ISSN 2278-3091

Volume 8, No.4, July – August 2019

International Journal of Advanced Trends in Computer Science and Engineering

Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse16842019.pdf

https://doi.org/10.30534/ijatcse/2019/16842019

Big Data Processing Platform on Intelligent Transportation Systems

Saida EL MENDILI [1], Younès EL BOUZEKRI EL IDRISSI [2], Nabil HMINA [3]

[1] Systems Engineering Laboratory,National School of Applied Sciences Ibn Tofail University of Kenitra, Morocco, elmendili.saida@uit.ac.ma

[2] Systems Engineering Laboratory,National School of Applied Sciences Ibn Tofail University of Kenitra, Morocco y.elbouzkeri@gmail.com

[3] Systems Engineering Laboratory, National School of Applied Sciences Ibn Tofail University of Kenitra, Morocco, hmina@univ-ibntofail.ac.ma

ABSTRACT

Nowadays, with the incredible demographic explosion that we have witnessed in the last few decades, management of transport is of paramount importance. The reason for this is that we have to face the management of problems relating to traffic detection, traffic jams created by urban public transport, data on motorway tolls, meteorological data and traffic safety, etc. These types of traffic data are numerous and enormous. Traditional tools are now unable to solve these problems. With the rapid development of Big Data technologies, the new way of thinking about intelligent transport has become an obligation; as a result, new architectures are mainly needed to work with big data. . In order to overcome this problem, it is essential to create a Big Data modeling approach for ITS, which pays particular attention to the creation of multiple layers. Among these we find Management and Processing layer which in its turn contains three levels: processing, analyzing and storing. In this paper, we are interested in the processing level, which attracts the attention of researchers. In fact, we will propose a Big Data processing design applied to Intelligent Transportation Systems. We will adopt a data modeling approach that treats both the transmission and the processing data.

Key words : Smart city, Data processing, Big Data, Internet of things, Intelligent Transportation Systems, and flink.

1. INTRODUCTION

Nowadays, cities are the common choice for living. In their proposed work, they represent a complicated system in which governments must respond properly to the needs of citizens and ensure economic, social and environmental sustainability challenges [1]. A smart city is based on the need to understand the citizens, mainly their relevance in government and their services using big data applications. Because of the growth of the population, the development of the Internet of things, the management of data faces so many challenges. Besides, the generated data from heterogeneous sources with different formats will complicate the data management, and fast time responding needed for real-time applications. So, many problems such as traffic services, the emergency systems, the faultiness of environment monitoring and so on are due to the low utilization of urban data. In addition, intelligent transport systems play an essential role in the creation of an intelligent urban city. Today, many applications of ITS are deployed in smart cities, different devices generate a huge amount of data each day. These data are generated from multiple sources (smart phones, traffic monitoring, smart parking, information services on public transport [train, bus, taxi, GPS and people, surveillance, etc.]), the management of these data has now become a promising challenge because collecting the information present in the ITS environment in order to integrate it in the system and transmit it from one layer to another using middleware's technologies, will lead to feeding this data into applications that will help in decision-making or to launching actions. The extraction of useful information and hidden dependencies from these data is an issue that is of interest to researchers today, in order to provide better services To citizens and to support decision-making processes. However, to extract valuable information for the development of intelligent Transportation Systems at the city level, we need to set up a data modeling platform which will integrate and analyze the huge amount of ITS data, to produce useful information to help decision-makers plan for any expansion of Smart City services. To model these data, we can distinguish between three layers: Collection layer, Management and Processing layer and Application layer. In this paper, we are interested in the data processing level, which aims to analyze and manage the data generated by the collection layer. We apply our approach on a case study of data simulation of traffic accidents. The remainder of this paper is structured as follows: Section II represents the design and implementation of a smart transportation big data processing platform, the proposed platform and the related articles in this field; the existing solutions are reviewed and reported in Section III. Trends of future research and open issues are presented in Section IV.

2. DESIGN AND IMPLEMENTATION OF SMART TRANSPORTATION BIG DATA PROCESSING PLATFORM

2.1. Functional and Non-Functional Requirements of ITS Processing Design

Smart transportation devices continuously generate sizable data, which is gathered from traffic. In addition, since the rate of data generation varies according to the equipment, data processing with different generation rates is a major challenge. For example, frequency of GPS sensor sup dating is measured in seconds, while the frequency of updates for temperature sensors may be measured hourly. Whether the data generation rate is high or low, there always exists the danger of important information loss [14] . With the rapid development of the Internet of Things (IoT) and new Internet technologies in its context, a large amount of data is generated by, a large amount of data is generated from heterogeneous sources with different formats, a fact which complicates the data management, and fast time responding for real-time applications. This makes the development of a data processing platform more challenging. Considering all these aspects, the main purpose of this paper is to propose a data processing design for ITS. Our analytic architecture is based on the flow of the data generated by the data collection layer. As shown in Fig 1, the use case diagram explains a set of scenarios that describing the interaction between the ITS user and our system. Our use case diagram captures the functional requirement of the system and its interaction between the actor and the system.

