do Valle et al. "Big Data Platform for Analysing Crime Evidences." ***2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService)*****. IEEE, 2020.**

BibTeX:

```
@inproceedings{do2020big,
  title={Big Data Platform for Analysing Crime Evidences},
  author={do Valle, Joao Marcos and Souza, Gabriel and Fidelis, Samuel and Ara{\'u}jo,
Adelson and Cacho, N{\'e}lio and Silva, Iaslan and Budke, Jaine and Sales, Henrique and
Filgueira, Max William and Lopes, Frederico and others},
  booktitle={2020 IEEE Sixth International Conference on Big Data Computing Service and
Applications (BigDataService)},
  pages={113--119},
  year={2020},
  organization={IEEE}
}
```

Big Data Platform for Analysing Crime Evidences

João Marcos do Valle, Gabriel Souza, Samuel Fidelis, Adelson Araújo*, Nélio Cacho, Iaslan Silva,
Jaine Budke, Henrique Sales, Max William Filgueira, Frederico Lopes, Daniel Araújo, Rivaldo Silva Junior*
Digital Metropolis Institute
Federal University of Rio Grande do Norte
*Public Ministry of the State of Rio Grande do Norte**
Natal, Brazil
jmarcos.araujo96@gmail.com, gabriel_feg@hotmail.com, samico@ufrn.edu.br, adelsondias@gmail.com,
neliocacho@dimap.ufrn.br, iaslannascimento@gmail.com, jainebudke@hotmail.com, hd5@ufrn.edu.br,
maxwilliam780@gmail.com, fred@imd.ufrn.br, daniel@imd.ufrn.br, rivajunior19@hotmail.com

***Abstract*—During a criminal investigation, the evidence collection process produces an enormous amount of data. These data are present in many medias that are extracted as crime evidences (USB flash drives, smartphones, hard drives, computers, etc). Due to this data volume, the manual analysis is slow and costly. This work fulfills this gap by presenting a data extraction and processing platform for crime evidence analysis, named INSIDE. Our proposed platform leverages a lambda architecture and uses a set of tools and frameworks such as Hadoop HDFS, Kafka, Spark and Docker to analyze a big volume of data at an acceptable time. We also present an example of the proposed platform in use by the Public Ministry of Rio Grande do Norte(Brazil), where some evaluative tests have been carried out.**

***Index Terms*—distributed systems, lambda architecture, crime evidences, digital evidences**

I. Introduction

With the advance of WEB 2.0 and the development of information and communication technologies (ICT), the amount of produced and collected data quickly rose in the last years. The plunging cost of hardware and software for storing data, accelerated by cloud computing, has enabled the collection and storage of huge amounts of data [1]. Big data analysis consists of the discovery and management of information, patterns or conclusions from this data [2]. In 2011, a report by the McKensey Global Institute defined big data analytics as a defining factor of the next wave of economic innovation [3]. Nevertheless, when working with big data, the analytics can prove itself to be a complex and onerous process. In this context, many Big Data Applications are found in the literature. There are a large number of proposed big data frameworks and platforms in the healthcare sector [4] [5], power distribution [6] and government sector [7].

In the Justice System, the speed of the analysis of crime evidences collected in an investigation is of paramount importance. Normally, the most critical period for an investigation are the first 48 hours [8]. As the prosecutor has the demand of verifying the crime evidences, forensic specialists need to extract, decodify and make an analysis of collected medias. This is necessary for prosecutor to make the legal requirements for the lawsuit and the legal complaints of the crime suspects. The problem is that the available time is relatively short for the possible amount of data generated by each investigation.

Investigations could comprise many suspects, in which, for each one is done the analyse of various medias (cell phones, tablets, Hard Drives, etc). Leads usually are found in specific files, and the analyst has to verify all the evidence to find a single lead. Also, there is the possibility that the analyst may have to analyse evidence acquired in flagrant, where the defendant was incarcerated and the analyst has access to the evidence just a few days before the judging. Therefore, the analysis process must be done quickly.

However, the collected media files have a large amount of data. Counterpoint Research [9] shows that the average global memory capacity of smartphones at the end of 2017 was 43 GB and was observed a constant increase. In view of this problem, it is necessary a system that can give indications to analysts of files that presents leads, like suspicious messages and images, to lead a faster result.

