Crisis Analysis from Social Media

Using Hadoop MapReduce

Gaurav Shad
CIDSE, ASU
gshad@asu.edu

Abhishek Vellanki
CIDSE, ASU
avellank@asu.edu

Nitesh Gupta
CIDSE, ASU
ngupta40@asu.edu

***Abstract*—Social Media contains wide variety of data that can be analyzed to get some meaningful information which can be used for predictions, advertisements, marketing etc. The amount of data that needs to be analyzed is so big that its processing is inefficient on a single machine using limited resources. This problem can be addressed by merging cloud computing and Hadoop to do distributed computing of data. The goal of this project is to develop an application that analyzes twitter data related to crisis (including natural disasters, terrorist attacks, etc) and represent in an interactive format the areas that are prone to some crisis. The scope of this project includes setting up Hadoop in a private cloud cluster and implementing MapReduce to process this data. The main tasks include designing of Mapper and Reducer that will help generate some quality output.**

Keywords—Hadoop, MapReduce, FLUME, Twitter, Data Visualization, Big Insights, Spark

I. Introduction

Social media connects people around the globe and hence turned itself into a platform where people share every event happening in their lives. Twitter's ease and speed of communication has made it the heart of social networking with millions of tweets every day. Because of its popularity, it has led to the development of applications and research in various domains [1]. Disaster detection is one such domain where researchers have already worked in the past and successfully predicted their occurrences."The 2014 earthquake in Napa was detected by USGS in 29 seconds using Twitter data, likely due to the tech savvy population that dominates the area" [2]. In this project we will analyze previous tweets for chosen time period (like 5 years) relating to crisis and return end user, graphical information based on the search criteria (location or crisis or both).

The problems addressed here are computing time and scalability. By processing data in a multimode cluster using MapReduce, we are enabling parallel processing which decreases the computing time and increases the scalability.

The big data that needs to be analyzed can be unstructured which traditional databases cannot handle, query and analyze. There comes a need for a system that can handle and analyze raw unstructured data efficiently. Since this big data is increasing exponentially, if not analyzed, this data remains meaningless.To extract some information from it, requiresthis data to be processed.

Technologies used:

Apache Hadoop – It's a framework for processing large data sets.

MapReduce – It's a programming paradigm which performs highly parallelized processing of large data sets over multiple clusters of nodes.

Apache FLUME – This is a service that helps in extracting data with the use of Twitter API and storing it into HDFS.

Data Visualization – IBM Insights or Google Maps API or KDE or Apache Spark will be used on the output set to visualize it.

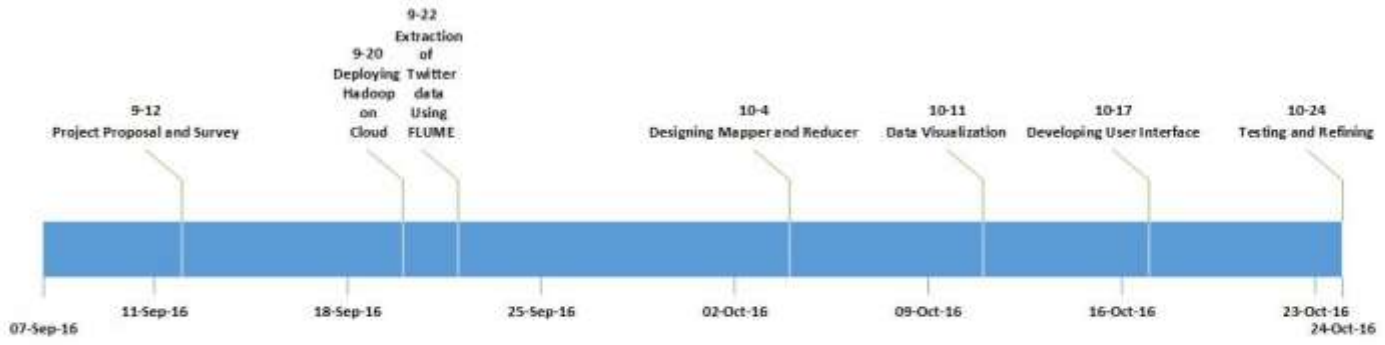
Expected Outcomes:

By working on this project, we expect to develop a data analytics application that will help visualize big data. The user interface will be developed to help end user in finding crisis prone areas in particular region.

The team comprises of three students:

1. Gaurav Shad
2. Abhishek Vellanki
3. Nitesh Gupta

We will work together in all the tasks (Tasks 1 -7) so that all team members learn everything involved in the project. Timelines for the project are as follows:



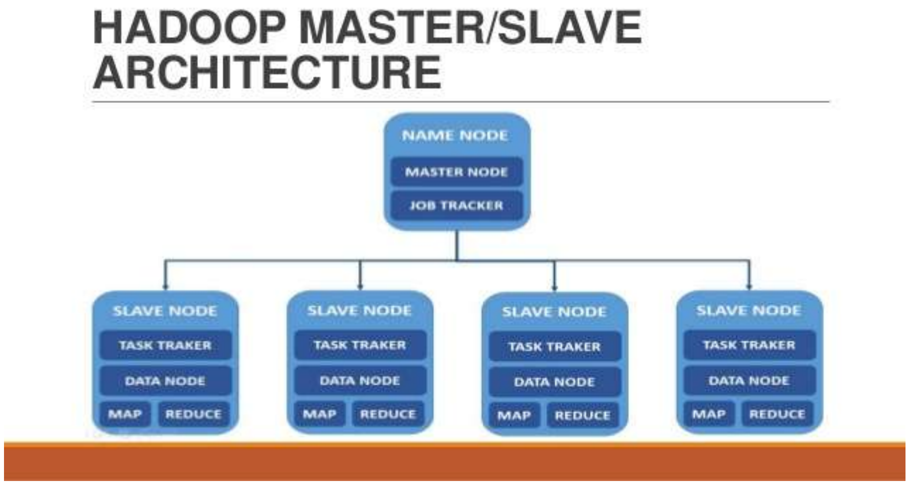
1. Timeline

II. System Models

A. System Model

The model consists of setting up a Hadoop Multi node cluster in a cloud. All the nodes, including the Master Node will reside in the cloud environment. The files need to be configured so as to interact and perform processing over different data nodes present in the cluster.