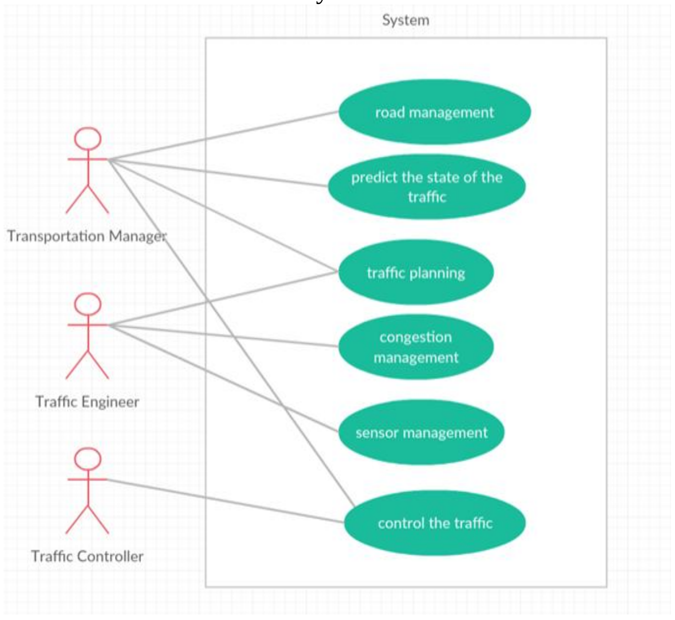

Figure 1: Traffic actors use case diagram

This paper presents a novel big data processing design for data flow handling specific to Smart Cities and applied on ITS. The design of this platform is based on answering the following research questions: What are the characteristics of big data processing frameworks and Big Data Transmission tools applied in ITS? What are the essential design principles that should guide this ITS architecture? In answering these questions, we adopted a systematic literature review on big data processing frameworks in ITS. The proposed platform introduces much functionality that makes good use of the collected data. We describe the functions and the components of each step and identify specific technologies. The value of the proposed platform is discussed in comparison to traditional knowledge discovery approaches supported by a comparative study of eac h functionality with an experimental test applied to traffic accident data. In doing so, our purpose is to check the efficiency of the proposed design.

The functional requirement mentioned above requires that our platform must also meet the following non-functional requirements:

- Management of a large size of traffic data increasing exponentially.
- Management of traffic data in real time.
- Providing low latency between the storage of traffic data and availability for treatment.

Using data collection tools, the data collection layer monitors, vehicles, traffic data and roads data, etc., the transformed traffic data, which includes structured data, semi-structured and unstructured data, is transmitted to the data management layer via middleware's.

As shown in Fig 2, the data collected by ITS sources can be:

- Structured: data that have been organized in a formatted repository (road site, sensor, detector sensor, GPS);
- Semi-structured: data that have not been organized into a specialized repository, as is the case in a database, but which nonetheless contain related information such as metadata, which makes them easier to process than raw data (radio station reports);
- Unstructured: data that do not reside in rows and columns of a traditional database (traffic camera, social media).

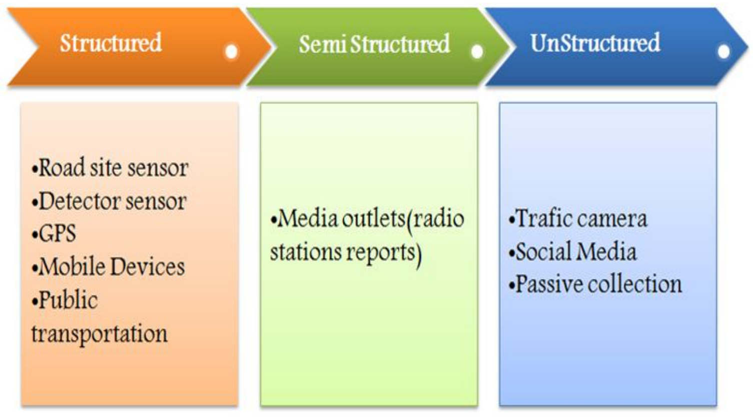

Figure 2: Type of data in ITS.

2.2. Big Data Processing Mode

Big data processing mode can be divided into stream processing and batch processing [4],[5]. The former is store-then-process, and the latter is straight through processing. In stream processing, the value of data reduces as time goes by which demanding real-time; on batch processing, data is stored first and then processed online and offline[16].Hadoop is the most representative of the batch processing method [6].

A. Batch Processing

Batch processing is the simultaneous processing of a large data flow. Data easily consists of millions of records for a day and can be stored in various ways (file, record, etc.) [18]. Operations are generally carried out in parallel in a sequential and uninterrupted chronological order. An example of a batch processing task is the set of transactions that a financial company can submit during a week. It can also be used in payroll processes, line item invoices, supply chain and order management [18].

B. Stream Processing

Continuous data processing is the process of analyzing data that moves from one device to another almost immediately [18]. This continuous calculation method occurs when data is transmitted through the system without a mandatory time limit on the output. With the near instantaneous throughput, systems do not need to store large amounts of data [18].

3. THE PROPOSED PLATFORM

3.1. Related Works

Cities always demand services to enhance the quality of life and make services more efficient. In the last few years, the concept of smart cities has played an important role in academia and in industry, the purpose of building smart cities is to improve services like traffic management, as well as the quality of life for citizens. Researchers have invested a lot of efforts on smart transportation, focusing on the development of services and applications, in fact, there are a few studies that have already explored smart transportation data processing design. In this section, about 20 articles in the field of smart transportation platform architecture designs are reviewed, so we present some existing studies and related work:

We indicate here research on developed architecture for Big Data processing in ITS, [13] proposed a cluster-based platform called "sipresk" to collect process and storage data for historical analysis.. Their platform is based on the Godzilla conceptual framework; the platform is a cluster-based and leverages the cloud to achieve reliability, scalability and adaptability to the changing operating conditions. It can be leveraged for both online and retrospective analysis; it is also validated in a case study where it is used to estimate the average speed and the congested sections of a highway.

In another study [10] Spark was used to clean and synchronize data from different sources, MongoDB is used to manage different data in real time and in batches. The DATEX-II data model is adopted, in order to harmonize the traffic data provided by road operators [10] .Similarly, in[3], different ITS actors (drivers, detectors, actuators, operators, etc.) have a role as editors or subscribers to Kafka topics. Kafka is used as a layer that separates publishers and subscribers from the analysis engine. Once a publisher publishes a new data element, it is sent to Kafka and saved in a Hadoop Distributed File System (HDFS) data warehouse for later analysis.

From the above, we can say that the literature applying Big Data processing approaches on ITS remains limited. In our work, we try to ameliorate this field by proposing architecture and a platform for Big-Data driven batch and real-time processing of ITS data and traffic control. However, before presenting the proposed approach, a comparison between these solutions in terms of performance is mandatory (as shown in table 1).