In this context, this work aims to present a data extraction and processing platform for crime evidence analysis, named INSIDE. Our platform leverages a lambda architecture and uses a set of tools and frameworks such as Hadoop HDFS, Kafka, Spark and Docker to analyse a big volume of data at an acceptable time. We also present an example of the proposed platform in use by the Public Ministry of Rio Grande do Norte (Brazil), where some evaluative tests have been carried out.

This paper is organized as follows. Section II presents the related works. Section III lists the requirements for the Big Data platform for crime evidence analysis. Section IV presents an overview of the big data platform and implementation details. Section V depicts the results obtained whilst testing the platform. Finally, section VI presents the final remarks and section VII the future works.

II. Related Work

International academic circles investigating big data forensic applications have often reported two main problems in their

This project is funded and supported by the INSIDE project, Public Ministry of the State of Rio Grande do Norte (MPRN), and by the SmartMetropolis project.

motivations: (i) variety in data structure and (ii) volume of data that must be processed. Because forensic computing may be in its infancy [10], the big data solutions presented have been in the process of incremental improvement. For example, Bhoedjang et al. [11] presented in 2012 the XIRAF tool, which was later adapted by Van Beek et al. [8] to include requirements for a big data platform and give rise to HANSKEN. Guided by design improvements, big data forensic analysis solutions have articulated functional and non-functional requirements to deliver maximum value by presenting data extracted from electronic evidence.

In this sense, many tools have emerged to facilitate the process of digital screening. To prioritize relevant evidence, digital triage would be a way of replacing a full forensic examination. One example is the work of Shaw and Browne [12], who proposed bootable software that reduces the amount of data from electronic evidence up to 0.2%. The purpose is to avoid investigative agents manually inspecting unnecessary items. Sanchez et al. [13] reports that these tools may not be very suitable for complicated cases, but argues that constant and mass exposure of abusive materials can also bring psychological problems for investigators. Marturana et al. [14] presented a machine learning approach to digital triage. Our work has taken a step forward and described a methodology of categorization of evidence images coupled into the highly scalable evidence data import pipeline.

Besides, the big data solutions for the forensic analysis presented in recent research papers have pointed out to reasonable justifications for using Apache tools such as Hadoop and Kafka [8], [15]. In this article, we describe how various technologies proposed in related studies can operate efficiently in a forensic big data pipeline. Moreover, [16] proposed a conceptual digital forensic framework based on artificial intelligence to optimize the performance of the digital forensics process. In [17], a framework model is proposed to help on forensic analysis to investigate illegal activity, such as suspect's behaviours on data contained on smartphones. The aim is to expose terrorists or criminal network. [18] describes a system that improves the ability to find relationships between big heterogeneous data using artificial intelligence models. For instance, it allows to identify correlation between evidences from multiple sources. The system is organized with three layers: data acquisition, data examination, and data analysis. [19] describes the digital forensic process and explains how Digital Forensics as a Service (DFaaS) approach developed at the Netherlands Forensics Institute helps analysts on the investigation process. In [20], a study on different techniques that helps the analysis of small datasets is presented. They describe how can be used to help investigators to remove noises from audio, video and images, and visualize the information on the context of big datasets. Our proposal platform builds upon many of these works but, to the best of our knowledge, is the first one to combine pluggable machine learning algorithms with a big data pipeline.

III. Application Requirements

The diversity and complexity of big data is challenging digital forensic in many ways, and the analysis of electronic evidence has become a task to be scaled. From the big data perspective, one may adopt the 5V's (variety, volume, velocity, veracity and value) framework to describe a digital forensic application that support faster investigations. Still, international academic circles have described an abundant set of design principles specifically for the digital forensics' peculiarities. For example, the problem with the volume of data is better solved when traduced as a problem of case lead time minimization [8] or even part of a digital triage process [12]. In this section, we describe some requirements and design principles inspired by [8] and adopted to define the INSIDE platform.

Minimize case lead time. According to Van Beek et al. [8], investigative agents usually have less than 48 hours to find relevant traces. The volume of available data has forced these agents to prioritize some evidence over others, often without any defined methodology [12]. Due to the limited time available, Van Beek et al. further states that the traces found are used to validate hypotheses rather than help to form them [8]. Decreasing the time to import evidence data is a key requirement for achieving more efficient forensic analysis and faster case executions.

Maximize Coverage. The growing variety of evidence data is also a factor to consider. Among images, chat messages, web histories, among others, various file formats can emerge. A big data tool for forensic analysis must be robust enough to cover as many of these formats as possible to achieve quality data extraction from the device. Without this coverage, the variation in data format may result in not all traces being found [8].

