Volume 5, Issue 9, September 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Effective Image Analysis on Twitter Streaming using Hadoop Eco System on Amazon Web Service EC2

Gautam Goswami
Irisidea Technologies Pvt. Limited,
India

Abstract:** Twitter is becoming the most popular online micro blogging network of real time post that enables users to send and read short 140-character messages called "tweets". Registered users can read and post tweets, but unregistered users can only read them. Today's Twitter is now less focused on what are you doing but has emerged as a source for discovery, with a focus on sharing relevant information and engaging in conversation. Sharing various visual information in the form of images/photos are becoming very popular where all the follower can see what images/photos have been posted/twitted instantly. In this paper I am going to explain how effectively registered users shares/uploads images among the followers. This information/statistics would be of great value for any organization/company when they launch their new product in market. If a particular image/photo sharing is high among tweeter community, organization/company can be assured that their product is penetrating more in the market. Here I have analyzed the momentum of visual information propagation . So that followers can be aware of that something new have been lunched in market and subsequently will have the curiosity to dig more on it. In this paper, the collected twitter steaming which has been (rich amount of data in semi structure format JSON for an interval of time) referred to as big data are processed efficiently to achieve mentioned output. With the available traditional software like RDBMS, MVC architecture framework like Struts etc, it's impossible to achieve the desire goal. To leverage the cloud computing, I have used the Amazon Web Service EC2 [2]where Hadoop cluster has been created to analyze the twitter streaming data[3]. Other components of Hadoop eco system viz. Apache Flume[9], Hive [6] have also been used . Using Flume[9], twitter streaming[3] has been collected for a particular interval of time and subsequently stored in Hadoop Distribution File System (HDFS)[1,13,14] for further analysis where traditional RDBMS are not compatible. I have used Hive for mining the stored data filtered through Map Reduce phase[1,13,14]. Only Map[1,13,14] has been used to parse the semi structured Streaming data (JSON)[7].

***Keywords:** Analysis, BIGDATA, Comment, Flume, Hive, HQL, Structured, Semi-Structured, Twitter, Tweets, Un-Structured, HDFS (Hadoop Distributed File System), JSON(JavaScript object notation), Mapper (Map-Reduce).*

I. INTRODUCTION

With the deep penetration of internet into human society, social media over internet has started a new dimension from the early 2006. Various blogs of individuals, Facebook, Twitter are mainly leading a revolutionary environment in the society. If we consider Twitter, it is an way to learn about the world through another person's eyes. Besides, it is becoming a marketing and advertising tool where modern internet savvy user slowly neglecting other advertising channels like TV, hoarding etc. Now-a-days most of the big companies use twitter as an advertising medium. They circulate their newly launched product information with short notes and images via twitter. Based on that, user shares the same among his followers with comments etc and again from a particular follower to his followers like a chain reaction. From different views and comments on the streaming data, companies can analysis the information on their products which indeed help them to take further decisions . The size of the data, continuously collected from the twitter stream for a particular duration being huge , can be referred to as BIG DATA. Using Hadoop and its components[1,13,14], it is together referred to as Hadoop Ecosystem[1,13,14] and is used for decision making step. In this paper image statistics are analyzed to know how many user shared or posted the same image across their community over Twitter.

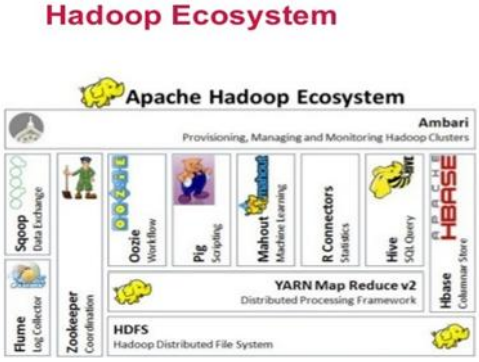


Fig.1 Hadoop Eco System

The above figure depicts how Hadoop eco system[1,13,14] is sound enough to solve and analyze any BIG DATA. Apache Flume[9] which is part of ecosystem, has been used to collect the twitter streaming data and stored in HDFS[1,13,14]. The Map has been used to filter out the raw streaming data which is semi structured[4] (JSON). Filtered information which is the output of Mapper[1,13,14], has been stored in Apache Hive[6]. HQL and UDF have been used to collect the images from bad and good tweets. Finally a freeware tool Anti-Twin[10] is used for detecting duplicate images. From this we can figure out the statistics of image sharing over twitter.

II. PROBLEM STATEMENT

A. Existing System

We are already familiar with and pointed out the limitation of traditional/existing software to analyze the BIG DATA. Present RDBMS is incapable of storing the twitter streaming data where size would be more than Petabytes. Also data is not in structured format. Filtering out the streaming data with required parameters is another challenge with traditional RDBMS and available frameworks. Distributed computing is mandatory in clustered environment to process the data. Otherwise time consumption for processing will be beyond our design assumption and time estimation. In comparison to traditional data warehousing tools/software, Apache Hive is an excelling data warehousing component running on HDFS for data mining for a very large size of data.

B. Proposed Architecture or System

Having listed out all the drawbacks and incapability of analyzing BIG DATA (Here Twitter steaming) using traditional software , as an alternative and effective framework, I have used Hadoop Ecosystem[1,13,14]. Apache Flume[9] has been configured to collect the continuous flow of streaming data from twitter and saved into HDFS[1,13,14]. From those raw semi structured data, Map[1,13,14] is executed in cluster environment to leverage the benefit of distributed processing in Amazon web service EC2[2]. In this way we can eliminate the complexity and the procurement of hardware to setup the Hadoop cluster[1,13,14]. The following diagram depicts clearly the architectural view of proposed system where the entire flow diagram of data movement as well as processing in various component of Hadoop Ecosystem[1,13,14] is shown.

