Twitter Streaming Application Architecture

This document describes the system architecture of an application that collects streaming data from Twitter and counts the number of times each word appears in the Twitter feed. This application is primarily built using Apache Storm technology supported by scripting in Python and PostgreSQL.

Application Idea

Capturing and analyzing live twitter data around a business interest area can provide a deeper understanding of the current social trends and demands. Historical data can provide information on the mainstream trends over a certain period of time, but live data can provide immediate and real-time insight. For example, a person who manages the ads for the TV programs can capture the live Twitter trends at the time of the live show to engage more TV viewers during a popular TV program (i.e. contextual advertising).

To demo such an application, this system collects tweets in real time, filters then based on certain conditions (such as remove hashtags, Retweets, URLs and other special characters) and stores them in a database. Then, simple analytics are run on the stored tweets to obtain the word counts of the words found in twitter feed.

Description of Architecture

The Figure below shows the basic architecture used in the Apache Storm framework for tweet collection and processing. First, a Tweet-spout is configured to collect tweets using the Twitter API in real time. Twitter provides developers with access to the real time tweets after signing up for access via an API. In this case, three spouts are turned on for data collection. Then, two bolts are configured to do the tweet processing. First, a parse bolt is configured such that it parses each tweet into a set of words after filtering any hashtags, special characters and any URLs. Second, a count bolt is configured to obtain counts of each word appearing in the twitter stream. The spouts and bolts are setup in a “shuffle” mode meaning that any tweet obtained by any spout can be sent to any of the three parse bolts. Similarly, all the three parse bolts can send data over to any of the two counting bolts which allows for parallelism in processing. The mechanism of this architecture is implemented using the topology file in the source code.

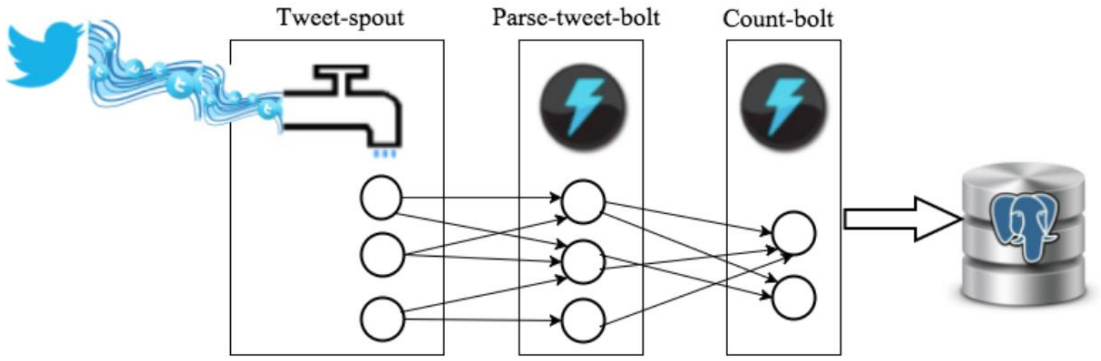

Figure 1 Topology for Tweet Collection

Directory and File Structure

The source code for this application has the following folder and file structure:

- Extweetwordcount – Name of the project folder
  - Topologies – Contains the topology file that controls the number of bolts and spouts and their configuration details
  - Src – source code folder for all python scipts
    - bolts – Two bolt files parse.py that parses the tweets into words and wordcount.py that counts the number of times each word occurs.
    - spouts – one tweet-spout.py file that collects, filters and sends the tweets to bolts for further processing.
    - Serve – serving scripts that can be run pre-determined queries against the database
  - Twitter Credentials.py : File that contains the twitter credentials
  - Postgres Database: Postgres database called 'tcount'

Instructions to Run

1. This project is run using the Amazon EC2 instance using the community AMI (UCB Spring 2016).
2. Here are the steps to recreate the results:
3. First, postgres needs to be started using the following command:
   sudo -u postgres pg_ctl -D /data/pgsql/data -l /data/pgsql/logs/pgsql.log start
4. Then, we need to create empty database and empty table for collection of tweets:
   su - postgres
   psql

   CREATE DATABASE tcount;

   \c tcount;

   CREATE TABLE tweetwordcount(word TEXT PRIMARY KEY NOT NULL, count INT NOT NULL);

5. Download the github repository to the EC2 instance.
6. cd to the exercise home directory
7. The twitter project can be started by using the command "sparse run –n tweetwordcount"
8. After letting the twitter stream run for a few minutes, you can terminate the program by ctrl+c.
9. Now, the database must be populated with all words in twitter stream and their counts. The serving scripts can be run now to see the specific results.
10. For example, cd src/serve/ and then run "python final_resultsts.py" shows all the words and their corresponding counts. "python plot_top20.py" shows the histogram of counts of the top 20 words.