Architecture for Postgres Database Populated with Word Counts from Twitter

Timothy Hurt

Application Overview

The application works by accessing the twitter stream, via python with the package tweetpy, using my personal credentials. Once in the twitter stream, python functions work together to listen to the tweet-stream and, every 0.1 seconds, pull out a tweet and emit it to the parse-tweet-bolts. As per the pictured architecture below, there are three spout bolts the run in parallel to one another. The Parse-tweet-bolt receives a tweet from a spout and determines which aspects of the tweet are valid words and the emits these words to the Count-bolt. As per the pictured architecture, there are three Parse-tweet-bolts that run in parallel to one another. A Count-bolt receives the valid words from a Parse-tweet-bolt and checks each word to see if that word is in the Postgres database. If it is, the table is updated by adding 1 to the previous count. If the specific word is not in the table, then the word is added to the table and initialized with a count of 1. There are two Count-bolts that feed into the PostgreSQL database and table. The Count-bolt also prints a running log of all the tweeted words and their count it is receiving in real-time, except that that count restarts every time the application is run whereas the database simply updates every time the application is run. The topology described so far is kept in a clojure file in the topologies directory. (See the directory tree at the end of this document)

Finally, to perform analysis on the table without accessing the database and table in PostgreSQL, there are two python function, finalresults.py and histogram.py, that can be used to analyze the data. Finalresults.py allows the user to see how many times a given word was counted and histogram.py allows a user to see how many words were used within a specific range of counts.

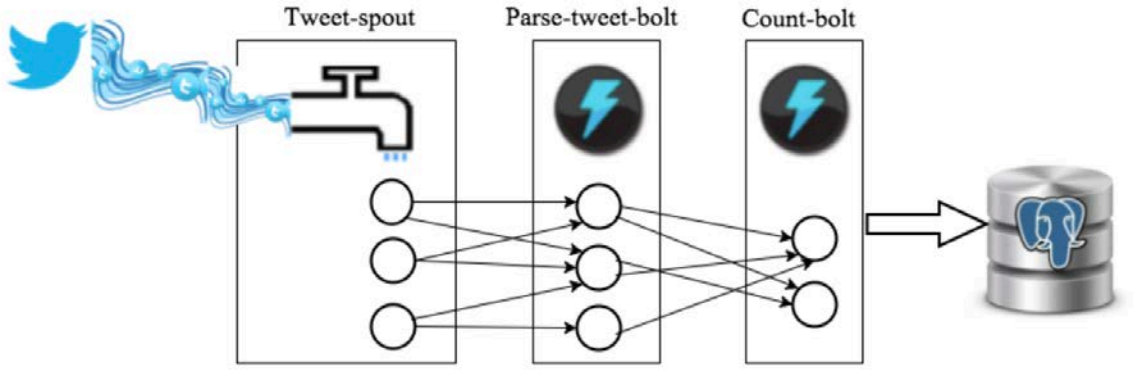

Figure 1: Application Topology

Figure 1 Taken from the 205 Exercise 2 Markdown file

How to use the Application

Start with an Amazon Instance built off the UCB MIDS W205 EX2-FULL AMI. Make sure to mount the volume with /data from previous activities. (To mount the drive, type: *mount -t ext4 /dev/xvdf /data*)

Make sure hadoop and postgres are running. (You can check using *ps -ef|grep hadoop* and *ps -ef|grep postgres*) If they are not running, while in root run: *./start-hadoop.sh* and then: */data/start_postgres.sh*

Make sure psycopg is installed (if it isn't, type: *pip install psycopg2==2.6.2*)

Change into the w205 user (*su – w205*)

git clone the exercise_2 from Timothyjhurt's w205 repository (*git clone https://github.com/timothyjhurt/w205.git*)

Change to the exercise_2 directory (*cd w205/exercise_2*)
Inside the exercise_2 directory is another directory named: extweetwordcount.
Change into that directory (*cd extweetwordcount*)

Once in the directory, to run the application, type: *sparse run*

The application will run until you hit: **Ctrl+C**

At this point, if you would like to view results or word counts from postrges, you can change back to the exercise_2 directory (*cd ..*) and type: *python finalresults.py* to see all of the words and their counts in alphabetical order.

If you want to look up a single word's count, type: *python finalresults.py word*
Where word is the word you want to look up.

If you want to find words that occurred a certain number of times.
Type: *python histogram.py X,Y*
Where X is the lower bound (inclusive)
and Y is the upper bound (inclusive)

File Structure

To aid in navigation through the files and directories, below is a file structure such that when a user is in w205/exercise_2 they can easily change directories into any file they would like:

./README.txt
./histogram.py
./Plot.PNG
./finalresults.py
./extweetwordcount
./extweetwordcount/README.md
./extweetwordcount/config.json
./extweetwordcount/project.clj
./extweetwordcount/Twittercredentials.pyc

./extweetwordcount/tasks.py
./extweetwordcount/psycopg-sample.py
./extweetwordcount/Twittercredentials.py
./extweetwordcount/hello-stream-twitter.py
./extweetwordcount/fabfile.py
./extweetwordcount/src/bolts
./extweetwordcount/src/bolts/wordcount.py
./extweetwordcount/src/bolts/__init__.py
./extweetwordcount/src/bolts/parse.py
./extweetwordcount/src/spouts
./extweetwordcount/src/spouts/tweets.py
./extweetwordcount/src/spouts/__init__.py
./extweetwordcount/topologies/tweetwordcount.clj
./extweetwordcount/virtualenvs/wordcount.txt
./screenshots/screenshot_HistogramResults.PNG
./screenshots/README.txt
./screenshots/screenshot_FinalResults.PNG
./screenshots/screenshot_TwitterStream_and_Postgres

Teaser of Results

Below is Plot.PNG which shows the top 20 words that came up over the time that my application was running. If I had to guess, I would guess that “the” is probably the word the occurs the most for more people.