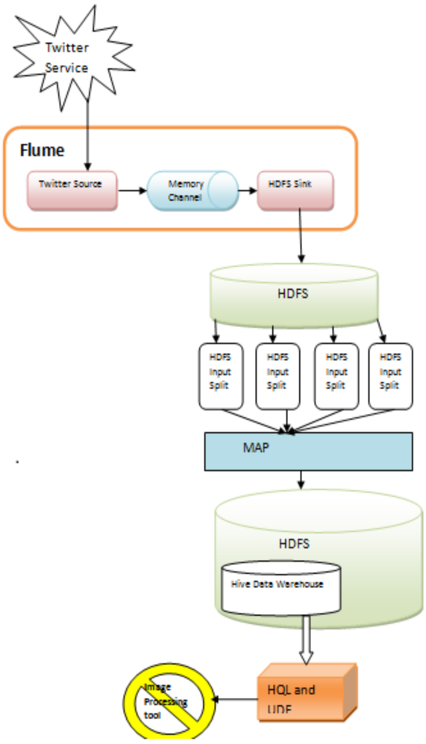


Fig.2 Proposed architecture

To find out the counting of same image posting over tweets, a freeware tool Anti-Twin is used (http://www.anti-twin.com).

III. EXECUTION METHODOLOGY

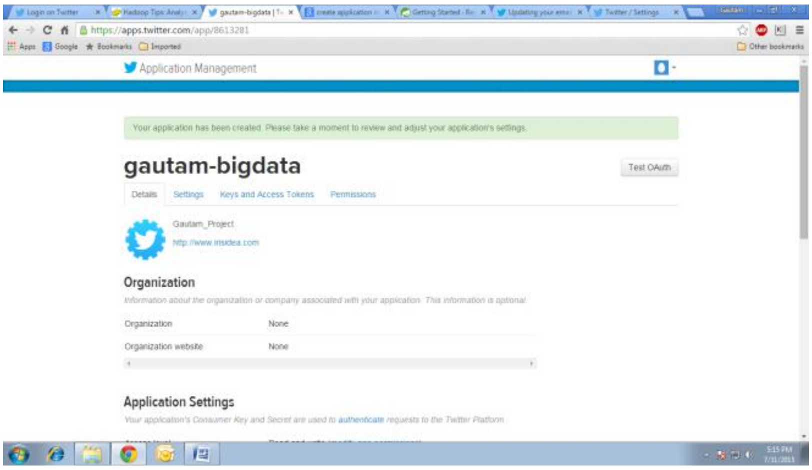
The above architectural diagram shows the complete flow of twitter streaming from Twitter service to image processing tool which is excellent enough to execute the problem statement. As an initial step, we need to collect the continuous flow of twitter streaming. To solve the problem statements, in this paper the following steps are executed:

- Create Twitter application
- Create the Hadoop cluster[1,13,14] in Amazon Web Service EC2 cloud[2].
- Connected to Twitter to receive the streaming/data which is in JSON format using Apache Flume[9]
- Configure Flume[9] with HDFS[1,13,14] so that streaming data can be stored in HDFS[1,13,14].

- Developed Map-Reduce[1,13,14] programming model. Here, to parse and filter the JSON data , only Map has been used as shown in diagram and subsequently stored in HDFS[1,13,14] in comma separated value (csv) format.
- Developed Hive User Define Function (UDF)[6] to extract the image URL from good and bad tweets and stored in file.
- Developed Java Utility class to download the images from the posted URL as mentioned above in sequential manner.
- Finally, free image processing tool Anti-Twin[10] has been used to get the statistics of images.

A. Creating Twitter Application

To retrieve the continuous flow of twitter streaming data[3], an account in twitter development portal is mandatory. In https://dev.twitter.com/apps/ an application is created and then generated the required keys. Below figure shows how the created application looks like in web browser.



https://apps.twitter.com/app/8613281

Application Management

Your application has been created. Please take a moment to review and adjust your application's settings.

gautam-bigdata

Gautam_Project

http://www.insidea.com

Organization

Information about the organization or company associated with your application. This information is optional.