HDFS Architecture:



2. HDFS Architecture

Master Node:

- It is the master of the system which maintains and manages data on slave nodes.

Slave Node:

- These nodes are deployed for actual storage and data processing.

Job Tracker:

- It resides in the master node which determines the execution plan and manages all the tasks.

Task Tracker:

- It resides in all the slave nodes which keeps track on the task and keeps job tracker updated.

B. *Software*

1. Apache Hadoop

   It's an open source framework for storage and processing of large data sets. It consists of four modules:

   - Hadoop Common
   - Hadoop Distributed File System (HDFS)
   - Hadoop YARN
   - Hadoop Map Reduce

2. Open Stack (modified)

   It's an open source cloud computing platform for public and private clouds. It lets user deploy virtual machines and other instances that handle different tasks for managing the cloud environment.

   Open Stack is not required for our project since the major task is performed by the Hadoop Yarn (Resource Manager) i.e. parallel processing. Initially we thought that we might require open stack capabilities to dynamically control the nodes (name and data) which later on we found that Hadoop YARN can take care of.

3. Apache FLUME

   It helps in efficiently collecting and moving large amounts of streaming data directly into HDFS. In our project, it will build a connection by using Twitter API to collect data.

4. Twitter API

   It helps to extract tweets based on a specific query. It can restrict the number of tweets to the given language, location and number of tweets allowed per page.

5. Data Visualization Tool

   We will use Big Insights or Spark to visualize the data that is received from MapReduce.

III. Project Description

The project concentrates on the data set retrieved from Twitter based on time filtering (like last 5 years) and process this data to represent end user how prone a region is to a crisis. Region can be a city, state or country.Performing crisis analysis on tweets can be trickier than normal text or paragraph because tweets are short messages which contain hash tags, slangs etc.

The main task of this implementation is to automatically filter which tweets that contain the crisis information required in this project. The MapReduce will use certain conditions to perform this task. In a research done by USGS National Earthquake Information Center (NEIC), they built similar kind of analytics application by filtering tweets on the following conditions [2]:

~~1. Remove the tweets containing more than seven words~~

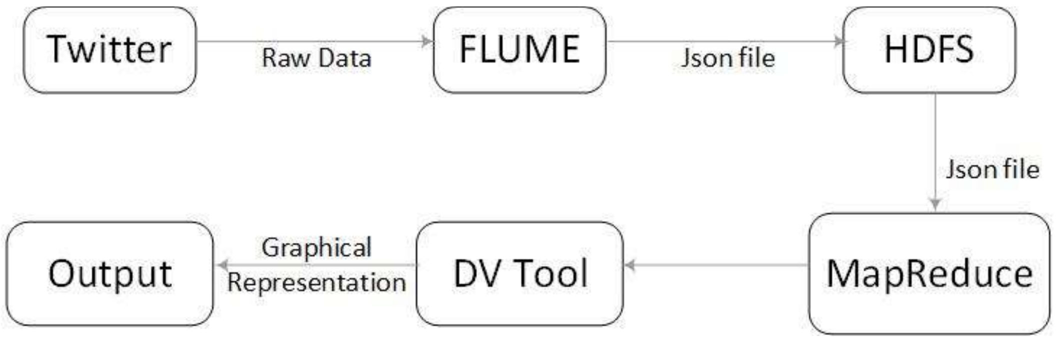
~~2. Remove the tweets that contains any links~~

1. The tweet needs to contain less than 10 words

(People caught up in a crisis don't write long tweets).

2. The tweet should not contain a link to any website
   (Someone might just be sharing an article).
3. It should not be a retweet
   We only need the original tweets to get accurate location of the crisis).
4. Tweets containing some keywords need to be ignored, for example, prayer, RIP, Magnitude etc.
   (People tweeting for others with the keywords mentioned. Magnitude won't be known by the people in crisis).
5. To get the location there are two ways if the geo feature is enabled, we will directly get the latitude and longitude or else, we will store the full name from the place parameter.

We will use the similar conditions to perform basic filtering and generalize it for all types of crises. After the basic filtering, we will apply more conditions to remove the tweets that contain crisis information of some other location. Finally, the processing will extract geo locations and data visualization tool will help build a graphical representation of it. And the end user will use the user interface of this application.



