**Project Summary**

Analyze the popularity and public visibility of the leading candidates for the Presidential election 2016, utilizing real time data from Twitter. The analysis employed a distributed data processing system known as Apache Hadoop using several EC2 instances on Amazon Web Services

**Description**

**1. Download large amount of data**

1.1]

The advent of twitter allows us to mine and analyze Tweets at scale, in real time, in order to make predictions and gain insights on a variety of topics. Twitter has made available a streaming API which allows us to gather tweets in real time. The data, in addition to the tweet, also includes an array of supplementary information regarding the user of the tweet. The particular features from this considerable assortment of data that we will be concerned with are:

1. Status – The actual tweet status

2. Follower Count – The number of followers of the user who posted the tweet.

3. Retweet Count – The number of times that particular tweet was retweeted by other users

4. Location – The location of the user who posted the tweet.

1.2]

The approach taken for data download is as follows:-

* Register for a developer account with Twitter. This generates a set of authorization tokens and keys.
* We have selected a set of keywords i.e. the names of the candidates for the Presidential elections in United States of America scheduled next year.
* Tweepy library from python is utilized to obtain streaming data.
* The streaming program in Python takes as input the authentication keys to connect to Twitter.
* Once authentication is established, the tweets are filtered based on the keywords specified.
* Tweets are downloaded and piped to a plain text file as JSON objects.

**2. Store data in the cloud**

2.1]

The Hadoop Distributed File System (HDFS) has been used to store data on the AWS cloud. HDFS is a distributed file system, which stores data in a distributed manner in the form of blocks. The blocks are stored in various nodes on the basis of a principle known as Network Locality, which treats Network bandwidth as the most precious resource, and tries to minimize it.

Input files are stored in the directory :- /project/input. All text files from the directory will be picked up and processed by Hadoop MapReduce jobs. The output files have also been stored in HDFS in plain text format at /project/output.

This file storage system is used as it is robust,and sufficiently fast for the amount of data we are analyzing. In addition, it is native to Hadoop and therefore easy to utilize as well as efficient.

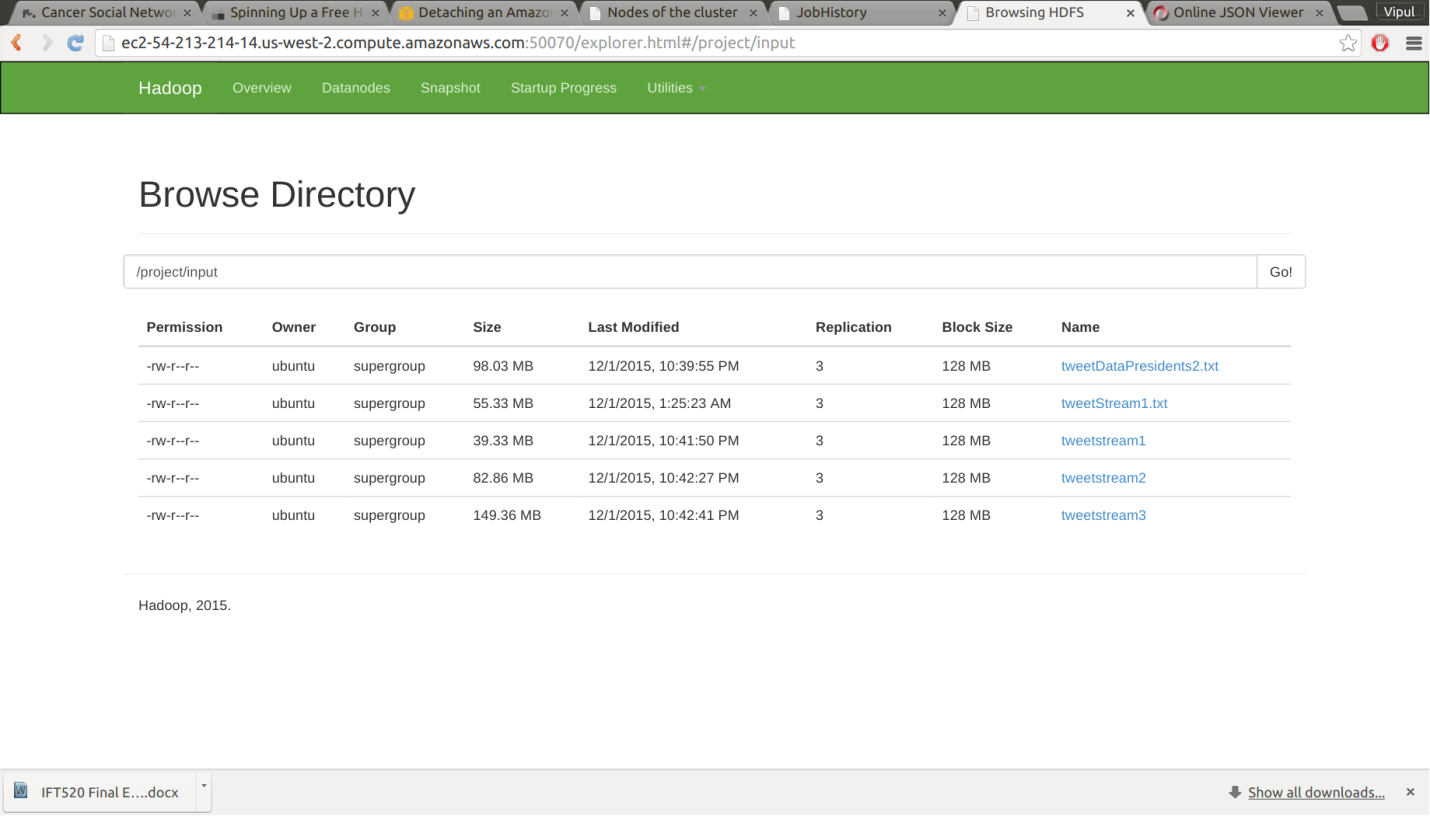
2.2]

Steps taken to store the data into HDFS: -

* Input
  + Route streaming data from the Python program to a file on the local machine.
  + The input is a text file with JSON objects.