Table 1 : Comparison of existing solutions

	Strong points	**Weak points**
Sipresk: A Big Data Analytic Platform for Smart Transportation	Supports various types of analytics on different data sources	- Inconsistency between sahara project and other components - No test for real-time processing
An Architecture for Big Data Processing on Intelligent Transportation Systems	Provides a unified mobility framework for data cleansing Enable the coupling of machine learning methods Provides a platform for heterogeneous merging and harmonization of dynamic flows of transport data	- No test for real-time processing
Big Data Analytics Architecture for Real-Time Traffic Control	Supports data analysis in streaming mode Provides a simple way to specify a data analysis query	Possibility of hardware data loss, disconnections problem with real time with high speed transmission

3.2. Description of the ITS Platform

The ITS data processing platform can process massive data coming from the collection layer. The requirements of the platform are:

- To build a data transmission system; This system mainly makes it possible to move the data generated by the collection layer in real time. It must be evolutionary, reliable and fault-tolerant.

- To build a data processing and analysis system; this system is used to process data in real time and offline by Apache Flink.
- To build a streaming system; this system plays the role of an intermediary between the transmission system, the data processing and analysis system in order to deliver the data generated by our transmission system to the processing system in a continuous manner. It must be a real-time, scalable and fault-tolerant public service.

3.3. The main functions of the platform

Fig.3 illustrates the overall architecture of the proposed platform, which is the key element in the data management layer that acts as an intermediary between data sources and applications of the smart city. It comprises two components: real-time processing and batch processing:

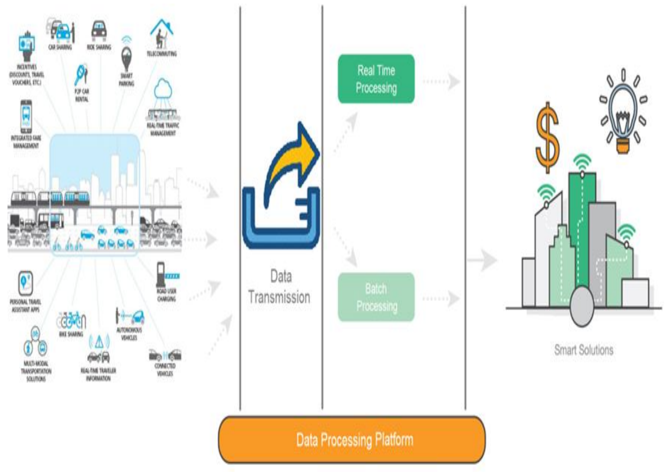

Figure 3: Conceptual data processing architecture.

Real-time processing module: allows us to process data in real time when it arrives and to quickly detect conditions in a short period of time from the receipt of the data. Real-time processing allows us to feed data into analytical tools as soon as they are generated and to obtain instant analytical results. We therefore read the data as it is generated from the data transfer module; with this approach, we can process the received data in real time almost any time. Real-time processing reads and writes data to different systems, including those that generate and use a constant data flow.
The batch-processing module: The batch processing module is the place where data blocks that have already been stored for a specified period of time are processed. This data contains millions of records for a day that can be stored as a file or record, etc. This particular file will undergo processing at the end of the day for various analyses that firm wants to do. Obviously, it will take a considerable amount of time for this file to be processed, which corresponds to batch processing.

3.4. The Key Technologies

Although the conceptual layer provides a coherent view of how data can be processed and made it available in an ITS environment, it is relevant to map potential technologies that may play a significant role in instantiating the proposed approach. A technological detail is recognized as one of the main contributions of the proposed architecture.
But it should be note, that there are several technologies of big data processing, for that, and in answering this question, we adopted a systematic review of the literature on Big Data Frameworks and Big Data Transmission tools, the value of this part is to make a comparative study between existing technologies based on real simulations to extract the most appropriate and the most efficient technologies.

A. Experimental Environment

In this paper, experiments are based on a PC with the following hardware configuration: Intel (R) Core(TM) i7-6500U, CPU @3.0GHZ*8, 16.00GB RAM and 1TB hard disk, 64-bit Operating System. The software environment uses the same configuration: Linux operating system (Ubuntu 16.04).

B. Data Set

The data used in these experiments are taken from traffic accident statistics provided by the National Statistics Institute (NIS) for the Flemish region (Belgium)] (Bart Goethals, 2014).
A total of 340,184 road accident records are included in the data set. A total of 572 attribute values are represented in the data set. On average, 45 attributes are completed for each accident in the data set. Traffic accident data contain a wide source of information on the different situations in which the accident occurred: Accident course (type of collision, road users, injuries), traffic conditions (maximum speed, priority control, environmental conditions (weather, lighting conditions, time of accident), road conditions (road surface, obstacles), human conditions (fatigue, alcohol) and geographical conditions (location, physical characteristics [7].

C. Big Data Frameworks Comparison

-Apache Spark
Apache Spark is a powerful processing framework that provides a user-freindly tool for efficient analytics of heterogeneous data. It was originally developed at UC Berkeley in 2009 [22]. Apache Spark is the next generation batch processing framework with stream processing capabilities. Built by using many of the same principles of Hadoop's MapReduce engine, Spark focuses primarily on speeding up batch processing workloads by offering full in-memory computation and processing optimization. Spark can be deployed as a standalone cluster (if paired with a capable storage layer) or it can hook into Hadoop as an alternative to the MapReduce engine [9].As shown in Fig 4, a Spark cluster is based on a master/slave architecture with three main components.

