FUDMA Journal of Sciences (FJS)
ISSN online: 2616-1370
ISSN print: 2645 - 2944
Vol. 3 No. 3, September, 2019, pp 527 –534

A REAL-TIME DATA STREAM PROCESSING MODEL FOR A SMART TRAFFIC APPLICATION, LEVERAGING INTELLIGENT INTERNET OF THINGS (IOT) CONCEPTS

***[1]Yakubu, M. I., [2]Okorafor, E., [3]Momoh, K. O.**

[1]Department of Computer Science, Kano University of Science and Technology, Wudil, Nigeria.
[2]Department of Computer Science, African University of Science and Technology, Abuja, Nigeria.
[3]Department of Physics, Ahmadu Bello University, Zaria, Nigeria.

*Corresponding Author's Email: muleekat246@gmail.com

ABSTRACT
Smart City of Smart systems is becoming ubiquitous. Improvements in miniaturization and networking capabilities of sensors have contributed to the proliferation of the Internet of Things (IoT) and continuous sensing environments. Data streams generated in such settings must keep pace with generation rates and be processed in real-time to gain insight quickly and make decisions that are in most cases critical and time-sensitive. The challenge lies in not only being able to process vast amounts of data in a given time but also being able to make data-driven decisions quickly or in many cases proactively. Handling the amounts of data generated could be very difficult especially when making data-driven decisions. The difficulty is being diminished using some big data methods to perform real-time stream processing. Among the different dimensions that improve the quality of life of people in a smart city environment, one of the important ones is transportation. In this work, a real-time data stream processing model for a smart traffic application was proposed and data streaming trends were used to monitor traffic which enables people to know if the roads in an IoT environment are congested.

***Keywords**: Smart traffic, IoT, Real-time, Data stream.*

INTRODUCTION

Quintillion bytes of data are being generated daily and handling these enormous amounts of data is becoming more tedious every day. These bytes of data are generated by people using some devices such as Mobile phones, connected vehicles, Laptops, smart devices, and these devices are connected to the internet so as to be able to identify themselves to other devices (Gehlot, 2016). According to (Santana, Chaves, Gerosa, Kon, & Milojicic, 2016), Smart City is a city in which it's social, business, communication, and technological aspects are supported by Information and Communication Technologies such as intelligent internet of things and data collection sensors to improve the experience of the citizen within the city. As the sensor senses the environments continuously, Data streams generated must be processed in real-time so as to gain insight quickly because the data generated are in many cases critical and time-sensitive. Smart cities are built on internet of things. (Nuaimi, Neyadi, Mohamed, & Al-jaroodi, 2015) said the big data systems will store, process, and mine smart city applications information in an efficient manner to generate information to improve and enhance different smart city services. In addition, the big data will help decision-makers who will use the data generated by sensors, to plan for any expansion and extension such as making data-driven decisions in either smart city services, resources, or areas. The Internet of things (IoT) environment have three components which include sensors which senses the movement of objects, actuators and embedded communication hardware; a middleware, which analyses and stores data and information generated by the hardware and a presentation layer, in which users use to access, view, manipulate, and visualize data extracted from the hardware (Santana et al., 2016). There is a wide range of services and applications. These services cover fields such as transportation (intelligent road networks, smart mobility, smart traffic lights, smart parking systems, connected cars and public transport), public utilities (smart electricity, water and gas distribution), education, technology, health and social care, public safety (Radek Kuchta, Kuchta, & Kadlec, 2014). This paper focuses on Smart traffic which is an important area of smart city. One of the problems that most cities face is the problem of traffic management. As the day goes by, the world's population tends to grow high which leads to congestion on the road because of the high population of people in the urban areas. The smart traffic application makes life easier by leveraging the Internet of things concepts to monitor the traffic thereby reducing the traffic congestion on the road and avoiding traffic jams. It is an IoT data processing and monitoring application that uses Spark Streaming and Kafka. The data generated by the sensors are used to monitor the traffic on how the traffic jam can be prevented. So to process this kind of data, we use Apache Hadoop. According to (Nagdive, 2018) Hadoop is an open-source software framework for storing data and running applications on clusters of commodity hardware. It provides massive storage for various kinds of data, enormous processing power and the ability to handle virtually limitless concurrent tasks. All these parts are analyzed in parallel and the results of the analysis are regrouped to produce the final output. Kafka provides a publish-subscribe messaging service, as illustrated in figure 1.