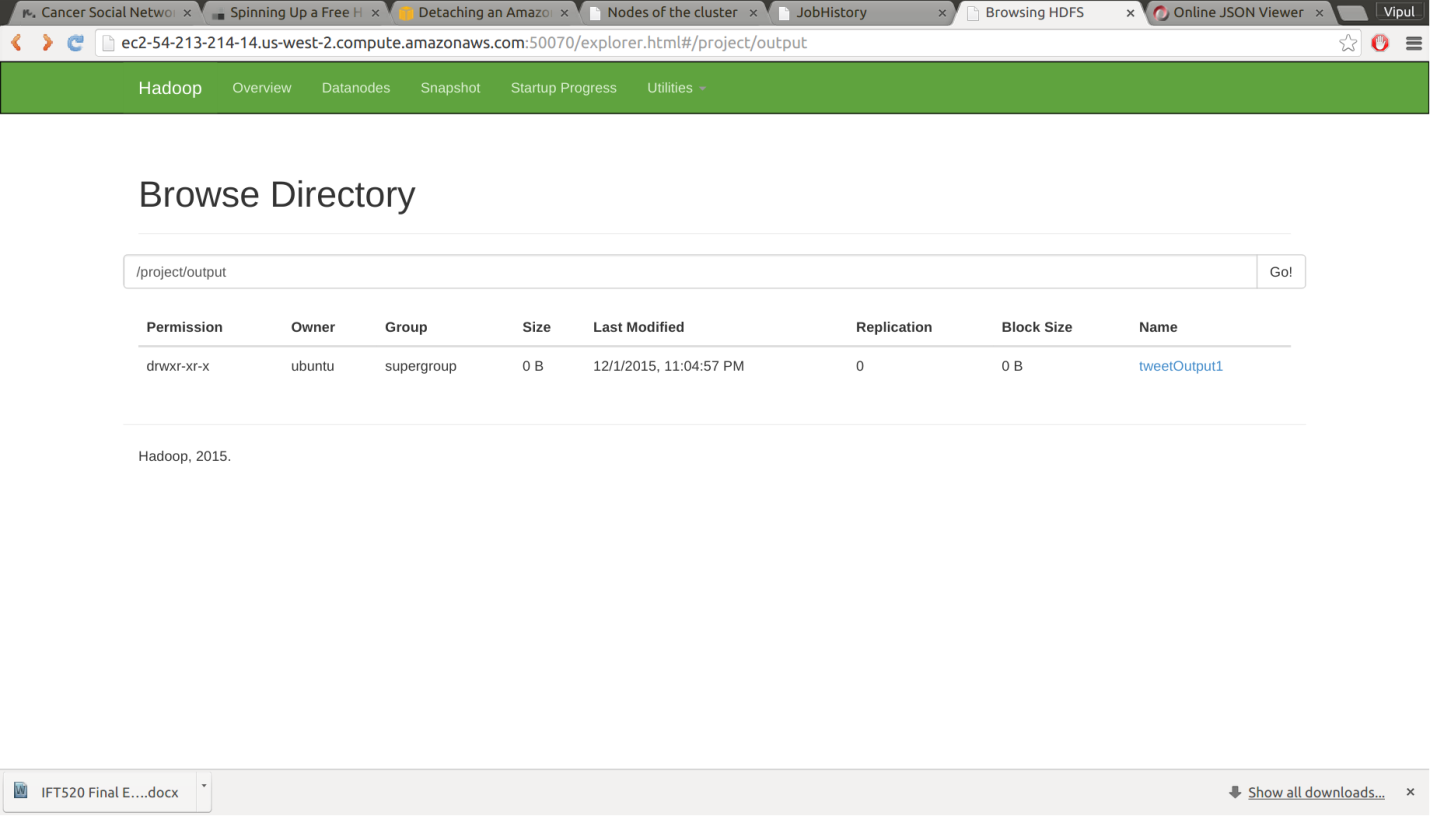
(This can also be done on the cloud however t2.micro instances are limited in their computing and memory capacity.)

* + Upload the data from local instance to the cloud using secure copy command (scp)
  + Upload file manually using the following command to hdfs (which will distribute it across nodes)  
    hadoop fs -copyFromLocal {input file name} {output file directory}



**Screenshot 1: - Input directory in the HDFS**

* + Output
    - Defined output data directory in the main class of the MapReduce framework (i.e. App.java)
    - The output file from the reducer is created as part-r-00000 (when run first ) on the directory specified in the program. The output is in plain text format.



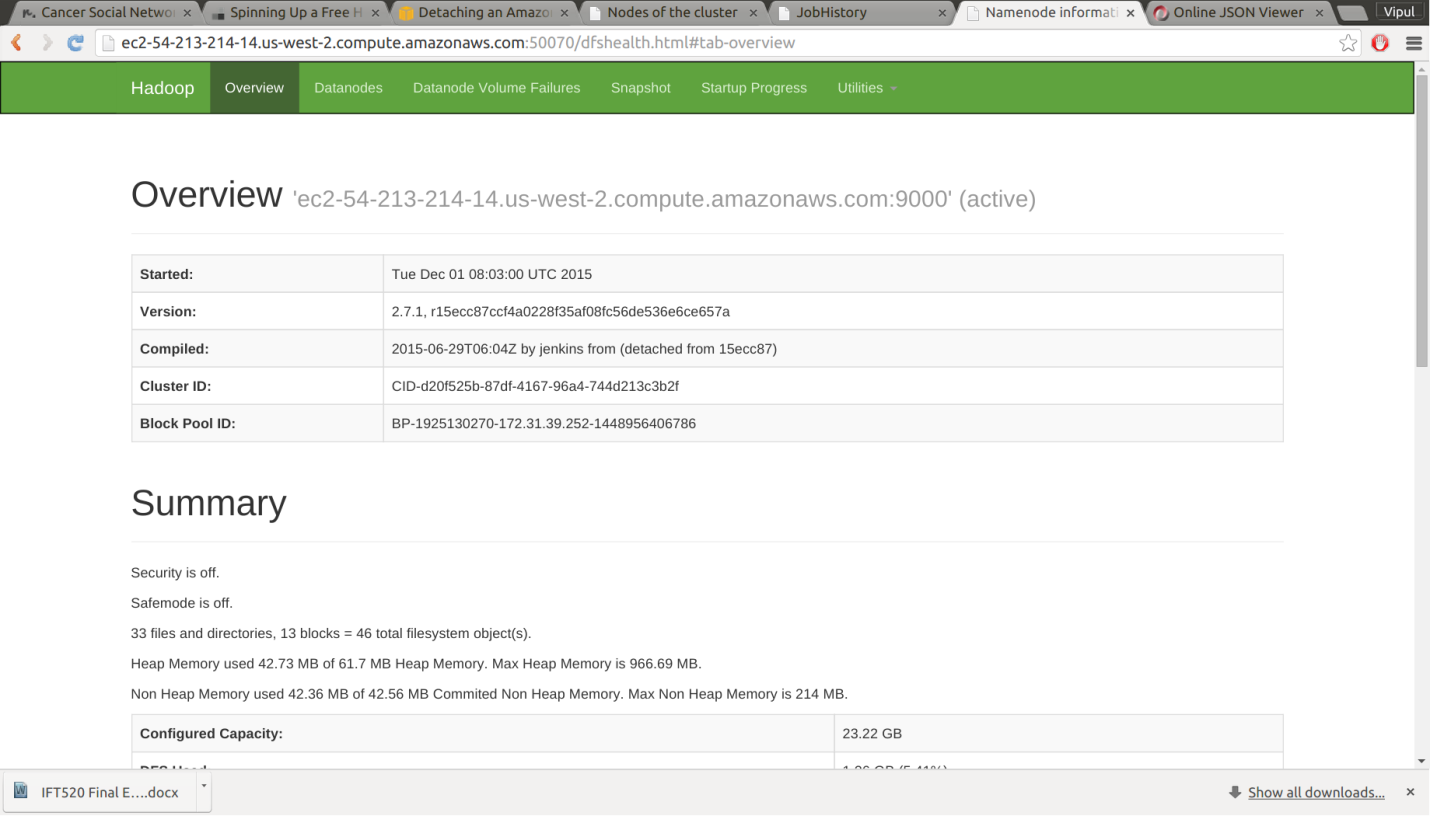
**Screen Shot 2: -Output Directory HDFS**

