Live Streaming Data Analysis By Distributed Technology Hive

Thirupathi Gandla (002343333)
Saivivek Therala (002343954)

System requirements

- Environment: IBM Info Sphere Big Insights v3.0
- NoSQL Tool: Hive
- Visualization: IBM Big Sheets.
- Programming Languages: Python, Java
- Data Source : Twitter Data on keyword “android”
- Data Size : 3.87 GB
- Data Format : json

Design Architecture

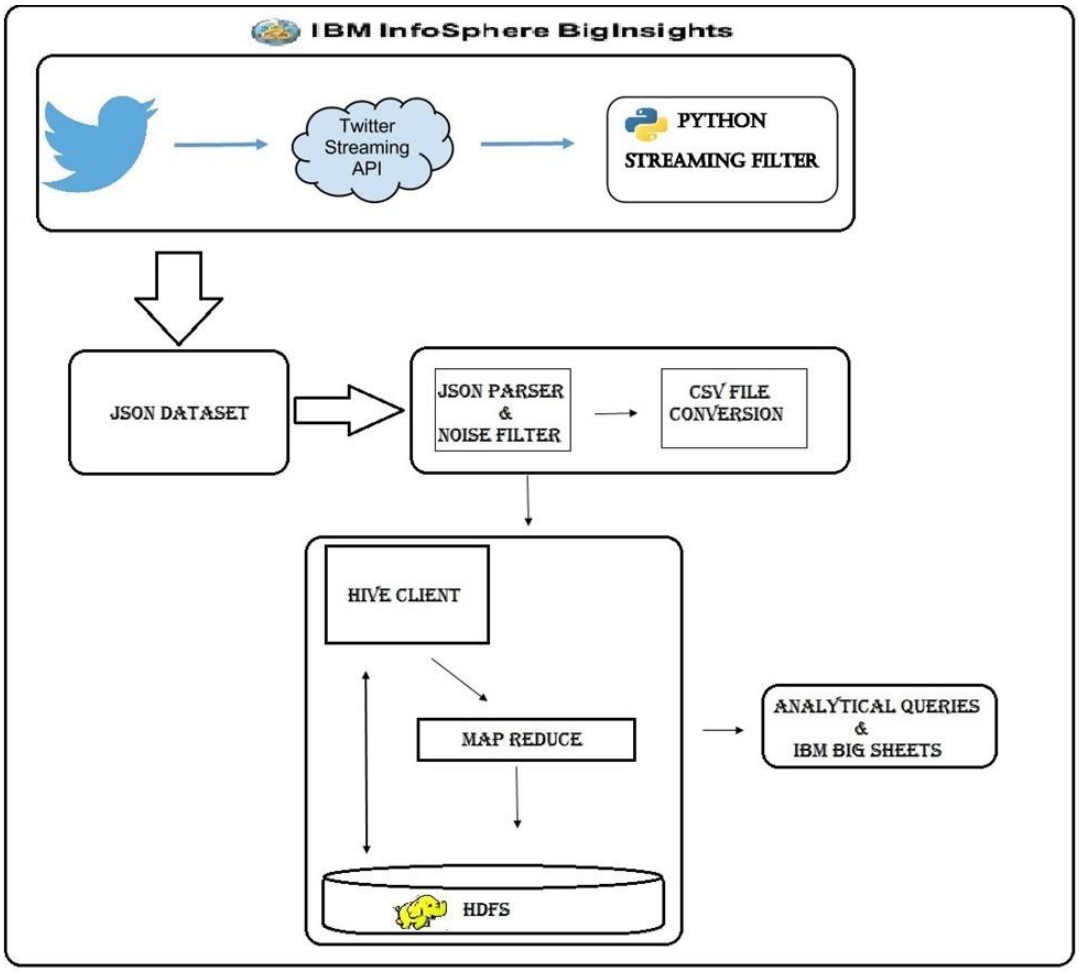


Fig 1: System Architecture

Implementation

Procedure:

The core idea of the project is to gather tweets using a keyword “android” from a twitter streaming API and to analyze the data.