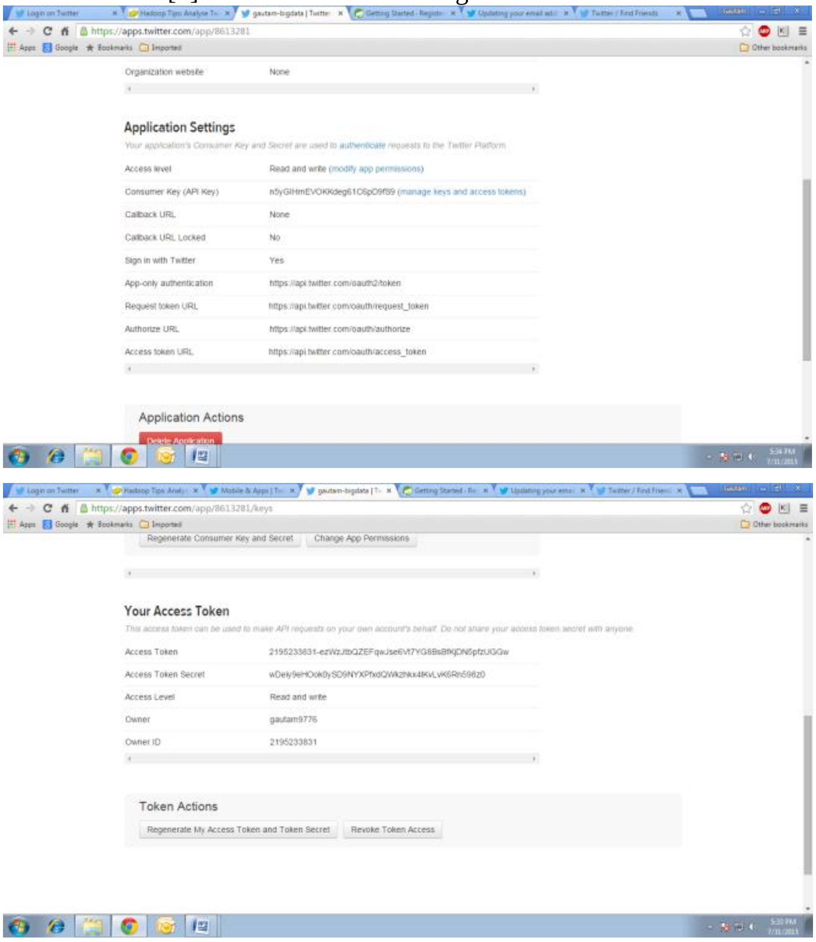
Organization	None
Organization website	None

Application Settings

Your application's Consumer Key and Secret are used to authenticate requests to the Twitter Platform.

Fig.3

From the created application, we need to retrieve all the required keys and access token which has to be provided in Flume configuration file so that Flume[9] can access the streaming data.



https://apps.twitter.com/app/8613281

Organization website	None

Application Settings

Your application's Consumer Key and Secret are used to authenticate requests to the Twitter Platform.

Access level	Read and write (modify app permissions)
Consumer Key (API Key)	n5yGIHmEVOKKdeg61O5pD9fS9 (manage keys and access tokens)
Callback URL	None
Callback URL Locked	No
Sign in with Twitter	Yes
App-only authentication	https://api.twitter.com/oauth2/token
Request token URL	https://api.twitter.com/oauth/request_token
Authorize URL	https://api.twitter.com/oauth/authorize
Access token URL	https://api.twitter.com/oauth/access_token

Application Actions

https://apps.twitter.com/app/8613281/keys

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	2195233831-ezWzJtbQZEFqwJse6Vt7YG8BsBfKjDN5pfzUGGw
Access Token Secret	wDeiy9eHOok0ySD9NYXPfxdQWk2hkx4lKvLvK6Rn59820
Access Level	Read and write
Owner	gautam9776
Owner ID	2195233831

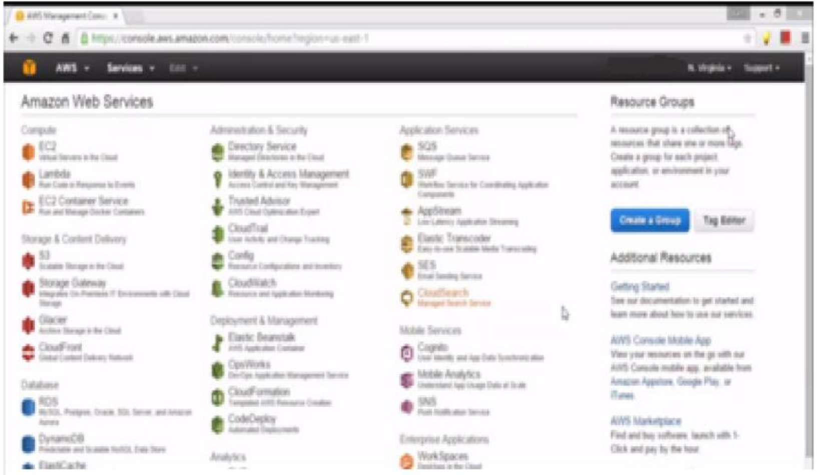
Token Actions

Fig-4

The above figure shows clearly all the generated keys and access token. We need to have the consumer key, consumer Secrete, access Token, and access Token Secret from the created application.

B. Create the Hadoop cluster in Amazon Web Service EC2 cloud

Amazon Web Service cloud EC2[2] is to leverage the complete cloud base service which eliminates the time consumption, setup and configuration of individual nodes (Name nodes, Data nodes) physically. Here with pay per hourly basis Hadoop cluster has been created. Using Ubuntu 14.04.2 two data nodes and one name node are created. Below are the few figures showing cluster creation on Amazon Web Service EC2[2].



