**Twitter #hashtag Analysis for Trending Topics**

**using Apache Spark**

|  |  |
| --- | --- |
| Ishwar Bhat Kattur | Prof. Tom Pearson |
| Anirban Bhattacharya |  |
| Rasika Chafekar |  |
| Priyanka Pakhale | 12/08/2016 |

Overview

The objective of the project was to:

1. Download a large amount of data
2. Store the data on Cloud
3. Analyze the Big Data:
   1. Build a distributed data-processing system on Cloud
   2. Perform the data analysis using the distributed system
   3. Get tangible results from the analysis

Approach

We decided to use Apache SparkTM 2.0.2 as a base for our Cluster Computing System, for the following reasons:

* Ease of Use – Write applications quickly (We used Python 2.7 for our project) and get the system up and running
* Speed – Can run programs up to 100x (!) faster than Hadoop MapReduce
* Popularity – Spark popularity is hugely increasing among developers (and this is going to continue for the future), especially over the past couple of years

Part 1: The Data Source

To get a large amount of data, we had two options:

1. Download a processed and readily available data from public datasets (like Kaggle, data.gov...)
2. Pull data from Big Data sources on our own

Even though option (i) seemed simple and easy, we decided to go with (ii).

Why?

* Freedom to choose our own dataset and the quantity of data
* Added benefit of getting familiar with public APIs

But with this came challenges like:

* Clean the data to get sensible and analyzable dataset
* Parse the data to make it available for cluster processing

What Data?

We used Twitter API to pull the tweets tweeted related to any given event for last 7 days.

We developed a Python Script, which asks Twitter (via the API) to provide a given number of tweets related to a particular topic (e.g. “Presidential Elections”, “LeBron James”, “Donald Trump”).

Some key points:

* Twitter limits automated fetching of Tweets to couple of thousand tweets per hour; which means we had to run the script overnight to get a sizeable amount
* The tweets are pulled and stored as a JSON file

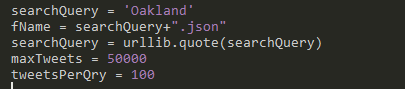
What analysis?

We would analyze the tweets dataset for most popular Hashtags used, and graphically display the results in decreasing order of popularity.

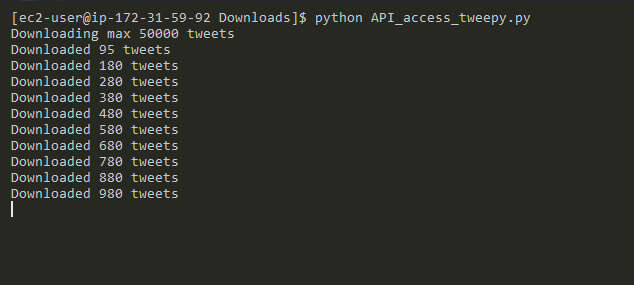
Downloading the Data: Steps Taken

1. Set up an EC2 **t2.micro** instance (with an upgraded storage of 40 GB to handle large JSON files) just to run the Python script
2. The script fetches **n** number of tweets for a particular topic from Twitter
3. A resulting JSON file containing tweet data is stored on the EC2 instance where the script is run

*To download a dataset of 4.4 GB (containing almost 900,000 tweets), we had to run the EC2 instance for around 12 hours.*



*Figure 1: Search topic (searchQuery = ‘Oakland’) and number of Tweets (maxTweets = 50000)*



*Figure 2: Script in action*

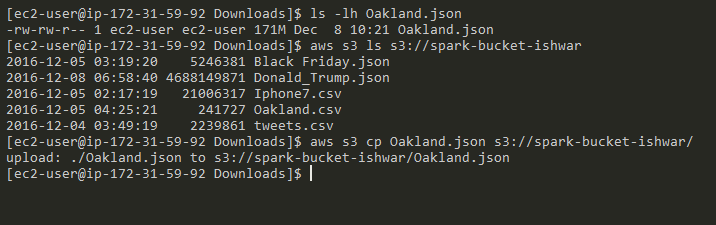
Part 2: Store the Data on Cloud

Since we had a JSON file which we would be analyzing in Spark, AWS S3 was the obvious choice for storage.