Implementation steps:

Note:

All the below stages were implemented in IBM Big Platform

Stage 1:

- A twitter application is created to access the streaming tweets from twitter website.
- Required authorizations were created for connecting to the API’s from python code.
- A library named tweepy from python is used to download streaming tweets from twitter with a streaming filter i.e., ‘android’.
- All the streaming tweets were collected into a json file to the local machine.

Stage 2:

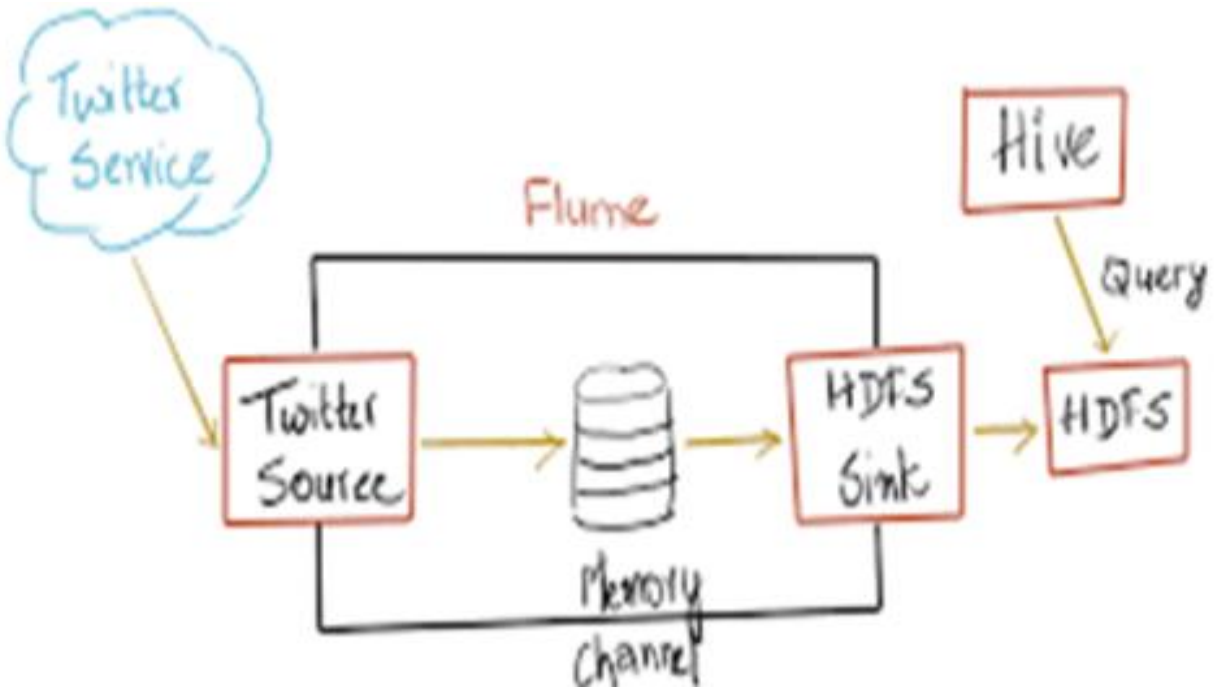
- A JSON2CSV.java class file is developed to filter the noise and convert the json file to csv format.

Stage 3:

- A Hive table named **master_tweets** is created using Row Format as csv serde.
- All the Tweets were loaded into the above hive table.
- Again, three tables named **data_tweet, user_info, user_count_info** were created to store the individual data from master table.
- Analytical queries were executed in hive shell to view and analyze the data.

Stage 4:

- All the Analytical queries output is stored in external hive tables.
- External table data is copied from Hadoop file system to local system using hadoop fs – copyToLocal command.
- We used IBM Big Sheets for visualization of the results. Once the results are available in local, We uploaded that results in to the Infosphere files tab.
- In the tool we convert the comma separated text format data in to sheet format. Then we select the line reader as comma separated value data.
- We save this as a master workbook then the control is redirected to bigsheets tab. Here we select the required visualization for our results data.