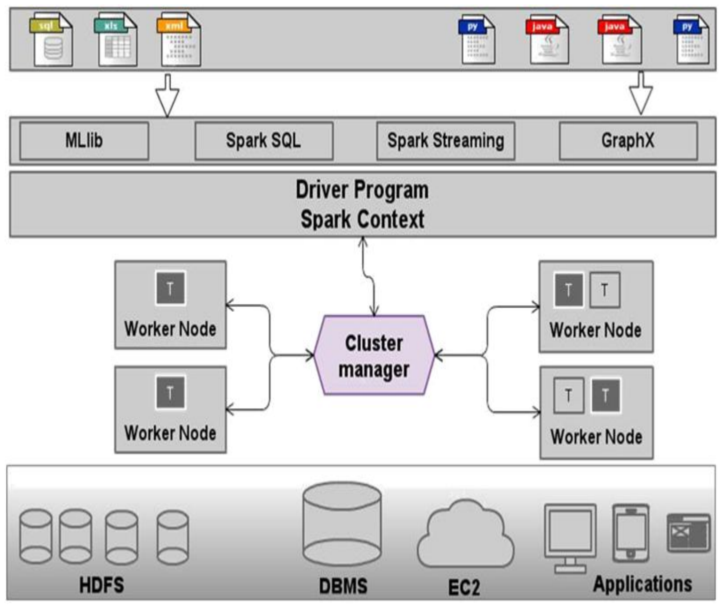

Figure 4: Spark Architecture [22] .

-Apache Storm

Storm [21] is an open source framework for processing large structured and unstructured data in real time. Storm is a fault tolerant framework that is suitable for real time data analysis, machine learning, sequential and iterative computation. Storm is geared for real time applications. As shown in Fig 5, a Storm program/topology is represented by directed acyclic graphs (DAG) [12]. The edges of the program DAG represent data transfer. The nodes of the DAG are divided into two types: Spouts and bolts. The spouts (or entry points) of a Storm program represent the data sources. The bolts represent the functions to be performed on the data. Note that Storm distributes bolts across multiple nodes to process the data in parallel.

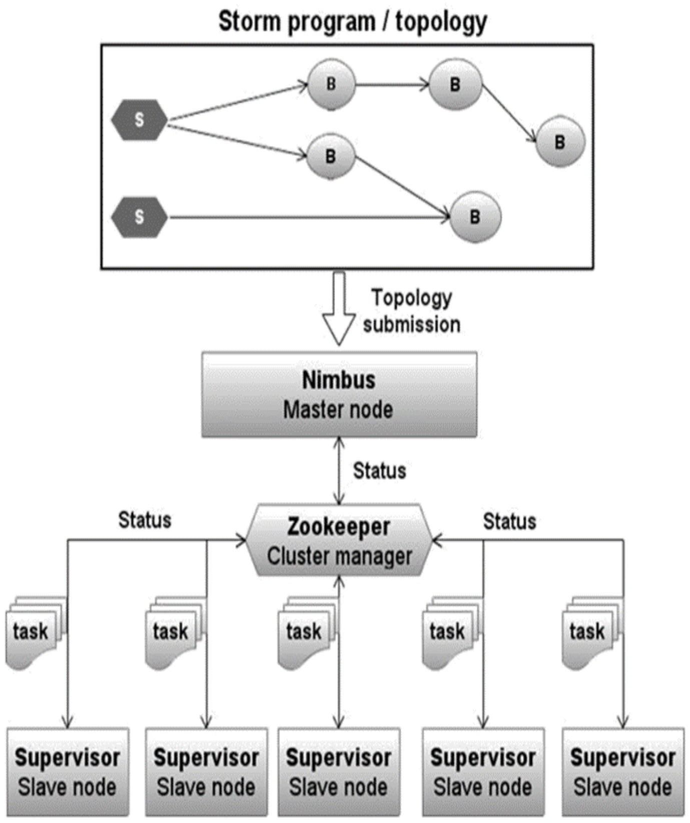

Figure 5: Storm Architecture [21]

-Apache Flink

Flink [2] is an open source framework for processing data in both real time mode and batch mode. It provides several benefits such as fault-tolerant and large-scale computation. The programming model of Flink is similar to Map Reduce. By contrast, to Map Reduce, Flink offers additional high-level functions such as join, filter and aggregation. Flink allows iterative processing and real-time computation of flow data collected by various tools such as

Table 2:. Comparison of big data processing frameworks

	Data Type	Data sources	Processing mode	Iterative computation	Performance
Flink	Key values	Kafka, kinesis	Batch and streaming	Yes	Memory: 8 GB, CPU:4-8 Cores
Spark	RDD	HDFS, DBMS and kafka	Batch and streaming	Yes	Memory: 16 GB, CPU:8-16 Cores
Storm	Key values	HDFS, DBMS and kafka	streaming	Yes	Memory: 24 GB, CPU:8-16 Cores

Flume and Kafka. It allocates APIs to launch a distributed calculation in a clear and simple way. Flink ML is a machine-learning library that provides learning algorithms for
creating fast and scalable Big Data applications. As shown in fig 6 there are different layers in the Flink ecosystem diagram.

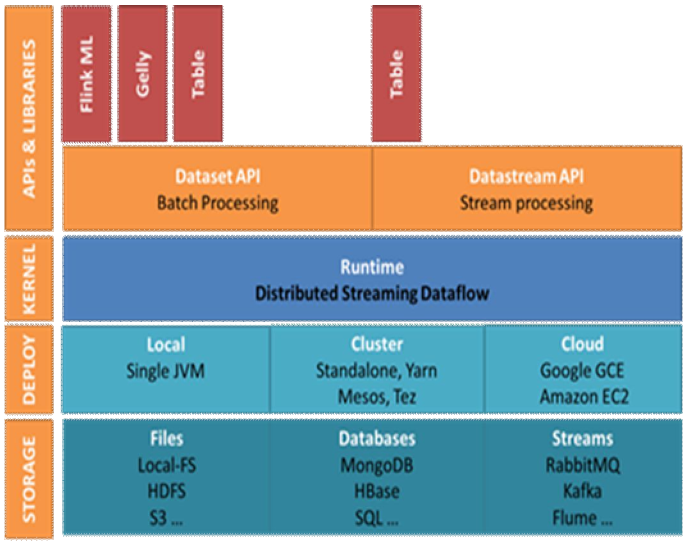

Figure 6: Flink ecosystem [2]