Data Retention. In a forensic analysis, extracting data from physical evidence is a time-consuming process that should happen only once. Thus, it is desirable that data be retained in agency environments for forensic consultation. A digital evidence data analysis application should consider data retention strategies that prevent sensitive information from being lost in case of failures or errors [8].

Security and Privacy. Seized evidences contain personal information such as identities of suspects and their peers. Also, it can contains victims information that must be protected. Digital evidences can have sensible data as crime pictures, videos, audios, detailed conversations and others. The access to this data must be restrict only to investigation analysts and detectives and the privacy of every person involved must be protected. System administrators must not be able to verify which data is being accessed and by who, but they still need to have knowledge on general information about the systems. In contrast, inside the teams, a level of transparency is required for then to be able to see which person in the team accessed certain files and what they were searching. This can be seen as a process of sharing clues or key data for the investigation with the team. This implies that there has to be different levels of clearance inside the system, maintaining the privacy of

victims, suspects and also the analysts and investigators.

Maintainability. We must create a software platform that is able to be improved in a incremental manner by integrating new components (machine learning algorithms, visualization techniques, etc) . Hence, we give a particular attention to modularity and changeability principles, such as low coupling, high cohesion, conciseness, and separation of concerns.

IV. INSIDE: Integration, aNalySis, vIsualization of Data for invEstigation

Based on the platform requirements elicited in the previous section, we have proposed *INSIDE* (*Integration, aNalySis, vIsualization of Data for invEstigation*). INSIDE is a big data platform for crime analysis capable of powering scalable, real-time and data-driven applications. This platform was conceived to process big data in the context of crime evidence, and speed up crime evidences analysis procedures, a very time consuming process.

Considering the INSIDE's requirements, we have selected a set of open source technologies that are used to implement it, namely:

- **Apache Hadoop**: a framework for distributed data processing which also provides a distributed storage in the Hadoop Distributed File System (HDFS)
- **Apache Kafka**: a scalable publish-subscribe messaging system, suitable for both offline and online messaging consumption [21] [22]. In recent years, it has become a highly use data ingestion tool used to ingest streams of data into processing platforms [23] [24].
- **Apache ZooKeeper**: quorum-based, centralized service for group coordination in distributed systems. Frequently used with Apache Kafka for the coordination of the messaging system brokers.
- **Apache Spark**: general-purpose cluster-computing framework that provides an API for using an entire cluster processing power, with implicit data parallelism and fault-tolerance. It is capable of speeding up processing time at a very high rate due to it's principal abstraction, the Resilient Distributed Dataset (RDD).

By choosing those technologies, we fulfil the aforementioned requirements in the following way: (i) *Minimize case lead time* and *Maximize Coverage* are obtained by using Spark to handle scalability of computational power, (ii) *Data Retention* is accomplished by using HDFS as a distributed file system, (iii) *Security and Privacy* are achieved by using Kerberos server to handle authentication in the whole platform, (iv) and *Maintainability* was satisfied by using Apache Kafka to decouple components in separated docker containers.

A. Architecture Overview

In order to improve modularity, INSIDE platform comprises many layers, each one with specific purpose. The data workflow inside the platform is as follows: The data extracted from digital evidences is sent to an *Importation Module*, which provides transformed data and metadata to a *Lambda Layer*. The Importation Module is capable to handle different types and sizes of data. In a similar fashion, the Lambda Layer comprises three other layers: the *Batch*, *Speed* and *Serving* layers.