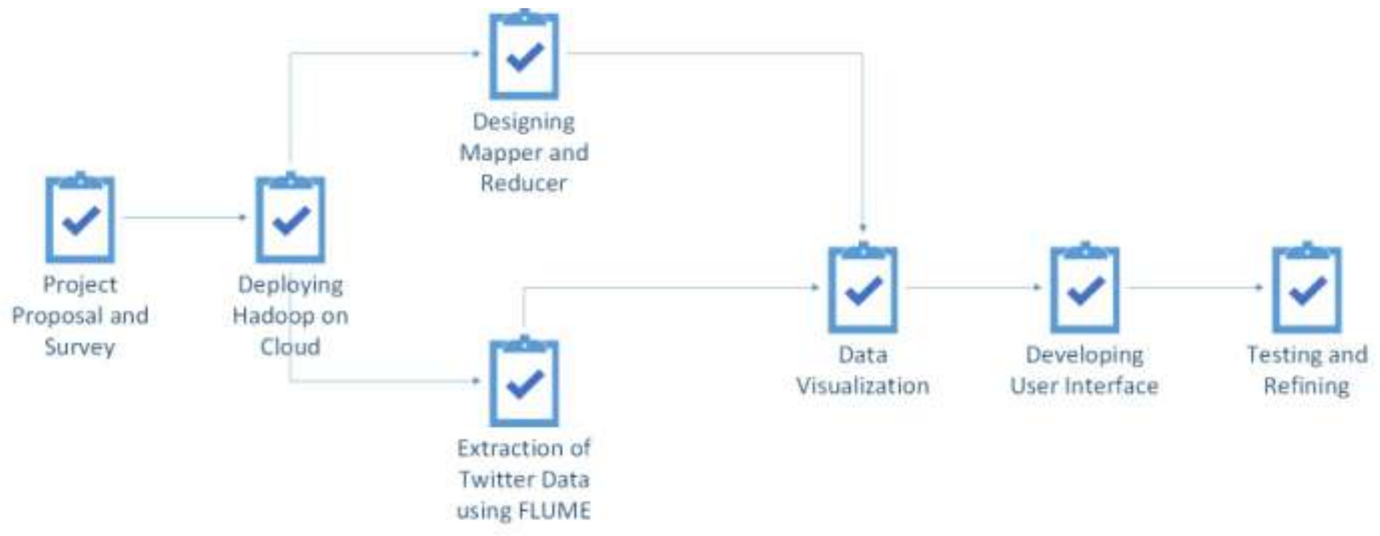
3. Workflow Diagram

A. Project Overview

The project is implemented in 6 tasks. **First task** is to install and configure Hadoop in a private cloud. **Second task** is to retrieve data from Twitter using Apache FLUME and sink this data to HDFS. **Third task** is to design Mapper and Reducer to process the data. **Fourth task** is to visualize data generated in the last task. **Fifth task** is to develop a user interface for better communication. **Sixth task** is to test the application and refine it.

Mid-Term goal for this project is to complete till task 3, designing Mapper and Reducer. We will use these Mapper and Reducer to analyze data and return outputin a readable format.

Final-Term goal for this project is to complete all the tasks that involve generating a final graphical representation of the output and a user interface.


Project Proposal and Survey
Deploying Hadoop on Cloud
Designing Mapper and Reducer
Extraction of Twitter Data using FLUME
Data Visualization
Developing User Interface
Testing and Refining

4. Task Dependencies

B. Task 1 : Deploying Hadoop on cloud

This task involves setting up a working environment for this project. It involves understanding Hadoop and its configuration files and making a multi node cluster on a private cloud which is the main requirement to process the data.

C. Task 2: Extracting Twitter data using FLUME

This task involves collecting tweets using FLUME. FLUME provides this service by establishing a connection with the help of twitter API for collecting streaming data. This data will further be used by Mapper and Reducer.

D. Task 3: Designing Mapper And Reducer

This task involves designing of Mapper and Reducer which will filter tweets based on certain conditions similar to the ones used by NEIC (National Earthquake Information centre) to improve the efficiency is identification of crisis. This part will also filter out the tweets that are generated from a location that is different from the crisis location. This will give only those tweets that further needs to be processed for determining the type of crisis and its geo location.

E. Task 4: Data Visualization

This task involves using the data generated in the previous step to be represented on a graph, or a heat map by using some kind of data visualization tools like Big Insights or Spark. The Reducer will generate the data in such a format that can be directly given as an input to the above mentioned tools.

F. Task 5: Developing User Interface

This task involves creation of user interface which will enable the user to just select a region or a crisis and see the end results in graphical format. If the user enters a region, it will display details for all crises that happened in that region. And if the user enters a crisis, it will return regions.

G. Task 6: Testing and Refine

This task involves testing of the developed application for performance and efficiency testing as well as any potential bugs.

H. Project Task Allocation

	Gaurav Shad	Nitesh Gupta	Abhishek Vellanki
Deploying Hadoop on Cloud	34%	33%	33%
Extraction of Twitter Data using FLUME	33%	33%	34%

Designing Mapper and Reducer	33%	34%	33%
Data Visualization	33%	33%	34%
Developing User Interface	33%	34%	33%
Testing and Refining	34%	33%	33%

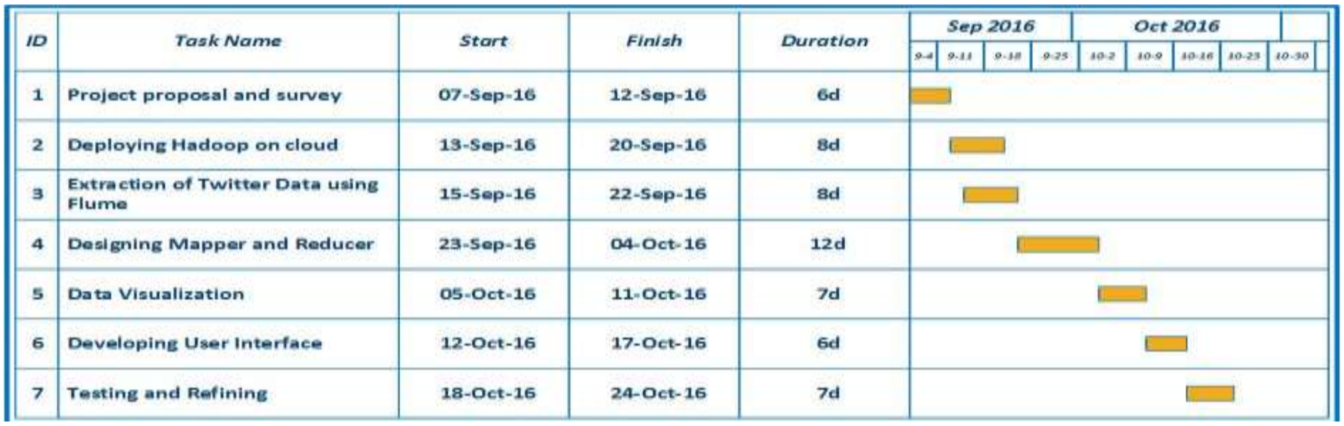
5. Task Allocation

I. *Deliverables*

Outcomes of the project

- ➢ Working Hadoop Map Reduce Environment in a private cloud.
- ➢ User Interface designing to receive input from user.
- ➢ An application of big data analysis using twitter data.
- ➢ Data Visualization on processed data

J. *Project Timeline*



ID	Task Name	Start	Finish	Duration
1	Project proposal and survey	07-Sep-16	12-Sep-16	6d
2	Deploying Hadoop on cloud	13-Sep-16	20-Sep-16	8d
3	Extraction of Twitter Data using Flume	15-Sep-16	22-Sep-16	8d
4	Designing Mapper and Reducer	23-Sep-16	04-Oct-16	12d
5	Data Visualization	05-Oct-16	11-Oct-16	7d
6	Developing User Interface	12-Oct-16	17-Oct-16	6d
7	Testing and Refining	18-Oct-16	24-Oct-16	7d

6. Gantt Chart

PROJECT PROGRESS

Project Status: Till Midterm

GIT Link: http://gitlab.thothlab.org/gshad/StormRiders-Crisis-Analysis.git

Project has progressed as planned initially during the project proposal. Tasks 1, 2 and 3 have been completed successfully.