-Discussion

As shown in Table 2, iterative and incremental processing, micro-batch processing capabilities, and memory features are the highlights of spark. Spark-Shell is an interactive tool proposed by Spark which allows the exploitation of the Spark cluster in real time. After creating the interactive applications, they can then be interactively run in the cluster. Due to the Resilient Distributed Datasets (RDD) concept and DAG-based programming model, Spark is known to be very fast in certain types of applications. Flink shares many features with Spark. Complex Big Data structures, such as Flink graphics, offer great processing performance; they have APIs and specific tools for machine learning, predictive analytics, and graph flow analysis. Flink allows iterative processing and real-time computing on the flow of data collected by various tools such as Flume and Kafka.

To compare the three frameworks, we conducted three experiments using the data set presented below. The first experiment allows for comparing the performance as a function of time; Fig 7 shows that Flink and Storm have better processing rates compared to Spark, especially for large messages.

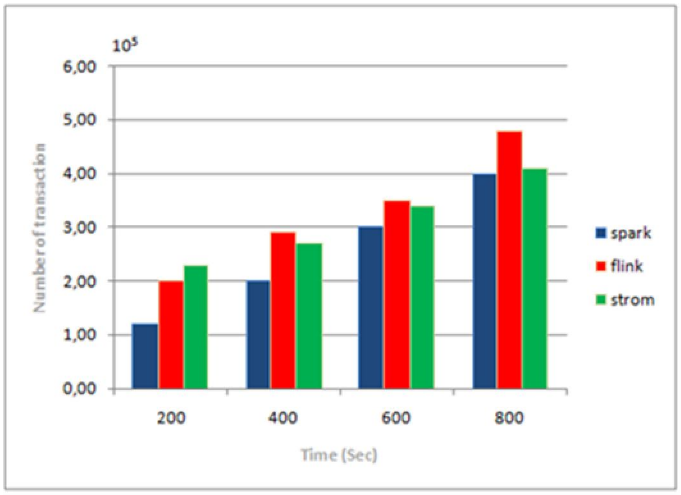

Figure 7: Number of transaction test.

In the second experiment, we compare the CPU consumption resources for the three frameworks. As shown in Fig. 8, Flink CPU consumption is low compared to Storm and spark. Flink uses about 18 of the available processors, while the use of the Storm processors varies between 35 and 41. This means that Flink can offer better results than other frameworks when CPU resources are more exploited.

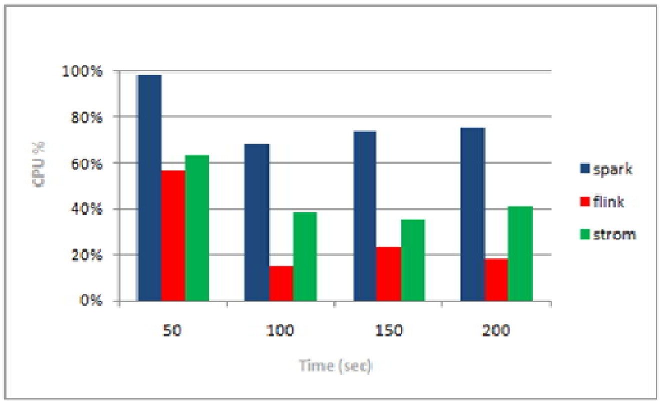

Figure 8: CPU consumption.

For the third experiment, the data processing rate is compared with the number of nodes. As shown in Fig. 9, Apache Flink has a higher throughput than the others. Even for two Flink nodes already shows better results than Spark and Storm. Spark and Storm have almost linear growth.

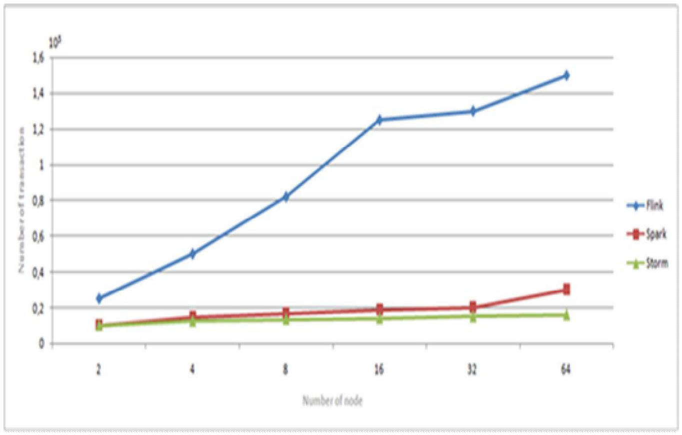

Figure 9: Number of nodes.

D. Big Data Transmission tools

There are several tools for data transfer that offer almost the same task. In this case it is important to answer this question: how these tools transform the data from a source to another. In this section, we present a comparison of some tools for importing and exporting data in and out of our proposed platform, which depends on the nature of the data being processed, the mode of data transfer, hardware and operating system platforms, user interface, reliability and fault tolerance. However, providing a complete list of tools is out of reach of this article. Here we present the most used tools[15] These tools include Sqoop and Flume.

-Apache Sqoop

Sqoop [20] and sqoop is a connection oriented, non-event based and open source software program for moving data between structured data stores and distributed file systems. Sqoop is mainly used for moving structured data (relational tables) stored in MySQL, Oracle or Microsoft SQL Server databases on a periodic basis as shown in Fig 10.