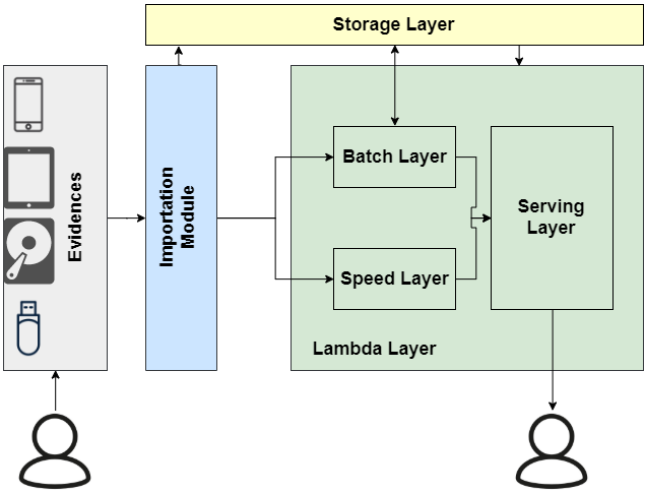

Fig. 1. INSIDE architecture

The *Batch layer* is in charge of receiving data to be persisted in the *Storage layer*. Additionally, batch-analysis processing can be executed in the Batch layer. The *Speed layer* is also known as the Real-time layer and caters to real-time analysis requirements, handling data streams from the ingestion module. The *Lambda layer* will also contain various specialized APIs, that will acquire different types of data for processing, using the Speed and Batch layers processing power. The Lambda layer also has a *Serving layer*, responsible for delivering data to the final user in different ways. INSIDE's architecture can be seen at Fig. 1. Hereafter, the pipeline layers will be described in detail.

B. Importation Module

The Importation Module is responsible for extracting and pre-processing evidence data. This module receives a media extraction as an input. The media extraction is a compressed file in a format that cannot be disclosed. This file contains two types of data: (i) an XML file containing all the metadata collected from the evidence, along with all the textual data present in the evidence, and (ii) a directory tree with a variety of other data types from the evidence (images, audios, videos, databases, configuration files, etc.). The extracted data and metadata from the evidence are then fed to the *Lambda Layer*.

C. Lambda Layer

After the data is ingested in the Lambda layer, it is fed to both Batch and Speed layers. The Batch layer receives data transferred via the WebHDFS API in the data ingestion module. The received data is written on the HDFS, and the necessary blocks for its storage is created. Hadoop handles the data replication and persistence of files in the HDFS, and this data is available at the file system to be used by other applications. The batch layer is where raw data is

stored as is in the rawest format possible. This is the store where master data in an immutable state is also available and used by various analyses going forward. Since the data is immutable, update and even delete are forbidden operations. Data is always appended (added) with a timestamp so that when some data is required, it can be queried with the highest timestamp to get the latest record. Delete is also forbidden because then many analyses would require these deleted record details. Even though HDFS is a very robust system, data transfer is considerably slow because everything must be written to the cluster's machine physical disk. Since one of the main objectives of this work is to minimize case lead time, the speed of the process is of great importance to the pipeline. Even though the batch layer has an acceptable speed, considering this work requirements, a faster speed layer was also implemented.

The speed layer is composed of an Apache Spark cluster for fast processing, and specialized applications in charge of handling different types of data. The Spark cluster is achieved via Docker containers, used to provide better scalability to the data processing environment. The Spark cluster architecture is composed of a master node and worker nodes. The master node is in charge of handling the requests for processing sent to it by the specialized applications, scheduling operations in the cluster, managing the cluster processing power and returning the outputs of the processing. The workers nodes receive data from the master node in the RDD format, a type of data stored in the cluster's distributed memory. Workers also receive parallel operations required for them to execute on the RDDs. The operations are executed using the maximum amount of available processing power in the cluster, thus, speeding up the process. In this layer, data is sent to Spark master using Kafka messages. The tests made for comparing the two layers can be seen at Section IV.

D. Image Classification API

Frequently, analysts receive great volumes of data that may be used for investigation. As aforementioned, these data normally are extracted from various devices, and of different types, and within these, images are a very frequent and important type of data. The process of image analysis is a lengthy and exhausting process, which can entail in accumulation of data to be analysed and in delay to the investigation results. Given the necessary time spent to examine each device in its entire set of images and that the huge number of devices needed to be analysed, there is a great need of tools that can streamline the process.

Starting from the need of aid and speed up the image analysis process, instead of directly checking all the image files in a device, through the image classification API it is possible to classify objects of illicit character present in the evidences. At this point, the analyst will come across images classified as having an incriminatory or illicit character. The image classification API operation can be divided in 3 parts: image receiving, classification and results submission.

The API's working flow begins through a Kafka consumer which receives a message from an "Image" topic. The consumed message is referring to a new image that has yet to be classified. The image then is processed by a Deep Learning (DL) algorithm, which was trained with a specific dataset for the detection of illicit objects. This has the goal to detect items that fit the descriptions already seen by the DL. The positive results are then encapsulated and sent to a Kafka producer, responsible for taking forward this information.