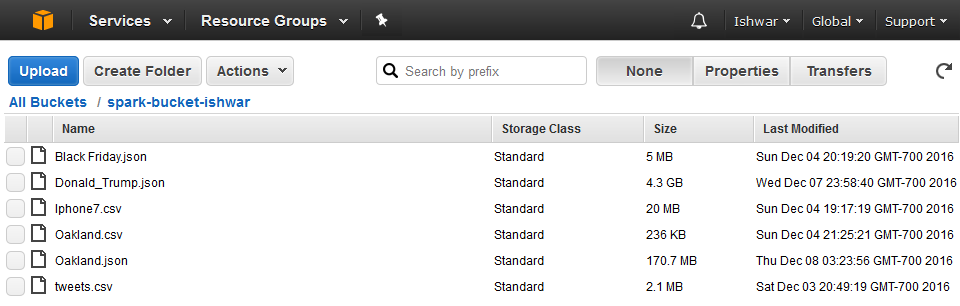
Also, since Spark works seamlessly by directly loading a file onto its memory for processing and can handle a JSON file, we didn’t have to use a database for storage.

Steps

1. Set up a dedicated S3 bucket
2. Get the AWS access key (AWS\_ACCESS\_KEY\_ID and AWS\_SECRET\_ACCESS\_KEY pair) to access our bucket on S3
3. Using the secret access key, use AWS CLI (Command Line Interface) to upload data from EC2 to S3 bucket



*Figure 3: Using the AWS CLI to upload our JSON file (Oakland.json) from EC2 to S3 bucket (spark-bucket-ishwar)*



*Figure 4: S3 bucket with the JSON file. Also shown is another JSON file (Donald\_Trump.json)*

Part 3: Build a Distributed Data Processing System on Cloud

To spin up our Spark Cluster, we used **four** EC2 instances of type **m4.large**.

Each of the m4.large instances has:

* 2 CPU cores
* 8 GB of RAM
* 8 GB of SSD storage
* 450 Mbps of dedicated bandwidth

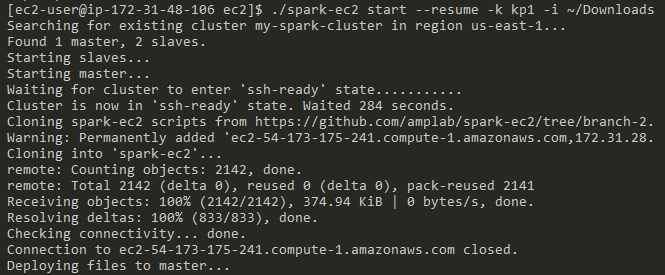
Steps

An EC2 instance was used to spin up an Apache Spark cluster, with:

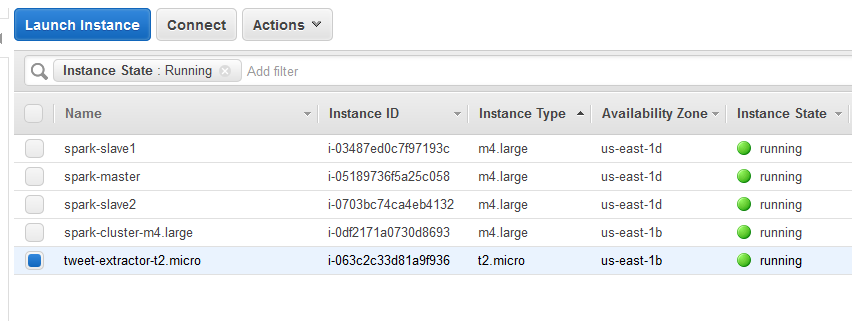
* One Master Node
* Two Slave Nodes

A script **spark-ec2** was used to deploy the cluster.

