Web Log Analysis

G.Santhoshi[1], H.Meenal[2], Dr.D.Shravani[3]

[1]M.Tech. Software Engineering Scholar Dept. CSE Stanley College of Engineering and Technology for Women SCETW O.U. Hyderabad TS India

[2]Assistant Professor Dept. CSE Stanley College of Engineering and Technology for Women SCETW O.U. Hyderabad TS India

[3]Associate Professor Dept. CSE Stanley College of Engineering and Technology for Women SCETW O.U. Hyderabad TS India

Abstract: **In the paper, Web Logs Streaming is used to analyze streaming data and batch data. It can read data from Web Server and analyze the data based on the scenario. This basically implements the Streaming Data Analysis for Data Error extraction, Analyze the type of errors from log files and store in one host. The solution providing for streaming real-time error logs / IP addresses of the systems who are accessing the website. It provides a file which contains the keywords of error types for error identification or IP addresses of the people who are accessing the website using spark processing logic. After processing the data result file be placed in AWS Oracle table. Processing logic is written in Spark Eco systems and with Scala language. Spark SQL and Spark Streaming has been used in the project to get desired output. Kafka has been used for sending and receiving the data from the webserver. Kafka internally using Zookeeper for producing the data from the input file. With the help of Zookeeper, Kafka producer will be producing the data and Kafka Consumer will be receiving on the basis of given Topic and will be sending to Spark Streaming. Spark Streaming will be creating DStream and processing the seamless.**

Keywords: **Web Logs, Hadoop, Scala, Spark-streaming, Kafka, Zookeeper.**

I. INTRODUCTION

In this paper, we look after the weblog files that are used to interaction of the user with the web pages that are logged in a single record as a text file is known as web log file. Some various technologies are used such as Hadoop, Scala, Spark, Kafka, and Zookeeper. Hadoop was one of the first popular big data technology. It is an open source, Java-based programming framework. It is part of the Apache project developed by the Apache Software Foundation. It is a scalable, fault-tolerant system for processing large datasets into different cluster servers. Internal components of Hadoop are HDFS & YARN with map reduce. Apache Spark is used for large-scale data processing. Spark is a Hadoop ecosystem. It extends the Map Reduce (processing) capabilities of Hadoop to overcome challenges faced by Hadoop Eco systems. It is 100% faster than Map Reduce in memory and 10% faster in disk It consolidated interface/framework to process large amount of data. It process efficiently Batch data, streaming (micro-batch), and iterative, interactive unified stack like YARN. Scala is a purely object-oriented language. The rest of the paper deals as follows: Section II, deals with the technology of Spark is presented. In section III, Introduction to Kafka. In section IV, Implementation and results is presented. Finally, the section V, concludes paper.

II. SPARK

Spark is a Hadoop ecosystem. It extends the Map Reduce (processing) capabilities of Hadoop. Extremely fast (in-memory).Consolidated interface/framework to process wide range of workloads: Batch, streaming (micro-batch), iterative, interactive unified stack like YARN.

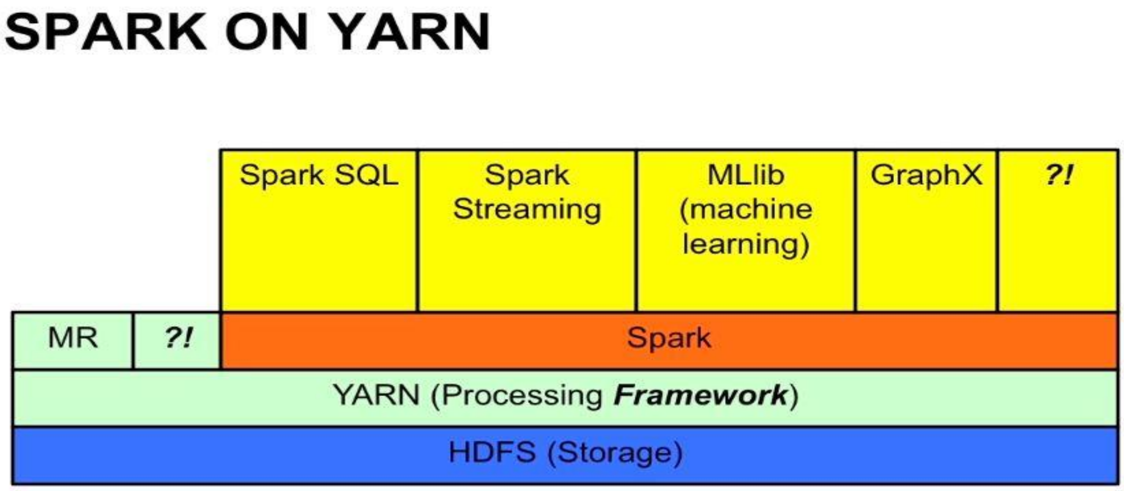


Fig 1: Spark on Yarn

Spark on yarn describes the different types of layers the first layer describes about the HDFS. It stores the different types of files. The second layer is the Yarn, it is the processing framework. The third layer is the spark, it is the Hadoop ecosystem it extends the map reduce capabilities. The fourth layer describes about the different technologies used for streaming the data.

