Real-time Processing of IoT Events with Historic data using Apache Kafka and Apache Spark with Dashing framework

Godson Michael D'silva
Information Technology Department, St. John College of Engineering and Technology, Palghar, India
dsilvagodson@gmail.com
,

Azharuddin Khan
Information Technology DepartmentSt. John College of Engineering and Technology, Palghar, India

Mr. Gaurav Joshi,Information Technology DepartmentSt. John College of Engineering and Technology, Palghar, India

Mr. SiddheshBari
Information Technology Department, St. John College of Engineering and Technology, Palghar, India

***Abstract*— IoT (Internet of Things) is a concept that broadens the idea of connecting multiple devices to each other over the Internet and enabling communication between these devices. Traditionally, the packets are sent over the network for communication only if both, the sender as well as the receiver, are online. This forces the sender and the receiver to be online 24×7; which is not achievable in each and every environment the devices communicates in. Considering the humongous data generated in the communication, it is necessary to store and process this data so that data insights can be identified to improve the organizational benefits. This generated data can be in two forms, real-time as well as existing or historical data. When this data is obtained in real-time and it is processed, even traditional big data technologies do not perform up to the mark. Hence to process this real-time data, streaming of this data is required; which is not a feature of traditional big data technologies. To achieve these objectives, the proposed architecture uses open source technologies such as Apache Kafka, for online and offline consumption of messages, and Apache Spark, to stream, process and provide a structure to the real-time and existing data. A framework known as Dashing is used to present the processed data in a more attractive and readable manner.**

Keywords—IoT, Real-time Data Processing, Apache Kafka, Apache Spark, Dashing, Azure Hdinsight.

I. INTRODUCTION

Internet of Things, termed as IoT, is basically actions or communication between things over the internet. These 'things' in IoT can be referred to as anything, from an electronic smart device to a human being. The concept of IoT is to access the devices remotely through an existing network infrastructure. It results in an improved integration of the real physical world to computer systems. The output provides improved efficiency and accuracy in the actions performed, with an added advantage of reduced human intervention. The modern day application of IoT is as simple as a Smart Home wherein all the devices are connected to one another as well as the owner over the internet. This application of IoT has scaled up to developing smart cities in the future. The Technological experts or scholars have briefly termed these 'things' in IoT as 'a tangled collection of software- hardware along with servicing of data'. The devices termed as 'things', receive or collect data from various sources using existing modern technologies, store them and help them to flow over the network. Out of the many examples of real-world applications of IoT, some include ventilation, washers /dryers, robotic vacuums etc. These all applications use Wi-Fi for remote monitoring of the devices involved. Objects in IoT not only mean objects with sensors but also objects which involve actuators such as locks, fans etc. Now, considering the humongous inclusion of various devices as 'things' in IoT; the consideration of data of these many things connected with each other is an obvious issue.

When trying to process this connected data, traditional big data technologies such as Hadoop under-perform. But Apache Kafka, an open source event processing technology developed by Apache Software Foundation is used since it has a storage layer which is massively scalable and is capable of handling large number of real-time data feeds. Apache Spark, again one of the big data technology is used to process the data with the help of cluster computing. When compared to traditional big data technologies such as Apache Hadoop; Spark performs actions 10 to 100 times faster. Also, Spark overcomes the problem of managing (processing) big and complex connected data to a larger extent. It is highly fault-tolerant. Since, we are dealing with real-time data feeds; Spark Streaming, a fundamental in Apache Spark; is used to stream and process the data.

A new open-source technology named Dashing can be used to build user-friendly and attractive dashboards. It uses an API to push data to dashboards. This data can be anything, ranging from attributes of a node to a set of commands required to operate an IoT device. The integration of these three open source technologies, namely Apache Kafka, Apache Spark and dashing; over the public clouds such as Microsoft Azure and AWS (Amazon Web Services) makes the architecture faster and scalable than the existing traditional applications.

II. LITERATURE SURVEY

In the paper proposed by MrLokesh Babu Rao and Mr C. Elayaraja, titled 'Image Analytics on Big Data in Motion – Implementation of Image Analytics CCL in Apache Kafka and Storm'; Apache Kafka is an open-source big data messaging system and Apache Storm, another open-source data processing system are used together to solve various issues regarding continuous and centralized image as well as video processing. Further, to demonstrate the applicability of image analytics along with big data stream processing; CCL (Connected Component Labelling) algorithm is used. It is an application in the form of algorithm for the graph theory of connected components which are uniquely labelled.