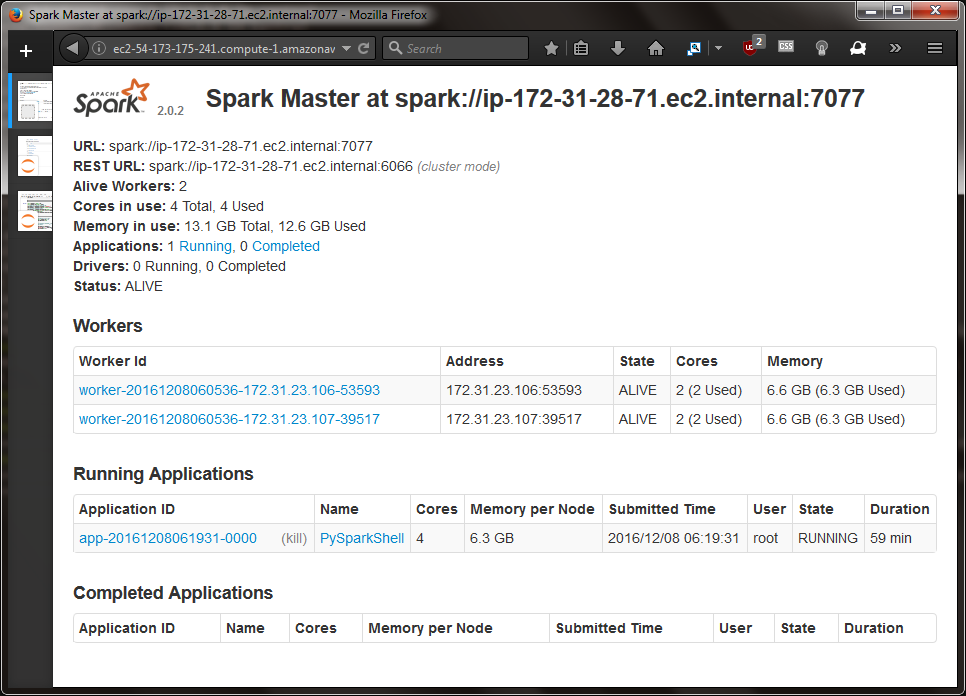
Once set up, the cluster can be stopped and started anytime, via command line.



*Figure 5: Starting our Cluster (my-spark-cluster)*



*Figure 6: EC2 machines which make up the Cluster*



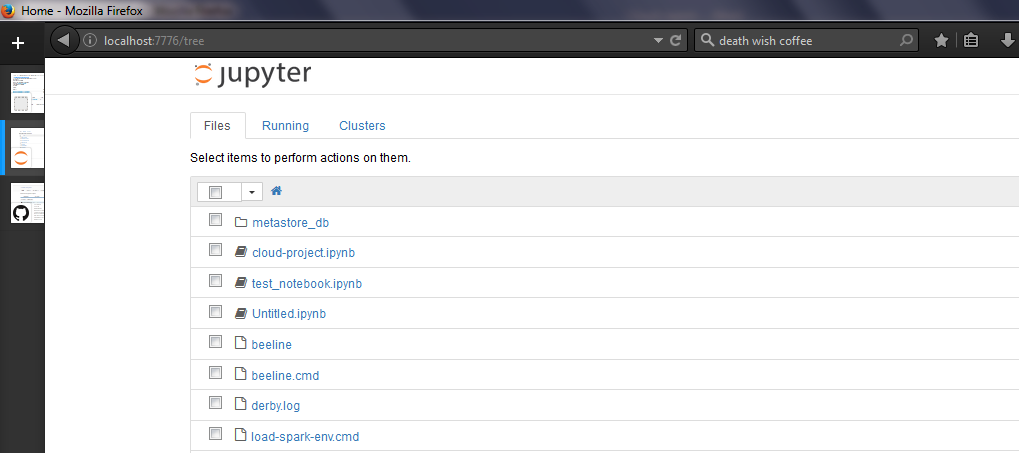
*Figure 7: Web UI of the Spark Master, with two Workers (Slaves)*

