Architecture of extweetwordcount

Dependencies

Amazon EC2 (it is recommended to use the ami-d4dd4ec3 for running this software. This will have pre-installed Python, Apache Storm, and Streamparse. If not using this AMI image, please manually install the software.)
Twitter API (please refer to Appendix I for specific information on setting up the twitter API)
Postgres
PsycoPG (pip install psycopg2==2.6.2)
Tweepy (pip install tweepy)

Note: Additional information on these softwares can be found in Appendix II. Python version is 2.7 during development of this application.

Overview

Capturing and analyzing live twitter data can provide a company valuable information regarding current social trends and demands. Historical data may be informative for trend assessment. However, live data can provide real-time insight, especially important in the fast evolving industries.

Figure 1 shows the overall architecture and the Storm topology implemented in this application. This uses the Tweepy library to read the live stream of tweets from Twitter. This information is fed into the tweet-spout component. This data then moves to the parse-tweet-bolt bolt which parses the tweets and extracts individual words, which then moves the words to the count-bolt. The count-bolt then tracks a count of each word from the stream. Finally, that information is then updated in the tcount Postgres database.

In this topology, once the data move from the Twitter stream to the Tweet spout via the Tweepy library, 3 processors are used. The data is then passed on the parse-tweet-bolt, which again uses 3 processors. Finally, the count-bolt uses 2 processors, then connects to the Postgres database and updates after each word is processed.

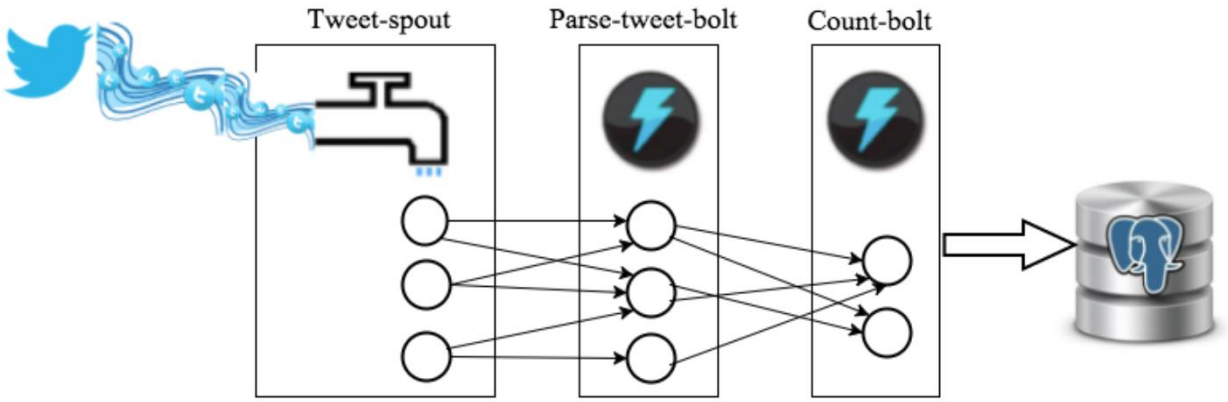

Figure 1: Application Topology

Folder Structure
The folder structure looks as follows (bold indicates file names, brackets represent folders, italics are descriptions).

README.txt: *Instructions on how to successfully run through the setup, streaming data, and obtain results*
Architecture.pdf: *this file which contains information about all pre-installed software, dependencies, and folder structure*
plot.png: *example bar graph of top 20 words from a twitter stream*
[extweetwordcount]
[extweetwordcount/src]: contains *the spout and bolts needed for the project*
[extweetwordcount/src/bolts]: *contains the bolts needed for the project*
extweetwordcount/src/bolts/**parse.py**: *parses the tweets and extracts words*
extweetwordcount/src/bolts/**wordcount.py**: *updates counts for each word and stores into tcount, a Postgres database*
[extweetwordcount/src/spouts]: *contains the spouts needed for the project*
extweetwordcount/src/spouts/**tweets.py**: *collects live stream of tweets from Twitter*
[extweetwordcount/topologies]: *this folder contains the code to describe how to use the bolts and spouts*
extweetwordcount/topologies/**tweetwordcount.clj**: *code that informs the project how to use the bolts and spouts*
[extweetwordcount/virtualenvs]
extweetwordcount/virtualenvs/**wordcount.txt**: *always required for streamparse projects*

extweetwordcount/**config.json**: *project configuration*
extweetwordcount/**fabfile.py**: *can use this to perform custom actions prior to or post topology submission*
extweetwordcount/**project.clj**: *description of project*
extweetwordcount/**tasks.py**: *can use this to perform custom actions prior to or post topology submission*
extweetwordcount/**README**: *contains instructions on how to use this*
extweetwordcount/**finalresults.py**: *script to output a full list of words with at least 1 count or search for a specific word*
extweetwordcount/**histogram.py**: *script will output a list of words between specified min and max counts*
extweetwordcount/**prerun.py**: *this will drop the tcount data if it exists, and create a fresh new version.*

[screenshots]
screenshots/**screenshot-setup.png**: *example output of a successfully run prerun.py*
screenshots/**screenshot-stream.png**: *example stream output from running 'sparse run' using the extweetwordcount topology*
screenshots/**screenshot-finalresults.png**: *example output of a successfully run finalresults.py with and without a word input*
screenshots/**screenshot-histogram.png**: *example output of a successfully run histogram.py using 3 and 8 as the minimum and maximum, respectively*

Appendix I: Twitter API Setup

Setting up a Twitter application will allow this application to access the Twitter data and use the access keys provided in the program. A Twitter account must be set up first by navigating to https://www.twitter.com

After a Twitter account is set up, please visit http://apps.twitter.com
Click "Create New App" and set up a name and description (website is required, but is not verified).
Click on "Keys and Access Tokens" tab, then "Create my access token".

This will provide four pieces of information:
- consumer key that identifies the application
- consumer secret that mimics a password
- access token that identifies your authorized access
- access secret that mimics the password for the access

Appendix II: Software Resources

Amazon Web Services (AWS): https://aws.amazon.com/what-is-cloud-computing/?nc2=h_l2_cc
Apache Storm Documentation: http://storm.apache.org/releases/current/index.html
Streamparse Documentation: https://streamparse.readthedocs.org/en/latest/api.html
Twitter Stream API: https://dev.twitter.com/streaming/overview
Tweepy Documentation: http://docs.tweepy.org/en/v3.5.0/
Tweepy How-To: http://docs.tweepy.org/en/v3.5.0/streaming_how_to.html
Psycopg Documentation: http://initd.org/psycopg/
Python: https://www.python.org/
Postgres: https://www.postgresql.org/