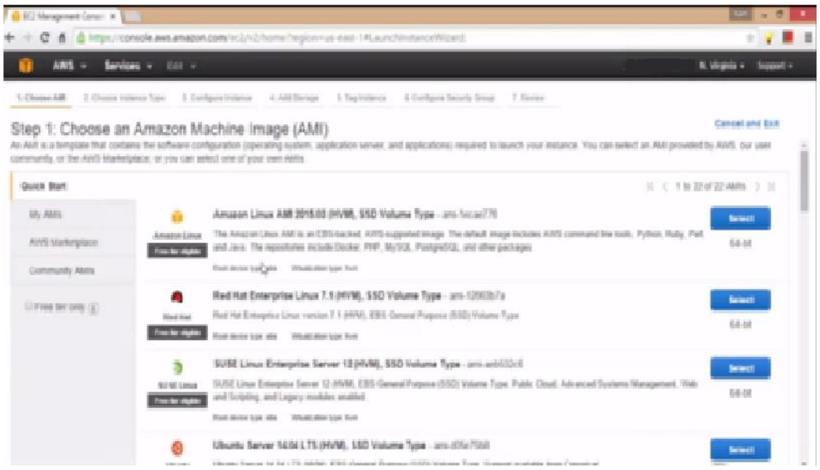
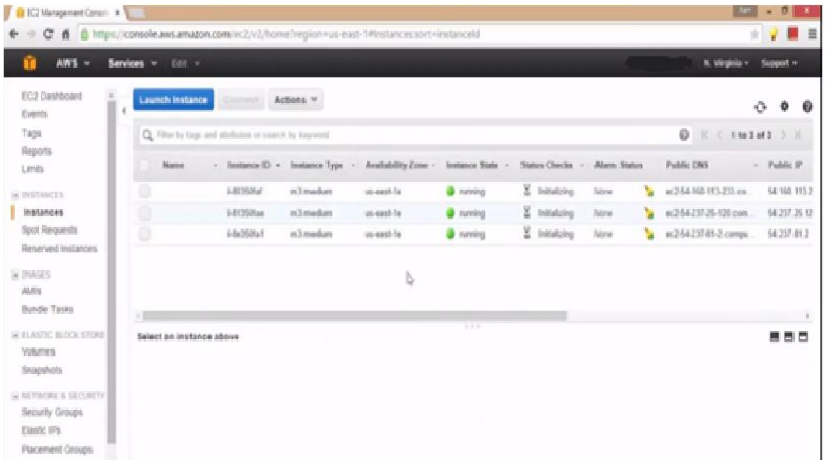

https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:
AWS
Services
Edit
N. Virginia
Support
1. Choose AMI
Step 1: Choose an Amazon Machine Image (AMI)
Cancel and Exit
Quick Start
My AMIs
AWS Marketplace
Community AMIs
Amazon Linux AMI 2015.03 (HVM), SSD Volume Type
Red Hat Enterprise Linux 7.1 (HVM), SSD Volume Type
SUSE Linux Enterprise Server 12 (HVM), SSD Volume Type
Ubuntu Server 14.04 LTS (HVM), SSD Volume Type
Select
64-bit

Fig-5

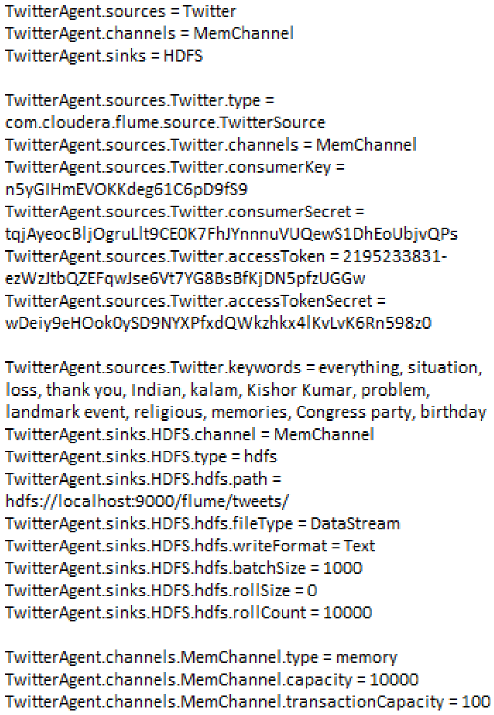


Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP
	i-80350faf	m3.medium	us-east-1a	running	Initializing	None	ec2-54-160-115-233.co...	54.160.115.2
	i-81350fae	m3.medium	us-east-1a	running	Initializing	None	ec2-54-237-26-120.com...	54.237.26.12
	i-8a350fa1	m3.medium	us-east-1a	running	Initializing	None	ec2-54-237-81-2.compu...	54.237.81.2

Fig-6

C. Twitter streaming extraction

Flume[9] has been installed and configured in Hadoop cluster[1,13,14] to retrieve the twitter streaming (data is in JSON format). Below is the Flume configuration file to connect with Twitter.



```
TwitterAgent.sources = Twitter
TwitterAgent.channels = MemChannel
TwitterAgent.sinks = HDFS

TwitterAgent.sources.Twitter.type =
com.cloudera.flume.source.TwitterSource
TwitterAgent.sources.Twitter.channels = MemChannel
TwitterAgent.sources.Twitter.consumerKey =
n5yGIHmEVOKKdeg61C6pD9fS9
TwitterAgent.sources.Twitter.consumerSecret =
tqjAyeocBljOgruLlt9CE0K7FhJYnnnuVUQewS1DhEoUbjvQPs
TwitterAgent.sources.Twitter.accessToken = 2195233831-
ezWzJtbQZEFqwJse6Vt7YG8BsBfKjDN5pfzUGGw
TwitterAgent.sources.Twitter.accessTokenSecret =
wDeiy9eHOok0ySD9NYXPfxdQWkzhkx4lKvLvK6Rn598z0

TwitterAgent.sources.Twitter.keywords = everything, situation,
loss, thank you, Indian, kalam, Kishor Kumar, problem,
landmark event, religious, memories, Congress party, birthday
TwitterAgent.sinks.HDFS.channel = MemChannel
TwitterAgent.sinks.HDFS.type = hdfs
TwitterAgent.sinks.HDFS.hdfs.path =
hdfs://localhost:9000/flume/tweets/
TwitterAgent.sinks.HDFS.hdfs.fileType = DataStream
TwitterAgent.sinks.HDFS.hdfs.writeFormat = Text
TwitterAgent.sinks.HDFS.hdfs.batchSize = 1000
TwitterAgent.sinks.HDFS.hdfs.rollSize = 0
TwitterAgent.sinks.HDFS.hdfs.rollCount = 10000

TwitterAgent.channels.MemChannel.type = memory
TwitterAgent.channels.MemChannel.capacity = 10000
TwitterAgent.channels.MemChannel.transactionCapacity = 100
```