Benefits of Spark:

The key features of Spark includes easy to use (programmer friendly), fast in memory access, general purpose, optimized fault tolerant, unified platform.

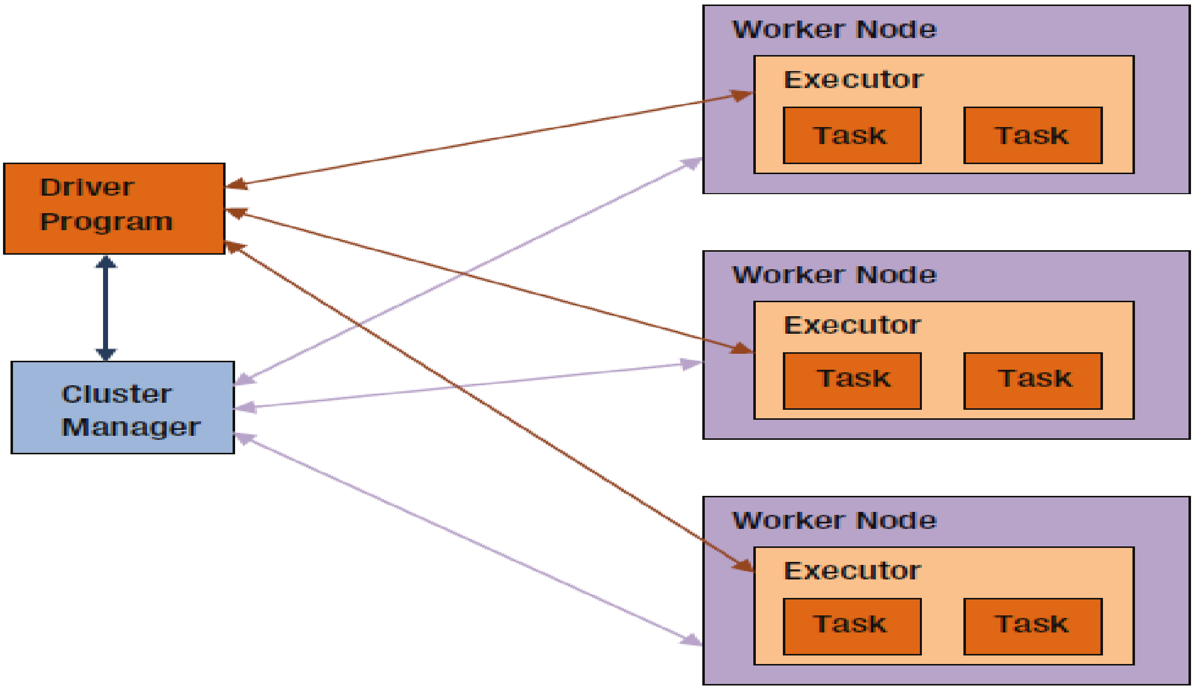


Fig 2: SPARK ARCHITECTURE

The key entities in the architecture are driver program, cluster manager(yarn),worker node(task and executor).

DRIVER PROGRAM:

It defines the transformations and actions.Where the driver program is placed to process, that node is called Driver node.

CLUSTER MANAGER (YARN):

It's a distributed OS. It's schedule the tasks and allocate the resources in the cluster.

It allocates RAM and CPUS to Executors based on Node manager request

WORK NODE:

In Hadoop terminology, it's also called node manager. It's manage the executors. If executors cross limits, node manager kill the executors.

TASK:

A task is the smallest unit of work that sends to an executor. It is executed by a thread in an executor on a worker node. Each task performs some computations to either return a result to a driver program or S3/hdfs. Spark creates a task per data partition. An executor runs one or more tasks concurrently. The amount of parallelism is determined by the number of partitions. More partitions mean more tasks processing data in parallel.

EXECUTORS:

Spark acts as an executors on each nodes in the cluster, that performs some task act as a processes that runs and store data. it support in-memory concept.