Hive Architecture

Tables Information

Table1

Syntax

```
create table master_tweets(
        created_at string,
        id bigint, id_str string,
        text string, user_id
        bigint, user_id_str
        string, user_name
        string,
        user_screen_name
        string, user_location
        string, user_protected
        boolean, user_verified
        boolean,
        user_followers_count
        bigint,
        user_friends_count
        bigint,
        user_listed_count
        bigint,
        user_favourites_count
        bigint,
        user_statuses_count
        bigint, user_created_at
```

```
        string, user_utc_offset
        int, user_time_zone
        string,
        user_geo_enabled
        boolean,
        user_lang string,
        user_contributors_enabled
        boolean, user_is_translator
        boolean, geo_type string,
        geo_coord_lat float,
        geo_coord_long float,
        coordy_type string,
        cordy_coord_lat float,
        cordy_coord_long float, place_id
        string, place_type string,
        place_name string,
        place_ctry_code string,
        place_country string,
        place_full_name string,
        retweet_count int, favorite_count
        int, favorited boolean, retweeted
        boolean,  possibly_sensitive
        boolean,
        filter_level     string,
        lang             string,
        timestamp_ms
        string
)
ROW FORMAT serde
'com.bizo.hive.serde.csv.CSVSerde' with
serdeproperties("separatorChar" = "\","quoteChar" =
"\"") stored as textfile;
```

Load Statement:

```
load data local inpath "/home/biadmin/Desktop/PB2-2days/csv_output/output1.csv" into
table master_tweets;
```

Table 2:

data_tweet

Syntax:

```
create table data_tweet(
       created_at string,
       id bigint, id_str string,
       text string, user_id
       bigint, geo_type string,
       geo_coord_lat float,
       geo_coord_long float,
```

```
coordy_type string,
cordy_coord_lat float,
cordy_coord_long float,
place_id string,
place_type string,
place_name string,
place_ctry_code string,
place_country string,
place_full_name string,
retweet_count int,
favorite_count int,
favorited boolean,
retweeted boolean,
possibly_sensitive
boolean,
filter_level string,
lang string,
timestamp_ms string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' stored
as textfile;
```

Load Statement:

```
Insert overwrite table data_tweet select distinct
created_at,id,id_str,text,user_id,geo_type,geo_coord_lat,geo_coord_long,coordy_type,cordy
_coord_lat,cordy_coord_long,place_id,place_type,place_name,place_ctry_code,place_countr
y,place_full_name,retweet_count,favorite_count,favorited,retweeted,possibly_sensitive,filter
_level,lang,timestamp_ms from master_tweets;
```

Table 3:

user_info

Syntax:

```
create table user_info( user_id
        bigint, user_id_str
        string, user_name
        string,
        user_screen_name
        string, user_location
        string, user_protected
        boolean, user_verified
        boolean,
        user_created_at string,
        user_utc_offset int,
        user_time_zone string,
        user_geo_enabled
        boolean,
        user_lang string,
        user_contributors_enabled boolean,
        user_is_translator boolean
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' stored
as textfile;
```

Load Statement:

```
Insert overwrite table user_info select distinct
    user_id,user_id_str,user_name,user_screen_name,user_location,user_protected,user_verified
    ,user
    _created_at,user_utc_offset,user_time_zone,user_geo_enabled,user_lang,user_contributors_
    enabl ed,user_is_translator from master_tweets;
```

Table 4: user_count_info

Syntax:

```
create table user_count_info(
      user_id bigint,
       user_followers_count
       bigint, user_friends_count
       bigint, user_listed_count
       bigint,
       user_favourites_count
       bigint, user_statuses_count
       bigint
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' stored
```

as textfile;

Load Statement:

Insert overwrite table user_count_info select
user_id,max(user_followers_count),max(user_friends_count),max(user_listed_count),max(
user_fa vourites_count),max(user_statuses_count) from master_tweets group by user_id;

Analytic Queries and Visualization

Query 1:

select count(ti.text), ui.user_id, ui.user_name, ui.user_screen_name From data_tweet ti JOIN user_info ui ON (ti.user_id = ui.user_id) GROUP BY ui.user_id, ui.user_name, ui.user_screen_name;

Description 1: To find the users contribution towards tweets in

dataset.