Part 4: Analyze the Data

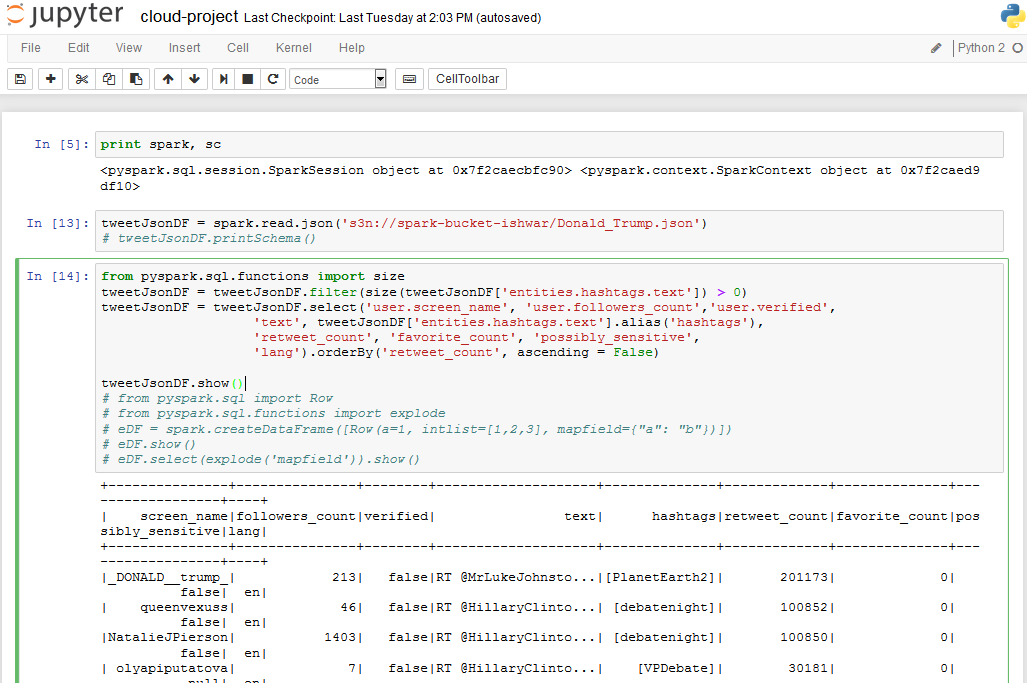
Now that we had our data as well as our Distributed Cloud Processing setup, we started to work on our data. We would be analyzing most popular hashtags related to the search topic used by the Twitter users.

Steps

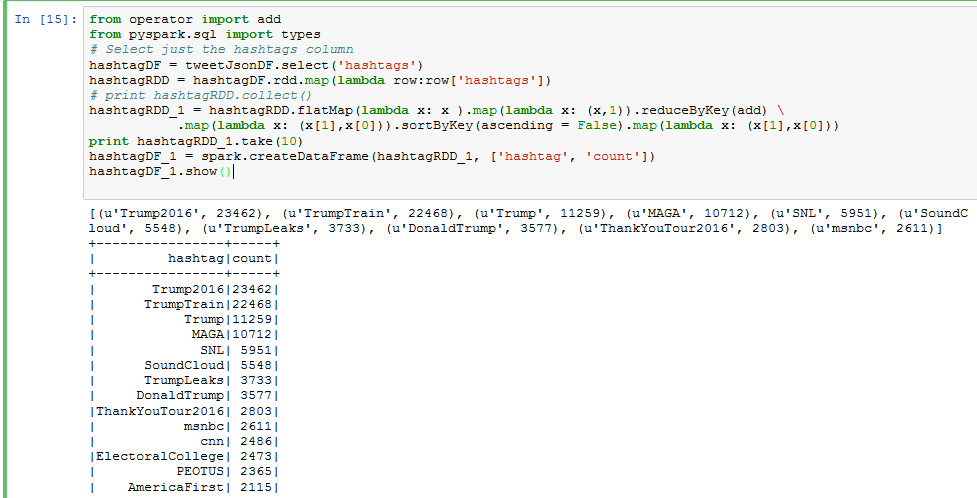
1. On our Spark Master, set up **Jupyter Notebook** (A Web Application for creating and running or Spark Code)
2. Once the Notebook is running, use SSH tunneling (using PuTTY) to access the Notebook on our local computer
3. Using the Spark Python API (PySpark), develop script to read and analyze the data