Spark Streaming

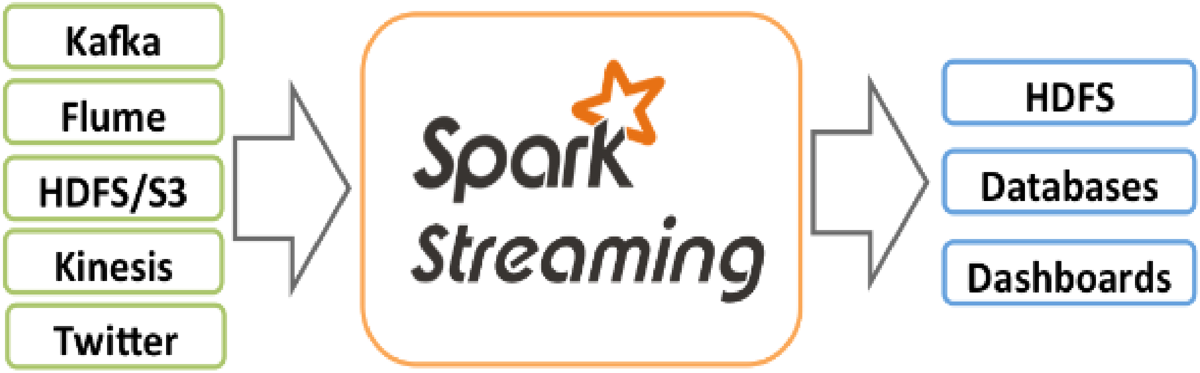


Fig 3: Spark Streaming

Spark Streaming receives input data streams and it generates spark engine by dividing the batches of input data at last it produces the batches of processed data.

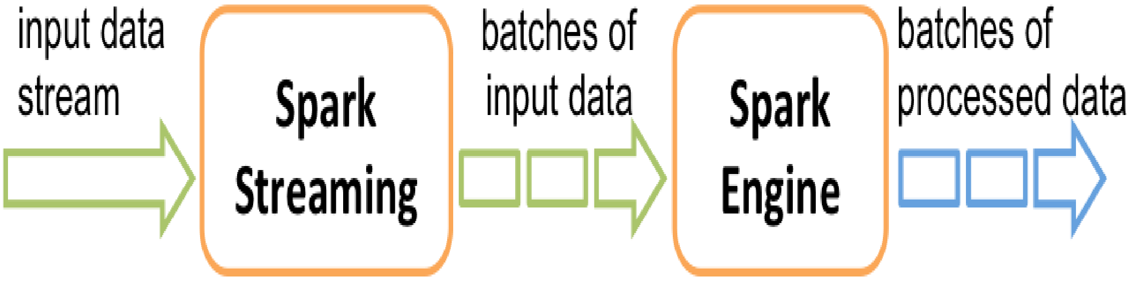


Fig 4: Spark Streaming in detail

SPARK STREAMING DATAFLOW:

