Twitter Big Data Analytics Using Spark

CS5540 Project Report

Project Members:

Sri Harsha Chennavajjala (SC9V9)

Teja Garidepally (TGWW4)

Raj Kiran Reddy Munnangi (RMYB9)

Introduction

Big Data:

Big data is the buzzing word in the present software industry. Huge amounts of data is being generated daily from various sources. Companies are trying to perform analytics on big data and get some valuable output which gives an edge over there competitors. In order to achieve this we need to program map reduce jobs in Hadoop ecosystem. It is very difficult to develop the code and reuse it for different business cases. On the other hand, People are very much comfortable to query data using SQL like queries.

Apache Spark:

Apache Spark is an open source cluster computing framework originally developed in the AMPLab at University of California, Berkeley but was later donated to the Apache Software Foundation where it remains today. Spark provides multi-stage in-memory primitives provides performance up to 100 times faster for certain applications.
Spark SQL is a component on top of Spark Core that introduces a new data abstraction called DataFrames, which provides support for structured and semi-structured data. Spark SQL provides a domain-specific language to manipulate DataFrames in Scala, Java, or Python. It also provides SQL language support, with command-line interfaces and ODBC/JDBC server.

Objective:

The goal of our project is to collect tweets using the keyword “#android” from the twitter and to analyze the collected data using Spark. We Selected “android” as our keyword because, the resulting tweets will be more and will be from throughout the world.

Environment Setup:

Environment: IBM Bluemix

Database: IBM dashDB

Frame Work: Apache Spark

Visualization: *matplotlib* from MatLab

Programming Languages: Python, Scala, Spark SQL

Data Source: Twitter Data on keyword “#android”

Volume of Data: 0.32 Million Tweets

Data Format: JSON

Introduction to IBM Bluemix:

IBM Bluemix is a cloud platform as a service (PaaS) developed by IBM. It supports several programming languages and services as well as integrated DevOps to build, run, deploy and manage applications on the cloud. Bluemix is based on Cloud Foundry open technology and runs on SoftLayer infrastructure. Bluemix supports several programming languages including Java, Node.js, Go, PHP, Python, Ruby Sinatra, Ruby on Rails and can be extended to support other languages such as Scala through the use of buildpacks.

Bluemix provides the following features:

- A range of services that enable you to build and extend web and mobile apps fast.
- Processing power for you to deliver application changes continuously.
- Fit-for-purpose programming models and services.
- Manageability of services and apps.
- Optimized and elastic workloads.
- Continuous availability.

The services we used in Bluemix are…

IBM Cloudant:

IBM Cloudant is a fully managed JSON document DBaaS that's optimized for data availability, durability, and mobility…perfect for fast-growing mobile & web apps. What makes Cloudant unique is its advanced indexing and ability to push data to the network edge, across multiple data centers and devices, for faster access and greater fault tolerance. It allows users to access data anytime, anywhere.

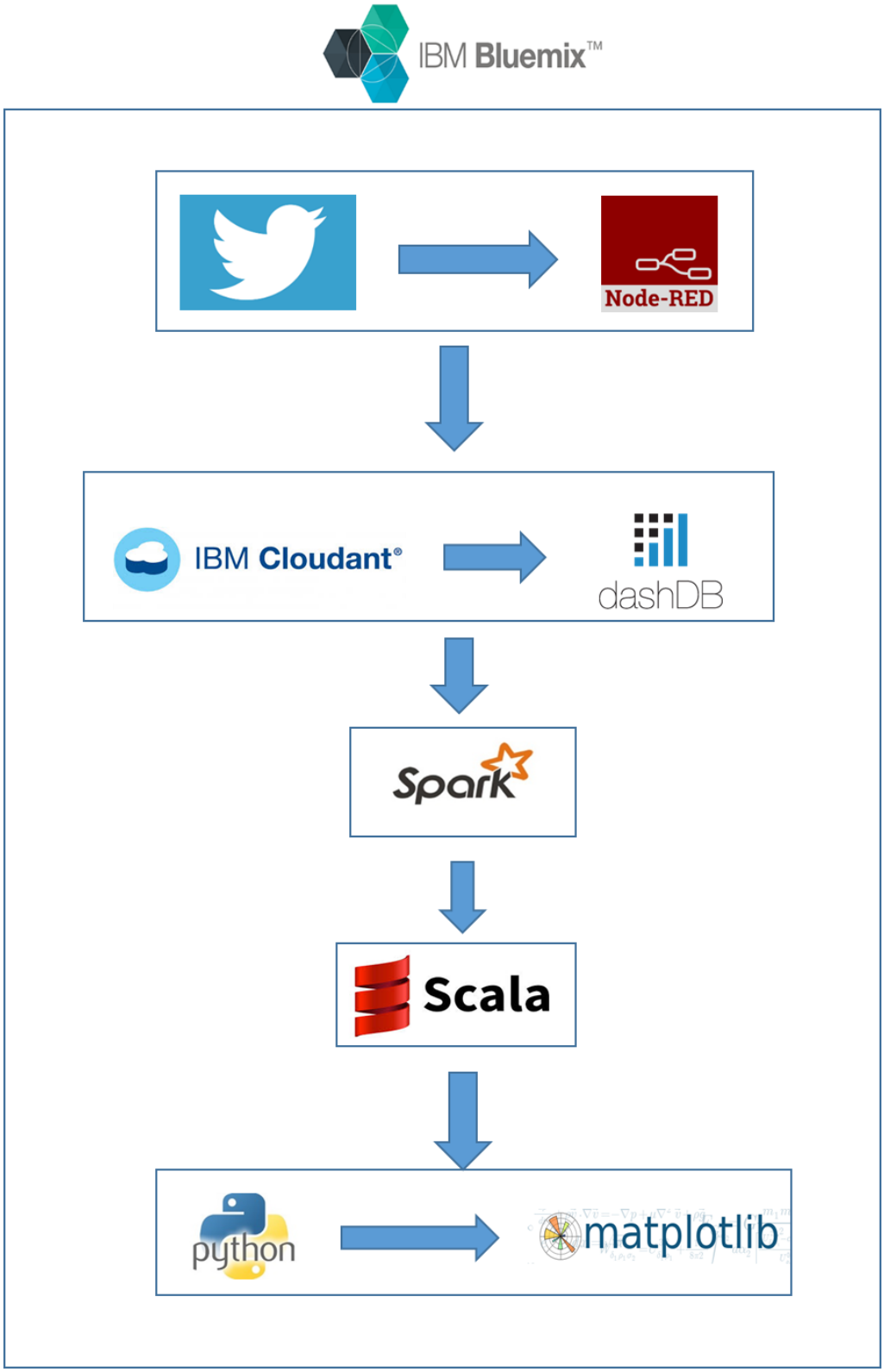
dashDB:

dashDB offers massive scalability and performance through its MPP architecture, and is compatible with a wide range of business intelligence toolsets and analytics. dashDB's integrated, in-database analytics let you quickly realize more value from your data.

Node-RED:

Node-RED provides a browser-based UI for creating flows of events and deploying them to its light-weight runtime. With built in node.js, it can be run at the edge of the network or in the cloud. The node package manager (npm) ecosystem can be used to easily extend the palette of nodes available, enabling connections to new devices and services.

System Architecture:



Each module of the above architecture diagram will be explained in the next section.

Approach:

The main idea of the project is to collect tweets using a keyword “#android” from a twitter streaming API and to analyze the data using Apache Spark.

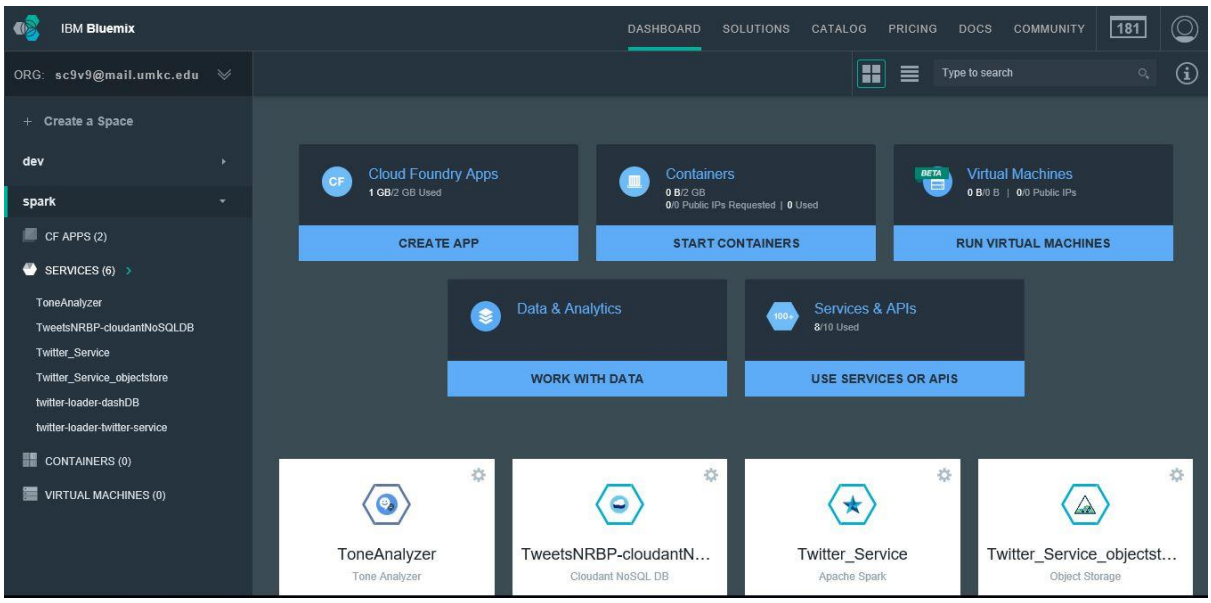
Implementation of Proposed Approach:

We used IBM Bluemix as our environment to perform the analytical queries on Twitter data.