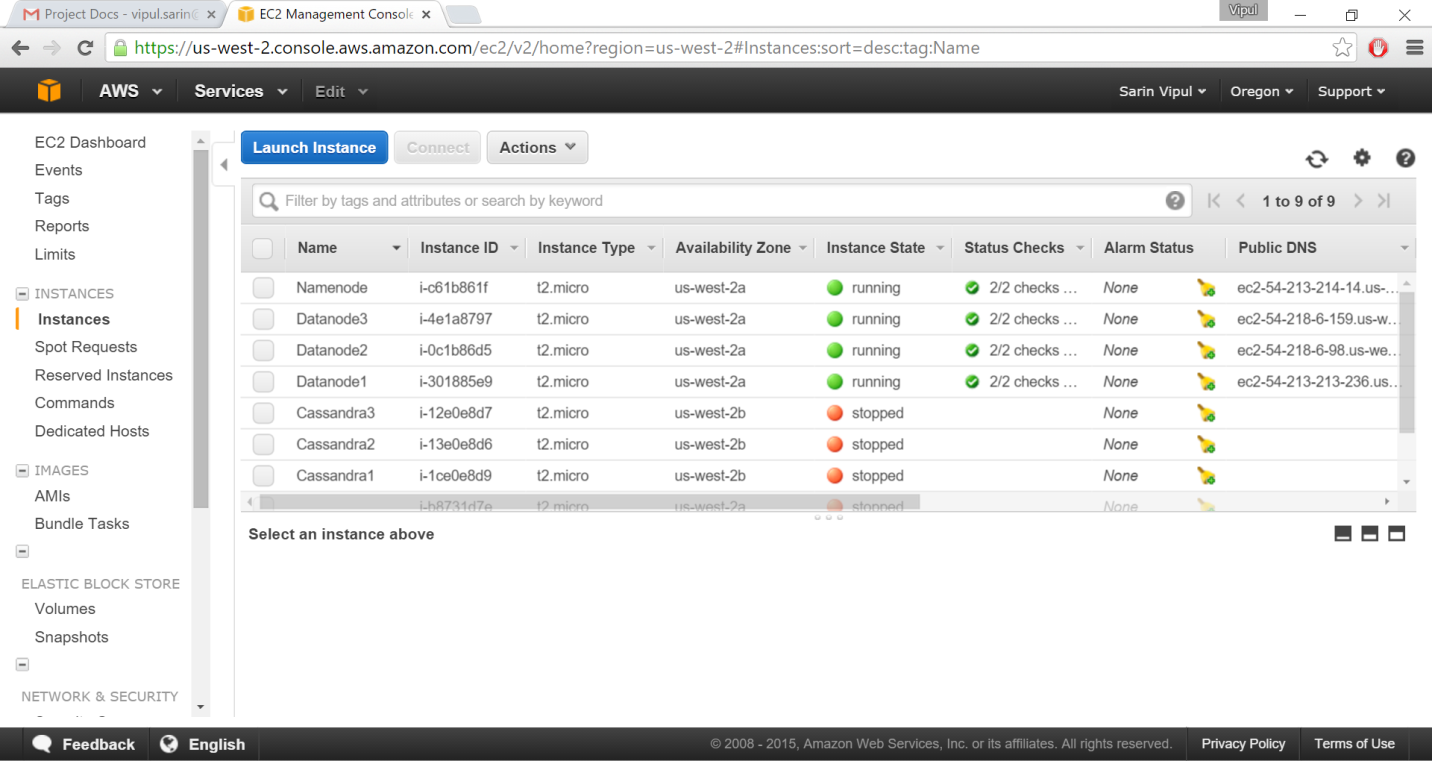
**3. Build a distributed data processing system in the cloud**

3.1] As mentioned in 2.1 we have used Hadoop Distributed File System to store our data.

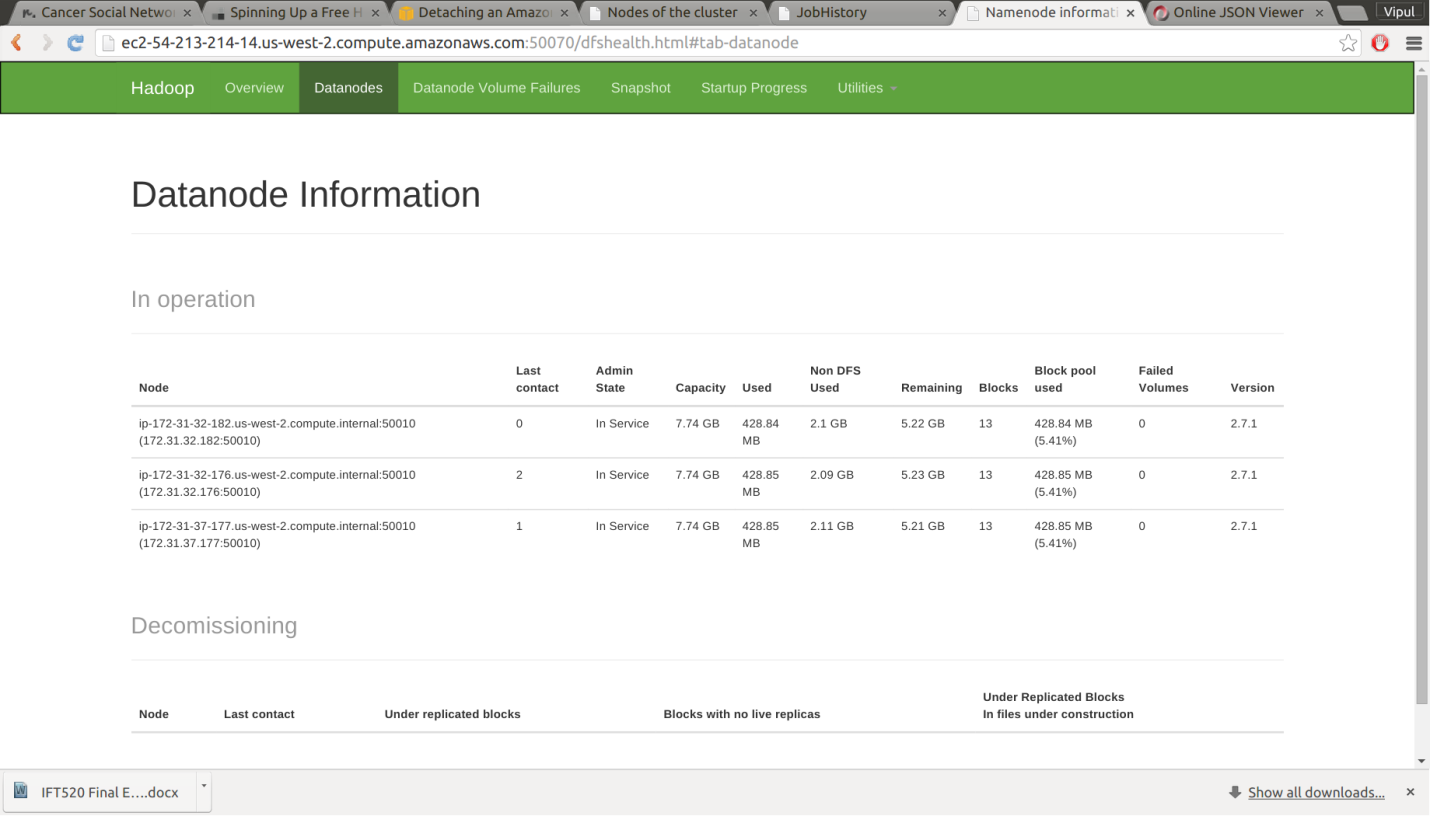
3.2] Hadoop MapReduce distributed data processing system has been used and deployed on the cloud for mining and analyzing data. These steps were followed to setup the distributed data processing system on the Amazon Web Services (AWS) Cloud.