Task Name	Status
Project Proposal and Survey	Complete
Deploying Hadoop on Cloud	Complete
Extraction of Twitter Data using Flume	Complete
Designing Mapper And Reducer	Complete(Add-On Required)
Data Visualization	Yet to start
Developing User Interface	Yet to start
Testing And Refining	Yet to start

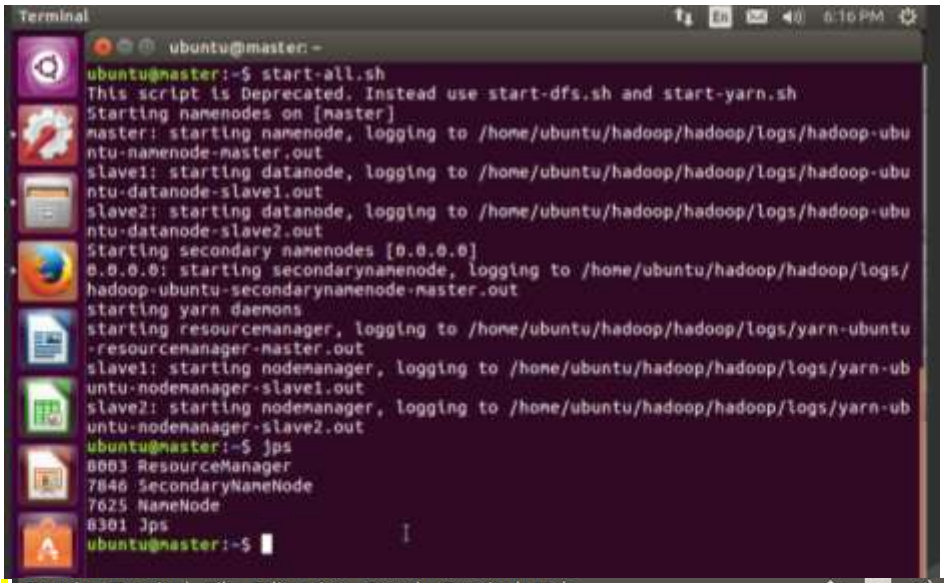
Tasks Completed
Task 1:
Deploying Hadoop on Cloud
In this task we setup Hadoop on private cloud in Thoth Lab and got it working. For this we understood how Hadoop works and then configured the XML files accordingly and setup the HDFS.

Task 2:
Extraction of Twitter data using FLUME
In this task we collected Twitter data using Twitter APIs. Twitter has 2 types of API to collect their data. The Streaming API gives us tweets in real-time. As the tweets are posted in twitter they get pushed into the HDFS. Apache FLUME has been used to integrate with this streaming API. After FLUME was installed, its configuration files were edited accordingly. It would push twitter data into the HDFS based on the parameters mentioned in its configuration files.
Since our application also requires historical data, we used a REST API by twitter to get some old twitter data as proof of concept. We used a Python Script to connect to the REST API which will search data from last 7 days.

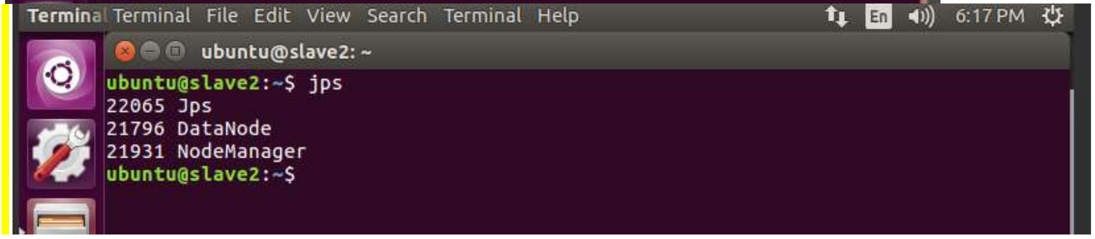
Task 3:
Designing Mapper and Reducer
We designed a MapReduce which is the important part of Hadoop framework to process the tweets. The tweets will be filtered out in the Mapper class. The Mapper class will give output as key value pairs which then be combined and used by Reducer class to generate final output.

Tasks detailed walkthrough
Deploying Hadoop on Cloud:
- Rename the nodes as master, slave1 and slave 2
- Edit the etc/hosts file on all nodes to mention ip address of each node with hostname
- setup **ssh**: generate **rsa keys** on master and transfer it to other data nodes to configure key based login in cluster
- Install Hadoop-2.7.3 on master node
  (Hadoop can be downloaded from Apache Software Foundation as tar file)
- Set up Hadoop environment variables in **bashrc file**
- Edit Hadoop configuration files in /etc/Hadoop