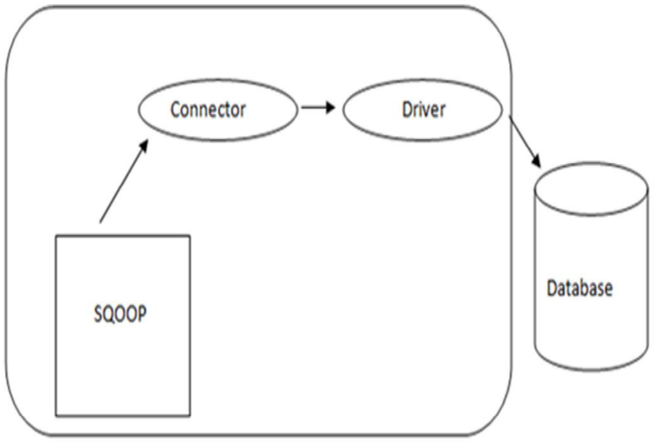

Figure 10 : Sqoop ecosystem[20] and sqoop1.

-Apache Flume

Apache Flume [11] is used for importing event-based data into Hadoop Distributed File System (HDFS). Unlike Sqoop, it is a one-way data collection, i.e., importing only. Apache Flume is a standard, straightforward, robust and flexible tool for streaming data ingestion into Hadoop. This Apache project, originally developed by Cloudera, received the status of incubator in 2012. It mainly consists of a set of agents, each having an instance of the Java Virtual Machine (JVM), requiring at least three components such as Flume Source, Flume Channel and Flume Sink. (as shown in Fig 11).

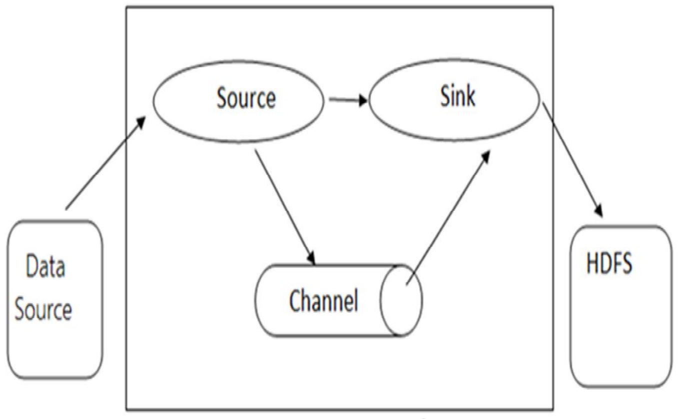

Figure 11: Flume ecosystem [11].

Flume Source is responsible for collecting incoming streams as events that are passed on to the Flume Sink via Flume Channel. Flume Channel uses in-memory and disk queues for storing events. Flume Sink later removes events from the Flume Channel after writing data into HDFS files. Flume is robust, self-contained, reliable, fault tolerant and event-based tool, which also supports filtering events by enabling multi-hope events transmission. Flume has been developed to handle streaming logs and data. But the data is being changed in the system from time to time. It is difficult to batch the data due to its dynamic nature. Configuring flume agents is an easy task that can be written in Java programming language. We can build custom sources by using Facebook or Twitter APIs for receiving data from each end and having provisions for sending received data to the Flume Channel and Flume Sink [17].

-Discussion

The summary of characteristics of the tools discussed above is shown in Table 3. Each tool is made for specific purposes. The similar characteristics are mainly due to the fact they are all playing with Hadoop ecosystem. Table 3 shows some use cases of the tools as discussed above. These tools help with migrating data from traditional data storage systems to HDFS. Sqoop is helpful in transferring structured data into HDFS. It can transfer bulk data and support bidirectional data transfer. Flume, on the other hand, is made for collecting different kind of log data and storing it into HDFS for further processing; it can be useful for data generated from different services of smart cities.

Table 3: Comparison of big data transmission tools

	Sqoop	**Flume**
Data Flow	RDBMS, NoSQL DB, Hive, HDFS	streaming data sources
Type of Loading	notdriven by events	completelyevent-driven
Used for	parallel data transfers, collecting and aggregating	collecting and aggregating data
HDFS Link	parallel For importing data, HDFS is the destination	data generally flow to through HDFS channels
Architecture	Connector based architecture	Agent based architecture
Fetch Data	structured data sources	streaming data
Performance	It reduces the excessive storage and processing loads by transferring them to other system and has fast performance	Flume is fault tolerant, robust and has tenable reliability mechanism for failover and recovery.
Use case	transfer between RDBMS and HDFS. Data transfer is only required to analyze and gain some intuitions from the data.	Analyzing huge amounts of log data helps identify unique threats or patterns. Flume helps collect, aggregate and moves this log data into HDFS. It is useful for sources, which generate a log and it can be useful for data generated from different services of smart cities.

E. Other Tools

Kafka [19] is a distributed messaging system. It has been used for collecting and delivering large volumes of log data with low latency [21]. Kafka provides the functionality of messaging system and it caters better throughput and fault tolerance. Kafka also performs some major tasks, which include monitoring operational data, log aggregation, website activity tracking, and event sourcing. Log aggregation collects log files from the web server and places them into a central storage area (HDFS). To prevent data loss, messages are stored on a disk and replicated within the cluster.

As a result, and following the comparative study presented above, figure 12 illustrates the technological layer, highlighting the most appropriate and the most efficient technologies.

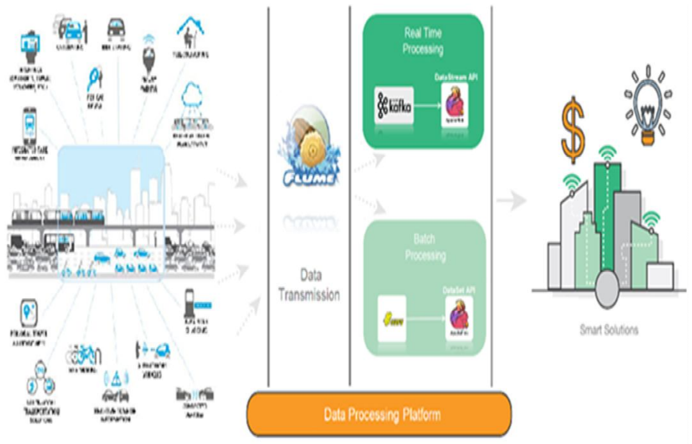

Figure 12 : Technological data processing architecture.