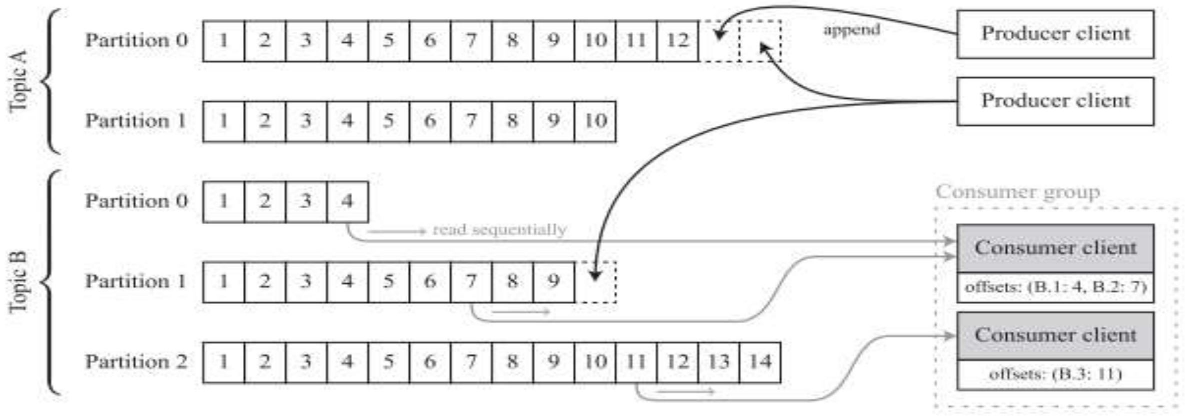


Fig. 1: A Kafka topic divided into partitions

Producer (publisher) clients write messages to a named topic, and consumer (subscriber) clients read messages in a topic (Kleppmann & Kreps). Apache Spark is a fast, in-memory data processing engine with classic and expressive development APIs to allow data workers to masterly execute streaming, machine learning or SQL workloads, graph processing that require fast continual access to dataset. Since the paper talks about the real-time data stream processing, the data needs to be processed as fast as possible. So as to achieve this, we need a fast platform (Conese, 2015). After the big data methods are being leveraged, then the data is used to make data-driven decisions to know which decision to take to monitor if a particular route is congested or not by knowing the number of vehicles on a particular route. The proposed system is used to monitor the traffic congestion on a particular road. Specifically, this research is concerned with the following:

Real-time integration of Apache Kafka with Spark to produce the data from the connected vehicles using Kafka producer and processing the data using Apache Spark;

Showing the processing speed and the number of jobs that were processed by Apache Spark;

Showing the number of Vehicles on different routes and the time at which the vehicles are available on the route.

RELATED WORKS

(Prathilothamai, Lakshmi, & Viswanthan, 2016) proposed a cost-effective model to predict the traffic jam to inform the public about the current traffic condition to all persons who are entering the same lane. The architecture they proposed was analyzed using Apache Spark and Apache Hadoop. The paper focused on the idea of traffic prediction based on mainly two factors such as the Speed of vehicle and the number of the vehicle. The sensor data was first collected, converted to a Comma-separated Value (CSV) file to retrieve the count of the vehicles passed, and the speed of the vehicles within a particular duration. After the retrieval, it was then loaded into Apache Spark and used to predict the traffic congestion using the decision tree model. In this paper, Timely Prediction of Road Traffic Congestion by processing of sensor data in Apache Spark is the main approach, it exploits. In (Lakshminarasimhan, 2016), advanced traffic management system was proposed which was implemented using internet of things. This paper focused on the communication between vehicles, processing unit and traffic lights. Vehicles are connected in such a way that all the information on the vehicles are captured so as to know the speed and to see the vehicles causing congestion on a particular road (Lakshminarasimhan, 2016). These connected vehicles generate all the information needed so as to know how it works. The data is being captured by the sensor, which transmits traffic data. The communication is established by a socket programming over Wi-Fi connection, so ports are like „mouth‟ and „ear‟ of sender and receiver consecutively. Vehicles „name‟ each other by calling RFID Reader name. The system also consists of a circuit embedded in each vehicle in commutation (Lakshminarasimhan, 2016). This paper analyses the ever-growing urban population around the globe and discusses the traffic systems in densely populated cities. Furthermore, an advanced traffic management system was proposed which was implemented using IoT. In this case, each vehicle acts as eye, which transmits traffic data. (Hahanov, 2015) described a cognitive traffic management system based on the internet of things approach. The Telecommunication technologies which can be used for the system development are analyzed. In the paper, smart traffic light integration was proposed as a replacement for the existing traffic lights. The system he proposed generates control signals based on the information obtained during the analysis of data obtained from cars and road sensors as well as alternative sources of user data such as social networks, the result of opinion polls, and others. This paper describes a CTMS based on the IoT approach. The telecommunication technologies which are used for the system development are analyzed. In the paper, smart traffic light integration was proposed as a replacement for the existing traffic lights. (Amini & Prehofer) proposed a comprehensive and flexible architecture based on distributed computing platform for real-time traffic control which is based on systematic analysis of the requirements of the existing traffic control system. In the architecture that was proposed, Big Data analytics engine informs the control logic. The system uses the distributed streaming platform Kafka and big data tools for the stream processing and building data pipelines. This paper gives an overview of both the existing system ITMS and the big data analytics approach. Kafka is a built-in mechanism hence it can tolerate hardware faults example failing computing machines. (Campus et al., 2017) said IoT based traffic signaling system is based on traffic density on road where the count of vehicles is done at each side of road by placement of sensor. In this system, three traffic lights, that is, Green amber and red are placed on junction of roadsides. Two pairs of sensor are placed across the roads which mark the distance for density zones.

EXPERIMENTAL RESULTS AND DISCUSSION

Real-Time Integration of Apache Kafka with Apache Spark

Kafka is a publish-subscribe messaging system used in messaging or ingesting data. It is used to fetch and collect the data from servers through and integrates with Spark to perform data analytics. After the data is been processed, it is pushed to the database for storage. Apache Kafka was used since it is a distributed streaming platform. Kafka supports multi-subscribers and automatically balances the consumer during failure. The proposed system is a data processing and monitoring system that processes data sent by a connected vehicle and uses the processed data to monitor traffic on a particular route. Data were generated randomly to test how the system works.