  - ➢ Edit **core-site.xml** to enter details of master node used for Hadoop instance
  - ➢ Edit **hdfs-site.xml** to enter details about replication, namenode and datanode
  - ➢ Edit **yarn-site.xml** to enter details about resource manager and node manager to configure yarn into Hadoop
  - ➢ Edit **mapred-site.xml** to enter details about job tracker and mapreduce framework
  - ➢ Edit **Hadoop-env.sh** to enter environment variables
- Transfer Hadoop files from master to all slave nodes using **scp**
- Create masters and slaves file in etc/Hadoop to enter hostnames
- Format namenode on master
- Start all the services and Hadoop is all set to go



```
ubuntu@master:~$ start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [master]
master: starting namenode, logging to /home/ubuntu/hadoop/hadoop/logs/hadoop-ubu
ntu-namenode-master.out
slave1: starting datanode, logging to /home/ubuntu/hadoop/hadoop/logs/hadoop-ubu
ntu-datanode-slave1.out
slave2: starting datanode, logging to /home/ubuntu/hadoop/hadoop/logs/hadoop-ubu
ntu-datanode-slave2.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/ubuntu/hadoop/hadoop/logs/
hadoop-ubuntu-secondarynamenode-master.out
starting yarn daemons
starting resourcemanager, logging to /home/ubuntu/hadoop/hadoop/logs/yarn-ubuntu
-resourcemanager-master.out
slave1: starting nodemanager, logging to /home/ubuntu/hadoop/hadoop/logs/yarn-ub
untu-nodemanager-slave1.out
slave2: starting nodemanager, logging to /home/ubuntu/hadoop/hadoop/logs/yarn-ub
untu-nodemanager-slave2.out
ubuntu@master:~$ jps
8003 ResourceManager
7846 SecondaryNameNode
7625 NameNode
8301 Jps
ubuntu@master:~$
```



```
ubuntu@slave2:~$ jps
22065 Jps
21796 DataNode
21931 NodeManager
ubuntu@slave2:~$
```

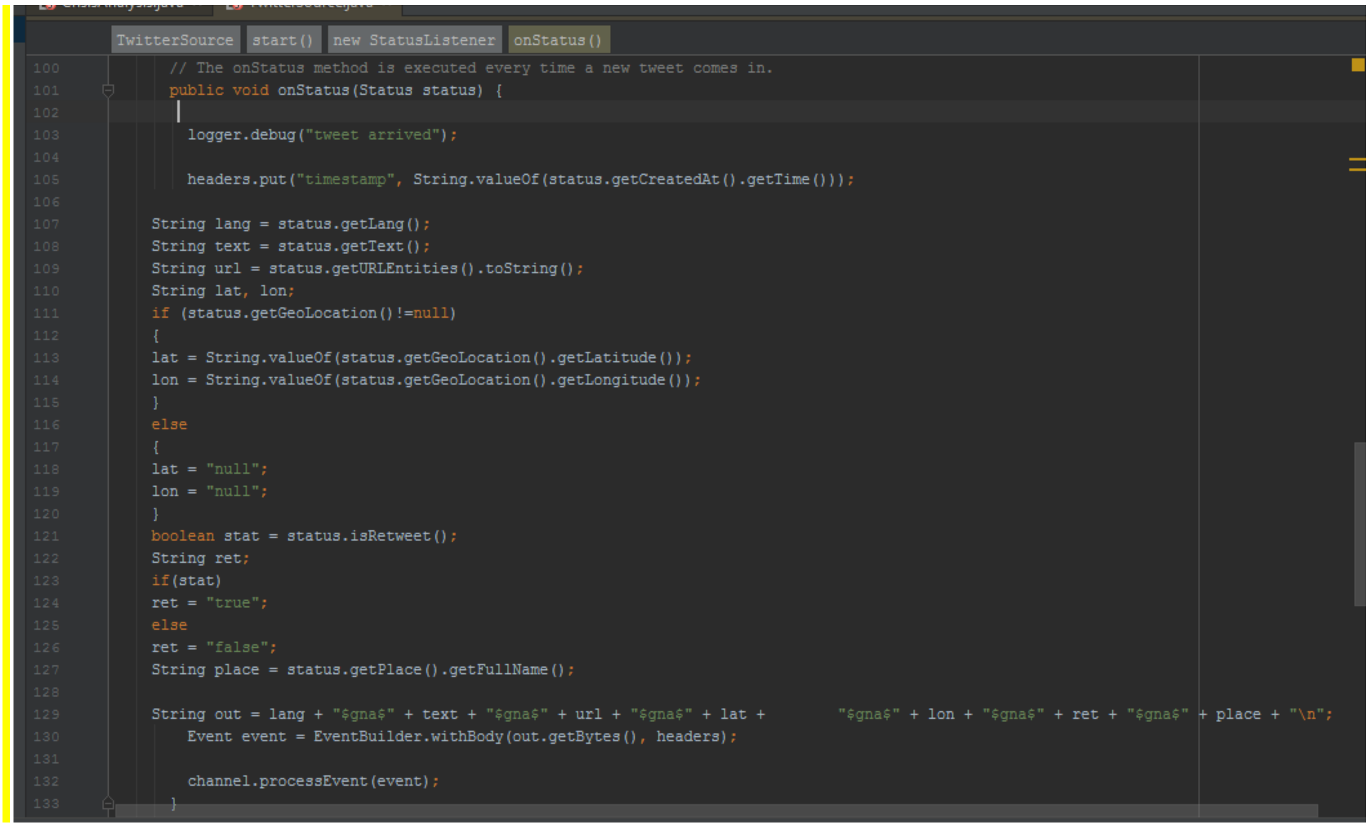
Extraction of Twitter data using FLUME
FLUME was setup on our Master node, it pushed the tweets into the hdfs path we mention in its configuration files. FLUME connects to the Steaming API of twitter and pushed tweets in real-time as they are published on Twitter.