3.5. Proposed Big Data Processing Platform for ITS

The proposed architecture of the data processing platform for ITS has many advantages: reliability, scalability, high fault tolerance and high efficiency using technologies such as Flume, Kafka and Flink. In this experimental part, we transported large amounts of accidents data using Flume, to the HDFS. The first step is to create an application and get the recordings using the experimental source provided by Apache Flume. then we use the memory channel to buffer these records and HDFS sink to push these DATA into the HDFS for batch mode processing. Finally, we send the data to kafka for streaming processing. The data generator (Accidents data generators), generates data that is collected by an intermediate node known as "Collector" that executes them. This data collector (which is also an agent) collects data from the Flume agent, which is aggregated and pushed into the HDFS. Our Flume Agent contains three main components, namely, the source, the channel and the sink (As shown in Fig 13).

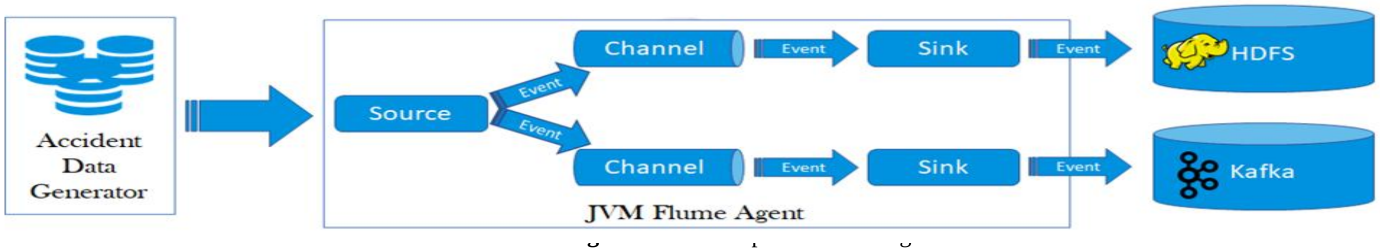

Figure 13: JVM Flame Agent

Data on accidents is stored in Kafka messages as a character string in JSON format. To achieve this, a script that contains a producer and a Kafka consumer must be created. The producer will receive Flume information every second. Each piece of the information contained in the Flume's response will be redirected to the Kafka topic. The consumer will store the information of different accidents. Once Flink gets the records, it provides special Kafka connectors for reading and writing data; it is now ready to use DataStream with the Kafka connector, so we can send and receive messages from Apache Kafka. Flink is the heart of our data processing system; it can process either streams of streaming data through Stream Builder and DataStream API, or batch data sets with batch optimizer and DataSet API. Its execution engine is scalable and distributed, allowing for the processing of massive data (streaming or batch), The execution of iterative operations, memory management and optimization of processing costs. Flink has a cache for storing data during processing. In addition, Flink autonomously manages its internal memory using its own data extraction and serialization components. It also optimizes network transfer and disk writing.

3.6. Experimental Results

In this analysis, the goal was to test the performance of our architecture in real time processing and batch processing on different datasets. The datasets were synthetically generated, i.e., 1 Go, 2 Go ,8 Go and run it on the platform cluster. We use acceleration as criteria to measure the performance of the proposed platform in the two cases (real time processing, Batch processing). When the size of the datasets increases by multiplying the number of transactions, as shown in Figure 14, the platform is more efficient in real time data processing. The reason real time processing is so fast is because it analyzes the data before it hits disk, so we can say that, in the point of performance, the latency of batch processing will be in a minute to hours while the latency of real time processing will be in sec.

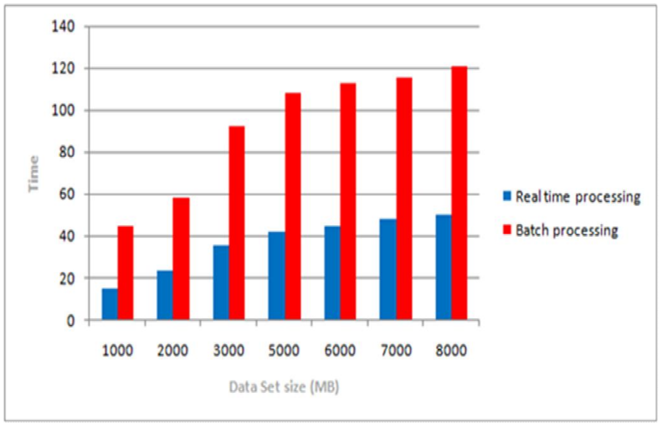

Figure 14: Experimental result.

4. CONCLUSION

In this paper, we have proposed a big data processing design for data flow specific to Smart Cities and applied on ITS. This platform will make good use of the collected data by adopting a data modeling approach. We have described the functions and components for each step and identified specific technologies. the proposed architecture is implemented in three modules: data transmission system, data processing and analysis system and a data streaming system. Experimentally, the proposed technical tools led to convincing results in terms of performance and speed using technologies such as : Hadoop ,Flume, Kafka and Flink. Although the proposed design introduces new features to big data analytics for ITS, the present work remains limited, because it's still in progress, with the need to implement and validate the other layers of the proposed architecture, namely the data collection layer and the application layer, as well as the other components of the data processing layer. As further work, we will first complete the architecture proposed by the different remaining levels and layers, then we will develop applications using the model proposed in our architecture according to different analyses. The overall purpose is to have users just call this application, change some parameters, and then can conduct various statistical analyses of data.