The Architecture of the Proposed System

Data is being captured from sensors and stored for processing. The amount of data generated by the sensors tends to be too large for processing in such a way that it cannot be handled by the file handling systems. So better ways are being looked out to utilize the vast amounts of data gotten from the sensors in real-time by stream processing. So, stream processing that can support real-time analytics is leveraged. To implement real-time data analytics, the system must be scalable and fault-tolerant while ensuring low latency and high availability. Stream Analytics is a processing engine that can process real-time events in real-time either from one data stream or from multiple streams. The events are carried from the sensors and a variety of sources. In an internet of things environment, Vehicles are connected in such a way that all the information of the vehicles are captured in order to know the speed, fuel level, longitude, latitude, and the particular road where the vehicles are moving in real-time to see the vehicles causing congestion on a particular road. These connected vehicles generate all the information needed so as to make decisions when an accident occurs when a vehicle over speeds and when a vehicle disobeys traffic rules. The data is captured by the sensor attached to the vehicles and once the data is captured, the big data tools are used for stream processing. In a smart city environment, to implement Traffic monitoring, vehicles are connected to capture information regarding their speeds and hence determine vehicles causing congestion thereby enabling the system to work. Data is captured by the sensors; then it is fetched from the sensor by Apache Kafka. Kafka captures the data from the sensors as soon as it is generated. It pushes the data to the brokers, the Kafka server then replicates the data to avoid data loss and also ensures that the data is delivered for processing. Apache Spark is a fast and general engine that is used for big data processing. It subscribes to the data from the Kafka topics and performs stream processing. Data is generated randomly by the application. Spark performs stream processing by counting the number of vehicles on a road. It then saves the data in the Cassandra tables. Cassandra is a NoSQL database that is used for storage processing. The data which is processed by Apache Spark is now persisted in the Cassandra tables. Then the Cassandra tables are used to push the data to the traffic monitoring system which displays the information of the vehicle. The keyspace and tables in Cassandra were created using the *cqlsh* command. The architecture of the proposed system is shown in figure 2.

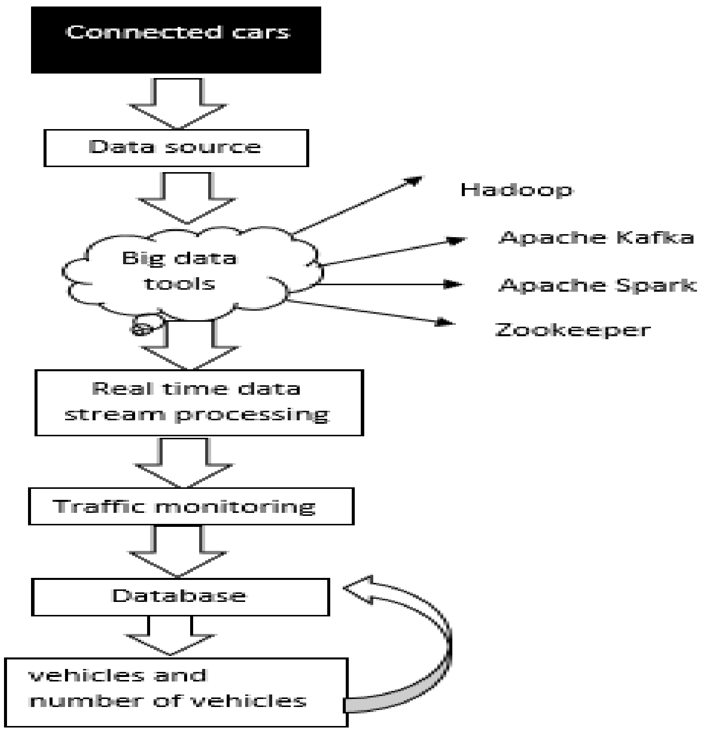


Fig 2: Architecture of the proposed system

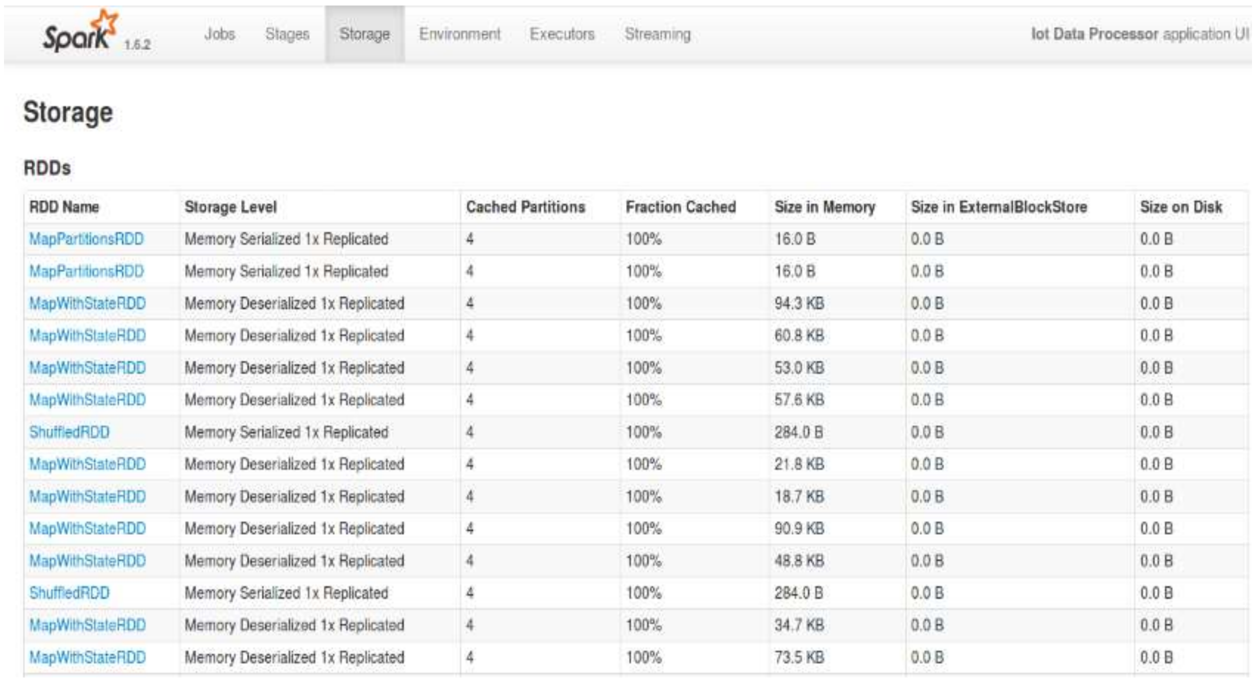
The System is categorically divided into three parts namely:
The producers, the consumers, and the results.