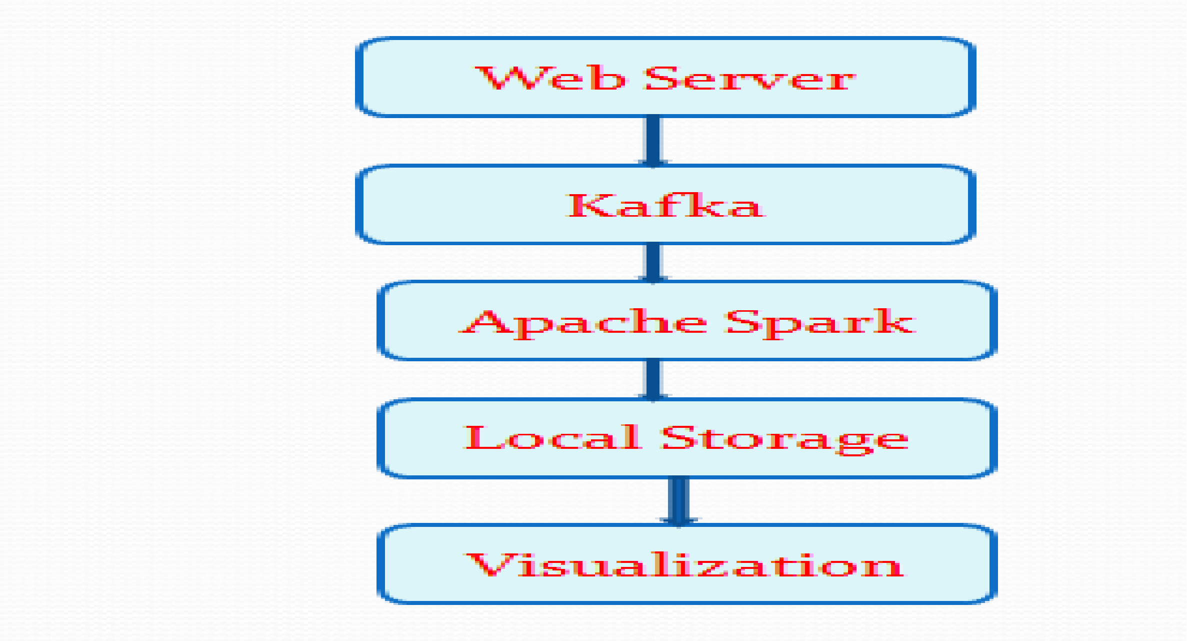


Fig 4: Spark Streaming Data Flow

III. KAFKA INTRODUCTION

Apache Kafka is a distributed streaming platform. It has three key capabilities:

1.Subscribe the streams of records, into a message queue.

2.Store the streams of records in a fault-tolerant.

3.Process the streams of records.

Kafka has four core APIs:

1.The Producer API

2.The Consumer API

3.The Streams API.

4.The Connector API.

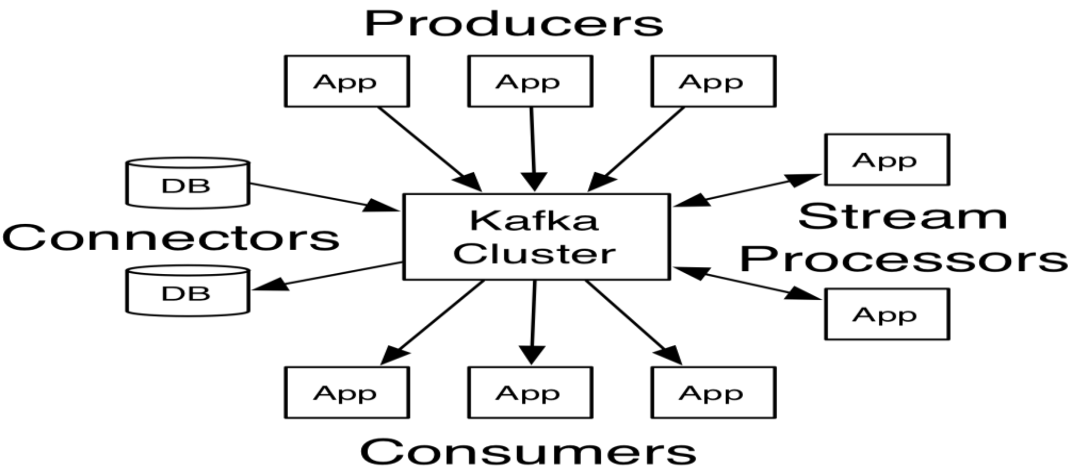
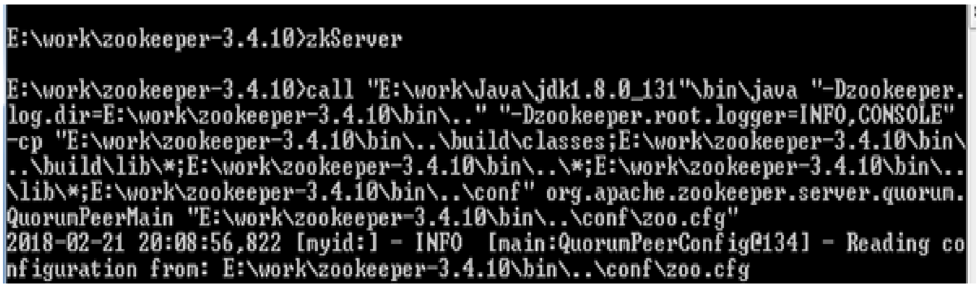


Fig 5: KAFKA Architecture

IV. IMPLEMENTATION AND RESULTS

Step1:Start Zookeeper server

```
bin/kafka-topics.sh --list --zookeeper localhost:2181
```