In the paper proposed by Mr Godson D'Silva, MrSanket Thakare and MrVinayakBharadi titled 'Real-time processing of IoT Events using Software as a Service (SaaS) with Graph Database', the issue or complexity of processing humongous 'connected' data from various IoT devices is addressed by using three open-source technologies namely; Ejabberd Server, Apache Spark and Neo4j graph database. To get a better and faster result all these three technologies have been deployed over Microsoft Azure Public Cloud. The issue of real-time analysis of humongous data connected with each other is being resolved up to a certain limit, but architecture starts to under-perform when data is being received in the form of real-time streaming data.

By sorting and selecting certain unique features from the above mentioned papers, we have designed a new architecture which includes the combination of three open source technologies such as Apache Kafka, Apache Spark and dashing. Since Resource utilization as well as system throughput is crucial, maximizing these is must. Visualization techniques are applied to the hardware resources which are being shared based on the Apache Spark jobs. In order to deal with real time and existing data, streaming operations are carried out using spark streaming and the result are generated & further displayed on the Dashboard. Thus, this proposed architecture works much faster than the existing modern day technologies used to store and process real-time data.

III. Related Theory

In this proposed architecture of communication of various IoT devices, public clouds AWS and Microsoft Azure Public Cloud are used to deploy the three open source technologies, namely, Apache Kafka, Apache Spark and Dashing. These open source technologies are discussed in detail below:

A. Apache Kafka

Apache Kafka provides log services which are based on distribution, partitioning and replication, also suitable for online as well as offline consumption of messages. Kafka includes broker, topics, producers and consumers. Broker is a Kafka cluster involves one or more servers. Topic in Kafka is a category which stores messages and even publishes those messages. Kafka also maintains partition log for messages, each message in this partition is identified by its unique id which is termed as offset. The topic can also contain multiple partition logs which contains messages that are replicated over a number of servers. These servers all together form the broker i.e. Kafka cluster.

Every partition has one server which acts as a leader and the other servers acts as followers to this leader. The leader is responsible for accepting and performing all the read/write operations within the server. The followers are passive replicas of the leader. As the leader server fails, one of the followers out of other servers is by default chosen to be the new leader. Processes are producers which publish data to a topic of their choice. It is possible for the producer to produce a particular message which comes under a chosen partition inside a topic. Producer performs this in any semantic manner which is most suitable according to the present condition. Now consumers come into the picture i.e. they consume the messages published under topics. It is always labelled with a consumer group name. Messages are delivered to the consumer instance within the subscribing consumer groups.

B. Apache Spark

Apache Spark is a big data technology used to process data on a large scale with the help of a concept known as cluster computing. Apache Spark can be termed as an updated version of Apache Hadoop. The advantages of Spark over Hadoop are that Spark performs the same actions in Hadoop 10 to 100 times faster. It provides the benefit of easy programming which makes it much more preferable for amateur programmers over Apache Hadoop. The issue of big and complex connected data that are to be stored and processed can be easily resolved in Spark.

When we consider real-time data being processed in Apache Spark, it has an extension feature called as Spark Streaming; which helps the users (programmers) to modify data in Apache Spark in real-time. The biggest advantage of using Spark Streaming is that it has a single execution engine i.e. it can handle both batch and streaming workloads and thus it is overcomes the limitations of traditional streaming systems. Spark Streaming helps Spark in increasing its core scheduling capability to perform streaming analytics on real-time data. Resilient Distributed Datasets (RDD), a fundamental data structure of Spark, is an immutable distributed collection of objects. RDDs contain any type of Python, Java or Scala objects with user-defined classes. RDD, an important feature in Spark streaming helps in computations when data processing is carried over multiple jobs. Also in terms of iterative distributed computing, it is fair to reuse or share data over the multiple jobs to be performed in clusters. But for this 'fair' thing to happen we need to store data in some intermediate stable storage and that makes the overall computations slower; lowering the performance of traditional big data technologies such as Apache Hadoop. RDD helps in overcoming this issue by enabling fault-tolerant distributed in-memory computations. Spark Streaming uses these RDDs to perform transformations over the mini-batches of data that are being ingested by Spark Streaming into the cluster.

C. Dashing

Dashing framework allows you to create a very attractive and beautiful dashboard as per your need. After developing a dashboard we require API which will push the entered data into your widget or widgets. Dashing allows you to rearrange the widgets and it is made possible with drag and drop function /interface. Widgets are small programs in html, css and coffee; where HTML is used for layouts, SCSS are used for Styles and Coffee is a coffee script file that handles the data and functionality. Batman bindings is used by widgets for their updation. Dashing comes under the MIT license.