Fig-7

D. Development of Map only job

Map[1,13,14] has been developed to parse and filter the incoming twitter stream data which is in JSON. During parsing, good and bad tweets are segregated into two different files which are stored again in HDFS[1,13,14] in csv format. Here are the few basic points to detect bad tweets based on reference[11]

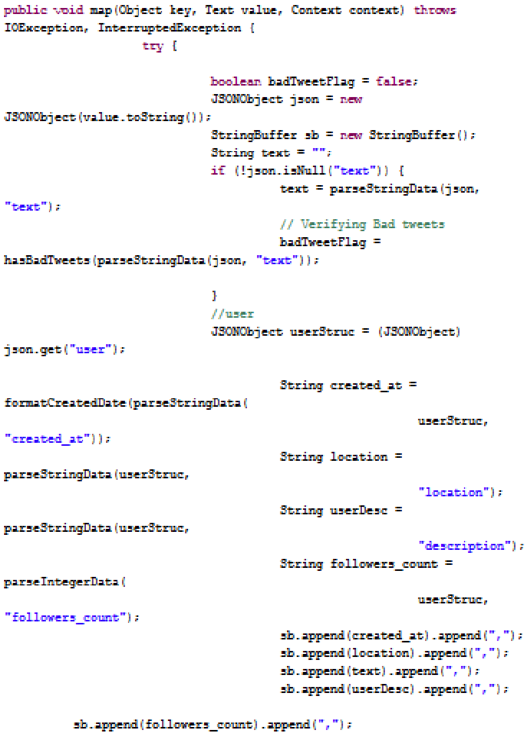
- Use RT in text (Re tweet)
- Too many hash tag(#) is obnoxious (Limit our self to 2 and not more than 3)
- Starting with @ in text, typing in all caps are rude and annoying. In the tech world, it's considered shouting.

org.json.jar is used as utility jar in java code to parse each JSON record inside Map method and it has been stored in distributed cache provided by Hadoop framework in installed cluster. Below figure depicts how raw twitter streaming data[3] in (JSON format) looks like:

{"filter_level":"low","retweeted":false,"in_reply_to_screen_name":null,"possibly_sensitive":false,"truncated":false,"lang":"en","in_reply_to_status_id_str":null,"id":628923003317825538,"extended_entities":{"media":[{"sizes":{"small":{"w":340,"resize":"fit","h":125},"thumb":{"w":150,"resize":"crop","h":150},"large":{"w":1024,"resize":"fit","h":378},"medium":{"w":600,"resize":"fit","h":221}},"id":628922815496859649,"media_url_https":"https://pbs.twimg.com/media/CLph9_DUMAE60ua.jpg","media_url":"http://pbs.twimg.com/media/CLph9_DUMAE60ua.jpg","expanded_url":"http://twitter.com/BootsPharmaJobs/status/628922815920607232/photo/1","source_status_id_str":"628922815920607232","indices":[125,140],"source_status_id":628922815920607232,"id_str":"628922815496859649","type":"photo","display_url":"pic.twitter.com/yRHjUokzC8","url":"http://t.co/yRHjUokzC8"}]},"in_reply_to_user_id_str":null,"timestamp_ms":"1438781903472","in_reply_to_status_id":null,"created_at":"Wed Aug 05 13:38:23 +0000 2015","favorite_count":0,"place":null,"coordinates":null,"text":"RT @BootsPharmaJobs: We reached a fantastic 5000 followers + we'd like to thank you all for your support #FeelGoodaboutBoots http://t.co/yR\u2026","contributors":null,"retweeted_status":{"filter_level":"low","retweeted":false,"in_reply_to_screen_name":null,"possibly_sensitive":false,"truncated":false,"lang":"en","in_reply_to_status_id_str":null,"id":628922815920607232,"extended_entities":{"media":[{"sizes":{"small":{"w":340,"resize":"fit","h":125},"thumb":{"w":150,"resize":"crop","h":150},"large":{"w":1024,"resize":"fit","h":378},"medium":{"w":600,"resize":"fit","h":221}},"id":628922815496859649,"media_url_https":"https://pbs.twimg.com/media/CLph9_DUMAE60ua.jpg**","media_url":"http://pbs.twimg.com/media/CLph9_DUMAE60ua.jpg"**,"expanded_url":"http://twitter.com/BootsPharmaJobs/status/628922815920607232/photo/1","indices":[104,126],"id_str":"628922815496859649","type":"photo","display_url":"pic.twitter.com/yRHjUokzC8","url":"http://t.co/yRHjUokzC8"}]},"in_reply_to_user_id_str":null,"in_reply_to_status_id":null,"created_at":"Wed Aug 05 13:37:38 +0000 2015","favorite_count":0,"place":null,"coordinates":null,"text":"We reached a fantastic 5000 followers + we'd like to thank you all for your support #FeelGoodaboutBoots http://t.co/yRHjUokzC8","contributors":null,"geo":null,"entities":{"trends":[],"symbols":[],"urls":[],"hashtags":[{"text":"FeelGoodaboutBoots","indices":[84,103]}],"media":[{"sizes":{"small":{"w":340,"resize":"fit","h":125},"thumb":{"w":150,"resize":"crop","h":150},"large":{"w":1024,"resize":"fit","h":378},"medium":{"w":600,"resize":"fit","h":221}},"id":628922815496859649,"media_url_https":"https://pbs.twimg.com/media/CLph9_DUMAE60ua.jpg","media_urlranslator":false}}.