REFERENCES

1. Al Nuaimi, E., Al Neyadi,H., Mohamed,N.,& Al-Jaroodi,J.(2015). **Applications of big data to smart cities**, J. Interne,t Serv. Appl, vol. 6 no 1, 1-15. https://doi.org/10.1186/s13174-015-0041-5
2. Alexandrov, A., Bergmann, R., Ewen, S., Freytag, J., Hueske, F., Heise, A., Kao, O., Leich, M., Leser, U., Markl, V., Naumann, F., Peters, M., Rheinländer, A., Sax, M., Schelter, S., Höger, M., Tzoumas, K., &Warneke, D. (2014). **The Stratosphere platform for big data analytics**. The VLDB Journal, 23, 939-964. https://doi.org/10.1007/s00778-014-0357-y
3. Amini, S., Gerostathopoulos, I., &Prehofer, C. (2017). **Big data analytics architecture for real-time traffic control**. 2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS). doi:10.1109/mtits.2017.8005605
4. CHANG, Ray.M., Kauffman,R.J.,KWON,YO. (2014). **Understanding the paradigm shift to computational social science in the presence of big data, Research Collection School Of Information Systems**, Volume 63, 67-80. https://doi.org/10.1016/j.dss.2013.08.008
5. Chen,J., Chen,Y., Du,X.,Li,C.,Lu,J(2013). **Big dataallenge: a data management perspective**. Frontiers of Computer Science, Volume 7(2), 157-164. https://doi.org/10.1007/s11704-013-3903-7
6. Dean,J.,Ghemawat,S.(2008)**MapReduce:Simplified data processing on large clusters**. COMMUNICATIONS OF THE ACM January, Vol. 51, No. 1, 137-150. https://doi.org/10.1145/1327452.1327492
7. EL Mendili,S., EL Bouzekri EL IDRISSI,Y., Hmina,N. (2017).**Association rules mining on MapReduce**, BDCA'17 Proceedings of the 2nd international Conference on Big Data, Cloud and Applications Article No. 58 ,Tetouan, Morocco aˆ March, ACM New York, NY, USA. https://doi.org/10.1145/3090354.3090414
8. ELMENDILI, s., el BOUZEKERI, y. and HMINA, N. (2019). **Big Data Processing Platform for Smart City**, IEEE - International Symposium on Advanced Electrical and Communication Technologies (ISAECT2018), 21-23 november, 2018, Morocco IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/document/8618812 [Accessed 2019]. https://doi.org/10.1109/ISAECT.2018.8618812
9. Ellingwood,I.(2016).Hadoop, Storm, Samza, **Spark, and Flink: Big Data Frameworks Compared**. Retrieved from https://www.digitalocean.com/community/tutorials/hadoop-stormsamza-spark-and-flink-big-data-frameworks-compared.
10. Guerreiro, G., Figueiras, P., Silva, R., Costa, R., & Jardim-Goncalves, R. (2016). **An architecture for big data processing on intelligent transportation systems**. An application scenario on highway traffic flows. 2016 IEEE 8th International Conference on Intelligent Systems (IS). doi:10.1109/is.2016.7737393
11. Hoffman,S.(2015) **Apache Flume: Distributed Log Collection for Hadoop**.Retrieved fromhttps://www.packtpub.com/big-data-and-business-intelligence/apache-flume-distributed-logcollection-hadoop.
12. Inoubli,W., Aridhi,S., Mezni, H., & Jung, A. (2016). **Big Data Frameworks: A Comparative Study.** CoRR, abs/1610.09962.
13. Khazaei, H., Zareian, S., Veleda, R., Litoiu, M. (2016). **Sipresk: A Big Data Analytic Platform for Smart Transportation**, Retrieved from https://doi.org/10.1007/978-3-319-33681-7.
14. Mahdavinejad,M.S.,Rezvan,M., Barekatain,M., Adibi,P., Barnaghi,P., Sheth,A.P.(2017) **Machine learning for Internet of Things data analysis:** A survey, Journal of Digital Communications and Networks.vol.4,161-175. https://doi.org/10.1016/j.dcan.2017.10.002
15. Marjit, U., Sharma, K., & Mandal, P. (2015). **Data Transfers in Hadoop: A Comparative Study**. Open Journal of Big Data(OJBD), (Issue 2 ed., Vol. 1, Ser. 2015).doi:ISSN 2365-029X.
16. Meng,X., Ci,X.(2013) **Big Data Management: Concepts, Techniques and Challenges**. Journal of Computer Research and Development., Volume 50(1), 146-169.
17. Rathee,S. (2013). **Big Data and Hadoop with components like Flume, Pig,Hive and Jaql**. International Conference on Cloud, Big Data and Trust ,78-82,RGPV.
18. Shiff,L.(2018).**Real Time vs Batch Processing vs Stream Processing: What's The Difference**?. Retrieved from https://www.bmc.com/blogs/batch-processing-stream-processing-real-time.
19. Thein,K.M.M.(2014). **Apache Kafka: Next Generation Distributed Messaging System**. International Journal of Scientific Engineering and Technology Research, Vol.03, Issue.47, 9478-9483.
20. Ting,K.,&Cecho,J. J. (2013).**Apache Sqoop Cookbook. O'Reilly Media**.Retrieved fromhttp://shop.oreilly.com/product/0636920029519.do
21. Toshniwal,A., Taneja,S., Shukla,A., Ramasamy,K., Patel,J. M.,.Kulkarni,S., Jackson,J., Gade,K., Fu,M., Donham,J., Bhagat,N., Mittal,S., &Ryaboy,D.(2014). **Storm@twitter**. In Proceedings of the ACM SIGMOD International Conference on Management

of Data, volume 14,147-156,New York, NY, USA, ACM.
https://doi.org/10.1145/2588555.2595641

22. Zaharia,M., Chowdhury,M., Franklin,M. J., Shenker,S., &Stoica,I. (2010).**Spark: Cluster computing with working sets**. In Proceedings of the 2Nd 40 USENIX Conference on Hot Topics in Cloud Computing, HotCloud10(pp. 10-17), Berkeley, CA, USA, 2010. USENIX Association.