Implementation steps:

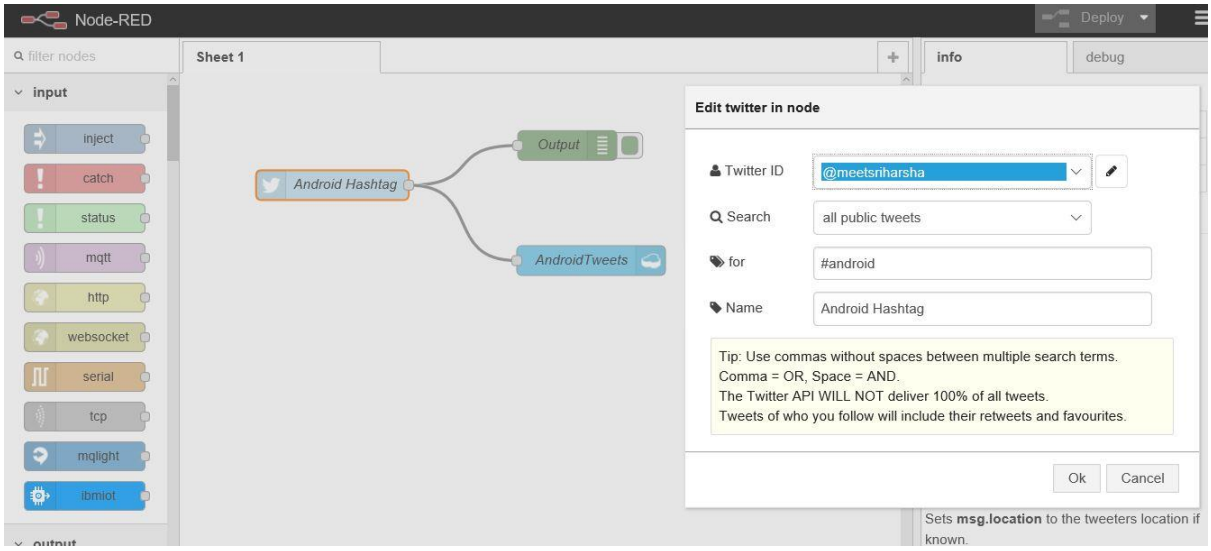
Stage 1:

- First we have created an account in IBM Bluemix.



User Dashboard Screen in Bluemix

- Next, we developed an application **TweetsNRBP** using Node.js language for collecting tweets based on the given keyword.
- This application collects tweets data with the keyword “#android”.

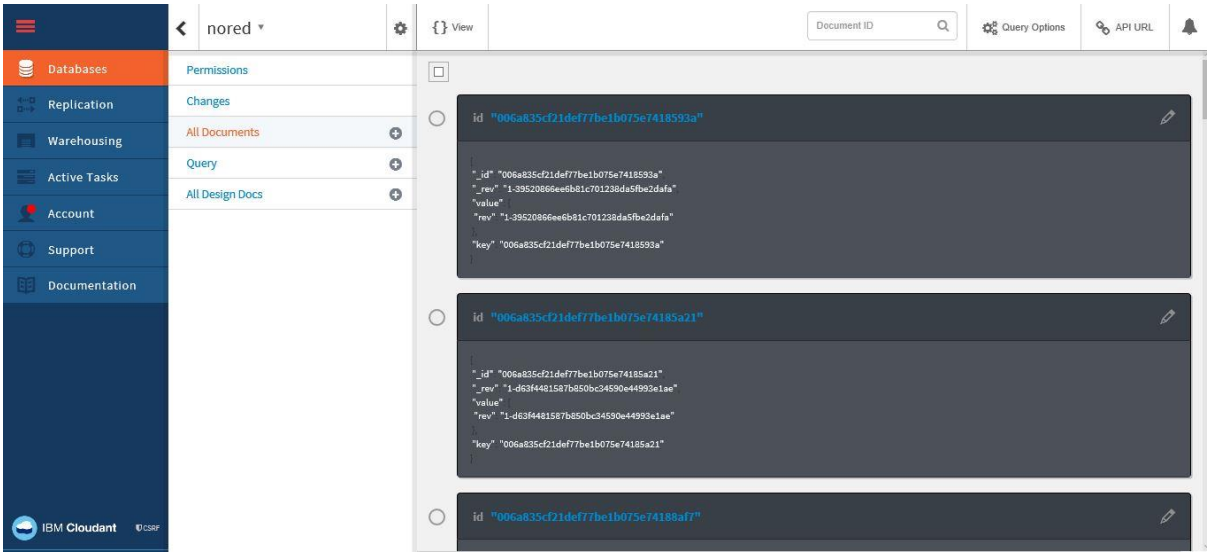


Node-RED interface for application workflow design

- The collected tweets are pushed in to IBM Cloudant data storage.