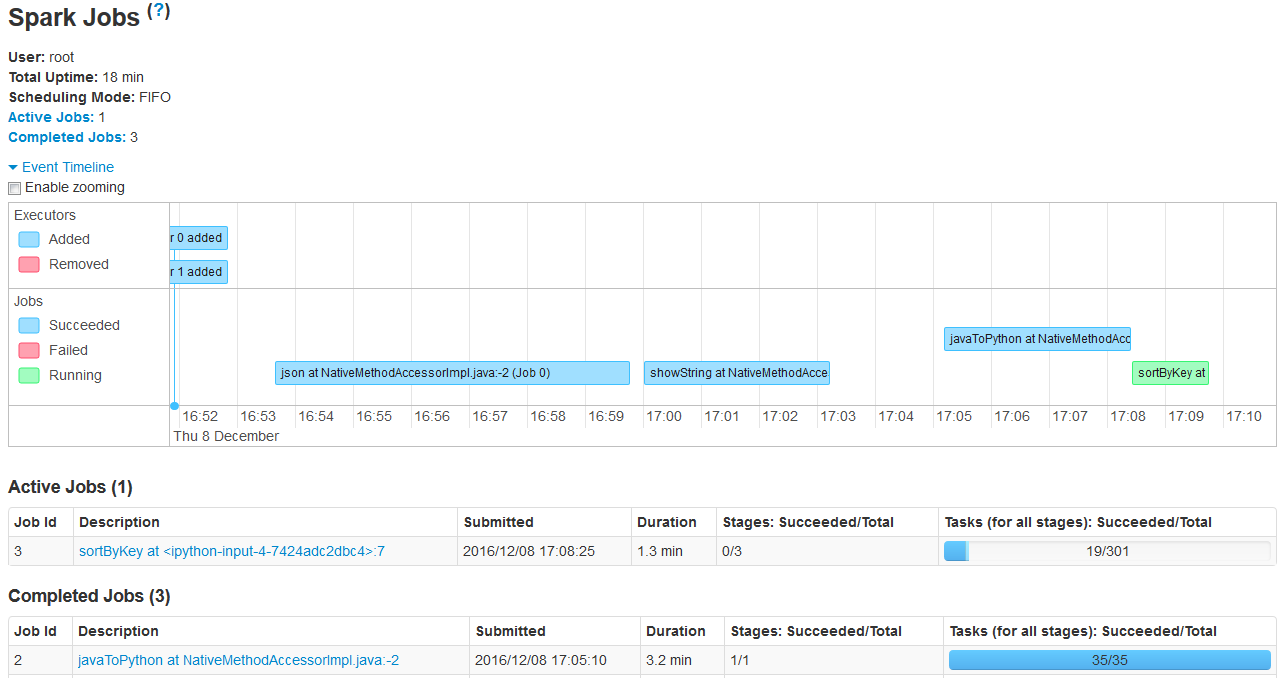
*Figure 8: Jupyter Notebook running on the local computer*