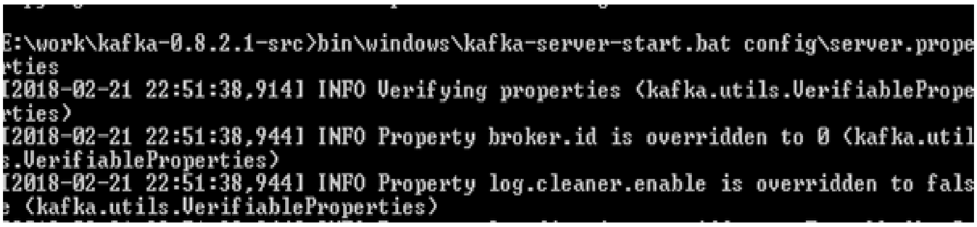


```
E:\work\zookeeper-3.4.10>zkServer

E:\work\zookeeper-3.4.10>call "E:\work\Java\jdk1.8.0_131"\bin\java "-Dzookeeper.
log.dir=E:\work\zookeeper-3.4.10\bin\.." "-Dzookeeper.root.logger=INFO,CONSOLE"
-cp "E:\work\zookeeper-3.4.10\bin\..\build\classes;E:\work\zookeeper-3.4.10\bin\
..\build\lib\*;E:\work\zookeeper-3.4.10\bin\..\*;E:\work\zookeeper-3.4.10\bin\..
\lib\*;E:\work\zookeeper-3.4.10\bin\..\conf" org.apache.zookeeper.server.quorum.
QuorumPeerMain "E:\work\zookeeper-3.4.10\bin\..\conf\zoo.cfg"
2018-02-21 20:08:56,822 [myid:] - INFO  [main:QuorumPeerConfig@134] - Reading co
nfiguration from: E:\work\zookeeper-3.4.10\bin\..\conf\zoo.cfg
```

Step2:Start kafka server

```
bin/kafka-server-start.sh config/server.properties
```

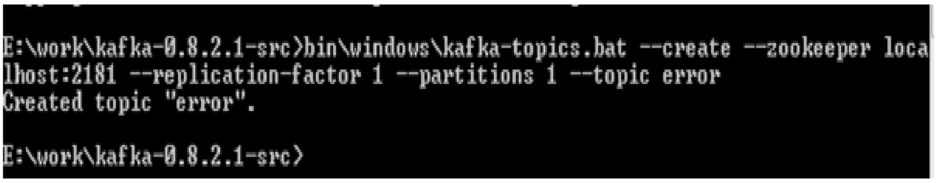


```
E:\work\kafka-0.8.2.1-src>bin\windows\kafka-server-start.bat config\server.prope
rties
[2018-02-21 22:51:38,914] INFO Verifying properties (kafka.utils.VerifiablePrope
rties)
[2018-02-21 22:51:38,944] INFO Property broker.id is overridden to 0 (kafka.util
s.VerifiableProperties)
[2018-02-21 22:51:38,944] INFO Property log.cleaner.enable is overridden to fals
e (kafka.utils.VerifiableProperties)
```

Step3:Create a topic

Take a new terminal

```
bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-
factor 1 --partitions 1 --topic error
```

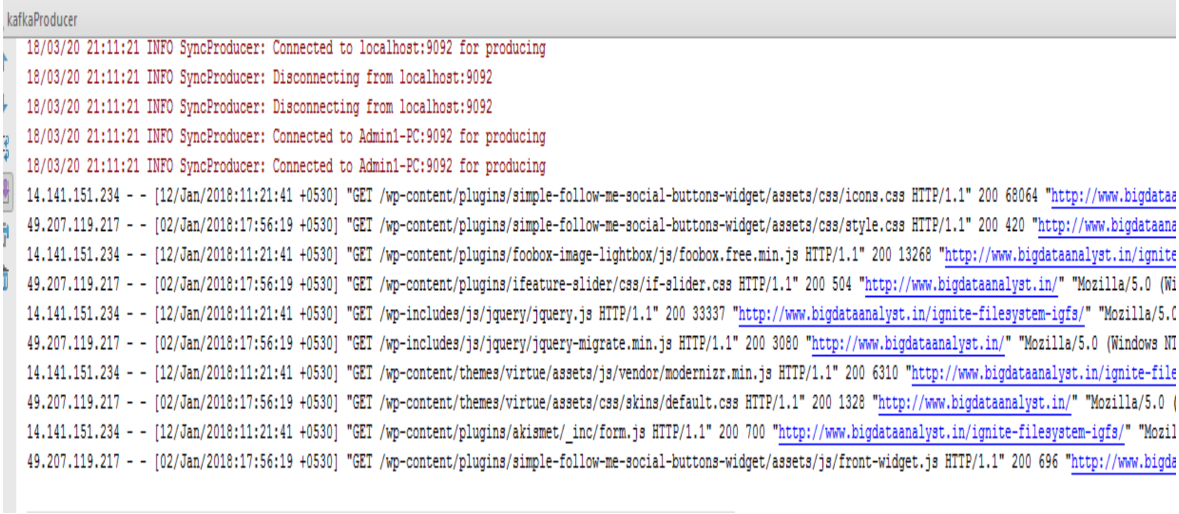


```
E:\work\kafka-0.8.2.1-src>bin\windows\kafka-topics.bat --create --zookeeper loca
lhost:2181 --replication-factor 1 --partitions 1 --topic error
Created topic "error".

E:\work\kafka-0.8.2.1-src>
```