These steps are executed for each image extracted from the evidence aiming to optimizing the image classification. Each image is processed with a specific classifier which can be tagged as belong to one or more classes. When the image is classified as document, identity or book, the textual information picted in the image are extracted and processed by Text Classification API. If the image is a person class, it is passed to another classifier, that extract the face and add it to database for facial recognition. Image classification uses YOLO (You Only Look Once) [25] [26] object detector. This tool is a convolutional neural network that addresses the detection task as a regression problem. Other object detection algorithms such as *R-CNN* and *Fast R-CNN* [27] analyzes the region of interest of the bounding boxes and only then trigger the convolutional network to classify these selected sections. Conversely, YOLO selects the area of interest and classifies it, doing both tasks in parallel.

The Image API contains classifiers offered by YOLO (*You Only Look Once*) for cover the classes: Person, Car, Motorcycle, Vehicle (truck, ambulance, bus, etc.); and also contains classifiers created by the members of this project. The implemented classifiers cover the classes: Firearms, Ammunition, Documents, and Identity; and the datasets used were built with images from real evidence.

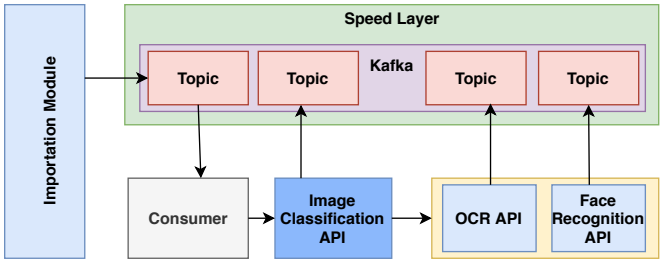

Fig. 2. Image Classification Pipeline

E. Message Classification API

Another essential requirement for an investigation is the analysis of the conversations between suspects in an evidence. In a chat, an analyst can find traces of crimes and illegal behaviours or legal proofs to incriminate a suspect. However, on a single smartphone, thousands of messages can be exchanged daily. Thus, a great number of messages need to be read by the analyst for a relevant message or chat to be detected. In this context, it is not difficult to realize that this is an exhausting and time-consuming process. To improve this, and to help analysts detect important messages, an API for text

classification was developed. This API uses Natural Language Processing for the Brazilian Portuguese to detect messages that can indicate possible crime.

Initially, a dataset for each type of crime was constructed with messages of real evidences. After, a classifier to detect suspect messages was implemented. In a literature review, different classifiers for text classification were found. Based on this review, we decided to test the Random Forest and Naive Bayes algorithms. In our experiments, Naive Bayes was faster, but Random Forest demonstrated better results when detecting suspect messages. Based on this result, the chose algorithm was the Random Forest. Initially, the algorithm is trained to detect two classes of crimes. We will refer to these classes as A and B.

The process of classifying a message starts with a Kafka consumer. After extraction, all evidence's messages are sent to a topic. The consumer subscribed to this topic consumes all messages in it and then sends it to the classification API. The API passes the message for each text classifier and gets the response as a common message, related to crime A or to crime B. The API then sends the message and its class to another Kafka's topic. This topic is then consumed by an application in the Serving Layer, and the message is saved into a relational database in the serving layer and able to be visualized by the analysts. This pipeline is shown in Figure 3.

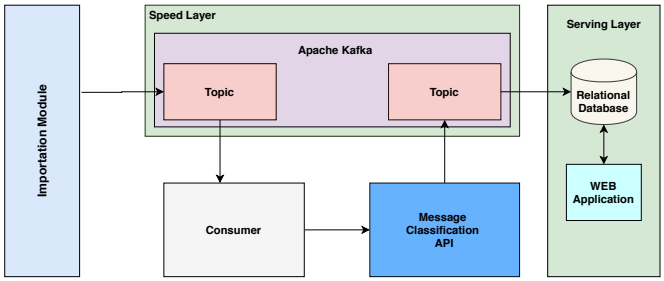

Fig. 3. Message Classification Pipeline

F. Serving Layer

As part of the lambda layer, the serving layer works as a visualization tool for the data present in the platform. In this case, serving layer is composed of a WEB service, and can contain data coming from the storage layer, or data resulting from the batch and speed layers. These results are then used to build views of the processed data, or, raw data present in the storage layer.

Besides data visualization, the Serving Layer has the function of giving feedback for the specialized data processing APIs. As those do not have a perfect accuracy, it is necessary to improve its metrics. Therefore, the analyst has the possibility to evaluate the classification result. If it is identified that, for example, an image is not of a particular class (weapon, document, etc.), the analyst can signalize in the interface that this specific result is wrong, and it also can correct it. Also, if the APIs do a partially correct classification (a bounding box out slightly out of place, for example), the user can also rectify it.

