Architecture Design

Application Idea

The goal of the application is to analyze live twitter stream data to identify trending topics in real-time. This can be useful for a deeper understanding of current social trends and demands, which can be applied back to your current work. This can be useful for advertising, developing marketing materials, determining hot topics to incorporate into product design, and many other areas.

The application reads the stream of tweets from the Twitter streaming API, parses them, counts the occurrences of each word in the stream of tweets, and writes the final results back to a Postgres database. These results can be queried, either through SQL in Postgres or using two custom-built python scripts.

Description of Architecture

Technologies used: Amazon EC2, Twitter API, Tweepy, Apache Storm, Streamparse, Postgres, PsycoPG, Python

Figure 1 displays the overall Application Topology, which starts with Twitter data and works through several streaming processing steps before being stored in a Postgres database for user access.

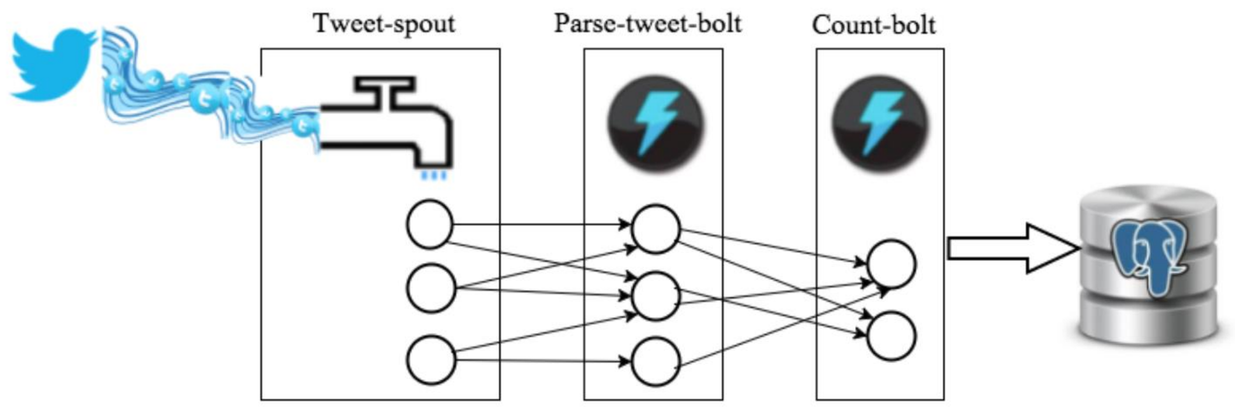

Figure 1: Application Topology

The application starts with a Twitter Application which can be accessed through the Twitter API to retrieve a live stream of tweet data.

The twitter stream is processed using Apache Spark through Streamparse. There is one tweet-spout which uses the Tweepy library to read the live stream of tweets through the Twitter API. This stream is then passed to a parse-tweet-bolt which parses the tweet contents to remove extraneous information (e.g., hashtags) and split the contents into words. These words are then sent to the count-bolt which counts the number of each word received, displays the updated count on the screen, and uses PsycoPG to update the count in Postgres table tweetwordcount in the database tcount.

Finally, the stored information can be accessed via psql querying the Postgres database or through two python scripts that use PsycoPG to query the Postgres database.

Directory and File Structure, and Dependencies

Note that this application is dependent on the Twitter application remaining running and set up with the current credentials.

The code is stored in https://github.com/PageJewel/W205/tree/master/Exercise2/code. Inside this directory, there are the two custom-built python scripts:

- 'finalresults.py' – returns count for a specified word, or if no word is specified it returns all words alphabetically
- 'histogram.py' – returns all words within a defined range of counts specified by the user

This directory also contains the folder 'extweetwordcount'. Most of the files inside this directory are background files set up by streamparse. The ones that have been edited for this application are:

- 'topologies/extweetwordcount.clj' – this creates the topology with one tweet-spout, one parse-tweet-bolt, and one count-bolt and the connections between them
- 'src/spouts/tweets.py' – this defines the tweet-spout
- 'src/bolts/parse.py' – this defines the parse-tweet-bolt
- 'src/bolts/wordcount.py' – this defines the count-bolt

Additionally, from https://github.com/PageJewel/W205/tree/master/Exercise2 there are screenshots of the working application at in the 'screenshot folder', a bar chart of the top 20 words for the current run of the application (Plot.PNG), and instructions on how to set up and run the application (Readme.txt).

Process to Run Application - note that these are also available in the Readme.txt file

1. Start up your AWS machine with the appropriate technology installed

[only needs to be done once] Follow steps 1-3 (through 'Download and Run the Setup Script') on https://github.com/UC-Berkeley-I-School/w205-fall-17-labs-exercises/blob/master/lab_2/Lab2.md to start an appropriate AWS instance. In Step 1, instead of using AMI 'UCB W205 Spring 2016' use the AMI 'UCB MIDS W205 EX2-FULL' which has streamparse already installed.

If you have already set up the AWS machine previously, you just need to start the EC2 instance, ssh in to it, and mount your drive as /data.

[only needs to be done once] 2. Install a couple additional components

Run the following for connecting python to postgres and for connecting to twitter:

pip install psycopg2

pip install tweepy

pip install requests[security]

3. Start Postgres running and set up basic table structure

First, run the following to start up postgres: /data/start_postgres.sh

Then, get a psql prompt with the following: psql -U postgres

[only needs to be done once] Run the following two commands to create your database and table:

CREATE DATABASE tcount;

CREATE TABLE tweetwordcount

```
(word TEXT PRIMARY KEY     NOT NULL,
count INT     NOT NULL);
```

[only needs to be done once] 4. Download the code into /data/exercise2/

All code to run the application lives in:
https://github.com/PageJewel/W205/tree/master/Exercise2/code

5. Start the application

Go to /data/exercise2/extweetwordcount

From the command line run the following: sparse run

When prompted, hit enter to continue running as root user

6. Access results

There are a couple ways to access the results as the application parses the twitter stream data:

- use your psql command line to run queries against the tweetwordcount table

- from the command line in /data/exercise2, run the following to get back the count for a word of interest: python finalresults.py <word of interest>

- from the command line in /data/exercise2, run the following to get back all words with a count in the range of interest: python histogram.py <inclusive lower bound> <inclusive upper bound>

7. Shut it down

From the command line where your streamparse app is running (from step 5), hit cntrl+c

To exit any psql prompts that you have open, run the following: \q

To shut down postgres, run: /data/stop_postgres.sh

Through the AWS console, stop the EC2 instance