The Producers

The application generates data randomly by itself so it acts like a real-time IoT platform. Vehicle type and the route are defined first comprising of three routes and five types of vehicles being used. The type of connected vehicles found in a city is truck, bus, car, taxi, private car. A random class is then used to generate values of the attributes of these connected vehicles. The attributes of these vehicles are the Vehicle_ID, longitude and latitude, time stamp, speed, and fuel level. So, in the first stage, where the data is produced each connected vehicle sends event information randomly while moving in a particular route. The producer acts as a data simulator application where data is produced. Once it produces the data, Apache Kafka now generates and captures the data for stream processing. The data is captured by the producer, pushed to the topics for storage and waits for the brokers to replicate the data to avoid data loss. Then, the data is now consumed by the consumer application. The producer produces the data in Javascript object notation format which is known as the (JSON). JSON is an open-standard file format that provides a method to store files in a readable format. Once everything is launched, the Kafka producer starts producing the messages in streams and saves everything to the Kafka topics. As it is saved, the consumer subscribes from the topics to process the data.

The Consumers

This is where Spark streaming is done. Spark is the consumer which subscribes to the data from the topics for streaming. As soon as the data is received, Spark streaming processes it to gain insight quickly and keep pace with generation rates. Once it is done with the processing, it is stored in the Cassandra database. The consumers try to calculate the vehicle count. To achieve this, it saves the previous records which are already processed and maps it with the current one which is been processed. The master node in the Spark cluster assigns tasks to the worker nodes and the tasks are run on the worker node on the Spark cluster for processing and once it is done, it is saved to the Cassandra database. This data that has been processed contains the latitude and longitude which is needed to detect the actual route. Java Random class is used to generate values of the attributes of the vehicles. Data is generated randomly by the producer application. Apache Spark consumes the messages from the topics and starts processing immediately. After each process, it pushes everything to the Cassandra database. The Cassandra database forwards the results processed by Kafka and the results are being queried from the database. Overleaf (figure 3) shows the Spark Resilient distributed data also known as RDD which is the representation of a set of data spread across the machines and allows in-memory processing and ensures that data is computed on different nodes on the Spark cluster.



Storage

RDDs

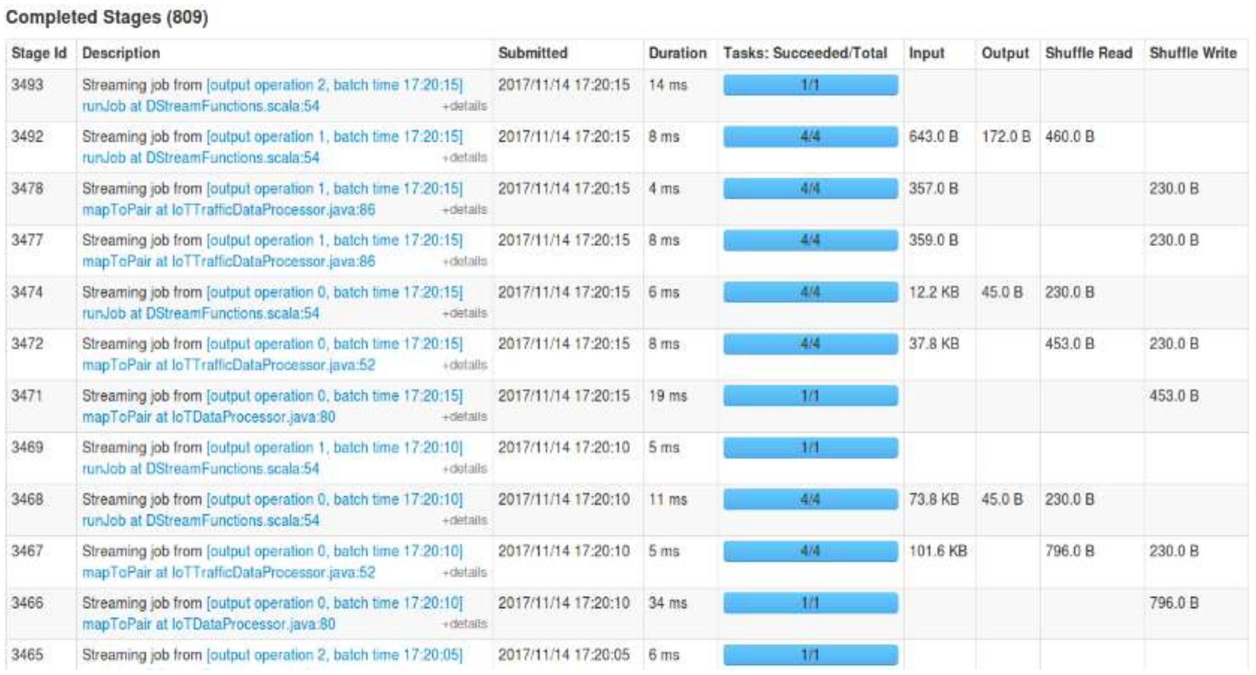
RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size in ExternalBlockStore	Size on Disk
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	16.0 B	0.0 B	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	16.0 B	0.0 B	0.0 B
MapWithStateRDD	Memory Deserialized 1x Replicated	4	100%	94.3 KB	0.0 B	0.0 B
MapWithStateRDD	Memory Deserialized 1x Replicated	4	100%	60.8 KB	0.0 B	0.0 B
MapWithStateRDD	Memory Deserialized 1x Replicated	4	100%	53.0 KB	0.0 B	0.0 B
MapWithStateRDD	Memory Deserialized 1x Replicated	4	100%	57.6 KB	0.0 B	0.0 B
ShuffledRDD	Memory Serialized 1x Replicated	4	100%	284.0 B	0.0 B	0.0 B
MapWithStateRDD	Memory Deserialized 1x Replicated	4	100%	21.8 KB	0.0 B	0.0 B
MapWithStateRDD	Memory Deserialized 1x Replicated	4	100%	18.7 KB	0.0 B	0.0 B
MapWithStateRDD	Memory Deserialized 1x Replicated	4	100%	90.9 KB	0.0 B	0.0 B
MapWithStateRDD	Memory Deserialized 1x Replicated	4	100%	48.8 KB	0.0 B	0.0 B
ShuffledRDD	Memory Serialized 1x Replicated	4	100%	284.0 B	0.0 B	0.0 B
MapWithStateRDD	Memory Deserialized 1x Replicated	4	100%	34.7 KB	0.0 B	0.0 B
MapWithStateRDD	Memory Deserialized 1x Replicated	4	100%	73.5 KB	0.0 B	0.0 B

Fig 3: Resilient Distributed Data on Spark.