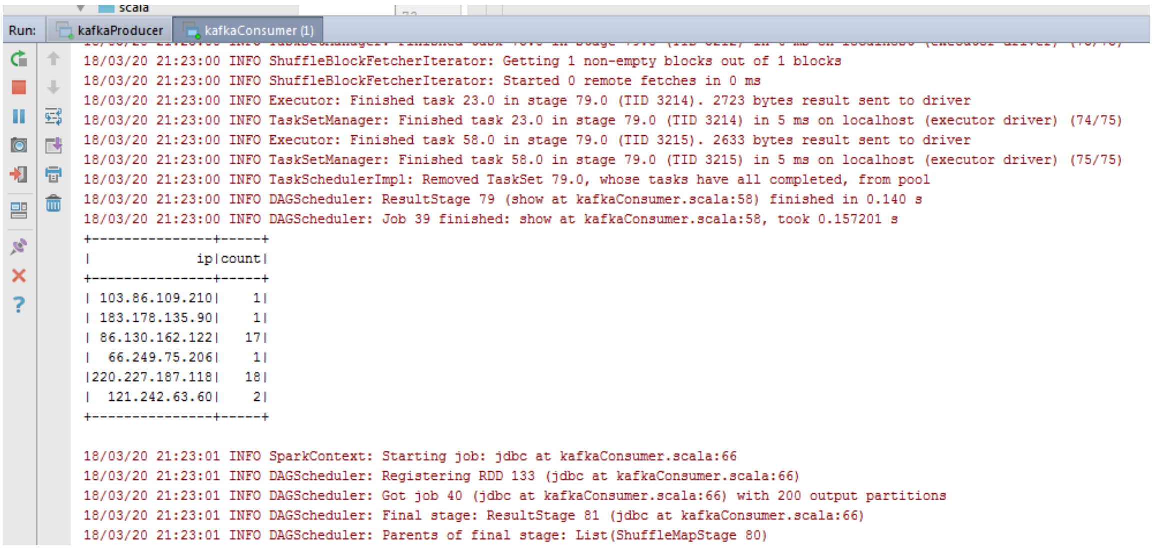
We can now see that topic if we run the list topic command

Kafka Producer Results:



```
18/03/20 21:11:21 INFO SyncProducer: Connected to localhost:9092 for producing
18/03/20 21:11:21 INFO SyncProducer: Disconnecting from localhost:9092
18/03/20 21:11:21 INFO SyncProducer: Disconnecting from localhost:9092
18/03/20 21:11:21 INFO SyncProducer: Connected to Admin1-PC:9092 for producing
18/03/20 21:11:21 INFO SyncProducer: Connected to Admin1-PC:9092 for producing
14.141.151.234 - - [12/Jan/2018:11:21:41 +0530] "GET /wp-content/plugins/simple-follow-me-social-buttons-widget/assets/css/icons.css HTTP/1.1" 200 68064 "http://www.bigdataa
49.207.119.217 - - [02/Jan/2018:17:56:19 +0530] "GET /wp-content/plugins/simple-follow-me-social-buttons-widget/assets/css/style.css HTTP/1.1" 200 420 "http://www.bigdataana
14.141.151.234 - - [12/Jan/2018:11:21:41 +0530] "GET /wp-content/plugins/foobox-image-lightbox/js/foobox.free.min.js HTTP/1.1" 200 13268 "http://www.bigdataanalyst.in/ignite
49.207.119.217 - - [02/Jan/2018:17:56:19 +0530] "GET /wp-content/plugins/ifeature-slider/css/if-slider.css HTTP/1.1" 200 504 "http://www.bigdataanalyst.in/" "Mozilla/5.0 (Wi
14.141.151.234 - - [12/Jan/2018:11:21:41 +0530] "GET /wp-includes/js/jquery/jquery.js HTTP/1.1" 200 33337 "http://www.bigdataanalyst.in/ignite-filesystem-igfs/" "Mozilla/5.0
49.207.119.217 - - [02/Jan/2018:17:56:19 +0530] "GET /wp-includes/js/jquery/jquery-migrate.min.js HTTP/1.1" 200 3080 "http://www.bigdataanalyst.in/" "Mozilla/5.0 (Windows NT
14.141.151.234 - - [12/Jan/2018:11:21:41 +0530] "GET /wp-content/themes/virtue/assets/js/vendor/modernizr.min.js HTTP/1.1" 200 6310 "http://www.bigdataanalyst.in/ignite-file
49.207.119.217 - - [02/Jan/2018:17:56:19 +0530] "GET /wp-content/themes/virtue/assets/css/skins/default.css HTTP/1.1" 200 1328 "http://www.bigdataanalyst.in/" "Mozilla/5.0 (
14.141.151.234 - - [12/Jan/2018:11:21:41 +0530] "GET /wp-content/plugins/akismet/_inc/form.js HTTP/1.1" 200 700 "http://www.bigdataanalyst.in/ignite-filesystem-igfs/" "Mozil
49.207.119.217 - - [02/Jan/2018:17:56:19 +0530] "GET /wp-content/plugins/simple-follow-me-social-buttons-widget/assets/js/front-widget.js HTTP/1.1" 200 696 "http://www.bigda
```

