Matt Acconciamessa
Exercise 2
W205

# Application Architecture

## *Overview:*

The tweetwordcount application is built to pull a stream of tweets from Twitter's API. It reads them, parses them, and then counts the number of appearances of each word in the stream. It then writes the results to a database on postgres. These results can then be queried with different python scripts.

An application of this nature can be used to identify trending topics on Twitter based on the appearance frequency of certain words. One could use this for sentiment analysis, gauging public opinion around certain events (e.g. political debates), and how people are receiving certain types of advertising (e.g. release of a movie trailer or a major sale event). The real-time nature of the stream makes it ideally situated for repeat use.

## *Topology Structure:*

The topology is set up in a 3:3:2 format as outlined by the image below – a tweet spout pulls tweets from the Twitter API and then pushes them to the parse tweet bolt. The bolt parses the tweets and extracts the words, pushing them to the count bolt. The count bolt then counts the number of appearances of each word pushed to it by the parse bolt. These counts update the tweetwordcount table in the postgres database labeled as "tcount".

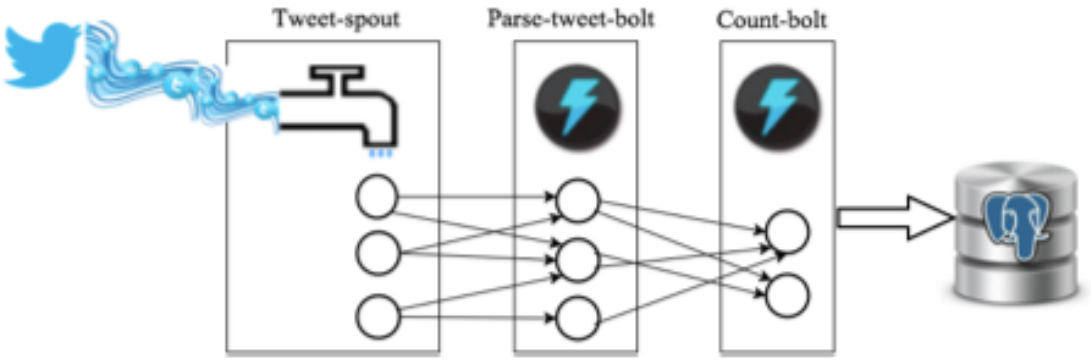


Figure 1: Application Topology

***Application Instructions:***

1) Clear table contents in postgres:

a. Log into postgres as user w205:

- psql --host=localhost --username=w205 --password --dbname=tcount

b. Clear existing tweetwordcount table:

- TRUNCATE TABLE tweetwordcount;

c. Disconnect from postgres:

- \q

2) Run EX2Tweetwordcount project:

a. cd /root/EX2Tweetwordcount

b. sparse run

c. To quit application, ctrl+c

3) Query for desired information:

a. python finalresults.py:

- This will return a list of words in the stream and the number of times each appears

b. python finalresults.py <word>:

- This will return the number of times that the <word> appears in the stream

c. python histogram.py k1 k2:

- This will return a list of words in the stream with a number of appearances between the k1 and k2 integer arguments

NOTE: Debugging difficulties:

In attempting to debug postgres and sparse difficulties, I followed the troubleshooting outlined on the ISVC course page (led by James).

I also rebuilt my AMI to ensure it was probably setup with streamparse and other requisite tools.

Unfortunately I continued to receive an execution error message after some sparse processing, halting the operation before a stream of tweets could be produced.

This was in contrast to the hello-stream-twitter.py file, which effectively produced a stream of tweets (refer to screenshots for a sample).

I sought a number of ways to remedy this issue, but ultimately have yet to find the debugging solution.

However, I believe the query files and topology/spouts/bolts are functional in the right environment.

***Directory Structure:***

***Instance Details:***

Instance ID: i-0f8b66dad64cb0052

Availability Zone: us-east-1a

Public DNS: ec2-54-164-79-182.compute-1.amazonaws.com

Public IP: 54.164.79.182

Key Name: UCB3.pem

Security group: launch-wizard-3

***Dependencies/Items to Install:***

For setup/running  this application, the following are required:

- Streamparse
- Tweepy
- psycopg
- postgres
- python 2.7
- Apache Storm
- Amazon EC2

***Directory Structure:***