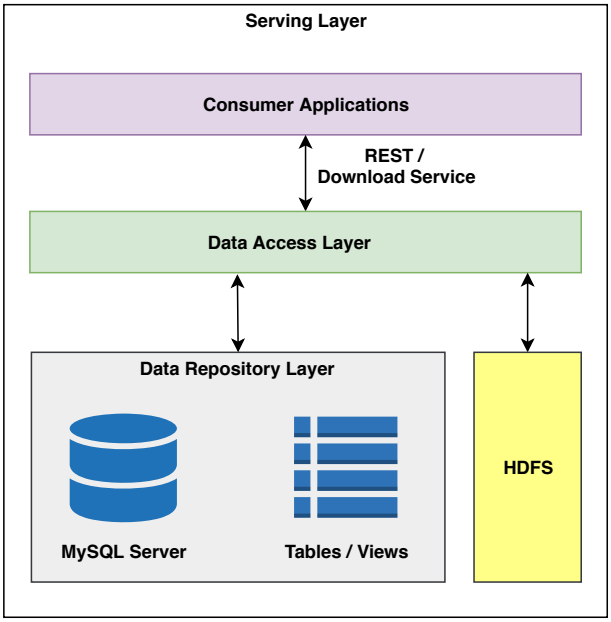

Fig. 4. Serving Layer Architecture

G. Storage Layer

The storage layer has the purpose of storing the data that went through the pipeline. This layer is in charge of storing, replicating and persisting all the data that is sent to the batch layer. This is achieved using HDFS, as the Hadoop cluster handles data persistence and software malfunctions using data replication. Hadoop replicates data present in a node to other nodes, so, if a node presents hardware failures, the data is safe elsewhere. In the proposed pipeline, to guarantee data persistence and at the same time not overload the storage, the data replication factor was set to two. Additionally, the serving layer can query data directly from the storage layer. Since the analysts in the serving layer can give feedback on the data importance, the storage layer is also able to receive data from the serving layer.

V. Evaluation

To assess the feasibility of the INSIDE platform, tests were conducted using real crime evidence data provided by the Public Ministry of Rio Grande do Norte, in Brazil. The tests used evidences with sizes ranging from 4.5GB to 7.5G and the evaluation of the platform was based on the requirements defined in Session III.

To maximize coverage, one of the main goals of the evaluation is to assess if all data from an evidence was processed and analysed. The developed Artificial Intelligence modules (message and image classification) classify each image and message in the evidence much faster and more effective than a human. New modules can be easily inserted in the future for new tasks such as processing video and audio. In the

classification context, tests were conducted to evaluate the Image Classification API processing in a serial manner and using Apache Spark. For the conducted tests, 5000 images were used as input and processed in each manner (serial and with Spark) using some of the steps of the pipeline shown in Figure 2. For each entry, the input was processed five times, and the value calculated for each entry is the mean of the input values. The tests results can be seen at Table I and II for both processing configurations (serial and Spark). For each entry, we have calculated the mean value of the time used in each step of the pipeline, for processing a single image. First, we have the time for the conversion to the base64 format. Second, the time for a Kafka consumer to receive an image and send it to the classifier. Then, we have the time for the Image Classification API to classify the image, and lastly the amount of time for the image to be produced to the Kafka topic.

TABLE I
COMPARISON BETWEEN SERIAL PROCESSING AND PROCESSING USING THE APACHE SPARK - TIME FOR CONVERSION TO BASE64 AND FOR KAFKA CONSUMER IMAGE INGESTION

	Serial		Spark	
Entry	**Base64**	**Consumer**	**Base64**	**Consumer**
1	0.5356	0.0172	0.0058	0.0043
2	0.5262	0.0162	0.0044	0.0044
3	0.5400	0.0137	0.0059	0.0059
4	0.5295	0.0198	0,0061	0.0061

TABLE II
COMPARISON BETWEEN SERIAL PROCESSING AND PROCESSING USING THE APACHE SPARK - TIME FOR CLASSIFICATION PROCESS AND FOR PRODUCER SEND IMAGES TO KAFKA

	Serial		Spark	
Entry	**Classification**	**Producer**	**Classification**	**Producer**
1	0.9700	0.00471	1.1002	0.0105
2	0.9827	0.00455	1.0005	0.0127
3	0.9975	0.00481	1.0246	0.0110
4	0.9176	0.00459	1.0360	0.0101

