEs as DLi and EST, for simplicity and clarity of graphical average of results obtained for these PEs is shown.
In general, all PEs have a similar service time. The process that performs the routing of the Color Layout is the shortest one which is followed by Join and JoinCL processes.

Figure 6 shows the utilization reported by PEs participants in the vector images construction process. As expected, the operation reported a high utilization in the process to determine if a new vector is selected as the pivot for the collection or not, using the SSS technique. This operation requires a lot of comparisons of distances, which tends to increase when increasing the size of the collection of indexed images.

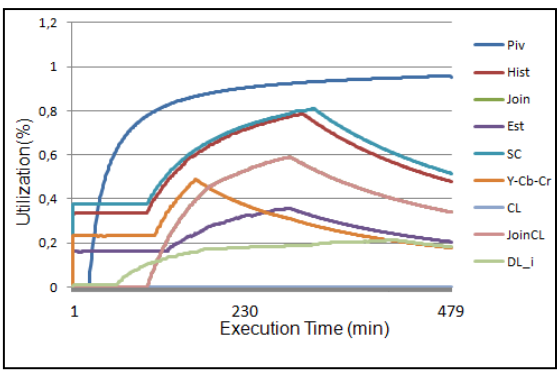

Figure 6: Utilization reported by responsible PEs for creating the vector image

Index construction evaluation

Next, we show the results obtained by the PE responsible for building the pivots index. Remember that an important objective of this paper is the PEs that build the pivot table and store the distances between the vector images and the lists of pivots do not exceed 50% utilization. In other words, when an PE Index reported a utilization of 50%, system creates a new PE Index for administering the new pivots.
Figures 7 and 8 show the utilization and service time in logarithmic scale respectively of the PEs responsible for building and maintaining the index pivots. Initially the system starts with only one PE Index. When it reaches 50% utilization, a second PE Index is created. To process all images a third PE Index is created at about 4 hours after the index system has begun to operate.

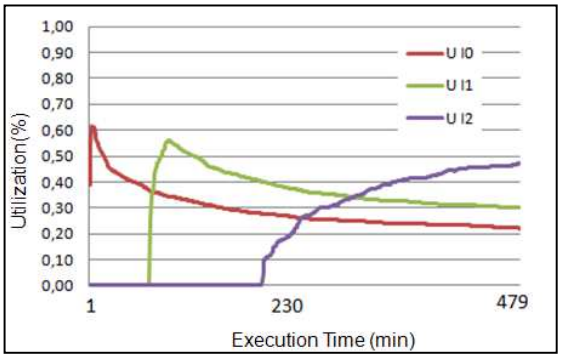

Figure 7: PEs Index utilization making index process.

Figure 9 shows the throughput measured as the number of vector images index per unit time in each PE Index. The graphic shows that PEs tends to start with a value of high throughput (700 for Ind0, 580 for IND1 and 390 for IND2). This throughput value tends to decrease as they enter more images to system and indexing work is divided among more PEs.
Figure 10 shows the average time to index every image in microseconds. As expected, this time increases as the system contains a greater amount of vector images, because the new image must be compared to more pivots in each index partition.

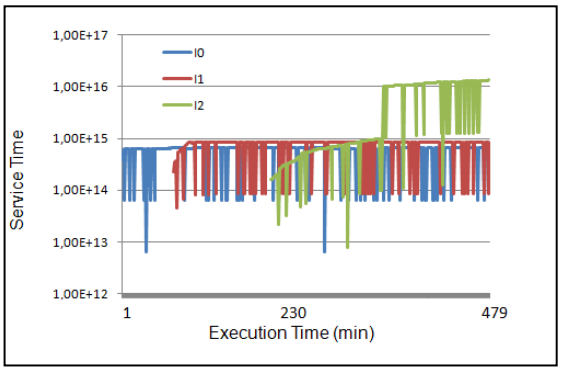

Figure 8: Service time in microseconds of PEs responsible for indexing the images.

Similarity search

In this section the performance of the proposed search of images in a system based on pivot index is evaluated. To

do this, given the collection of 101,000 images, 70% of these images were indexed and the 30% rest was used for similar searches.
A radius of 3 is used in the search system. The algorithm shows the images that are a distance of no more than 3 of the query image.