Steps to setup FLUME:

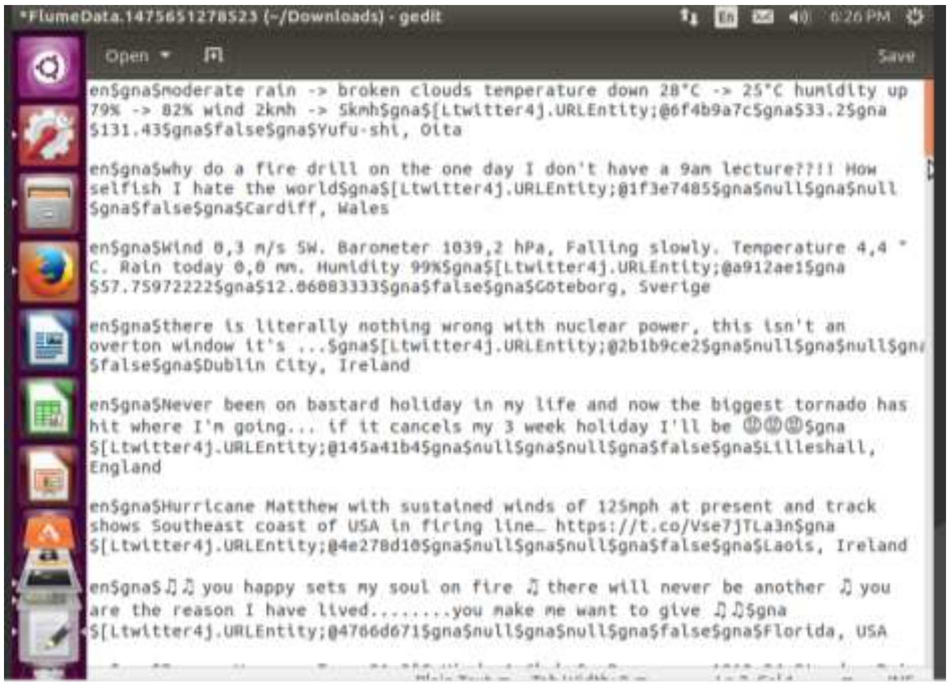
- Install Apache FLUME
  FLUME can be downloaded from the Apache website as a tar file
- Create an app on **apps.twitter.com** and get the Authentication tokens. There are 4 tokens Consumer key, Consumer Secret, Access Token, Access Secret Token
- Extract the FLUME tar file
- Edit flume.env.sh.template file and mention the Java path and save as **flume.env.sh**
- Create **twitter.conf** file in the conf folder, FLUME works as per this configuration file
  And mention details about the source, channel and sink
- Place **FULME-sources-1.0SNAPSHOT.jar**, **twitter-4j-core-4.0.5.jar**, twitter4j-stream-4.0.5.jar, twitter4j-media-support-4.0.5.jar in the lib folder inside the flume directory
- Run FLUME
- Tweets generated this way will be stored in the hdfs directory as mentioned in the twitter.conf

We require specific data from the tweets coming through flume i.e. language, text of tweet, url in tweet (if it exist), latitude, longitude, retweet status and place (will be used in case if there is no geo location).

To get this specific data we require to modify the **TwitterSource.java**, compile it and generate TwitterSource.class and TwitterClass$1.class. This file is uploaded in gitlab along with the TwitterSource.java file. After editing this file, we get data in text file with the required fields.



```
        // The onStatus method is executed every time a new tweet comes in.
        public void onStatus(Status status) {

          logger.debug("tweet arrived");

          headers.put("timestamp", String.valueOf(status.getCreatedAt().getTime()));

      String lang = status.getLang();
      String text = status.getText();
      String url = status.getURLEntities().toString();
      String lat, lon;
      if (status.getGeoLocation()!=null)
      {
      lat = String.valueOf(status.getGeoLocation().getLatitude());
      lon = String.valueOf(status.getGeoLocation().getLongitude());
      }
      else
      {
      lat = "null";
      lon = "null";
      }
      boolean stat = status.isRetweet();
      String ret;
      if(stat)
      ret = "true";
      else
      ret = "false";
      String place = status.getPlace().getFullName();

      String out = lang + "$gna$" + text + "$gna$" + url + "$gna$" + lat +        "$gna$" + lon + "$gna$" + ret + "$gna$" + place + "\n";
          Event event = EventBuilder.withBody(out.getBytes(), headers);

          channel.processEvent(event);
        }
```



```
en$gna$moderate rain -> broken clouds temperature down 28°C -> 25°C humidity up
79% -> 82% wind 2kmh -> 5kmh$gna$[Ltwitter4j.URLEntity;@6f4b9a7c$gna$33.2$gna
$131.43$gna$false$gna$Yufu-shi, Oita

en$gna$why do a fire drill on the one day I don't have a 9am lecture??!! How
selfish I hate the world$gna$[Ltwitter4j.URLEntity;@1f3e7485$gna$null$gna$null
$gna$false$gna$Cardiff, Wales

en$gna$Wind 0,3 m/s SW. Barometer 1039,2 hPa, Falling slowly. Temperature 4,4 °
C. Rain today 0,0 mm. Humidity 99%$gna$[Ltwitter4j.URLEntity;@a912ae1$gna
$57.7597222$gna$12.0608333$gna$false$gna$Göteborg, Sverige

en$gna$there is literally nothing wrong with nuclear power, this isn't an
overton window it's ...$gna$[Ltwitter4j.URLEntity;@2b1b9ce2$gna$null$gna$null$gna
$false$gna$Dublin City, Ireland

en$gna$Never been on bastard holiday in my life and now the biggest tornado has
hit where I'm going... if it cancels my 3 week holiday I'll be ☹☹☹$gna
$[Ltwitter4j.URLEntity;@145a41b4$gna$null$gna$null$gna$false$gna$Lilleshall,
England

en$gna$Hurricane Matthew with sustained winds of 125mph at present and track
shows Southeast coast of USA in firing line… https://t.co/Vse7jTLa3n$gna
$[Ltwitter4j.URLEntity;@4e278d10$gna$null$gna$null$gna$false$gna$Laois, Ireland

en$gna$♫♫ you happy sets my soul on fire ♫ there will never be another ♫ you
are the reason I have lived........you make me want to give ♫♫$gna
$[Ltwitter4j.URLEntity;@4786d671$gna$null$gna$null$gna$false$gna$Florida, USA
```