Visualization 1:

Fig 1: Tweet_Contribution_WordCloud

Query2

select ui.user_id,ui.user_name,ui.user_screen_name,uci.user_statuses_count from user_info ui JOIN user_count_info uci ON (ui.user_id=uci.user_id) ORDER BY uci.user_statuses_count DESC;

Description 2: To identify Active Users in twitter from the given

users dataset.

Visualization 2:

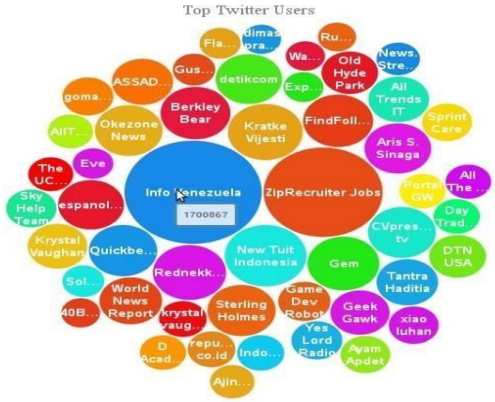


Fig 2: Active_Userschart_TagCloud.

Query 3:

select text,geo_coord_lat,geo_coord_long from data_tweet where geo_coord_lat is not null and geo_coord_long is not null;

Description 3: To plot the geographical locations of tweets from dataset.

Visualization 3:

Fig 3: Geoplot_Tweet_Map

Query 4:

```
select  ui.user_name,ui.user_screen_name,uci.user_followers_count from  user_info ui
JOIN user_count_info uci ON(ui.user_id=uci.user_id) ORDER BY
ci.user_followers_count DESC;
```

Description 4: To list out the popular users from the given dataset.

Visualization 4:

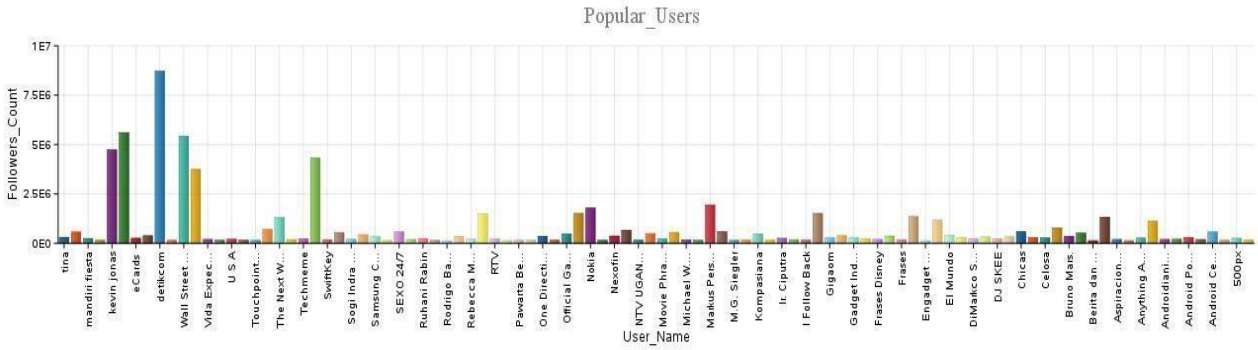


Fig 4: Popular_Users_Bargraph

Query 5:

```
select place_country country,count(*) count from data_tweet group by place_country;
```

Description 5: To find the tweet contribution from various countries.

