Architecture of Exercise 2 - Twitter-Storm-Postgres Application

Megan Jasek

Overview - Application Idea

This application is a streaming application that analyzes Twitter Tweets which are read from the Twitter streaming API, parsed and counted. The word count of the words in the Tweets is written to a Postgres database. The data in the Postgres database can be further analyzed with tools of the user's choosing. This application deploys two functions that output the number of occurrences of words and the most popular words.

Architecture

The overall architecture of the application is as follows (and is shown in the architecture diagram below). The application uses streamparse which enables it to use Apache Storm with python to collect and analyze Tweets from a stream of Tweets that is output from Twitter. The driver of the application is Apache Storm which uses a series of components called spouts and bolts to process the Tweets. This application uses one spout to collect the Tweets from Twitter, one bolt to parse the Tweets and a second bolt to count the words in the Tweets and store the results.

The spout component is called Tweet-spout and uses the tweepy python library to access the Twitter Streaming API to collect Tweets from Twitter in real-time. The tweepy.Stream.filter command is used to filter out Tweets containing the following words: "a", "A", "the", "THE", "i", "I", "you", "YOU", "u", "U". After the Tweets with these words are captured, this spout sends the Tweets to the Parse-tweet bolt using shuffle grouping which ensures each bolt component worker gets an equal number of Tweets. There is one Storm worker deployed for this spout.

The first bolt component is called Parse-tweet-bolt and consists of python code that parses the Tweets into valid words. The words are converted to lowercase and leading and trailing punctuation and symbols are removed. This bolt passes the valid words to the next bolt (the Wordcount bolt) using fields grouping by words which ensures that each word gets sent to the same bolt worker. This ensures that the counts of the words are correct. There are two Storm workers deployed for this bolt.

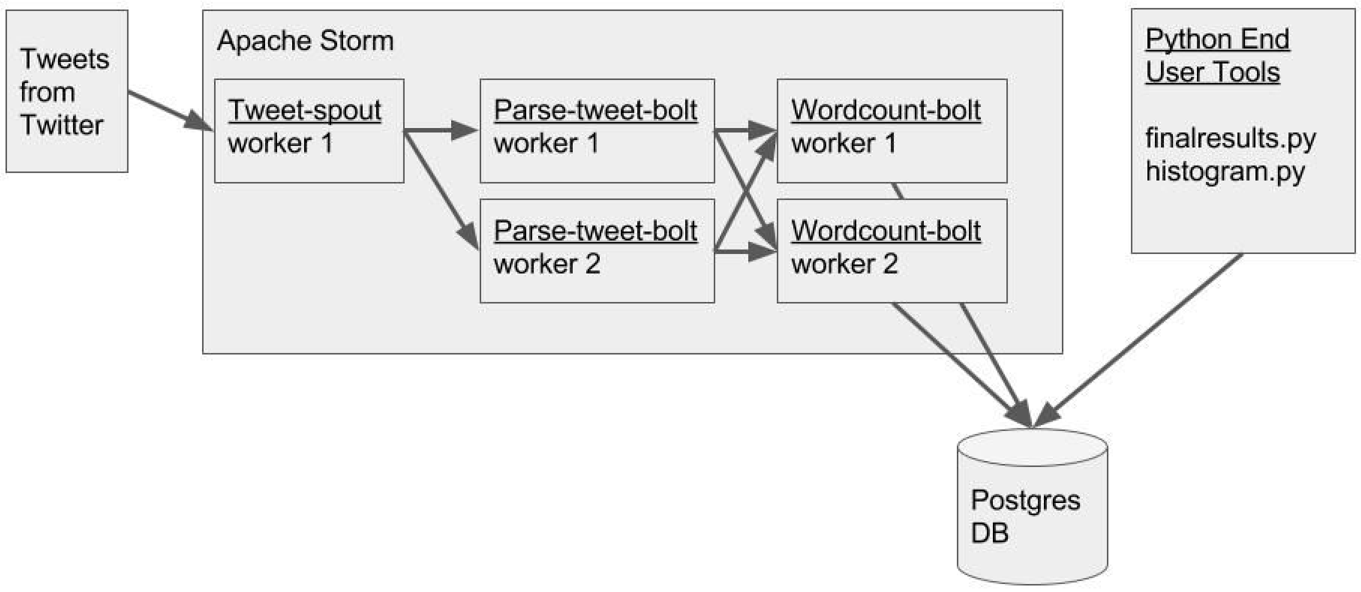
The second bolt component is called Wordcount-bolt. It uses python code to count the words and uses the psycopy2 python library to access Postgres and store the counts in a Postgres database. There are two Storm workers deployed for this bolt.