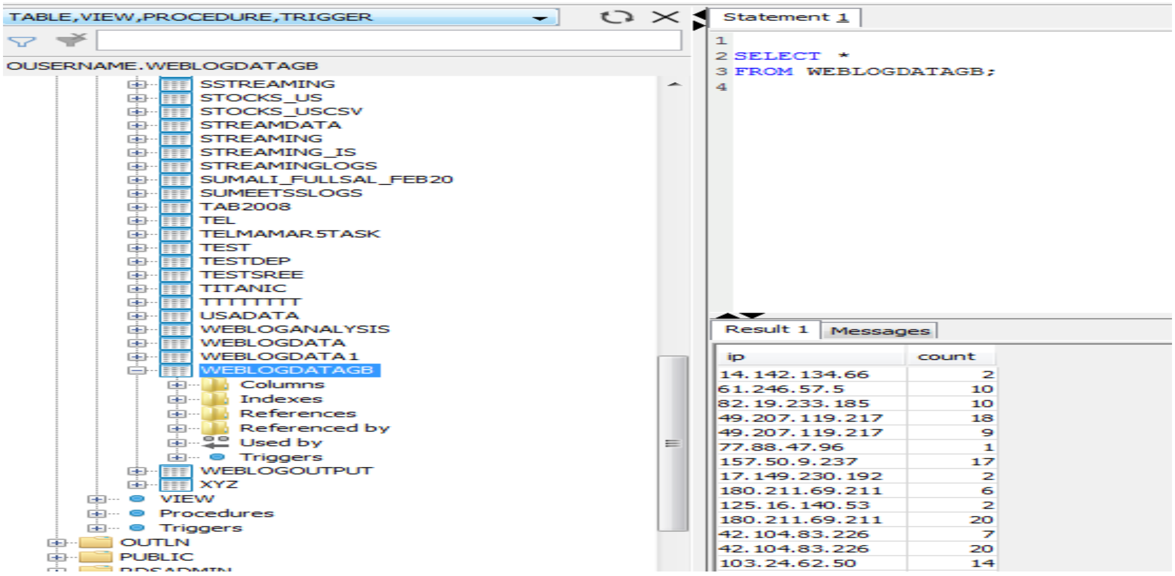
Kafka Consumer execution



```
18/03/20 21:23:00 INFO ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 1 blocks
18/03/20 21:23:00 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
18/03/20 21:23:00 INFO Executor: Finished task 23.0 in stage 79.0 (TID 3214). 2723 bytes result sent to driver
18/03/20 21:23:00 INFO TaskSetManager: Finished task 23.0 in stage 79.0 (TID 3214) in 5 ms on localhost (executor driver) (74/75)
18/03/20 21:23:00 INFO Executor: Finished task 58.0 in stage 79.0 (TID 3215). 2633 bytes result sent to driver
18/03/20 21:23:00 INFO TaskSetManager: Finished task 58.0 in stage 79.0 (TID 3215) in 5 ms on localhost (executor driver) (75/75)
18/03/20 21:23:00 INFO TaskSchedulerImpl: Removed TaskSet 79.0, whose tasks have all completed, from pool
18/03/20 21:23:00 INFO DAGScheduler: ResultStage 79 (show at kafkaConsumer.scala:58) finished in 0.140 s
18/03/20 21:23:00 INFO DAGScheduler: Job 39 finished: show at kafkaConsumer.scala:58, took 0.157201 s
+---------------+-----+
|             ip|count|
+---------------+-----+
| 103.86.109.210|    1|
| 183.178.135.90|    1|
| 86.130.162.122|   17|
|  66.249.75.206|    1|
|220.227.187.118|   18|
|   121.242.63.60|    2|
+---------------+-----+

18/03/20 21:23:01 INFO SparkContext: Starting job: jdbc at kafkaConsumer.scala:66
18/03/20 21:23:01 INFO DAGScheduler: Registering RDD 133 (jdbc at kafkaConsumer.scala:66)
18/03/20 21:23:01 INFO DAGScheduler: Got job 40 (jdbc at kafkaConsumer.scala:66) with 200 output partitions
18/03/20 21:23:01 INFO DAGScheduler: Final stage: ResultStage 81 (jdbc at kafkaConsumer.scala:66)
18/03/20 21:23:01 INFO DAGScheduler: Parents of final stage: List(ShuffleMapStage 80)
```