* + The EC2 t2.micro instances have been used to deploy Hadoop distributed system across a cluster.
  + Four separate instances were created, namely : - Namenode, Datanode1,Datanode2 and Datanode3.
  + Secure Shell (SSH) – a cryptographic network protocol is used to remote login to the instances.
  + In the Hadoop framework the Namenode controls all the slave nodes - Datanodes. HDFS is distributed across the Datanodes. Mapper job runs on all nodes in a distributed fashion. The mapper outputs a key-value pair, with the key being the presidential candidate and the value being the tweet object.
  + Once the mapper tasks end, the key value pairs are shuffled (or reduced by key) to combine multiple values on the basis of their key.
  + The output from the shuffle is passed to the Reducer job. The reducer job performs count analysis described in Section 4.
  + Hadoop framework was installed on each EC2 instance. The configuration files for Hadoop were set in accordance to the role of each node i.e. depending on whether it is Namenode or Datanode. Some files that were modified are : - hdfs-site.xml, yarn-site.xml, core-site.xml, mapred-site.xml.
  + Properties such as location of Namenode, replication factor, Java installation home were changed.
  + A master file containing the Namenode’s own hostname and a slave file containing hostnames of all Datanode also defined.
  + Public security keys shared from the Namenode to all Datanodes to authenticate connections between them when the job is running.



****

**Screenshot 3: -Cluster view and EC2 instances**



**Screenshot 4 : - Datanodes**

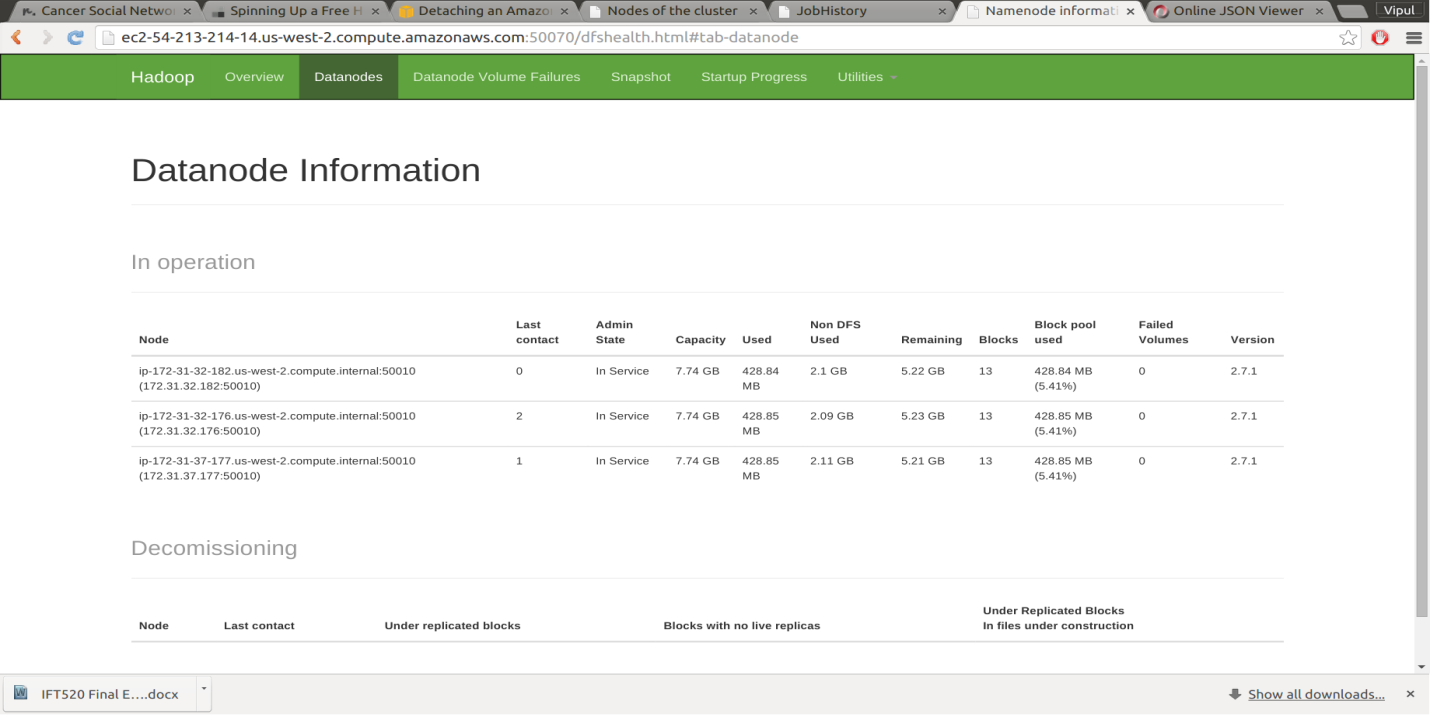
**4.** **Analyze the data**

4.1] United States of America has a scheduled election November 2016. We are analyzing tweets about and for the leading candidates running for office in these elections. This analysis is limited to quantitative factors. Sentiment analysis is out of scope for this project and a probable feature for future development. The following things were analyzed for six candidates