From the results present in Table I and Table II it is possible to assess that the only relevant difference in the classifier performance was at the base64 conversion step. The results indicate that Spark operations for data conversion into base64 is always faster than serial processing operations. Although the gain is significant in the conversion step, using Apache Spark has not provided a gain in time for the classification step. This happened because for the project evaluation, Spark was being executed at a cluster running on standalone mode, which means that was using its own resource negotiator. This resource negotiator is not able to execute Python scripts in a cluster, only locally, thus, not being able to parallelize the classification step in the Spark cluster. To address this problem, in future works, the classification step will be executed in a cluster using a resource negotiator such as YARN or Apache Mesos.

To retain data for as much time as necessary in a robust and fail-proof system, the proposed platform has an HDFS cluster containing all the data and metadata from the evidence with replication. At the same time, we still have part of the data saved on a relational SQL Server database for Service Layer consumption. Thus, if the relational database is deleted, the HDFS still has all data, preventing the necessity of other extraction. And if some HDFS nodes fail, we have the same data replicated in the HDFS and part of it in the relational database. To guarantee data security and the privacy of the investigation's target, the platform requires authentication and authorization for all operations. Each analyst has an account that uses to access the system and only the evidences that he has permission to access is available.

The pipeline architecture facilitates the addition of new modules for data processing. If there is the need for a new image or text classifier or a new module that will need evidence data as input, it can be done by adding a new Kafka consumer. This consumer process can also receive data from HDFS if there is the need to process older evidence.

VI. CONCLUSION

In our experiments, we provided some evidences that the use of big data technology has much to contribute to numerous areas of knowledge and their respective solutions, including crime evidence analysis. Once it enables the fast processing of data and the acquirement of important information that could help specialists achieve faster results without an accuracy reduction of the analysis, beneficial for the legal system, and the entire population. In this context, this work presented how a big data platform can be used in crime evidence analysis. We offered a big data pipeline for analyzing crime evidences, and developed a case study using it to prove it's applicability in a real-life scenario.

The use of Artificial Intelligence to process a big amount of data is an important resource to speed up an investigation. A great number of tools are available for parallel processing and many tools to classify data are found in the literature. The combination of these various tools provides the creation of a platform efficient to crime investigation and other areas such as industry, digital market, and any other area that uses a lot of data. In our study, many open-source tools are used to implement a platform able to withstand a great volume and processing of data. For storage, Hadoop HDFS proved to be a good ordered tool meeting all system requirements. The Apache Spark is a tool with great potential to be used in this platform, although there are still several components to be investigated, the few explored resources provided good results.

VII. FUTURE WORK

As future work, we plan to improve the data processing APIs by include modules to automatically detect suspects or important information contained in audios, and even, an audio transcription module. Similarly, a module for video analysis and processing is also an interesting module to implement in the future. Another important requirement is the addition of new classes to the Image Classification API to provide a

better experience to the analyst. Regarding the performance, the Apache Spark is a tool that can be better explored. The Spark cluster mode is a suggestion to increase speed up in the Image Classifier and future classifiers such as audio and video. Some techniques of parallel processing can be adding in the importation module and other modules of data processing.

VIII. Acknowledgment

This work is supported by Ministério Público do Rio Grande do Norte in the context of the INSIDE Project [1] and partially supported by the SmartMetropolis Project[2].

References

[1] A. Alexandrov, R. Bergmann, S. Ewen, J.-C. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M. J. Sax, S. Schelter, M. Höger, K. Tzoumas, and D. Warneke, "The stratosphere platform for big data analytics," *The VLDB Journal*, vol. 23, no. 6, pp. 939–964, Dec. 2014. [Online]. Available: https://doi.org/10.1007/s00778-014-0357-y

[2] C. Adrian, R. Abdullah, R. Atan, and Y. Y. Jusoh, "Factors influencing to the implementation success of big data analytics: A systematic literature review," in *2017 International Conference on Research and Innovation in Information Systems (ICRIIS)*, July 2017, pp. 1–6.

[3] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big data: The next frontier for innovation, competition," *Washington, DC: McKinsey Global Institute*, 2011.

[4] W. Raghupathi and V. Raghupathi, "Big data analytics in healthcare: promise and potential," *Health information science and systems*, vol. 2, no. 1, p. 3, 2014.