Fig-8

Here are the code snipped which are used inside map method in Map[1,13,14].



```
public void map(Object key, Text value, Context context) throws
IOException, InterruptedException {
                   try {

                             boolean badTweetFlag = false;
                             JSONObject json = new
JSONObject(value.toString());
                             StringBuffer sb = new StringBuffer();
                             String text = "";
                             if (!json.isNull("text")) {
                                      text = parseStringData(json,
"text");
                                      // Verifying Bad tweets
                                      badTweetFlag =
hasBadTweets(parseStringData(json, "text"));

                             }
                             //user
                             JSONObject userStruc = (JSONObject)
json.get("user");

                                      String created_at =
formatCreatedDate(parseStringData(
                                                          userStruc,
"created_at"));
                                      String location =
parseStringData(userStruc,
                                                          "location");
                                      String userDesc =
parseStringData(userStruc,
                                                          "description");
                                      String followers_count =
parseIntegerData(
                                                          userStruc,
"followers_count");
                                      sb.append(created_at).append(",");
                                      sb.append(location).append(",");
                                      sb.append(text).append(",");
                                      sb.append(userDesc).append(",");

          sb.append(followers_count).append(",");
```

Fig-9

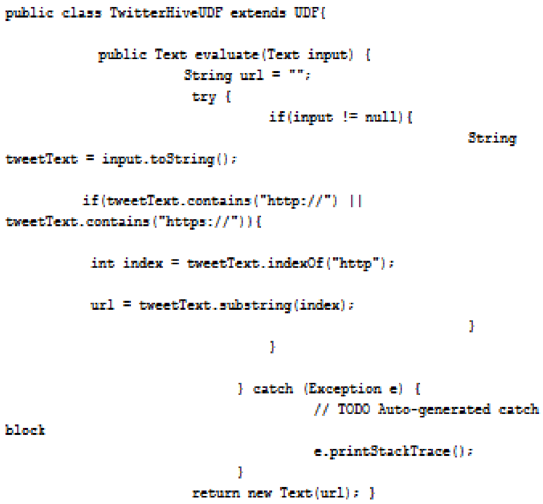
E. Loading data into Hive

In Hive, two external tables are created with the following columns where csv formatted data obtained from Map are uploaded. Use of JSONSerde[12] provided by Cloudera to load streaming data (JSON) into Hive tables directly is avoided.

- created_at
- location
- text
- media_url
- userDescription
- follower_count
- in_reply_to
- retweeted_text
- retweeted_user_location
- retweeted_user_name

F. UDF development in Hive

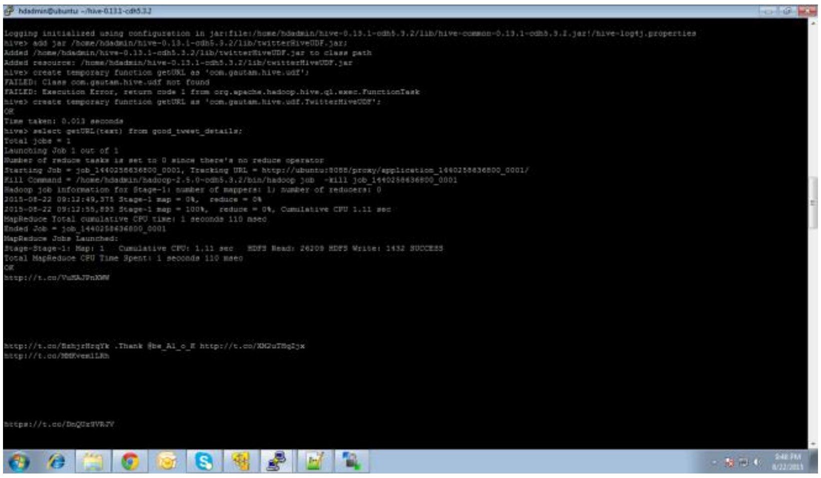
Here an user define function[5] has been developed to extract all the posted image URLs from user created tweets(both good and bad). The image URL was appended against the JSON key "media_url" in each row of JSON streaming data. In Fig 8, it is highlighted and shown as an example. Here are the code snippet of custom UDF.



```
public class TwitterHiveUDF extends UDF{

            public Text evaluate(Text input) {
                        String url = "";
                         try {
                                   if(input != null){
                                                             String
tweetText = input.toString();

          if(tweetText.contains("http://") ||
tweetText.contains("https://")){

           int index = tweetText.indexOf("http");

           url = tweetText.substring(index);
                                                             }
                                   }

                               } catch (Exception e) {
                                         // TODO Auto-generated catch
block
                                         e.printStackTrace();
                               }
                         return new Text(url); }
```

Fig-10

Hive UDF[6] has produced a file having all the URL for further processing. Below figure represents execution of UDF hive query against each Hive table and also the results .