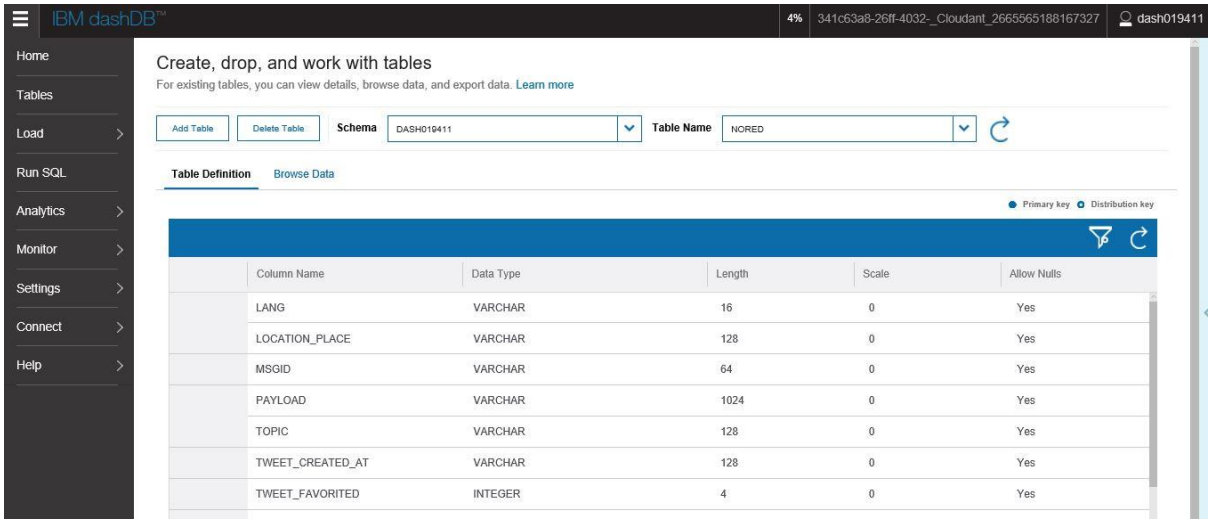
Stage 2:

- The collected data in IBM Cloudant is stored in JSON format.



Snapshot of Cloudant database

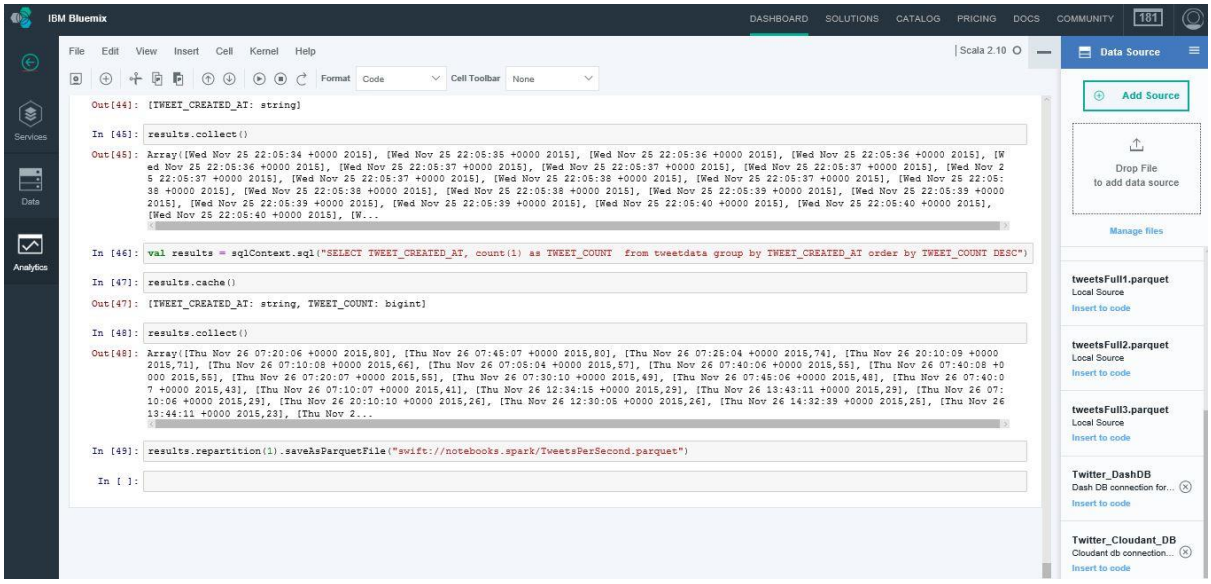
- The data in IBM Cloudant will be parsed and stored in a table format in dashDB.
- In dashDB, data is stored in table DASH019411.nored.



Snapshot of IBM dashDB table for storing tweets data

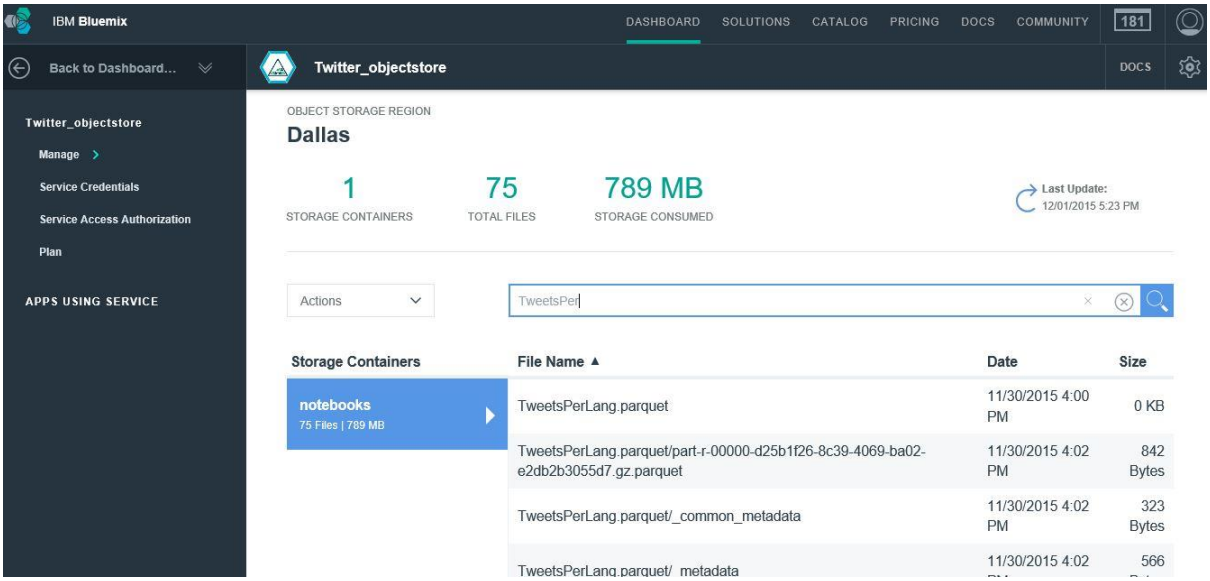
Stage 3:

- In IBM Analytics for Apache Spark we have created two notebooks in scala and python.
- Scala notebook is used to analyze the data using Spark SQL queries.



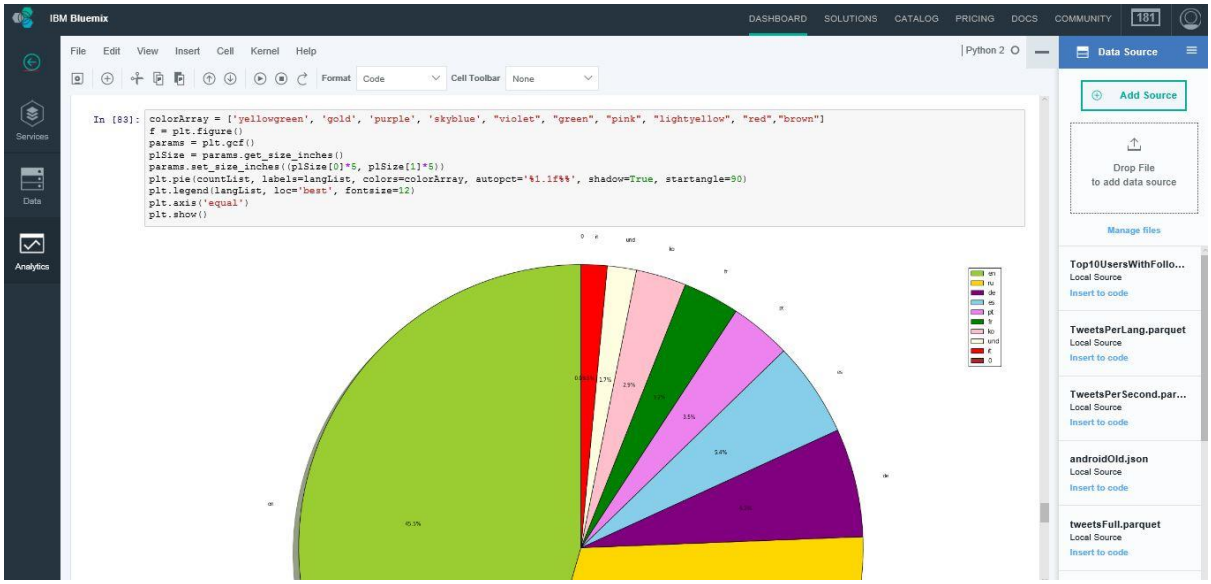
Scala notebook interface

- The results of analytical queries are stored in an object storage named **Twitter_objectstore** in form of parquet files.



Object Storage view with files and storage statistics

- In Python notebook, results of analytical queries are visualized.



Python notebook interface

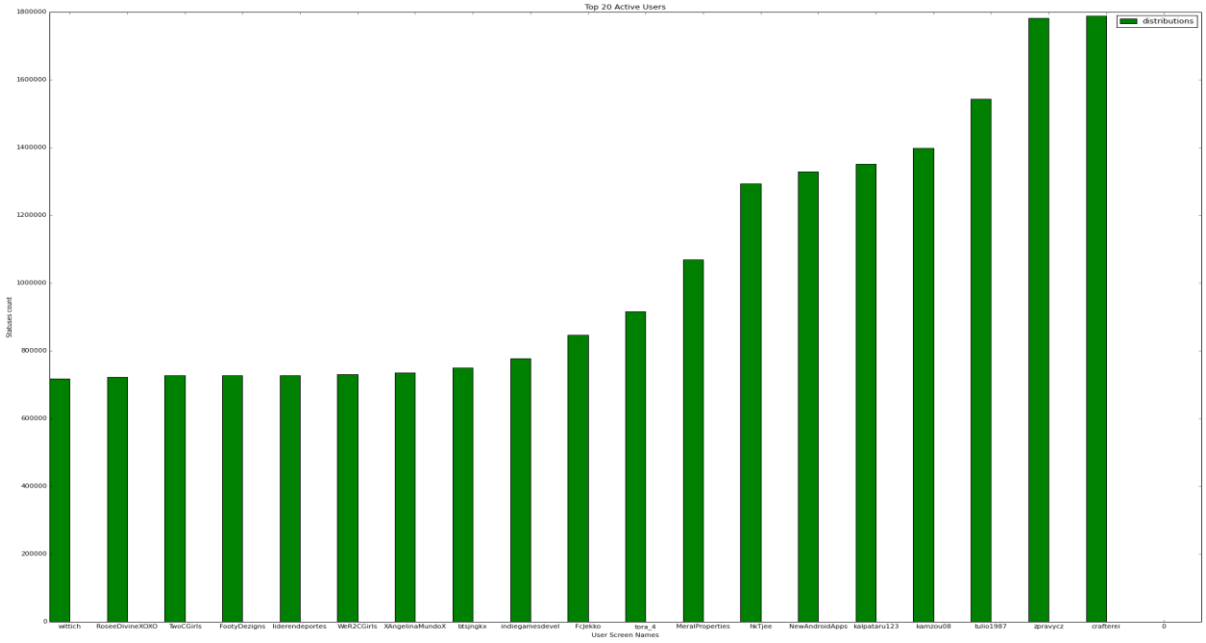
Analytic Queries and Visualization:

Query 1:

```
SELECT TWEET_USER_SCREEN_NAME, MAX(TWEET_USER_STATUSES_COUNT) AS
TWEET_USER_STATUSES_COUNT FROM tweetdata group by TWEET_USER_SCREEN_NAME
order by TWEET_USER_STATUSES_COUNT DESC LIMIT 20
```

Description: To find top active users in dataset

Visualization: Bar Graph



X-axis: User Screen names

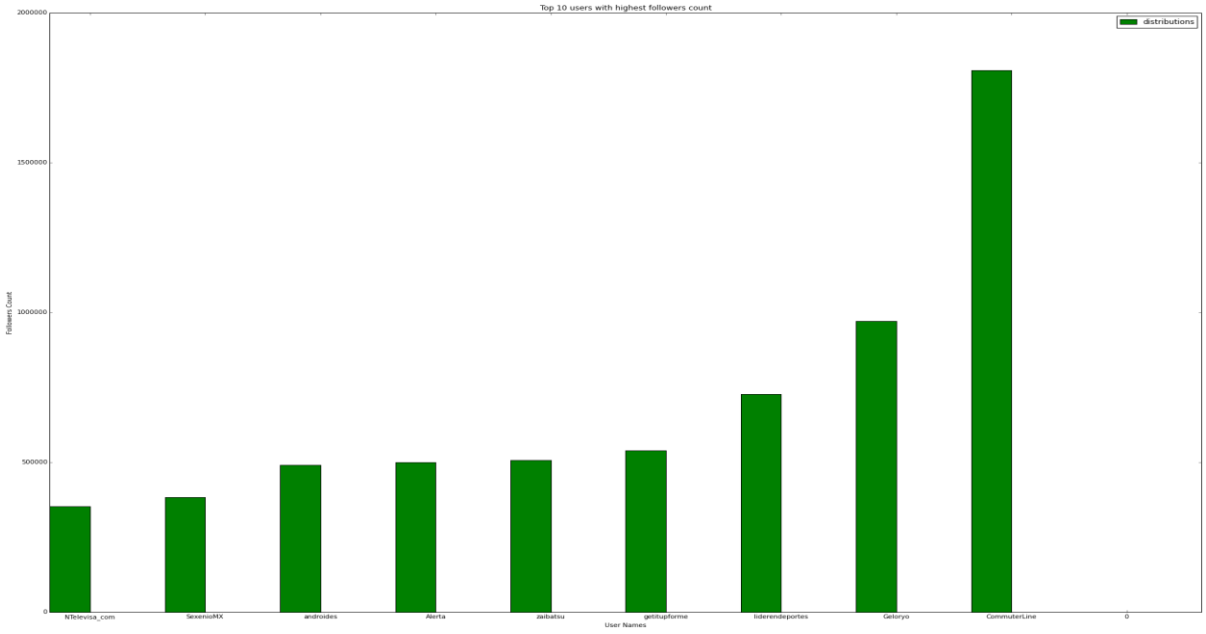
Y-axis: Statuses count

Query 2:

```
SELECT TWEET_USER_SCREEN_NAME, MAX(TWEET_USER_FOLLOWERS_COUNT) AS TWEET_USER_FOLLOWERS_COUNT FROM tweetdata group by TWEET_USER_SCREEN_NAME order by TWEET_USER_FOLLOWERS_COUNT DESC LIMIT 10
```