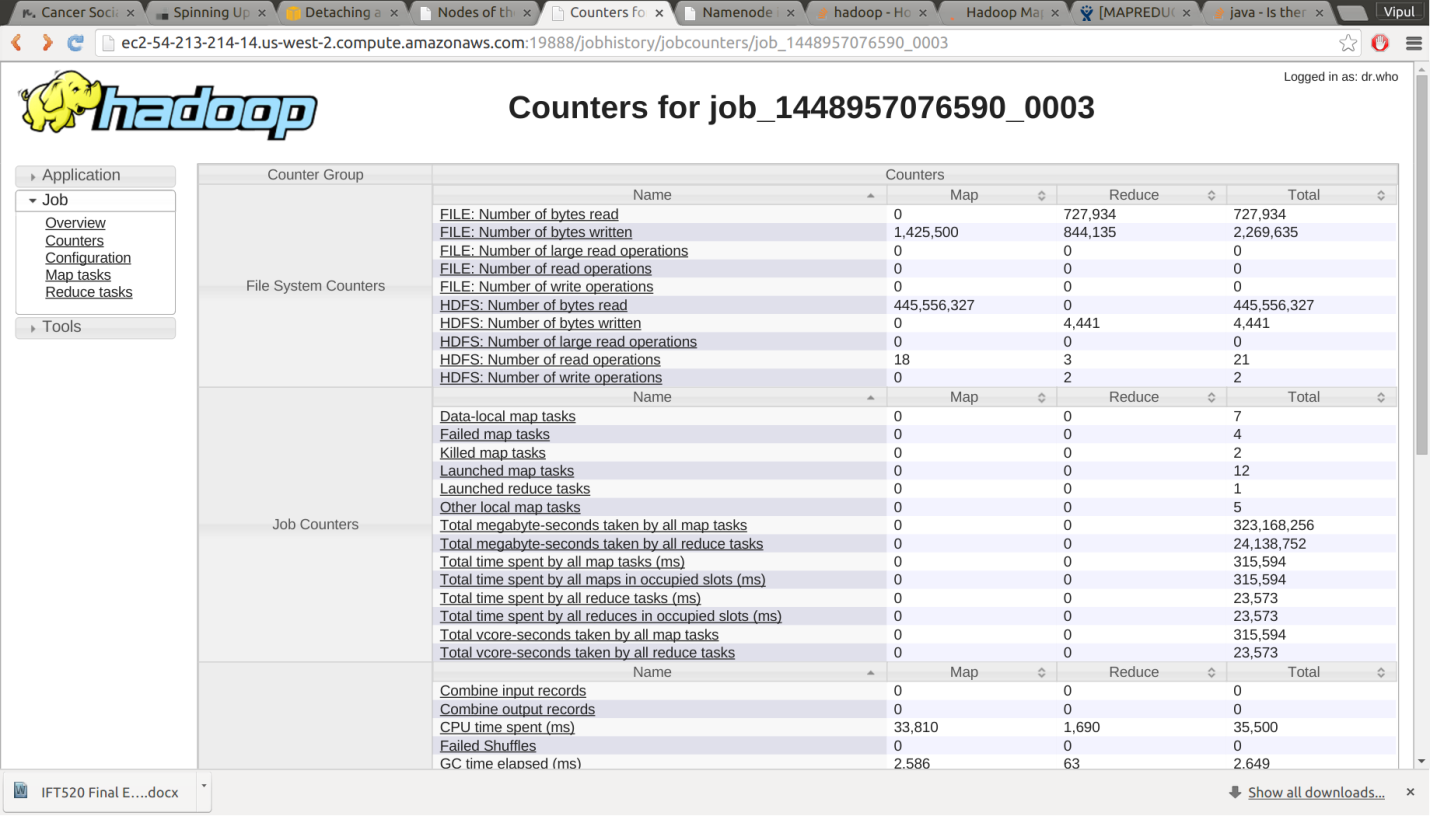
* **Total number of tweets for these candidates**
* **Top ten locations tweeting about these candidates. Note that these locations are those mentioned by Twitter users. Since Twitter is social site, there is no uniformity or standards to define locations and arbitrary location names are accepted. Our system accepts location as they are mentioned by the users.**
* **Average influence level for each candidate. This metric is calculated by summing up the follower counts of each user tweeting about a particular candidate and then dividing the number by the total tweet count. For example if Candidate X has a user with 42 followers and a user with 50 followers tweeting about him, influence level will be (42+50/2)=46**
* **Average retweet count for each candidate. This metric is calculated by taking the average of total retweets for each candidate.**

**4.2]**

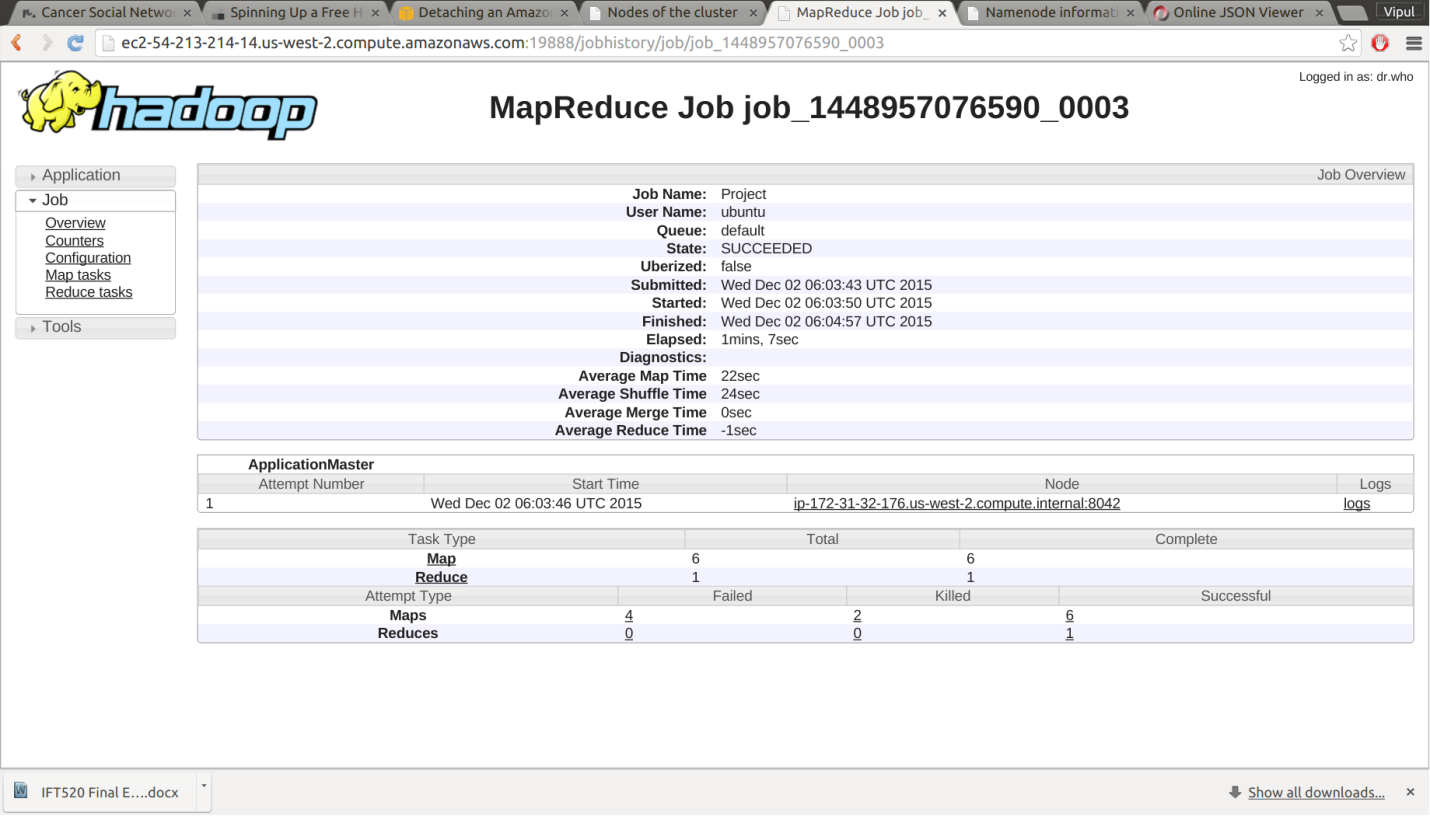
The following steps were followed to analyze the data :-