IV. Conceptual Framework

The proposed system architecture consists of main components such as IoT devices, apache kafka, apache spark and dashing framework as illustrated in Figure1.

The IoT devices are the front end of the architecture. They communicate with each other through the messages sent over the cloud. The messages they send to each other for communication are a set of real-time data. This real-time data is been passed on to Apache Kafka.The apache kafka has a storage layer which is massively scalable and capable of handling large number of real-time data feeds. It receives the messages sent by IoT devices, stores them by writing each message to its broker; internal component of Apache Kafka, which is a cluster having one or more servers. Further, it forwards these messages to Apache Spark. Blob Storage. This data is an existing data stored over the cloud which is being sent to the processing block of Apache Spark Streaming, internal component of Apache Spark.

It includes its internal components working separately in this architecture. The first one being Apache Spark Streaming. It helps in streaming real-time data and has a distinct feature of having a single execution engine, both streaming as well as batch workloads. The processing block processes this data received from Apache Kafka as well as the existing data, together and converts this whole data to operations which are understandable to the GraphX module, another internal component of Apache Spark. Now, these operations are being performed in in GraphX and the output is generated in graph format as well as relation format. The output is forwarded to dashing framework.This framework is being used in this architecture to present the data to the user in a more attractive and readable manner. This framework helps in presenting the data in the output on attractive dashboards.

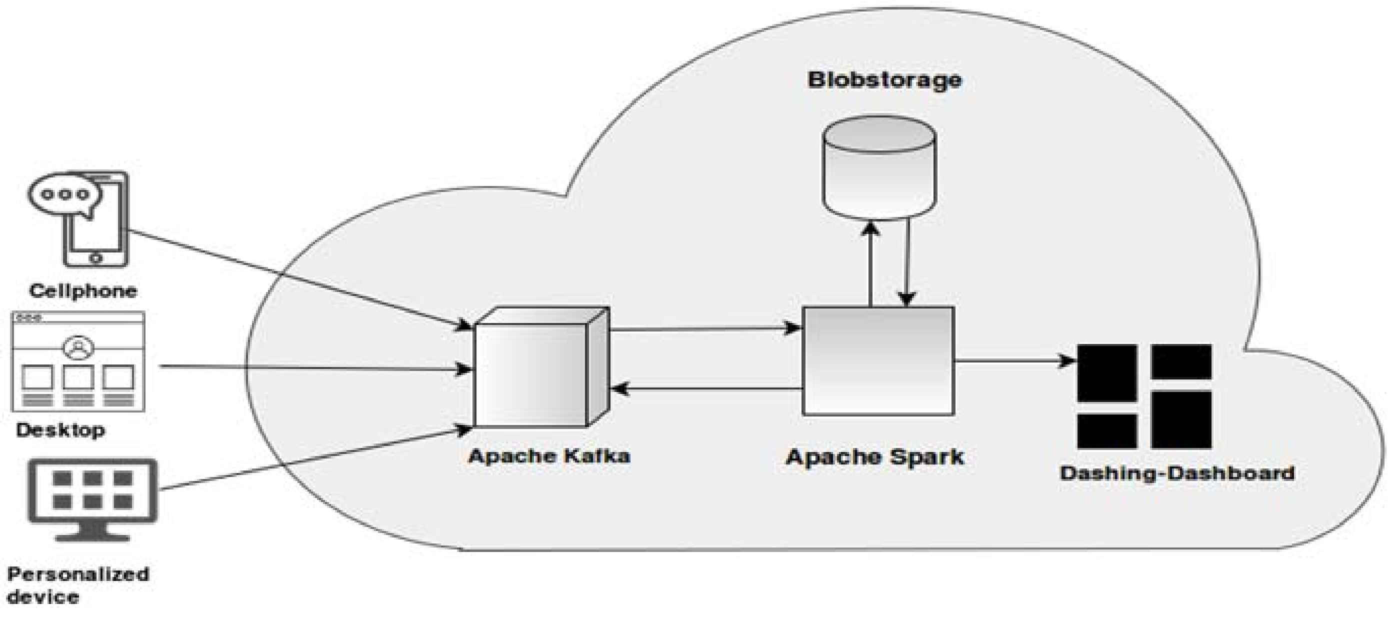


Fig. 1.Generalized Real-time Data Processing Architecture

V. IMPLEMENTATION

This proposed architecture is divided into three main categories first is dealing with real-time data, second is dealing with historic data and last one is displaying the data insights on the dashboards.