Description: To find Top Users with highest Followers Count

Visualization: Bar Graph



X-axis: User Name

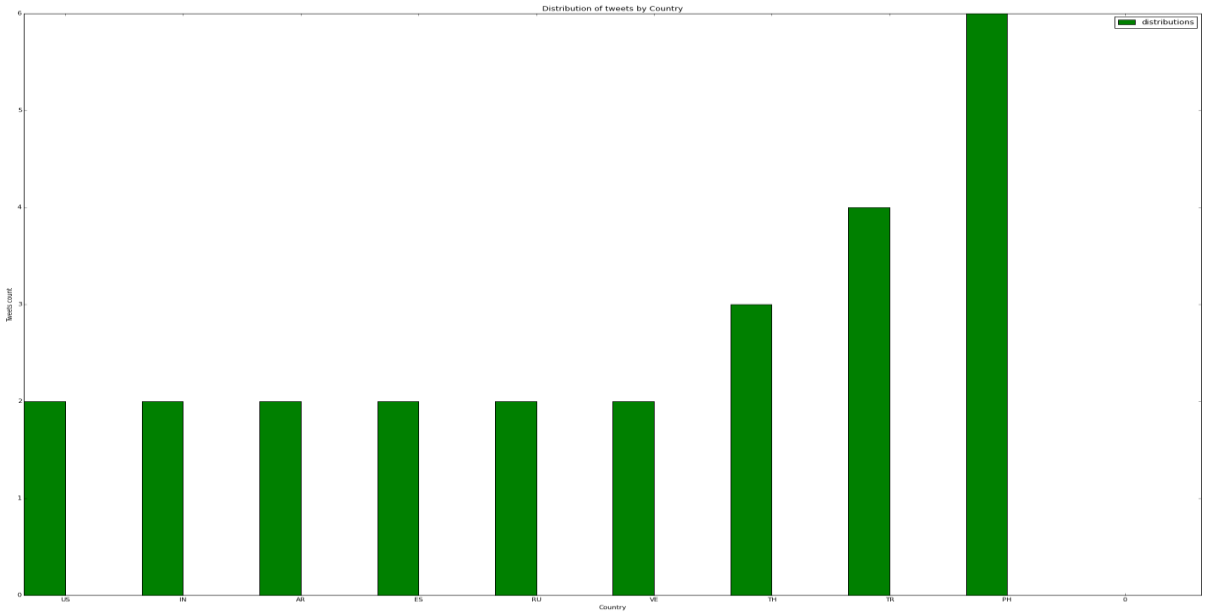
Y-axis: Follower count

Query 3:

```
SELECT TWEET_PLACE_COUNTRY_CODE, COUNT(1) AS TWEETS_PER_COUNTRY FROM tweetdata group by TWEET_PLACE_COUNTRY_CODE order by TWEETS_PER_COUNTRY DESC LIMIT 10
```

Description: To find tweets per country

Visualization: Bar Graph



X-axis:Country

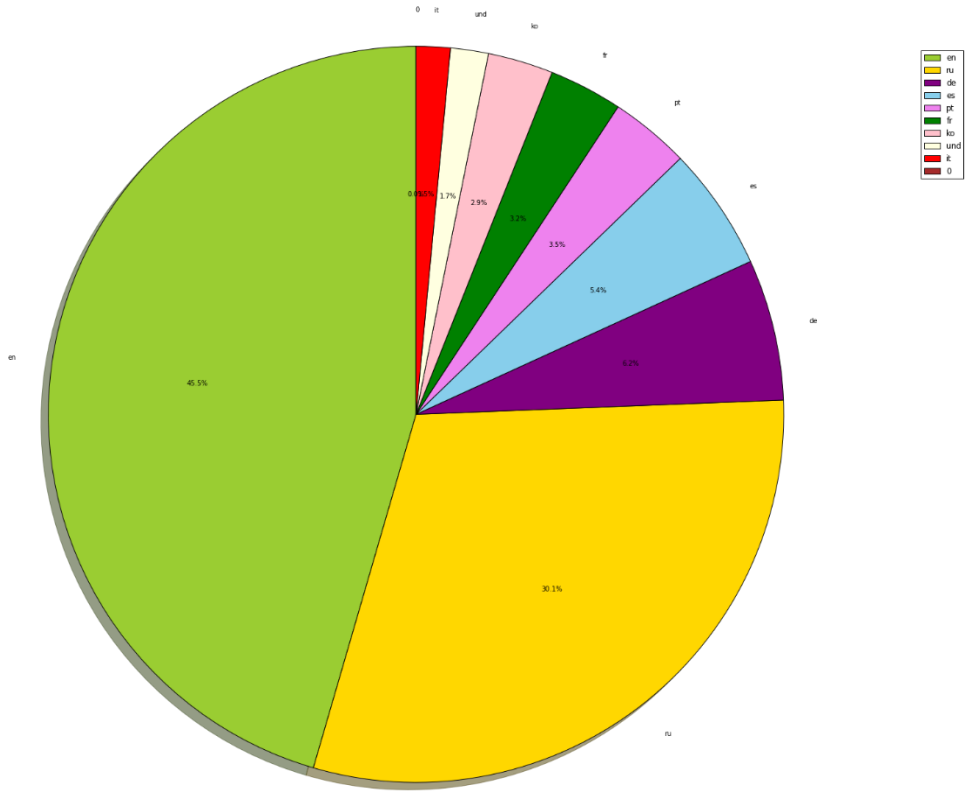
Y- axis:Tweets count

Query 4:

```
SELECT TWEET_LANG, count(1) as totTweets from tweetdata group by TWEET_LANG
order by totTweets
```