Visualization 5:

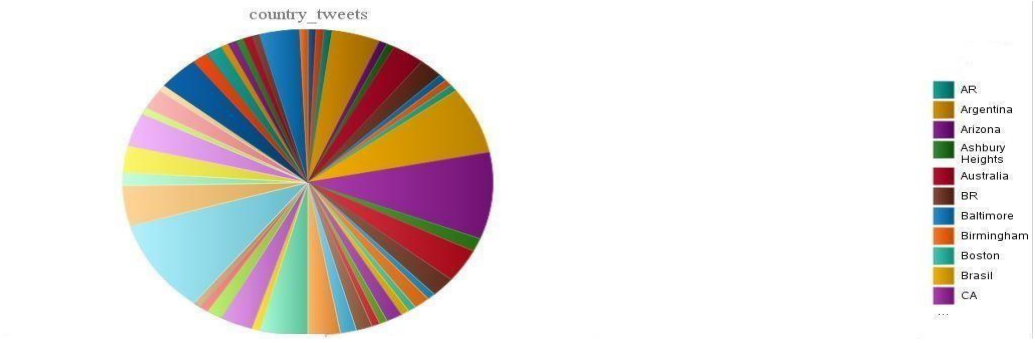


Fig 5: Country_Tweets_PieChart

Testing Details

1. Twitter Streaming Data from Python contains a lot of noise data in the tweets.
    a. Around 3.87 Giga Bytes of Data of 783876 tweets were collected but there is a noise data and no useful information in most of the tweets.
2. JSON Parser: Before converting the json file to csv file most of the tweets were filtered to get useful information.
    a. Around 19635 tweets were converted to csv file for hive tables.

3. A CSV File with 19635 is loaded to hive tables. All the data is loaded into master tables.

```
Time taken: 0.175 seconds
hive> load data local inpath '/home/biadmin/Desktop/PB2-2days/csv_output/output9.csv' into table master_tweets
Copying data from file:/home/biadmin/Desktop/PB2-2days/csv_output/output9.csv
Copying file: file:/home/biadmin/Desktop/PB2-2days/csv_output/output9.csv
Loading data to table default.master_tweets
OK
Time taken: 0.144 seconds
hive> select count(*) from master_tweets;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
```

```
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 6.98 sec   HDFS Read: 10287299 HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 980 msec
OK
19635
Time taken: 40.018 seconds, Fetched: 1 row(s)
hive>
```

4. From Master table master_tweets data is inserted into 3 other tables for extracting the individual records data.
    a. Data from master tables and 3 other tables were matched with the results count.

```
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 9.05 sec   HDFS Read: 10287398 HDFS Write:
Total MapReduce CPU Time Spent: 9 seconds 50 msec
OK
9572
Time taken: 53.54 seconds, Fetched: 1 row(s)
hive> select count(distinct user_id) from user_info;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
```

```
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 6.63 sec   HDFS Read: 1312091 HDFS Write:
Total MapReduce CPU Time Spent: 6 seconds 630 msec
OK
9572
Time taken: 31.504 seconds, Fetched: 1 row(s)
hive>
```

5. Our query 2 is useful to identify Active Users in twitter from the given users dataset which means the twitter page having highest number of tweets tweeted. This can be observed in the below two figures. In fig 3 we can see 1700867 tweets by Info Venezuela twitter page, we observed the tweet count in Info Venezuela(@Info_Ve) page as 1708271. This is a clear indication of successful execution of our joins queries.

Fig 6a: Query 2 Graphical output.

Fig 6b: Tweet count in official page.

Queries to be visualized:

1:
-- Language based tweets count from dataset

```
select lang, count(*) from data_tweet group by lang having lang is not null;
```

2:
-- Total no of Retweeted tweets in a dataset

```
select count(*) from data_tweet where text like "RT%";
```

3:
-- Top Friends Count from a given dataset

```
select ui.user_name,ui.user_screen_name,uci.user_friends_count
from user_info ui JOIN user_count_info uci
ON(ui.user_id=uci.user_id)
ORDER BY uci.user_friends_count DESC;
```