Apache Cassandra is launched; then Apache Zookeeper is started because Kafka cannot work without Zookeeper starting perfectly. Also, the Kafka and the zookeeper servers are started. Then Spark is started. In Spark, worker node and master nodes must all be started because that is where the jobs are submitted for processing. Also, Spark-submit is used to submit all the jobs to the Spark cluster for stream processing.

Spark Streaming

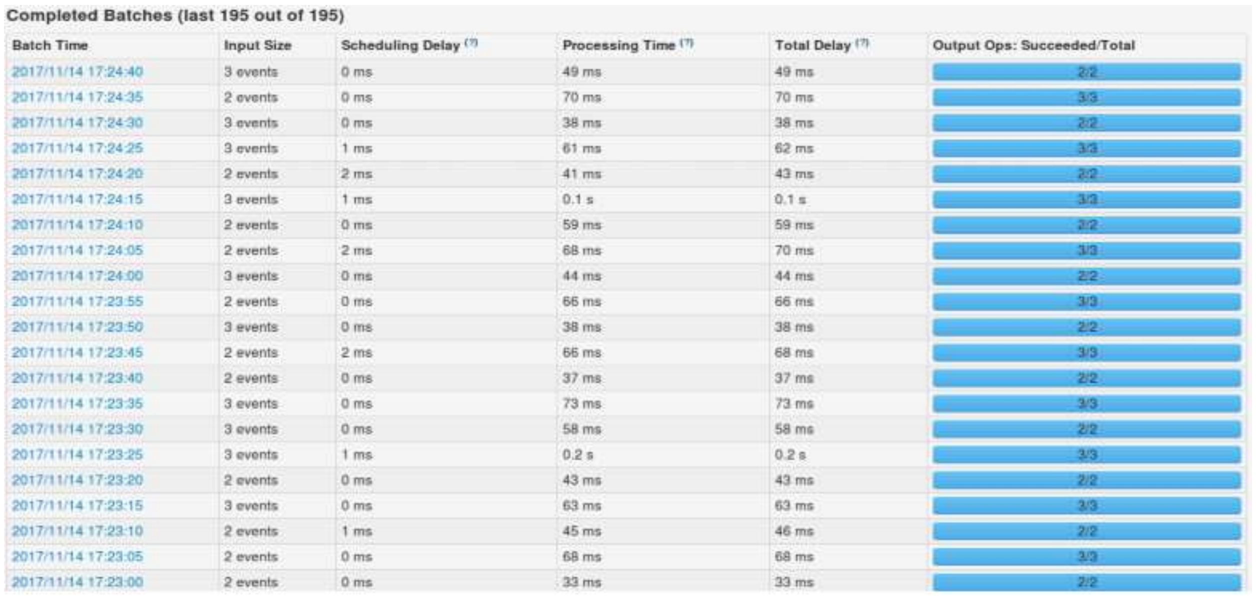
Apache Spark consumes the messages from the topics and starts processing immediately. After each process, it pushes everything to the Cassandra database. The Cassandra database displays the results processed by Kafka.



Completed Stages (809)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
3493	Streaming job from [output operation 2, batch time 17:20:15] runJob at DStreamFunctions.scala:54 +details	2017/11/14 17:20:15	14 ms	1/1				
3492	Streaming job from [output operation 1, batch time 17:20:15] runJob at DStreamFunctions.scala:54 +details	2017/11/14 17:20:15	8 ms	4/4	643.0 B	172.0 B	460.0 B	
3478	Streaming job from [output operation 1, batch time 17:20:15] mapToPair at IoTTrafficDataProcessor.java:86 +details	2017/11/14 17:20:15	4 ms	4/4	357.0 B			230.0 B
3477	Streaming job from [output operation 1, batch time 17:20:15] mapToPair at IoTTrafficDataProcessor.java:86 +details	2017/11/14 17:20:15	8 ms	4/4	359.0 B			230.0 B
3474	Streaming job from [output operation 0, batch time 17:20:15] runJob at DStreamFunctions.scala:54 +details	2017/11/14 17:20:15	6 ms	4/4	12.2 KB	45.0 B	230.0 B	
3472	Streaming job from [output operation 0, batch time 17:20:15] mapToPair at IoTTrafficDataProcessor.java:52 +details	2017/11/14 17:20:15	8 ms	4/4	37.8 KB		453.0 B	230.0 B
3471	Streaming job from [output operation 0, batch time 17:20:15] mapToPair at IoTDataProcessor.java:80 +details	2017/11/14 17:20:15	19 ms	1/1				453.0 B
3469	Streaming job from [output operation 1, batch time 17:20:10] runJob at DStreamFunctions.scala:54 +details	2017/11/14 17:20:10	5 ms	1/1				
3468	Streaming job from [output operation 0, batch time 17:20:10] runJob at DStreamFunctions.scala:54 +details	2017/11/14 17:20:10	11 ms	4/4	73.8 KB	45.0 B	230.0 B	
3467	Streaming job from [output operation 0, batch time 17:20:10] mapToPair at IoTTrafficDataProcessor.java:52 +details	2017/11/14 17:20:10	5 ms	4/4	101.6 KB		796.0 B	230.0 B
3466	Streaming job from [output operation 0, batch time 17:20:10] mapToPair at IoTDataProcessor.java:80 +details	2017/11/14 17:20:10	34 ms	1/1				796.0 B
3465	Streaming job from [output operation 2, batch time 17:20:05]	2017/11/14 17:20:05	6 ms	1/1				

Fig 4: Picture of submitted jobs.