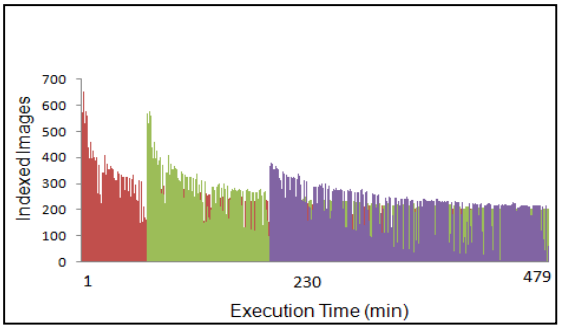

Figure 9: Throughput for PE Index.

Search process works as follows. Once we obtained the representing vector of the image in PE called Join, this vector is sent to each PE DLi to calculate the distance for each existing pivots to the new query vector. Then, distances with the vector query is sent to PE PIV. In this case, as the incoming event is a "QUERY" PE PIV redirects this event to PE Index. PE Index calculated the triangle inequality between images stored locally and vector image to discard the no similar images. Those images that cannot be discarded are directly comparable to the query. Finally, if the distance is $d(q, vi) \leq r$ then the image represented by the vector *vi* is reported as a result.

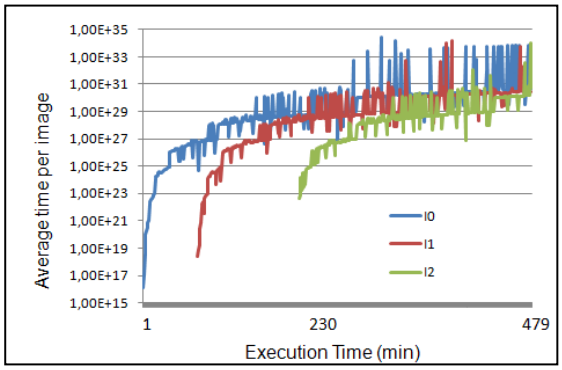

Figure 10: Service time in microseconds of PEs responsible for indexing the images.

Figure 11 shows the number of queries processed per minute in each PE Index. At the start of the implementation of PE Index reported a low number of processed queries. This is the effect of start-up system that can be discarded, because the system quickly reaches a state stable execution, where an average of 13 queries per minute were resolved by the PE Index.
Figure 12 shows the average number of candidate images reported every minute by the system to process queries. In general, it can be observed that the PE Index I3, is the one that discarded images the most.
Figure 13 shows the average time required to process a query in microseconds. The results show that the query processing time is directly related to the amount of candidate results. In other words, as shown in Figure 12, the PE Index IND2 reports less candidate images for queries which have a direct effect on the processing time for each query. IND2 is the shortest time reported in the Figure 13.

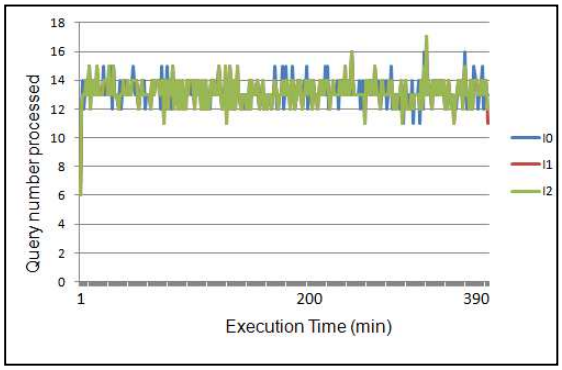

Figure 11. Query number processed by PE Index.

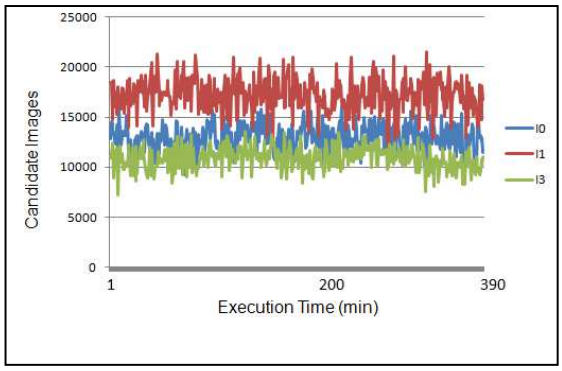

Figure 12. Average number of candidate images for queries.

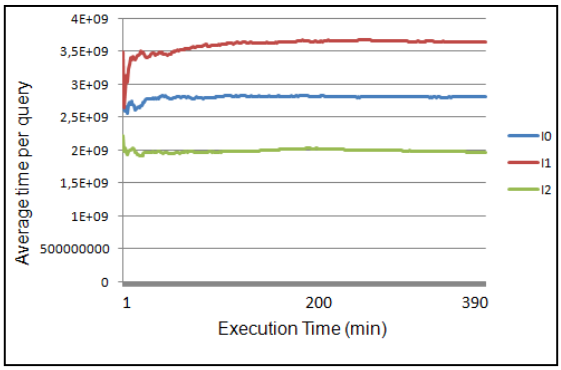

Figure 13. Average time to process a query.