* The Mapper task reads the input file from the HDFS.
* An input Java HashMap defines the variation of keywords against which each token of the status string is mapped. **This assisted in deciding about which candidate the user was tweeting.** Single tweet can have multiple candidates. To avoid repetitive entries for same candidate in same tweet HashMap is used to store the values.
* Value of each parameter i.e. place, follower count and retweet count is extracted from the tweet and serialized using a custom writable class. These values are then added to the Java HashMap output.
* All values from the HashMap are then written to Mapper Output as object of our Custom Writable class.
* Reducer takes as input Custom Writable class object containing all values for each candidate (Outputs are merged for each candidate prior to Reduce job by Hadoop Shuffle process).
* The reducer counts total number of tweets while iterating over Custom Writable objects.
* Reducer extracts all the attributes from the Custom Writable and calculates value for follower count and retweet count as described in Section 4.1
* For place attribute each place and its count is stored in a HashMap. A custom comparator class is defined that sorts a HashMapas per the values instead of the keys. The HashMap values are then put in a TreeMap that uses this comparator. This way we can iterate over the TreeMap that contains top 10 places.
* All output is written to a ArrayList of type Text. At the end context.write writes all output from the ArrayList in the output file.

**Screenshot 5: - Datanode Information with EC2 addresses**



**Screenshot 6: -Total jobs and time Statistics for each MapReduce Jobs**



**Screenshot 7: -Overall view of MapReduce jobs.**

**5. Display the results of Big Data Analysis**

5.1 We have displayed our results through a web browser running on the cloud. The web browser accepts requests for and displays several files showing the following analysis:

1. Total Tweet Count – An interactive Bar Chart showing the total tweet counts for all the candidates.

2. Total Follower Count – A Bar Chart to visualize the average number of followers for each candidate

3. Total Retweet Count – A Bar Chart to visualize the average number of retweets that each tweet received, on a candidate to candidate basis

4. Tweets By Location – An interactive, animated Pie Chart to visualize the spread of locations that Tweets were being posted from. It includes a dropdown to select the candidate to display the data on.

In addition to this, the raw data can be viewed in the HDFS output file.

5.2. The steps taken to display the data were:

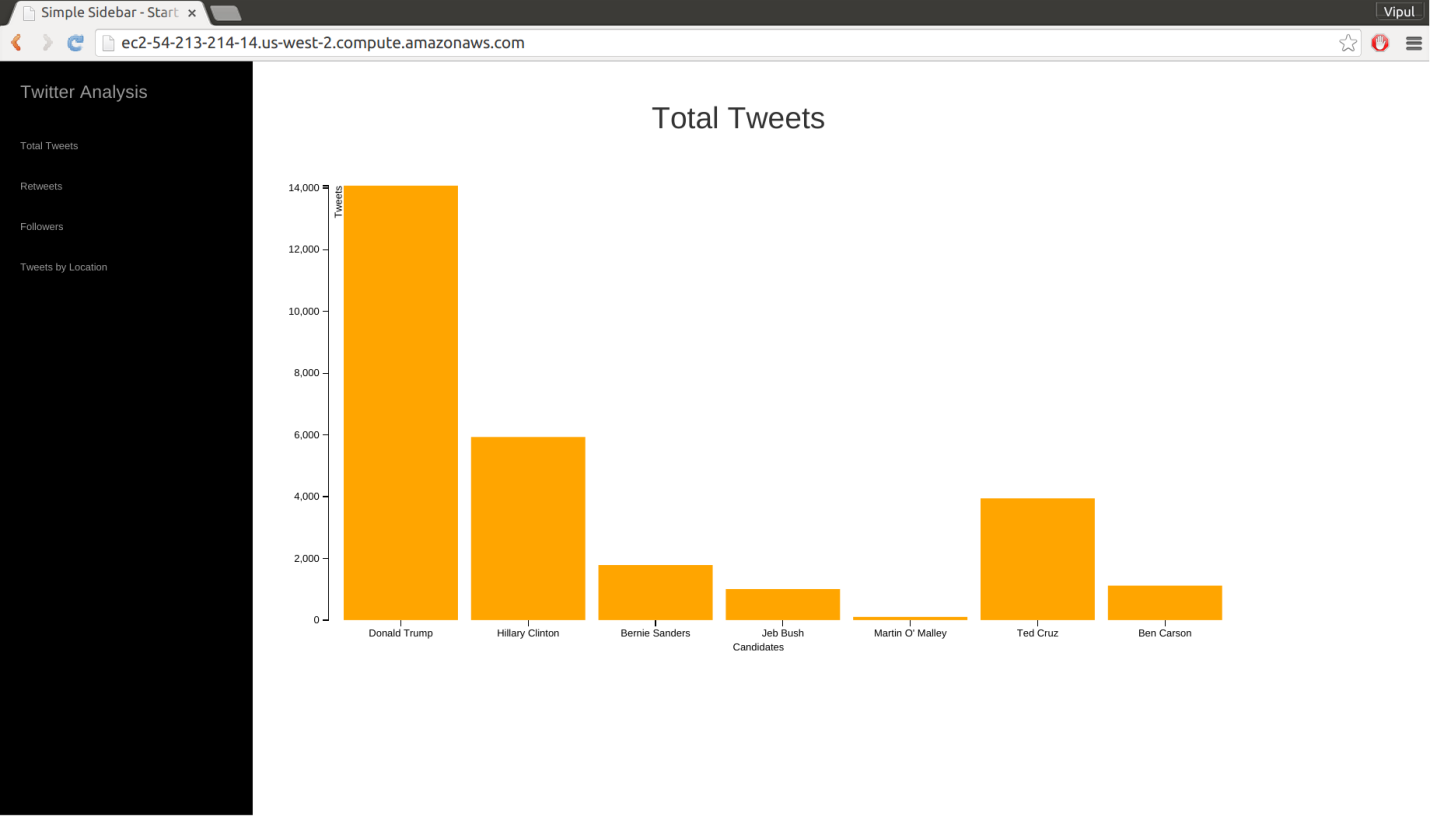
1. The Hadoop MapReduce Job was run first, which stores the output data into HDFS in the folder /project/output/tweetOutput1.