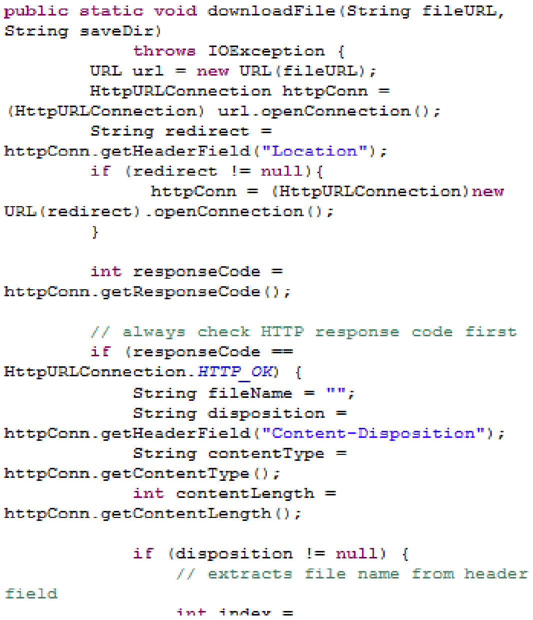


```
hdadmin@ubuntu: ~/hive-0.13.1-cdh5.3.2

Logging initialized using configuration in jar:file:/home/hdadmin/hive-0.13.1-cdh5.3.2/lib/hive-common-0.13.1-cdh5.3.2.jar!/hive-log4j.properties
hive> add jar /home/hdadmin/hive-0.13.1-cdh5.3.2/lib/twitterHiveUDF.jar;
Added /home/hdadmin/hive-0.13.1-cdh5.3.2/lib/twitterHiveUDF.jar to class path
Added resource: /home/hdadmin/hive-0.13.1-cdh5.3.2/lib/twitterHiveUDF.jar
hive> create temporary function getURL as 'com.gautam.hive.udf';
FAILED: Class com.gautam.hive.udf not found
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.FunctionTask
hive> create temporary function getURL as 'com.gautam.hive.udf.TwitterHiveUDF';
OK
Time taken: 0.013 seconds
hive> select getURL(text) from good_tweet_details;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1440258636800_0001, Tracking URL = http://ubuntu:8088/proxy/application_1440258636800_0001/
Kill Command = /home/hdadmin/hadoop-2.5.0-cdh5.3.2/bin/hadoop job  -kill job_1440258636800_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2015-08-22 09:12:49,375 Stage-1 map = 0%,  reduce = 0%
2015-08-22 09:12:55,893 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.11 sec
MapReduce Total cumulative CPU time: 1 seconds 110 msec
Ended Job = job_1440258636800_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 1.11 sec   HDFS Read: 26209 HDFS Write: 1432 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 110 msec
OK
http://t.co/VuSAJPnXWW





http://t.co/EshjrHrqYk .Thank @be_Al_o_E http://t.co/XM2uTHgZjx
http://t.co/MMKvemlLRh






https://t.co/DnQUxSVRJV
```

Fig-11

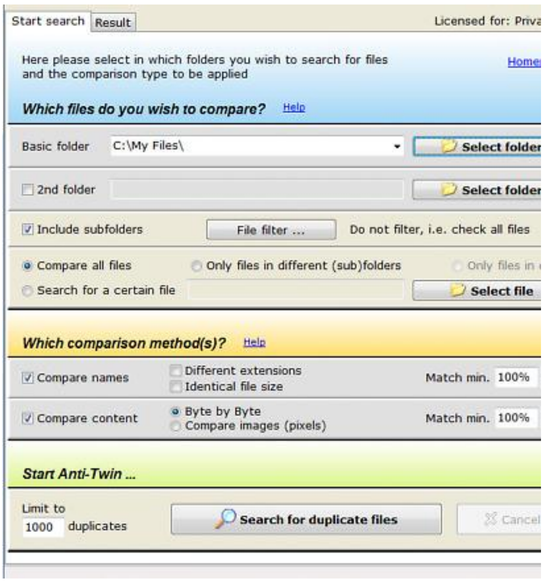
G. Image consolidation and analysis

Using Java, few utility classes are developed to download all the images in a sequential manner from the output file of Hive UDF[6] as explained above. Here is the code snippet to download images efficiently.



```
public static void downloadFile(String fileURL,
String saveDir)
            throws IOException {
        URL url = new URL(fileURL);
        HttpURLConnection httpConn =
(HttpURLConnection) url.openConnection();
        String redirect =
httpConn.getHeaderField("Location");
        if (redirect != null){
              httpConn = (HttpURLConnection)new
URL(redirect).openConnection();
        }

        int responseCode =
httpConn.getResponseCode();

        // always check HTTP response code first
        if (responseCode ==
HttpURLConnection.HTTP_OK) {
            String fileName = "";
            String disposition =
httpConn.getHeaderField("Content-Disposition");
            String contentType =
httpConn.getContentType();
            int contentLength =
httpConn.getContentLength();

            if (disposition != null) {
                // extracts file name from header
field
                int index =
```

Fig-12.