Description: To find number of tweets per language

Visualization: Pie Chart

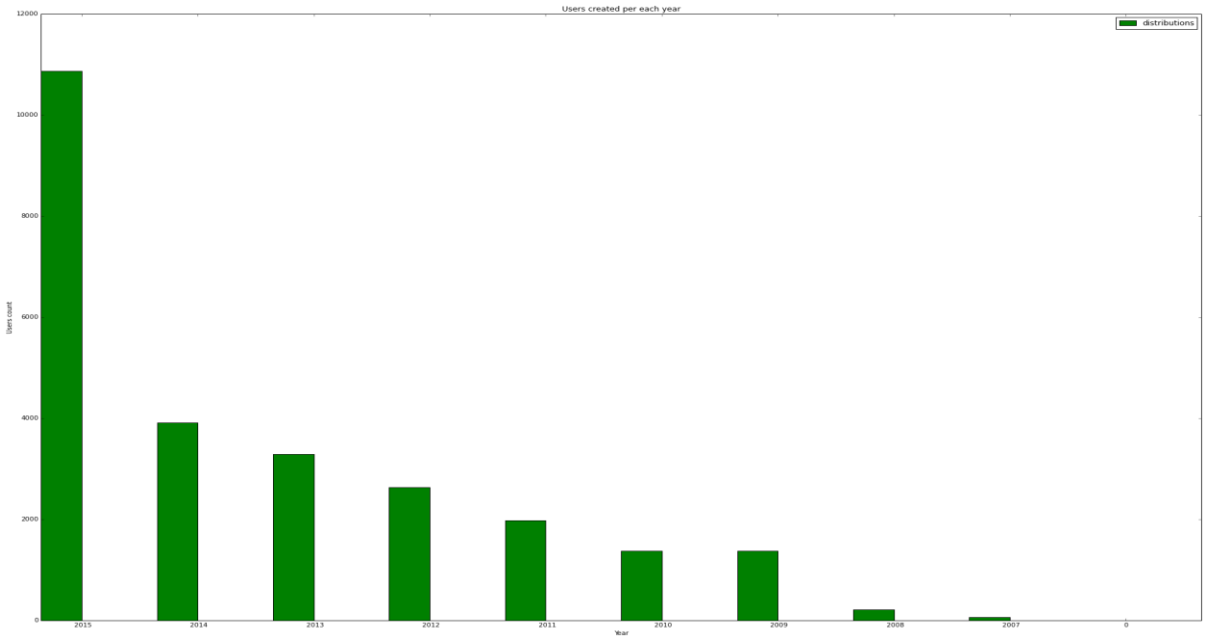


Query 5:

```
SELECT t1.YR as YR, count(1) as CNT FROM (select DISTINCT TWEET_USER_ID, substring(TWEET_USER_CREATED_AT,26) AS YR FROM tweetdata where TWEET_USER_CREATED_AT IS NOT NULL) t1 group by YR order by YR
```

Description: To find user accounts created per year

Visualization: Bar graph



X-axis: Year

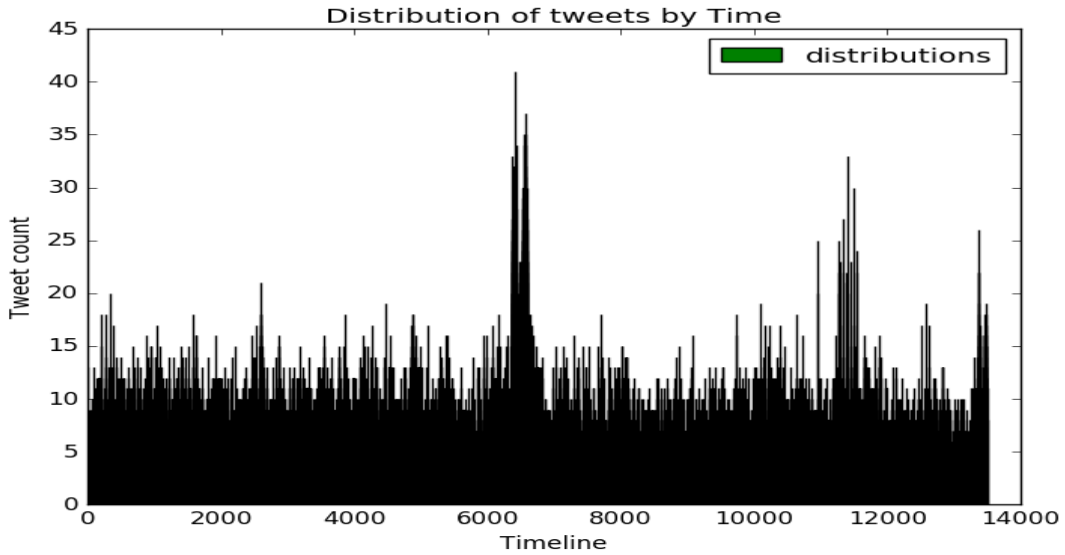
Y-axis: Users Count

Query 6:

```
val tweetCreatedDt = tweets.filter(_.nonEmpty).map(x =>
(extractTweetDate(x), 1)) val tweetCreatedDtCnt =
tweetCreatedDt.reduceByKey((a, b) => a + b)
tweetCreatedDtCnt.repartition(1).saveAsTextFile("D:/UMKC/Docs/Subjects/PBDM
/PB_Project/TweetsPerTime")
```

Description: To find users per year

Visualization: Bar Graph



X-axis: Timeline in seconds

Y-axis: Tweet count