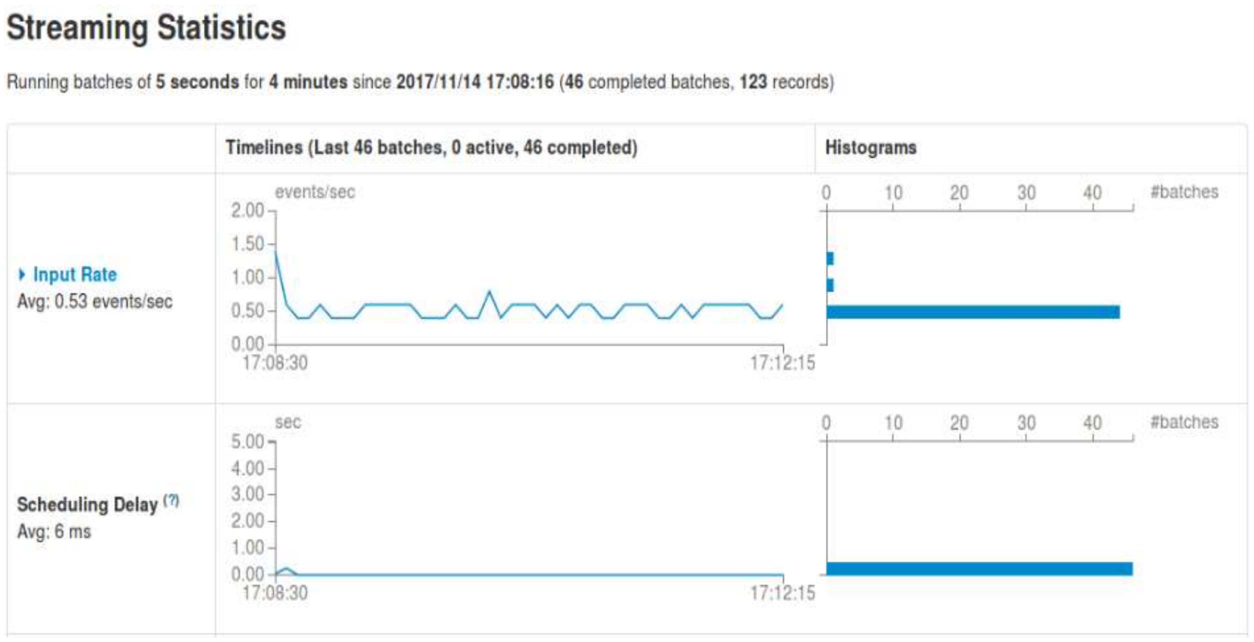
Completed Batches (last 195 out of 195)

Batch Time	Input Size	Scheduling Delay (?)	Processing Time (?)	Total Delay (?)	Output Ops: Succeeded/Total
2017/11/14 17:24:40	3 events	0 ms	49 ms	49 ms	2/2
2017/11/14 17:24:35	2 events	0 ms	70 ms	70 ms	3/3
2017/11/14 17:24:30	3 events	0 ms	38 ms	38 ms	2/2
2017/11/14 17:24:25	3 events	1 ms	61 ms	62 ms	3/3
2017/11/14 17:24:20	2 events	2 ms	41 ms	43 ms	2/2
2017/11/14 17:24:15	3 events	1 ms	0.1 s	0.1 s	3/3
2017/11/14 17:24:10	2 events	0 ms	59 ms	59 ms	2/2
2017/11/14 17:24:05	2 events	2 ms	68 ms	70 ms	3/3
2017/11/14 17:24:00	3 events	0 ms	44 ms	44 ms	2/2
2017/11/14 17:23:55	2 events	0 ms	66 ms	66 ms	3/3
2017/11/14 17:23:50	3 events	0 ms	38 ms	38 ms	2/2
2017/11/14 17:23:45	2 events	2 ms	66 ms	68 ms	3/3
2017/11/14 17:23:40	2 events	0 ms	37 ms	37 ms	2/2
2017/11/14 17:23:35	3 events	0 ms	73 ms	73 ms	3/3
2017/11/14 17:23:30	3 events	0 ms	58 ms	58 ms	2/2
2017/11/14 17:23:25	3 events	1 ms	0.2 s	0.2 s	3/3
2017/11/14 17:23:20	2 events	0 ms	43 ms	43 ms	2/2
2017/11/14 17:23:15	3 events	0 ms	63 ms	63 ms	3/3
2017/11/14 17:23:10	2 events	1 ms	45 ms	46 ms	2/2
2017/11/14 17:23:05	2 events	0 ms	68 ms	68 ms	3/3
2017/11/14 17:23:00	2 events	0 ms	33 ms	33 ms	2/2

Fig 5: Picture of completed batches showing the processing time.

Streaming Statistics

This depicts the Spark streaming, the running batches, the time and date it started, and also the completed batches. After this process, the results are queried from the database.