Output saved in Oracle DB



```
SELECT *
FROM WEBLOGDATAGB;
```

ip	count
14.142.134.66	2
61.246.57.5	10
82.19.233.185	10
49.207.119.217	18
49.207.119.217	9
77.88.47.96	1
157.50.9.237	17
17.149.230.192	2
180.211.69.211	6
125.16.140.53	2
180.211.69.211	20
42.104.83.226	7
42.104.83.226	20
103.24.62.50	14

V. CONSLUSION

In this paper we will be processing different types of errors / IP address from Web Server Logs. This can be used to process different websites like twitter data processing, Share Market Analysis. Spark Streaming on Hadoop YARN cluster processing messages from Apache using the new direct API. It is used for Business "things" such as IOT applications that are connected devices and sensors, predictive analytics, which are used to manage the risk and design the new business opportunities with the real time analytics.

REFERENCES

[1] The Complete Reference https://spark.apache.org/ (last accessed on 7 April 2019)

[2] https://spark.apache.org/docs/2.2.0/structured-streaming-programming-guide.html (last accessed on 7 April 2019)

[3] www.stackoverflow.com (last accessed on 7 April 2019)

[4] “SparkR: Scaling R Programs with Spark”, Shivaram Venkataraman, Zongheng Yang, Davies Liu, Eric Liang, Hossein Falaki, Xiangrui Meng, Reynold Xin, Ali Ghodsi, Michael Franklin, Ion Stoica, and Matei Zaharia. SIGMOD 2016. June 2016.

[5] “MLlib: Machine Learning in Apache Spark”, Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Zadeh, Matei Zaharia, and Ameet Talwalkar. Journal of Machine Learning Research (JMLR). 2016.

[6] “Spark SQL: Relational Data Processing in Spark.” Michael Armbrust, Reynold S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael J. Franklin, Ali Ghodsi, Matei Zaharia. SIGMOD 2015. June 2015.

[7] “GraphX: Unifying Data-Parallel and Graph-Parallel Analytics”. Reynold S. Xin, Daniel Crankshaw, Ankur Dave, Joseph E. Gonzalez, Michael J. Franklin, Ion Stoica. OSDI 2014. October 2014.

[8] “Discretized Streams: Fault-Tolerant Streaming Computation at Scale.” Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, Ion Stoica. SOSP 2013. November 2013.

[9] “Shark: SQL and Rich Analytics at Scale.” Reynold S. Xin, Joshua Rosen, Matei Zaharia, Michael J. Franklin, Scott Shenker, Ion Stoica. SIGMOD 2013. June 2013.Shark: Fast Data Analysis Using Coarse-grained Distributed Memory (demo). Cliff Engle, Antonio Lupher, Reynold S. Xin, Matei Zaharia, Haoyuan Li, Scott Shenker, Ion Stoica. SIGMOD 2012. May 2012. Best Demo Award.

[10] “Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing”. Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica. NSDI 2012. April 2012. Best Paper Award.

[11] “Spark: Cluster Computing with Working Sets”. Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica. HotCloud 2010. June 2010.

[12] “Discretized Streams: An Efficient and Fault-Tolerant Model for Stream Processing on Large Clusters.” Matei Zaharia, Tathagata Das, Haoyuan Li, Scott Shenker, Ion Stoica. HotCloud 2012. June 2012.

[13] https://spark.apache.org/research.html (last accessed on 7 April 2019)