Extraction of Twitter data using REST API

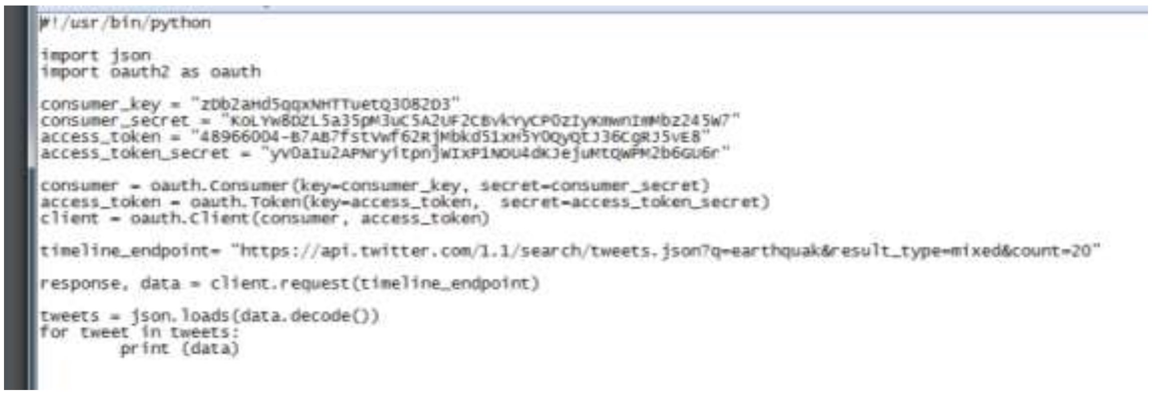
The REST API GET search/tweets can also be used for collection of tweets as an alternative. The REST API required the above mention tokens as well, the same tokens can be used.

To access this API, a python script was used. The script contained the 4 tokens and the API request.

The API allows us to mention the following parameters

q - the keywords based on which tweets are to be extracted
geocode - returns tweets of users in a certain geographical radius
lang – restricts tweets to given language
locale - language of the query we are sending
result_ type - allows us to decide if the tweets collected need to be recent, mixed or old
count - the number of tweets to be returned per page
until – returns tweets created before the given date
since_id - returns tweets with a result ID greater than the ID specified
max_id – returns tweets with an ID less than or equal to specified ID
include_entitites - the entities will be excluded when set to false
callback - if supplied the response will use the JSONP format with a callback of the given name

The output file with the required tweets and its attributes will be in json format and will be put in the directory mentioned in the Python Script.



```
#!/usr/bin/python

import json
import oauth2 as oauth

consumer_key = "zDb2aHd5qqxNHTTuetQ308203"
consumer_secret = "KoLYwBDZL5a35pM3uC5A2UF2CBvkYyCP0zIyKmwnImMbz245W7"
access_token = "48966004-B7AB7fstVwf62RjMbkd51xM5Y0QyQtJ36CgRJ5vE8"
access_token_secret = "yvOaIu2APNry1tpnjWIxP1NOU4dKJejuMtQWPM2b6GU6r"

consumer = oauth.Consumer(key=consumer_key, secret=consumer_secret)
access_token = oauth.Token(key=access_token,  secret=access_token_secret)
client = oauth.Client(consumer, access_token)

timeline_endpoint= "https://api.twitter.com/1.1/search/tweets.json?q=earthquak&result_type=mixed&count=20"

response, data = client.request(timeline_endpoint)

tweets = json.loads(data.decode())
for tweet in tweets:
        print (data)
```

Designing Mapper and Reducer

Mapper and Reducer are working on top of the text file generated by flume.

Mapper:

The mapper class reads data line by line from text file generated by flume.
It then extracts all the parameters language, text, url, latitude, longitude, retweet_status and place.

It then filters the tweets as per the following conditions:
- Language should be English
- Retweet_status should be false
- Word count should be less than 10
- No url should be there in text
- No words like prayer, RIP etc should be present

If a tweet satisfies all these conditions, then it will consider it for analysis.
It will check what crises are mentioned in that tweet using a dictionary given and give the location and name of crisis as key value pair to output collector

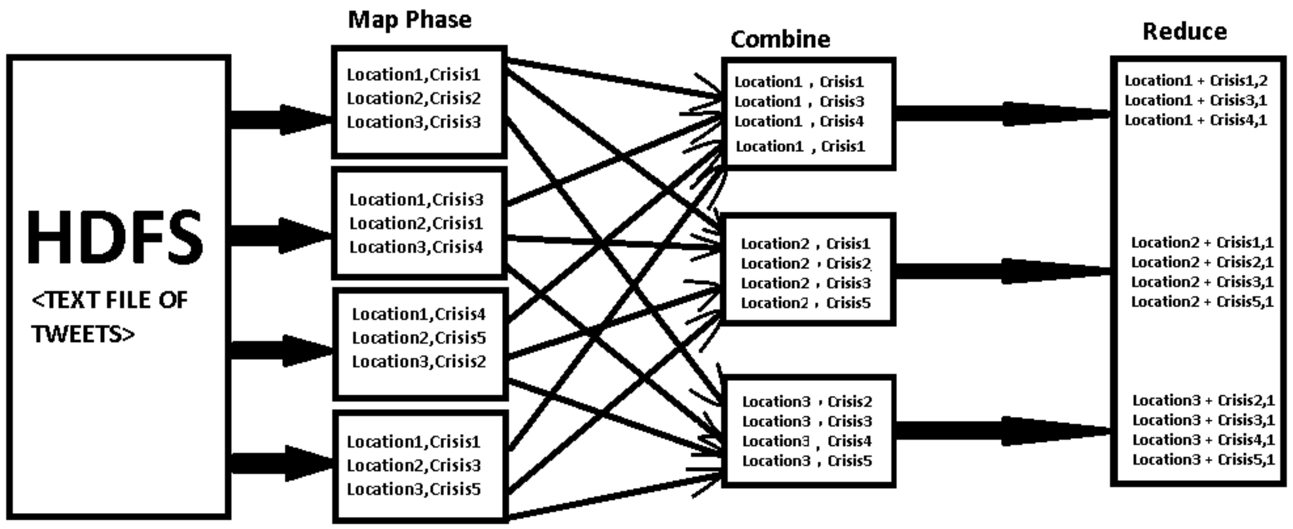
Output format: <location, crisis_name>

Reducer:

The reducer will use the output generated by mapper.
It will take key as a location and iterate over all the values(crisis) happened at that location.
It will use a hash map to temporarily store the count for each crisis happened at that location.
It will finally give location name, crisis name and count to output collector.