Streaming Statistics

Running batches of **5 seconds** for **4 minutes** since **2017/11/14 17:08:16** (**46** completed batches, **123** records)

Timelines (Last 46 batches, 0 active, 46 completed)

Histograms

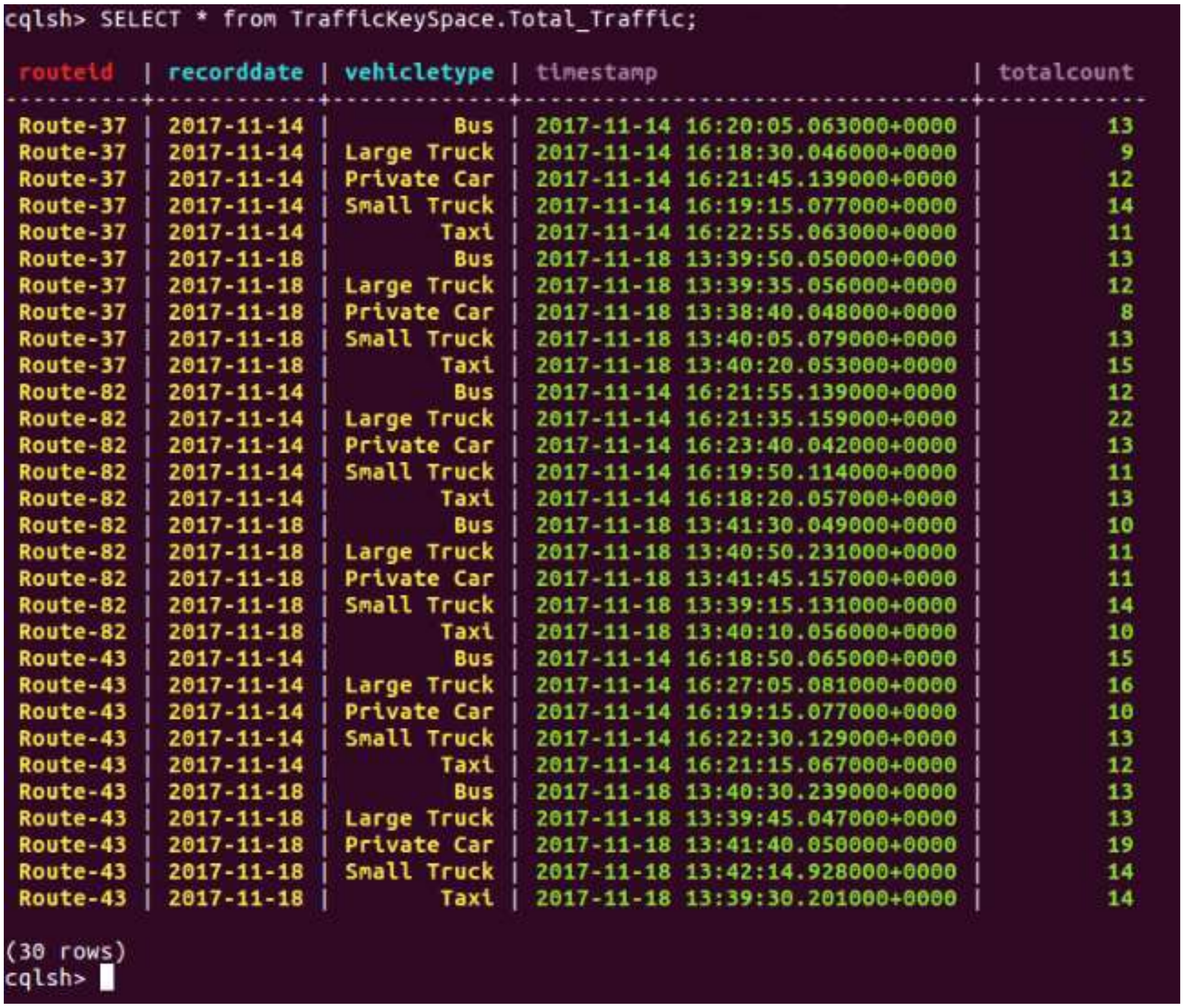
Input Rate
Avg: 0.53 events/sec

Scheduling Delay (?)
Avg: 6 ms

Fig 6: Picture of streaming statistics

The Results

This displays the data that has been processed by the consumer application showing the number of vehicles on different routes.



```
cqlsh> SELECT * from TrafficKeySpace.Total_Traffic;

 routeid  | recorddate | vehicletype | timestamp                       | totalcount
----------+------------+-------------+---------------------------------+------------
 Route-37 | 2017-11-14 |         Bus | 2017-11-14 16:20:05.063000+0000 |         13
 Route-37 | 2017-11-14 | Large Truck | 2017-11-14 16:18:30.046000+0000 |          9
 Route-37 | 2017-11-14 | Private Car | 2017-11-14 16:21:45.139000+0000 |         12
 Route-37 | 2017-11-14 | Small Truck | 2017-11-14 16:19:15.077000+0000 |         14
 Route-37 | 2017-11-14 |        Taxi | 2017-11-14 16:22:55.063000+0000 |         11
 Route-37 | 2017-11-18 |         Bus | 2017-11-18 13:39:50.050000+0000 |         13
 Route-37 | 2017-11-18 | Large Truck | 2017-11-18 13:39:35.056000+0000 |         12
 Route-37 | 2017-11-18 | Private Car | 2017-11-18 13:38:40.048000+0000 |          8
 Route-37 | 2017-11-18 | Small Truck | 2017-11-18 13:40:05.079000+0000 |         13
 Route-37 | 2017-11-18 |        Taxi | 2017-11-18 13:40:20.053000+0000 |         15
 Route-82 | 2017-11-14 |         Bus | 2017-11-14 16:21:55.139000+0000 |         12
 Route-82 | 2017-11-14 | Large Truck | 2017-11-14 16:21:35.159000+0000 |         22
 Route-82 | 2017-11-14 | Private Car | 2017-11-14 16:23:40.042000+0000 |         13
 Route-82 | 2017-11-14 | Small Truck | 2017-11-14 16:19:50.114000+0000 |         11
 Route-82 | 2017-11-14 |        Taxi | 2017-11-14 16:18:20.057000+0000 |         13
 Route-82 | 2017-11-18 |         Bus | 2017-11-18 13:41:30.049000+0000 |         10
 Route-82 | 2017-11-18 | Large Truck | 2017-11-18 13:40:50.231000+0000 |         11
 Route-82 | 2017-11-18 | Private Car | 2017-11-18 13:41:45.157000+0000 |         11
 Route-82 | 2017-11-18 | Small Truck | 2017-11-18 13:39:15.131000+0000 |         14
 Route-82 | 2017-11-18 |        Taxi | 2017-11-18 13:40:10.056000+0000 |         10
 Route-43 | 2017-11-14 |         Bus | 2017-11-14 16:18:50.065000+0000 |         15
 Route-43 | 2017-11-14 | Large Truck | 2017-11-14 16:27:05.081000+0000 |         16
 Route-43 | 2017-11-14 | Private Car | 2017-11-14 16:19:15.077000+0000 |         10
 Route-43 | 2017-11-14 | Small Truck | 2017-11-14 16:22:30.129000+0000 |         13
 Route-43 | 2017-11-14 |        Taxi | 2017-11-14 16:21:15.067000+0000 |         12
 Route-43 | 2017-11-18 |         Bus | 2017-11-18 13:40:30.239000+0000 |         13
 Route-43 | 2017-11-18 | Large Truck | 2017-11-18 13:39:45.047000+0000 |         13
 Route-43 | 2017-11-18 | Private Car | 2017-11-18 13:41:40.050000+0000 |         19
 Route-43 | 2017-11-18 | Small Truck | 2017-11-18 13:42:14.928000+0000 |         14
 Route-43 | 2017-11-18 |        Taxi | 2017-11-18 13:39:30.201000+0000 |         14

(30 rows)
cqlsh>
```