A. Processing real-time data operation

In this operation, the events are captured apache kafka in real-time which are emitted to apache spark were data processing is done and finally result is displayed on the dashboards to get useful insights as illustrated in below steps :

1. The data is received or sent to the cloud from IoT Devices. These IoT devices include Cell phones, Desktops and other personalized devices. The data sent over the cloud can be in the form of events, log files etc.
2. Apache Kafka, deployed over the cloud, receives this data in text format and stores it in records. Its storage layer basically performs like a massively scalable message queuing system which receives, stores and sends data. Apache Kafka uses the concept of clustering to store and process data.
3. The data is forwarded by Kafka to Apache Spark which processes the data to obtain a desired output. The Internal components of Apache Spark such as Spark Streaming and GraphX are in use in the architecture proposed by this paper.
4. The data is forwarded by Kafka to Spark Streaming, which accepts data in real-time and processes the data parallel on a cluster.
5. The data is treated as batch of RDDs by Streaming and is processed using RDD operations. The results of these RDD operations are returned in batches. These batches contains events, logs etc.
6. These events or logs are converted into operations that are easily understandable by GraphX. GraphX performs these operations and the generated output is stored in graph format. GraphX also has an additional feature of presenting the data to the user in relational format.
7. The final output generated by Apache Spark is forwarded for dashing. Since dashing helps in building dashboards, it helps us rearrange the widgets and present the data to the user in a more attractive manner.
8. Finally, the output generated by dashing is presented as it is desired by the user.

B. Processing existing data set operation

In this operation, the historic data stored in blob storage are processed by apache spark and the result is displayed on the dashboards to get useful insights as illustrated in below steps:

1. Now let us consider the scenario of existing dataset that forwards its data to the Apache Spark. Since it is not a real-time data; we will be directly passing it on to the processing block of Spark Streaming.
2. The data received by the processing block are treated as batch of RDDs and operations are performed on these datasets.