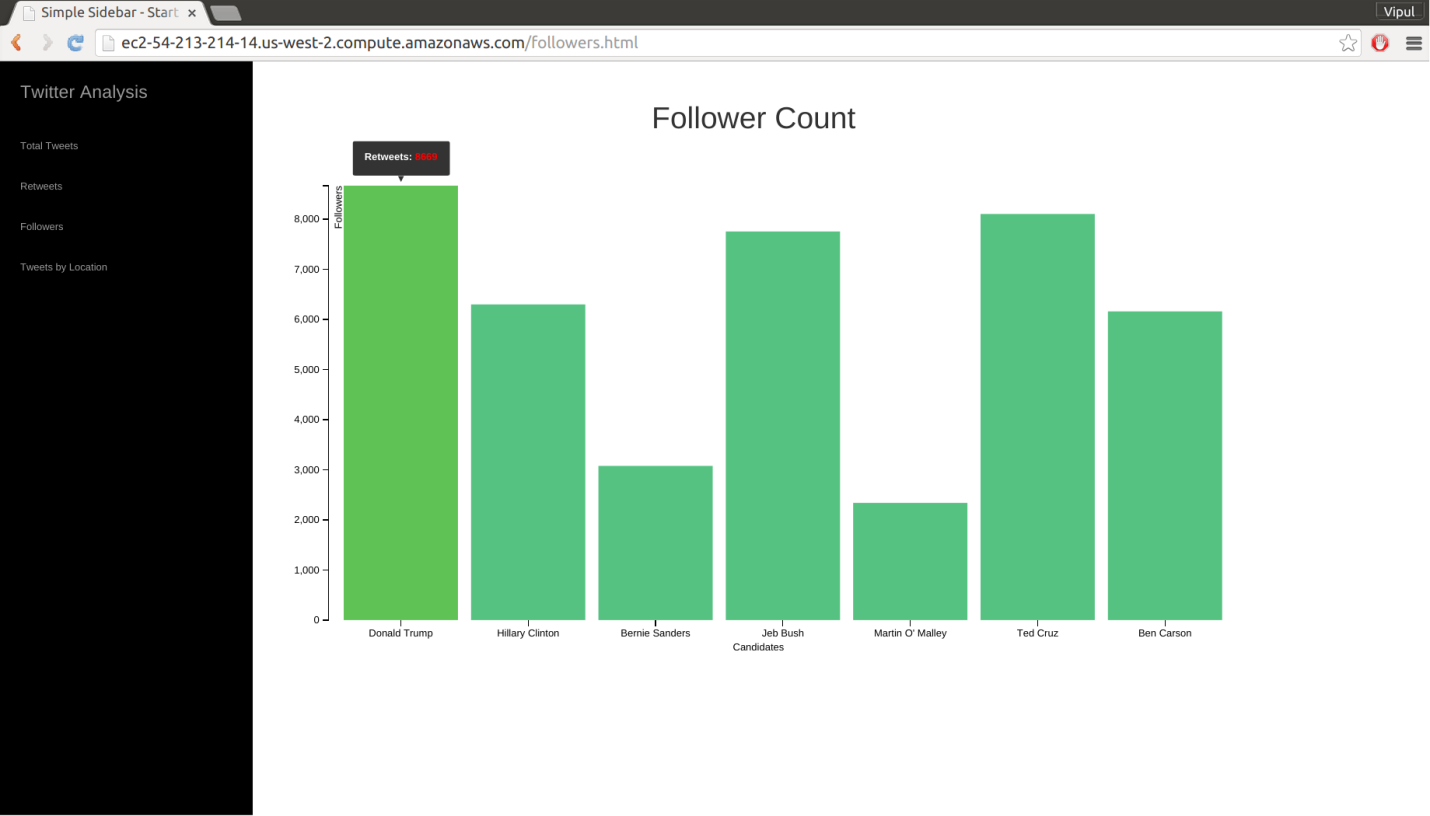
2. We installed an Apache Web Server onto our Namenode instance to display the dynamic graphs on the cloud.

3. We created our interactive graphs, using the Data Driven Documents (D3.js) javascript library. This library enables us to bind JSON data to a graph for visualization purposes.

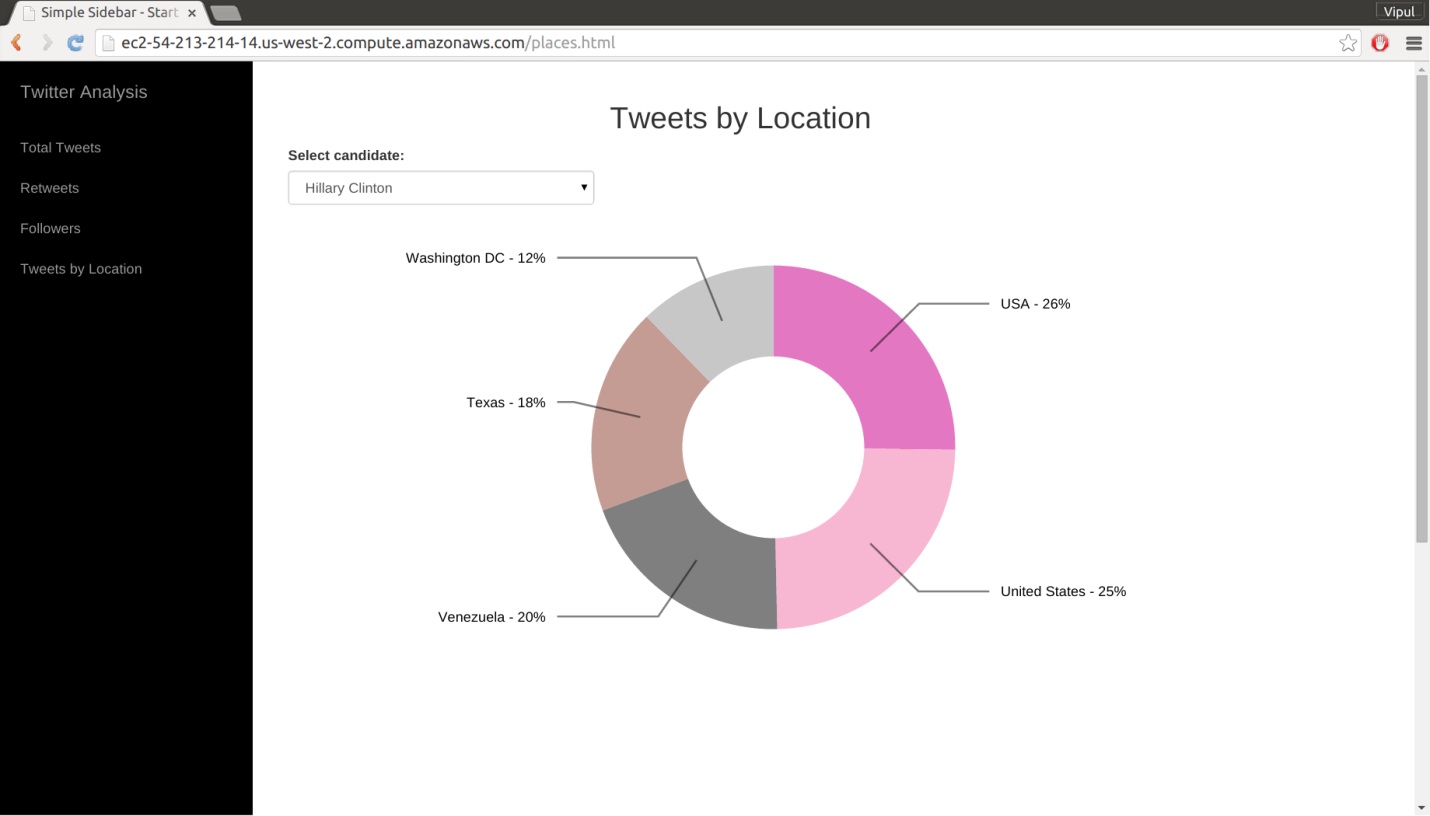
4. We took the data output from the MapReduce job, and bound the data to our D3.js graphs, which are stored in various html files