[5] E. Mezghani, E. Exposito, K. Drira, M. Da Silveira, and C. Pruski, "A semantic big data platform for integrating heterogeneous wearable data in healthcare," *Journal of medical systems*, vol. 39, no. 12, p. 185, 2015.

[6] N. Yu, S. Shah, R. Johnson, R. Sherick, M. Hong, and K. Loparo, "Big data analytics in power distribution systems," in *2015 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 2015, pp. 1–5.

[7] G.-H. Kim, S. Trimi, and J.-H. Chung, "Big-data applications in the government sector," *Communications of the ACM*, vol. 57, no. 3, pp. 78–85, 2014.

[8] H. Van Beek, E. van Eijk, R. van Baar, M. Ugen, J. Bodde, and A. Siemelink, "Digital forensics as a service: Game on," *Digital Investigation*, vol. 15, pp. 20–38, 2015.

[9] "Average storage capacity in smartphones to cross 80gb by end-2019," https://www.counterpointresearch.com/average-storage-capacity-smartphones-cross-80gb-end-2019/, accessed: 2019-12-20.

[10] Y. Wu and J. Zhang, "Building the electronic evidence analysis model based on association rule mining and fp-growth algorithm," *Soft Computing*, 2019.

[11] R. A. Bhoedjang, A. R. van Ballegooij, H. M. van Beek, J. C. van Schie, F. W. Dillema, R. B. van Baar, F. A. Ouwendijk, and M. Streppel, "Engineering an online computer forensic service," *Digital Investigation*, vol. 9, no. 2, pp. 96–108, 2012.

[12] A. Shaw and A. Browne, "A practical and robust approach to coping with large volumes of data submitted for digital forensic examination," *Digital Investigation*, vol. 10, no. 2, pp. 116–128, 2013.

[13] L. Sanchez, C. Grajeda, I. Baggili, and C. Hall, "A practitioner survey exploring the value of forensic tools, ai, filtering, & safer presentation for investigating child sexual abuse material (csam)," *Digital Investigation*, vol. 29, pp. S124–S142, 2019.

[14] F. Marturana, G. Me, R. Berte, and S. Tacconi, "A quantitative approach to triaging in mobile forensics," in *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2011, pp. 582–588.

[15] N. Kishore, S. Saxena, and P. Raina, "Big data as a challenge and opportunity in digital forensic investigation," in *2017 2nd International Conference on Telecommunication and Networks (TEL-NET)*. IEEE, 2017.

[16] P. H. Rughani, "Artificial intelligence based digital forensics framework." *International Journal of Advanced Research in Computer Science*, vol. 8, no. 8, 2017.

[17] H. Sachdev, L. Chen, C. Rebman *et al.*, "A new framework for securing, extracting and analyzing big forensic data," *Journal of Digital Forensics, Security and Law*, vol. 13, no. 2, p. 6, 2018.

[18] H. Mohammed, N. Clarke, and F. Li, "An automated approach for digital forensic analysis of heterogeneous big data," 2016.

[19] R. Van Baar, H. Van Beek, and E. Van Eijk, "Digital forensics as a service: A game changer," *Digital Investigation*, vol. 11, pp. S54–S62, 2014.

[20] S. Tahir and W. Iqbal, "Big dataan evolving concern for forensic investigators," in *2015 First International Conference on Anti-Cybercrime (ICACC)*. IEEE, 2015, pp. 1–6.

[21] G. Wang, J. Koshy, S. Subramanian, K. Paramasivam, M. Zadeh, N. Narkhede, J. Rao, J. Kreps, and J. Stein, "Building a replicated logging system with apache kafka," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1654–1655, 2015.

[22] K. Thein, "Apache kafka: Next generation distributed messaging system," *International Journal of Scientific Engineering and Technology Research*, vol. 3, no. 47, pp. 9478–9483, 2014.

[23] P. Le Noac'H, A. Costan, and L. Bougé, "A performance evaluation of apache kafka in support of big data streaming applications," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 4803–4806.

[24] K. Goodhope, J. Koshy, J. Kreps, N. Narkhede, R. Park, J. Rao, and V. Y. Ye, "Building linkedin's real-time activity data pipeline." *IEEE Data Eng. Bull.*, vol. 35, no. 2, pp. 33–45, 2012.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[26] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[27] R. Girshick, "Fast r-cnn," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[1] http://inside.imd.ufrn.br

[2] http://smartmetropolis.imd.ufrn.br