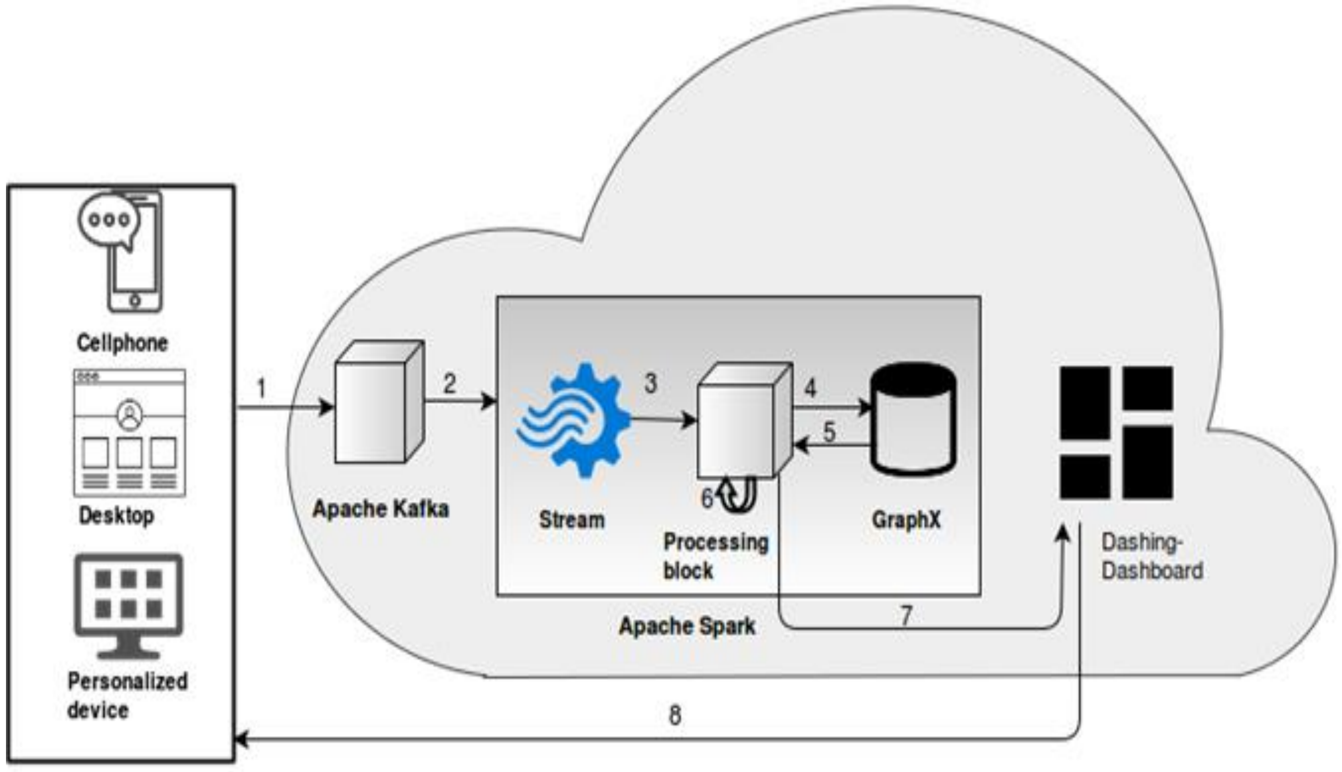


Fig.2: Processing real-time data operation

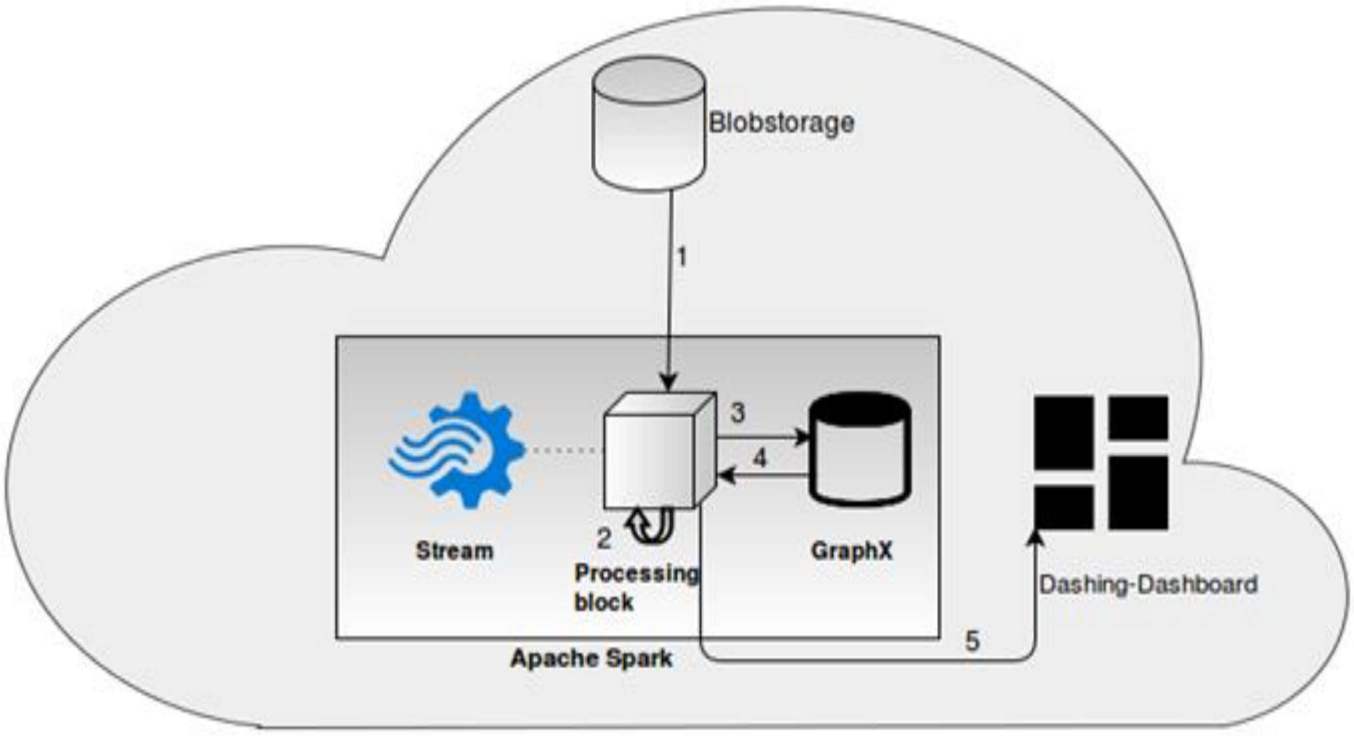


Fig.3: Processing existing data set operation

3. After processing, this data is then passed on to GraphX module in Apache Spark as operations.
4. GraphX module performs these operations and generates
5. the output in graph format. It also has the feature to generate the output in relational format.
6. The output of GraphX module is forwarded to the dashing framework so that it can present the data in the output in a more understandable manner in the dashboard.