6. CONCLUSIONS AND FUTURE WORK

Although there is a significant progress in Big Data technology for efficient processing of large volumes of unstructured data, today's world presents challenges which current platforms, by themselves, are not enough. The reason is the additional requirement for fast processing of the data to be useful in decision making organizations. Thus, the demands for *stream processing* techniques provide solutions that are designed to handle large volumes in real time, allowing the effect of

"analysis and subsequent reaction with data in motion". In this paper we used the stream processing platform, called S4 to accelerate image processing system, when a continuous flow of objects are expected. The design of the application architecture that is capable to analyze and act on a flow of images in real time is presented here. Finally, an experimental study of the performance of each step involved in the process is done. As to the construction of the feature vector, it is visible the impact of the SSS technique used to search, making this one of the points to continue to develop to optimize the proposed architecture. Regarding the construction of the index, the technique used to maintain the utilization of PE 50%, showing its positive impact on time indexing of objects-images. Finally, this paper analyzes the performance of the whole process of similarity search of images, showing how the process is favoured when it has a solid architecture, and dynamic time, enabling it to rule out candidates during the search process.

7. REFERENCES

[AGT14] Andrade H., Gedik B. and Turaga D., Fundamentals of Stream Processing Aplications Design, System and Analytics, Cambridge University Press, 2014.

[B13] Barlow, Real-Time Big Data Analytics: Emerging Architecture. Kindle Edition. O'Reilly Media Inc. 2013.

[BFPR06] Brisaboa N.R., Farina A., Pedreira O., Reyes N.: Similarity search using sparse pivots for efficient multimedia information retrieval. In: ISM. pp. 881–888 (2006).

[BNC03] Bustos B., Navarro G. and Chávez E. Pivot selection techniques for proximity searching in metric spaces, Pattern Recognition Letters, Volume 24, Issue 14, October 2003, Pages 2357-2366, ISSN 0167-8655.

[CNBYM01] Chávez E., Navarro G. and Baeza-Yates R., and Marroquín J L. Searching in metric spaces. ACM Comput. Surv. 33, 3 (September 2001), 273-321.

[Food] The Food-101 Data Set: http://www.vision.ee.ethz.ch/datasets extra/food-101/

[DGG15] A. De Mauro, M. Greco, and M. Grimaldi. "What is big data? A consensual definition and a review of key research topics", in AIP, 2015 pp. 97–104.

[DG04] Dean J. and Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters. *COMMUNICATIONS OF THE ACM*, 2008, vol. 51, no 1, p. 107.

[HKJR10] Hunt P., Konar M., Junqueira F.P. and Reed B. Zookeeper: wait-free coordination for internet-scale systems. In Proceedings of the 2010 USENIX conference on USENIX annual technical conference, USENIXATC'10, pp 11–11, Berkeley, CA, USA, 2010. USENIX Association.

[L01] D. Laney, "3D Data Management: Controlling Data Volume, Velocity and Variety", Gartner, 2001, pp. 1-4

[L13] Lux M. LIRE: Open Source Image Retrieval in Java, In Proceedings of the 21st ACM International Conference on Multimedia. New York, NY, USA , pp. 843-846 (2013). ACM.

[MSCJ13] Mayer-Schönberger V., Cukier K., and Jurado A.I.. Big data: La revolución de los datos masivos. Turner. 2013.

[NRNK10] Neumeyer L., Robbins B., Nair A. and Kesari A. "S4: Distributed Stream Computing Platform," Data Mining Workshops (ICDMW), 2010 IEEE International Conference on , vol., no., pp.170,177, 13-13 Dec. 2010.

[S414] S4- Distributed Stream Computing Platform: revisada en Julio 2014. http://incubator.apache.org/s4/

[S05] Samet H. Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling). 2005. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[SBMCA05] Spyrou E., Borgne H. L., Mailis T.P., Cooke E., Avrithis Y.S., and O'Connor N.E. Fusing mpeg-7 visual descriptors for image classification, in ICANN (2), 2005, pp. 847–852.

[ZADB06] Zezula P., Amato G., Dohnal V., Batko M. Similarity Search: The Metric Space Approach, Advances in Database Systems, Springer, 2006.