Output format: <location + crisis name, count>

The output generated tells us for each location, how many times a particular crisis has happened there.



Challenges and Roadblocks

1) **Historical Data:**
Twitter does not provide complete historical data with its APIs. At most one-week old data can be collected using the REST API.
As historical data is critical for our project we have to look for a twitter data dump posted by anyone online.
We found such data by archieve.org

2) **Configuring flume to give us the specific data:**
Accessing the json particular fields was a bottle neck since we didn’t need to analyze whole data from json. For this purpose, we had to overwrite the TwitterSource.java where we added code to return us only the data that we require. After successful compilation of the java we had two class files which were put into the flume-source-1.0-SNAPSHOT.jar.

To compile the TwitterSource.java it required to import packages like org. apache. flume, twitter4j, org. json. simple and slf4j after all the packages were locally added then we built the java file for class file generation.

Comments Addressed:

Comment 1: Why is Open stack required in your project? Will the results be different in a Map reduce environment without involvement of Open stack?

Reply: Open Stack is not required for our project since the major task is performed by the Hadoop Yarn (Resource Manager) i.e. parallel processing. Initially we thought that we might require open stack capabilities to dynamically control the nodes (name and data) which later on we found that Hadoop YARN can take care of.

Comment2:

a) How are cloud features such as scalability, on-demand etc. incorporated by project?

Reply: Hadoop supports distributed data storage and processing for big data sets across nodes in multiple clusters. Hadoop Common with the core libraries and utilities; the Hadoop Distributed File System (HDFS) for storing data on distributed machines; Hadoop YARN for managing and scheduling cluster computing resources; and Hadoop MapReduce, the programming language for big data processing.

b) What is size of data-set used?

Reply: Size of data set is variable since we are analysing data through flume that gives the live tweets which may vary from few megabytes to gigabytes.
However, the static data set that is json format is around 30 gigabytes.This data set is pretty big to prove the abilities of cloud computing processing.

c) What specific features are being used to identify crisis information?

Reply: To make the results accurate we need to filter out the tweets that won't tell us about the crisis that actually happened at that particular location. Some of the parameters we are using are same as parameters used by the NEIC (National Earthquake Information Center) and others are modified and created as per our needs. The parameters we established are:

- The tweet needs to contain less than 10 words
  (People caught up in a crisis don't write long tweets).
- The tweet should not contain a link to any website
  (Someone might just be sharing an article).
- It should not be a retweet
  (We only need the original tweets to get accurate location of the crisis).
- Tweets containing some keywords need to be ignored, for example, prayer, RIP, Magnitude etc.
  (People tweeting for others with the keywords mentioned. Magnitude won't be known by the people in crisis).
- To get the location there are two ways if the geo feature is enabled, we will directly get the latitude and longitude or else, we will store the full name from the place parameter.

d) Can the quality/trust of information related to crisis be justified using your analysis algorithm?

Reply: People have started to post their entire lives on social media, the food they eat, places they visit etc. It is safe to assume that a lot of people will tweet if their neighborhood gets hit by a crisis.

The results being produced from the application are as accurate as they can be. Some of the parameters of the application to filter tweets are based on the parameters developed by the NEIC in their research on the same topic. We added a couple of filters of our own to make the results more optimal.

However, the precision and reliability of this application cannot be used to get an accurate search on a crisis as the Twitter API do not give access to every tweet on their website. The streaming API gives access to only 1% of the tweets.

IV. Risk Management of the Project

Risk	**Mitigation Strategy**
Hadoop installation and configuration is a tricky. Issue can come up with the configuration of xml files.	To solve this issue, installation must be done with proper time and care.
Lack of large data sets from Twitter because of rate limit of Twitter API	Twitter API must be run in parallel and flume must be used to get more data
Distinguishing between normal tweets and useful tweets (crisis information)	Conditions needs to be improved so as filter out the data precisely

V. Conclusion

To summarize, this project results in data visualization from the output of MapReduce algorithm. It returns us a graphical representation of the crisis that already happened in a region, queried in by the user. The tweets are streamed using Hadoop, Flume and Twitter API. Mapper and Reducer will be used to filter out the tweets based on certain conditions and process the required data. A user interface will help user interact with this big data analysis application to gain knowledge on this topic.

Futurework:
In the Mapper and Reducer, K-means or some other clustering method can be implemented to improve the quality of data and efficiency of the application. Additionally, data from other social media like Facebook can also be extracted and processed to return more meaningful data.

Acknowledgment

We would like to express our gratitude towards our professor, Dr. Dijiang Huang and Teaching Assistant Mr. Ankur Chowdhary for assisting and giving us valuable inputs so that we could deliver it in time.

References

[1] Shamanth Kumar, Fred Morstatter, Huan Liu "Twitter Data Analytics" http://tweettracker.fulton.asu.edu/tda/TwitterDataAnalytics.pdf

[2] “How the USGS uses Twitter data to track earthquakes”
https://blog.twitter.com/2015/usgs-twitter-data-earthquake-detection

[3] “How-to: Analyze Twitter Data with Apache Hadoop”
http://blog.cloudera.com/blog/2012/09/analyzing-twitter-data-with-hadoop/

[4] Manoj Kumar Danthala, Dr. Siddhartha Ghosh "Bigdata Analysis: Streaming Twitter Data with Apache Hadoop and Visualizing using BigInsights"
https://www.ijert.org/view-pdf/13162/bigdata-analysis-streaming-twitter-data-with-apache-hadoop-and-visualizing-using-biginsights

[5] “Hadoop” http://hadoop.apache.org/

[6] “FLUME” http://flume.apache.org/

[7] “Open Stack” http://www.openstack.org/

[8] “Twitter Developers API” https://dev.twitter.com/docs/api/1.1/get/search/tweets