1. The compute lambda triggers the dynamodb for the count of the commuters present in each coach of the particular train.
2. The dynamodb returns the relevant count to the compute lambda.
3. The compute lambda performs computation by comparing present count of the commuters with actual capacity of the coach.
4. The computed result of the crowd analysis is further sent to the SNS.
5. The SNS notifies this result to the upcoming station's indicators to be displayed in graphical percentile representation.

VI. RESULTS

In this paper a highly robust, scalable, pluggable and faster architecture is proposed which tries to solve the issues of traditional Data processing approach by integration of apache kafka, apache spark, dashing and deploying it on windows azure public cloud.

A. Topics created in kafka and sending events

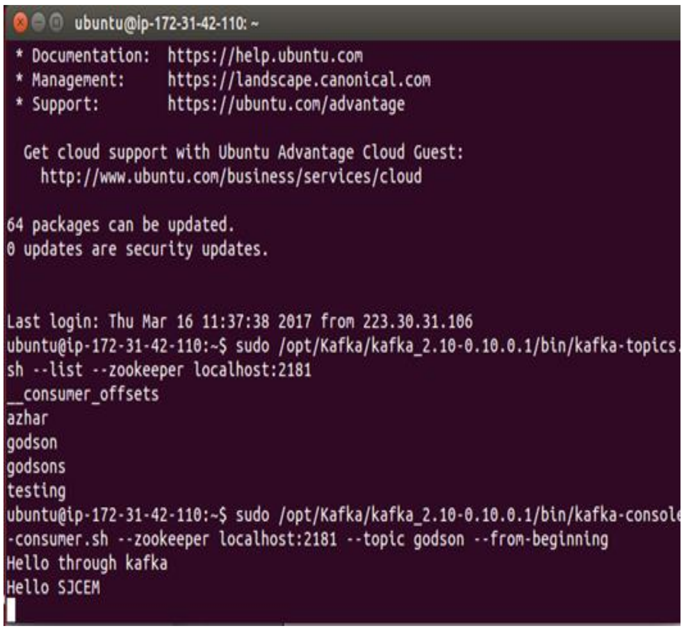
Four topics in Apache kafka are created i.e. "azhar", "godson", "godsons" and "testing". The event is sent under the topic godson from external source. Apache kafka receives the event and then it is being displayed with the help of zookeeper.

Similarly, data set is sent externally to kafka which can be seen in the image below. A total 1000 events were sent to Apache kafka externally under the same topic "godson". These events were then listed down in Apache kafka interface through its listen function as illustrated in Figure 4.

A total 1000 events were sent to Apache kafka externally under the same topic "godson". These events were then listed down in Apache kafka interface through its listen function as illustrated in Figure 5.

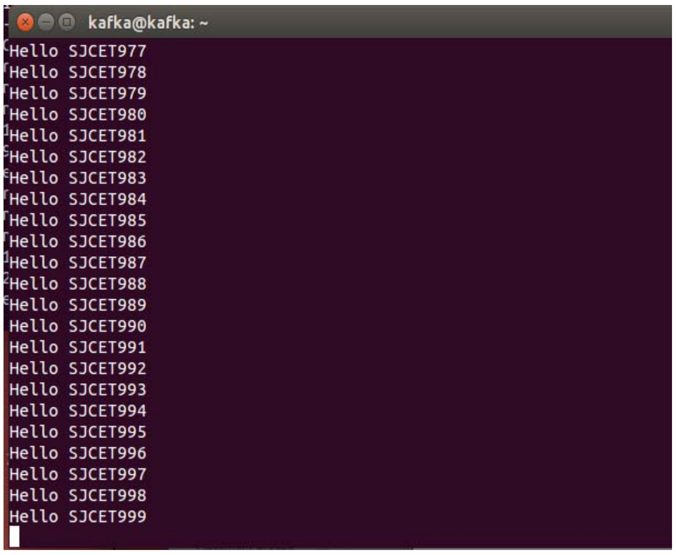
B. Events captured in Apache spark

In this proposed architecture a connection is made between apache kafka and apache spark. The Apache spark streaming module is configured to listen events on a topic created in Figure 4. After connection is established apache spark will able to listen to the events which were forwarded from apache Kafka. So whenever a new data is generated from the IoT devices its sends to kafka topic as illustrated in Figure 6 were 1000 random events are send to the Kafka topic.All this events are received by apache spark. The apache spark then reads the data and starts processing the data as per the application logic and lastly updates it on the dashing dashboards as illustrated in Figure 6.