Fig 6: Results showing the number of vehicles on each route and the types of vehicles.

SUMMARY

The paper discusses how to monitor the movement of traffic by using the connected cars in the IoT environment. The objectives achieved from the implementation are that real-time data stream processing was performed using Apache Kafka with Spark and also, to process data in the IoT environment using the information captured from the connected cars. The information from the cars was captured and used to monitor traffic. It monitors the movement of the vehicles, the longitude, and latitude which is used to calculate the distance and processes the data as soon as it is received using Apache Spark. The proposed system achieved the aim of performing real-time streaming integration of Apache Kafka with Spark which captures the data from the IoT devices, and also does the processing instantaneously.

CONCLUSION

This approach was taken by understanding how the smart devices work in an IoT environment, understanding how to generate data and capture the data from these devices and also to be able to process the data captured instantaneously. The approach was achieved by using Kafka which is a distributed streaming platform and a messaging system to publish messages for streaming.

Firstly, Kafka topics were created which was used to store the messages, then the brokers or the Kafka server was used to replicating the messages in order to avoid message loss and ensure that messages were delivered successfully for processing by Spark streaming. Spark streaming, the consumer fetches the messages from the topics and processes immediately. Also, it tries to understand the nature of the messages in order to be able to process the data.

Secondly, the Spark processor processes the data and pushes the data to the database. In the Cassandra database, key spaces were created in which the messages that were processed were stored and keyspace is used to define data replication on nodes.

FUTURE WORK

It is recommended that future work on traffic monitoring should use the Kaa IoT platform to run the application. Kaa IoT is a highly flexible, multi-purpose, 100% open-source middleware platform for implementing complete end-to-end IoT solutions, connected applications, and smart products. It generates real-time data from smart devices to develop IoT applications. The data displayed on the monitoring dashboard should be used to avoid traffic congestion and prevent road traffic in the smart city environment. It should also use the decision tree or random forest to predict probable future events and should compare the two methods to determine which will work faster and better.

ACKNOWLEDGMENT
The author is grateful to the African University of Science and Technology (AUST), Abuja and African Capacity Building Foundation (ACBF) for giving me access to its facilities and resources respectively.

REFERENCES

Amini, S., Gerostathopoulos, I., & Prehofer, C. (2017). Big data analytics architecture for real-time traffic control. 5th IEEE International Conference on Models and Technologies for Intelligent Transportation System (MT – ITS). http://doi.org/10.1109/mtits.2017.8005605.

Campus, K., Campus, K., Mani, V., Campus, K., Sankaranarayanan, S., & Campus, K. (2017). IoT Based Traffic Signalling System, *12*(19), 8264–8269.

Conese, A. (2015). Inferring Latent User Attributes in Streams of Multimodal Social Data using Apache Spark.

Gehlot, R. (2016). Storage and Retrieval of Data for Smart City using Hadoop, *3*(5), 85–89.

Hahanov, V. (2015). Smart traffic light in terms of the Cognitive road traffic management system (CTMS) based on the Internet of Things. Volodymyr Miz PhD student at Kharkov National University of Radio.

Kleppmann, M., & Kreps, J. (2016). Kafka , Samza and the Unix Philosophy of Distributed Data, 1–11.

Lakshminarasimhan, M. (2016). IoT Based Traffic Management System Advanced Traffic Management System Using Internet of Things, (March), 0-9.

Nagdive, A. S. (2018). A Review of Hadoop Ecosystem for BigData, *180*(14), 35–40.

Nuaimi, E. Al, Neyadi, H. Al, Mohamed, N., & Al-jaroodi, J. (2015). Applications of big data to smart cities. *Journal of Internet Services and Applications*. https://doi.org/10.1186/s13174-015-0041-5.

Prathilothamai, M., Lakshmi, A. M. S., & Viswanthan, D. (2016). Cost Effective Road Traffic Prediction Model using Apache Spark, 9(May). https://doi.org/10.17485/ijst/2016/v9i17/87334

Radek Kuchta, R. N., Kuchta, R., & Kadlec, J. (2014). Smart City Concept, Applications and Services. *Journal of Telecommunications System & Management*, *03*(02), 1–8. https://doi.org/10.4172/2167-0919.1000117.

Santana, E. F. Z., Chaves, A. P., Gerosa, M. A., Kon, F., & Milojicic, D. (2016). Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. *ArXiv Preprint ArXiv:1609.08089*, (October). Retrieved from http://arxiv.org/abs/1609.08089.