The Postgres database that the second bolt writes to is named 'tcount' and contains one table called 'tweetwordcount'. This table has two columns: word and count. The 'word' column stores the word name and the 'count' column stores the number of times that word occurred in the stream of Tweets up to that point.

The application also deploys two python programs called finalresults.py and histogram.py. Finalresults takes 0 or 1 arguments. If it has no arguments it outputs a sorted list (sorted alphabetically in ascending order) of each word in the Postgres database and its number of occurrences. If the user passes it a word as an argument then the number of occurrences of that word are returned.

The second python program is called histogram.py and it takes two arguments which are a low integer and a high integer. The program returns a sorted list (sorted in descending order based on number of occurrences of a word) of words whose number of occurrences fall between (and including) the low integer and the high integer.

Architecture of Exercise 2 - Twitter-Storm-Postgres Application



Directory and File Structure and Dependencies

The application is laid out in the following directory and file structure.

- **exercise_2 -** the top level directory. In this directory are the following files
  - Readme.txt - an explanation of how to get the application installed and started
  - finalresults.py - code for the finalresults program
  - histogram.py - code for the histogram program
  - install-script.sh - a script that installs required components on to an already setup ucbw205_complete_plus_postgres AMI from Amazon EC2.
  - rmdbtable.sh - removes the Postgres table ‘tweetwordcount’ from the ‘tcount’ database and creates a new one, so that the user can collect a new set of Tweets with a new set of word counts.
  - Architecture.pdf - this document. Describes the architecture of the application.
  - plot.png - a bar graph of the top 20 Twitter words output from the application at a point in time.
  - **EX2Tweetwordcount** - this directory has the standard directory structure of a streamparse application project. The relevant files that were changed for this project are as follows:
    - src/spouts/tweets.py
      - NOTE: you must add your Twitter application credentials to this file in order to get Tweets from Twitter Streaming API using tweepy.
    - src/bolts/parse.py
    - src/bolts/wordcount/py

        - topologies/tweetwordcount.clj - contains information to set the number of Storm component workers, which fields are passed between the components and which grouping strategy (ex: shuffle, fields) is used to pass the data between components
    - **screenshots** - directory that contains screenshots of the running application
        - screenshot-storm-components-spout.png - screenshot of Tweets output from the spout component
        - screenshot-storm-components-bolt-parse.png - screenshot of parsed words output from the parse bolt
        - screenshot-storm-components-bolt-count.png - screenshot of the output from the application
        - screenshot-storm-components-bolt-count 2.png - screenshot of the words and counts from the count bolt.

How to Run the Application

Basic steps to run the application. See the Readme.txt files for exact installation steps.

1. Install the ucbw205_complete_plus_postgres_PY2.7 with the easybutton instructions from the file: TheEasyButtonforYourAWSEnvironment_2_2.pdf. Make sure the correct security group rules are defined (see Readme.txt for details).
2. Change to the /data directory with the command
3. Make sure that the Postgres server is running. If it's not start it.
4. Clone my UCB-MIDS repository from github with this command (you may need to add your username before github.com, ex: https://<username>@github.com/mjasek114/UCB-MIDS)
    a. git clone https://github.com/mjasek114/UCB-MIDS
5. Pull the application code from github
    a. git pull origin hw_dev
6. Run the install script to install my application: run the install-script.sh script
7. Add Twittter credentials to the file UCB-MIDS/exercise_2/EX2Tweetwordcount/src/spouts/tweets.py
8. change to the streamparse project directory called: EX2Tweetwordcount and use the ‘sparse run’ command to start the application.
9. Press Ctrl-z when you want to stop the application
10. Move up 1 directory level to run the serving files with these commands (example arguments are given):
    a. python finalresults.py hello
    b. python finalresults.py
    c. python histogram.py 3, 8
11. To run the application again, change to the exercise_2 directory and run the following script to create an empty ‘tweetwordcount’ table in the ‘tcount’ database
    a. ./rmdbtable.sh