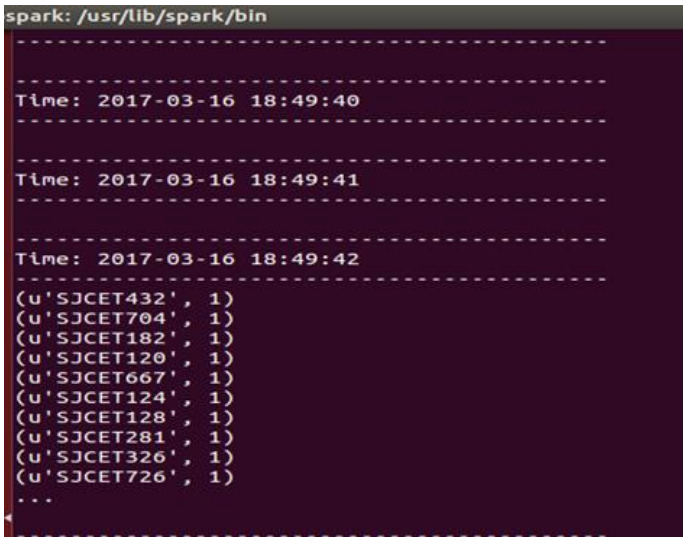
```
ubuntu@ip-172-31-42-110: ~
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

64 packages can be updated.
0 updates are security updates.


Last login: Thu Mar 16 11:37:38 2017 from 223.30.31.106
ubuntu@ip-172-31-42-110:~$ sudo /opt/Kafka/kafka_2.10-0.10.0.1/bin/kafka-topics.
sh --list --zookeeper localhost:2181
__consumer_offsets
azhar
godson
godsons
testing
ubuntu@ip-172-31-42-110:~$ sudo /opt/Kafka/kafka_2.10-0.10.0.1/bin/kafka-console
-consumer.sh --zookeeper localhost:2181 --topic godson --from-beginning
Hello through kafka
Hello SJCEM
```

Fig.4: Topics created in apache Kafka



```
kafka@kafka: ~
Hello SJCET977
Hello SJCET978
Hello SJCET979
Hello SJCET980
Hello SJCET981
Hello SJCET982
Hello SJCET983
Hello SJCET984
Hello SJCET985
Hello SJCET986
Hello SJCET987
Hello SJCET988
Hello SJCET989
Hello SJCET990
Hello SJCET991
Hello SJCET992
Hello SJCET993
Hello SJCET994
Hello SJCET995
Hello SJCET996
Hello SJCET997
Hello SJCET998
Hello SJCET999
```

Fig.5: Published Events



```
spark: /usr/lib/spark/bin
-------------------------------------------

-------------------------------------------
Time: 2017-03-16 18:49:40
-------------------------------------------

-------------------------------------------
Time: 2017-03-16 18:49:41
-------------------------------------------

-------------------------------------------
Time: 2017-03-16 18:49:42
-------------------------------------------
(u'SJCET432', 1)
(u'SJCET704', 1)
(u'SJCET182', 1)
(u'SJCET120', 1)
(u'SJCET667', 1)
(u'SJCET124', 1)
(u'SJCET128', 1)
(u'SJCET281', 1)
(u'SJCET326', 1)
(u'SJCET726', 1)
...
```

Fig.6: Events captured by apache spark

C. Data Visualized using dashing

This proposed architecture generated many outputs, from which one of the outputs of using dashing framework is as illustrated in Figure 7.

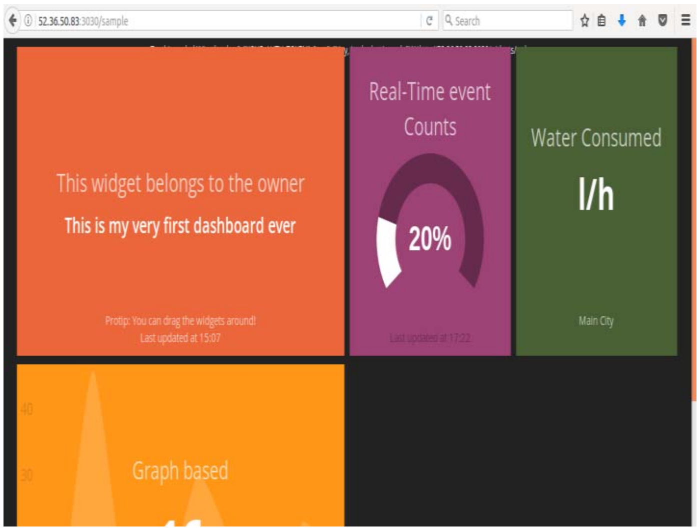


Fig. 7: Events captured by apache spark

Paper presents four widgets in the following dashboard, which includes text type, meter type, Sparkline, and graph type. In the Figure 7, there are three widgets which are of text type, meter type, and Sparkline. In the text type, the data streamed by the apache spark and processed by analytic part is directly feed into this widget, whereas meter type shows the percentage count of the real-time events coming from the source; Sparkline widget displays the Sparkline graphs based on the real-time events. The fourth and the final widget generated using dashing is graph based which keeps on changing based on the data being streamed