Start search | Result   Licensed for: Priva

Here please select in which folders you wish to search for files and the comparison type to be applied

Which files do you wish to compare? Help

Basic folder: C:\My Files\

□ 2nd folder

☑ Include subfolders   Do not filter, i.e. check all files

● Compare all files   ○ Only files in different (sub)folders   ○ Only files in

○ Search for a certain file

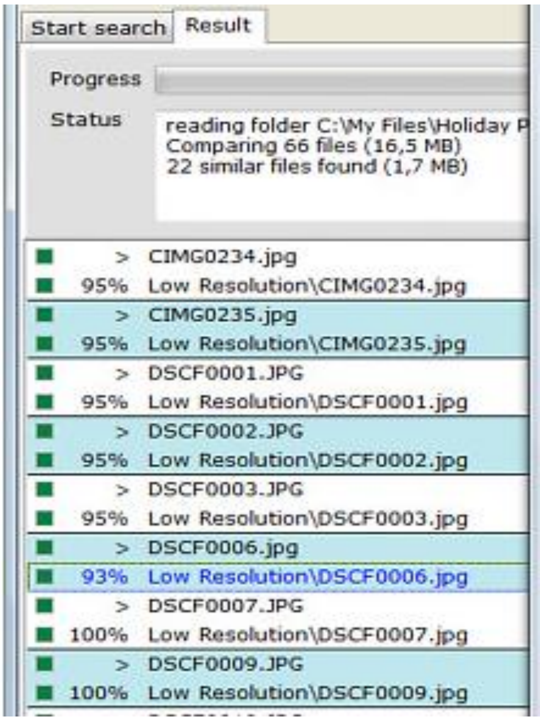
Which comparison method(s)? Help

☑ Compare names   □ Different extensions   □ Identical file size   Match min. 100%

☑ Compare content   ● Byte by Byte   ○ Compare images (pixels)   Match min. 100%

Start Anti-Twin ...

Limit to 1000 duplicates



```
reading folder C:\My Files\Holiday P
Comparing 66 files (16,5 MB)
22 similar files found (1,7 MB)

      >  CIMG0234.jpg
   95%   Low Resolution\CIMG0234.jpg
      >  CIMG0235.jpg
   95%   Low Resolution\CIMG0235.jpg
      >  DSCF0001.JPG
   95%   Low Resolution\DSCF0001.jpg
      >  DSCF0002.JPG
   95%   Low Resolution\DSCF0002.jpg
      >  DSCF0003.JPG
   95%   Low Resolution\DSCF0003.jpg
      >  DSCF0006.jpg
   93%   Low Resolution\DSCF0006.jpg
      >  DSCF0007.JPG
  100%   Low Resolution\DSCF0007.jpg
      >  DSCF0009.JPG
  100%   Low Resolution\DSCF0009.jpg
```

Fig-13

Using free image processing tool Anti-Twin[10], statistics of all the downloaded images are recorded. So in this manner, the entire paper has represented the complete analysis of shared/twitted/retwitted images.

IV. CONCLUSION

In this paper, incapability of traditional software system to process and analyze BIG DATA, has been outlined. Using Hadoop and its supporting components (Hadoop Ecosystem) on commodity hardware I have processed, analyzed a huge volume of data (BIG DATA) and achieve the desired result with more time efficiency . This methodology can be effectively adopted by any company to find the popularity analysis (a kind of advertisement) of a newly released product into market if social media is used as an alternative advertizing channel.

V. FUTURE PLAN

We shall analyze public sentiment by using various sentiment analysis tools viz. Stanford Core NLP[8] as the data dictionary, Machine Learning .

ACKNOWLEDGMENTS

I extend my gratitude and sincere thanks to my colleague Kislay Komal, Managing Director of Irisidea Technologies Pvt Ltd. and Dr. Chandrali Baishya, Asst. professor, Department of Studies & Research in Mathematics, Tumkur University, Karnataka for providing assistance and support.

REFERENCES

[1] T. White, *Hadoop The Definitive Guide: Storage and Analysis at Internet Scale*, 4th Edition O'Reilly Media, Inc., 2015.
[2] http://aws.amazon.com/articles/Amazon-EC2/
[3] https://dev.twitter.com/overview/api/tweets
[4] http://searchcio.techtarget.com/intelsponsorednews/Five-Things-to-Know-About-Hadoop-for-Managing Unstructured-Data#3593061869001
[5] http://hortonworks.com/hadoop-tutorial/how-to-process-data-with-apache-hive/
[6] http://hive.apache.org/
[7] http://jsonlint.com/
[8] http://nlp.stanford.edu/software/corenlp.shtml.
[9] http://flume.apache.org/
[10] http://www.joerg-rosenthal.com/en/antitwin/guide.html
[11] http://www.adweek.com/socialtimes/ 10-lessons-twitter-newbies/451563
[12] http://blog.cloudera.com/blog/2012/12/how-to-use-a-serde-in-apache-hive/
[13] J. R. Owens, B Femiano & J Lentz, *Hadoop Real World Solutions Cookbook*, PACKT Publishing, 2013.
[14] Russell Jurney, *Agile Data Science: Building Data Analytics Applications with Hadoop,* O'Reilly Media, Inc., 2014.

AUTHOR

Gautam is associated with Irisidea Technologies Private Limited as Technical Advisor. He is having around 14+ years of experience and in-depth knowledge on various technological platforms and business domains. Previously he worked with multinational giants like IBM India and Europe. He has expertise in design and development of financial products of Hyperion (Now Oracle). He has participated in multiple products' solution architecture of Hyperion to enhance their existing customer's knowledge on new releases and features in USA. Having very strong grip in E-Commerce domain where he has architected various E-Commerce applications to support B2C and B2B model for best retail companies based in USA and Canada. Gautam graduated in Mechanical Engineering, has completed certification in 'Comprehensive Software Project Management and Design' from Indian Institute of Science (IISc) Bangalore.