5. These html files are then hosted on the Apache web server where they can be accessed.

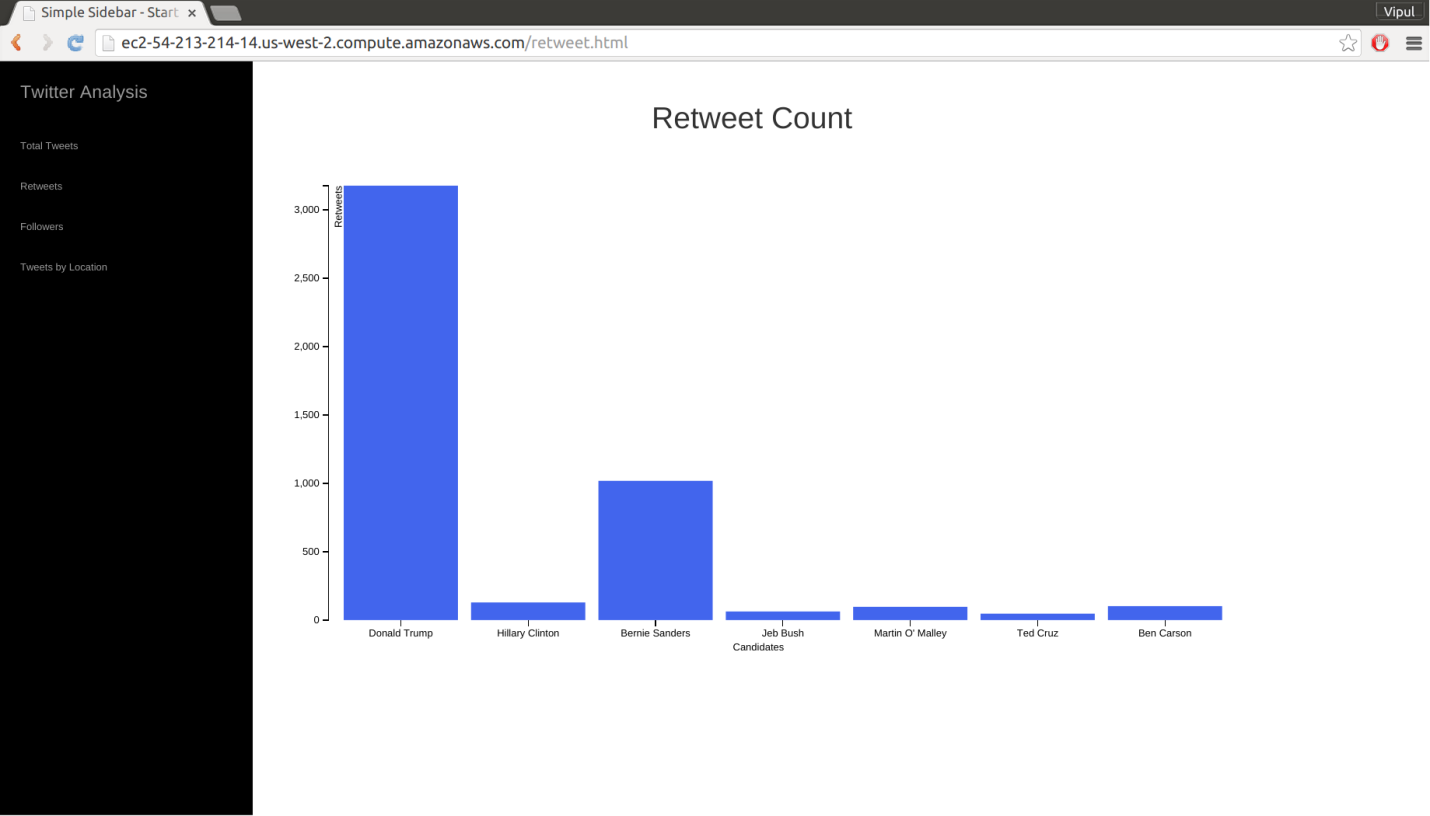
 **Screehshot 8: - Total Tweet Count**



**Screenshot 9: -Influency level**



**Screenshot 10: -Pie Chart for Locations**



**Screenshot 11: - Retweet Count**

**Conclusion**

This project successfully demonstrates the power of cloud computing in developing large scale distributed systems at a fraction of the costs generally associated with buying and building the infrastructure for such a system, in-house. The natural synergy between cloud computing and big data technologies can be leverages extremely efficiently and is proof of the growing maturity of these technologies. The visual format that the data is presented in allows for an easy appreciation of the data, even though a vast amount of tweets (over 30,000) were analyzed.

This project is but a proof of concept, for large scale data analysis on a variety of topics utilizing cloud computing. For example, a web application taking user input on events of interest can be developed to gather, analyze and run future MapReduce jobs for the general public. The proliferation of this technology can allow data analysts to uncover interesting trends and patterns, in otherwise abstruse data. We hope to extend this project and build such a system in the future.