VII. CONCLUSION

Irrespective of the operational environment the IoT devices communicates generates humongous amount of data that in real-time; and which needs to be streamed and processed. Also, the output needs to be converted in a well-structured and readable format. This paper aims to resolve the issues of streaming and processing real-time data by using Apache Kafka and Apache Spark for storing, streaming and processing the real-time messages generated during the communication of IoT devices. To visualize the data an get useful insights from it, an open source framework know as dashing is used. In order to make the architecture more efficiency, this architecture has been deployed on windows Azure public cloud.

REFERENCES

[1]Mr. Godson Michael D'silva, Mr. SanketThakare, Dr. Vinayak Ashok Bharadi, "Real-time Processing of IoT Events using a Software as a Service (SaaS) Architecture with Graph Database" in 2015 IEEE International Conference on Computing Communication Control and Automation, pp. 65–72, Feb. 2015.

[2] MateiZaharia, Tathagata Das, Haoyuan Li, Scott Shenker, Ion Stoica, "Discretized Streams: An Efficient and Fault-Tolerant Model for Stream Processing on Large Clusters" in University of California, Berkeley.

[3] Mohammed Ghesmoune, Mustapha Lebbah, HaneneAzzag, "Micro-Batching Growing Neural Gas for Clustering Data Streams Using Spark Streaming" in University of Paris 13, Sorbonne Paris City LIPN-UMR 7030 - CNRS 99, av. J-B Clement – F-93430 Villetaneuse, France.

[4] MateiZaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, Ion Stoica, "Discretized Streams: Fault-Tolerant Streaming Computation at Scale" in University of California, Berkeley.

[5] Arian Bär, Alessandro Finamore, Pedro Casas, Lukasz Golab, Marco Mellia, "Large-scale network traffic monitoring with DBStream, a system for rolling big data analysis" in FTW Vienna, Austria, Politecnico di Torino, Italy, University of Waterloo, Canada.

[6] Cao Ngoc Nguyen, Jik-Soo Kim, Soonwook Hwang, "KOHA: Building a Kafka-Based Distributed Queue System on the Fly in a Hadoop Cluster" in Nat. Inst. of Supercomput. & Networking at KISTI, Daejeon, South Korea.

[7] LokeshBabu Rao, Elayaraja, "Image Analytics on Big Data In Motion – Implementation of Image Analytics CCL in Apache Kafka and Storm" in Dhaanish Ahmed College of Engineering, Chennai.

[8] Joseph E. Gonzalez, Reynold S. Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin, Ion Stoica, "GraphX: Graph Processing in a Distributed Dataflow Framework" in UC Berkeley AMPLab.

[9] Apache Spark: http://spark.apache.org/ accessed on 26-02-2017 10:00 AM.

[10]Apache Kafka: https://en.wikipedia.org/wiki/Apache_Kafka/ accessed on 26-02-2017 05:00PM.

[11]Apache Kafka: http://blog.cloudera.com/blog/2014/09/apache-kafka-for-beginners/ accessed on 27-02-2017 11:00 AM.

[12]Apache Spark streaming: https://databricks.com/blog/2015/07/30/diving-into-apache-spark-streamings-execution-model.html/ accessed on 28-02-2017 11:00 AM.

[13]GraphX:http://ampcamp.berkeley.edu/big-data-mini-course/graph-analytics-with-graphx.html accessed on 01-03-2017 10:00 AM.

[14] Dashing: http://dashing.io// accessed on 01-03-2017 06.00 P.M.

[15] V. A. Bharadi and G. M. DSilva, "Online Signature Recognition Using Software as a Service (SaaS) Model on Public Cloud," in 2015 IEEE International Conference on Computing Communication Control and Automation, pp. 65–72, Feb. 2015.

[16] G. M. DSilva and V. A. Bharadi, "Modified Online Signature Recognition Using Software as a Service (SaaS) Model on Public Cloud," IEEE International Conference on Information Processing, December 16-19, 2015.

[17] Fox, Geoffrey C., SupunKamburugamuve, and Ryan D. Hartman. "Architecture and measured characteristics of a cloud based internet of things." Collaboration Technologies and Systems (CTS), 2012 International Conference on.IEEE, 2012.