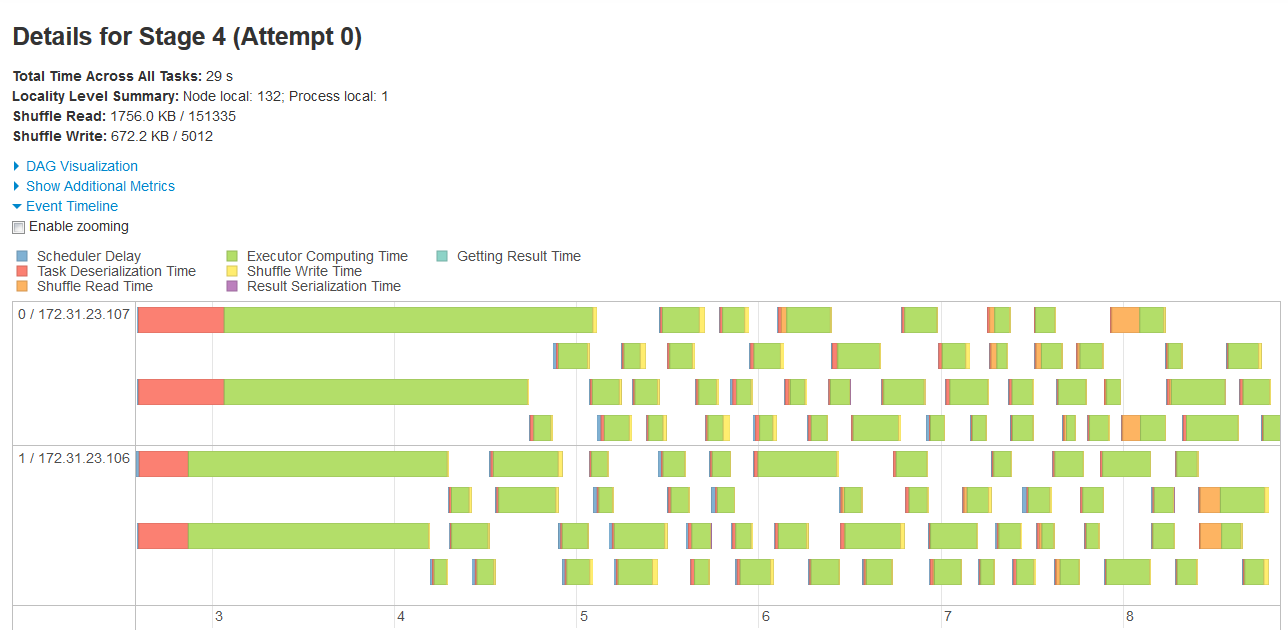
*Figure 9: Reading the JSON file from S3 and parsing it to get required information*



*Figure 10: MapReduce to find popular Hashtags and arrange in descending order*



*Figure 11: Timeline Visualization of our Spark Cluster performing a sortByKey operation*



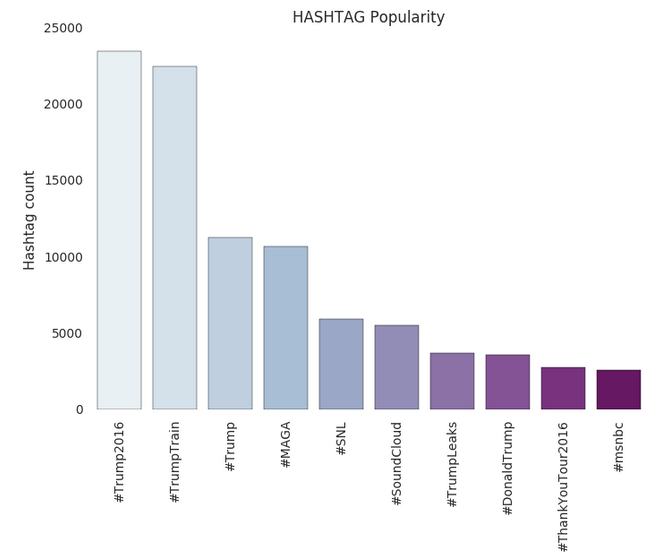
*Figure 12: Detailed Timeline for a specific Stage*

Part 5: Pretty Output

Once we get the popular hashtags, a bar chart showing the hashtag popularity was developed.

Steps

1. Covert Spark Dataframe of **top 10 Hashtags** to a Pandas Dataframe
2. Using the Pandas Dataframe and Seaborn, a Python graphing module, display the Hashtag Popularity Bar Chart



*Figure 13: Hashtag vs Count bar plot*

Results

Big data was downloaded and analyzed, and results were obtained.

# References

*Apache Spark - Lightning-fast cluster computing*. (n.d.). Retrieved from Apache Spark: http://spark.apache.org/

*Survey shows huge popularity spike for Apache Spark*. (n.d.). Retrieved from Fortune: http://fortune.com/2015/09/25/apache-spark-survey/