Architecture: Intro to Streaming Application

MIDS 205 Exercise 2

R. Timothy Weeks

Concept

This application uses Storm, Postgres and the Twitter API to capture real-time word counts from processed tweets.

Deployment

To deploy the application, start an AWS EC2 instance using the community AMI 'UCB W205 Spring Ex 2 Image.' Also, attach the required Elastic Block Store volume with device name = '/dev/xvdf/'. After connecting to the running instance via SSH, git clone the following repositories into the root directory (use screenshot-setup for reference):

- **https://github.com/rtimothyweeks/exercise_2**
- **https://github.com/UC-Berkeley-I-School/w205-spring-16-labs-exercises**

After both repositories have been successfully cloned, use bash to execute the script exercise_2/setup.sh. This script first mounts the attached EBS volume, updates Python and pip, and installs required packages: redis, virtualenv, streamparse, psycopg2 and tweepy. The script then creates a new Storm project entitled ‘EXTwoTweetwordcount’ and moves required components from the MIDS and base repositories into the correct directories within the Storm project. Finally, the script calls create_db.py, which creates the Tweetwordcount table within the default Postgres database.

After setup.sh has completed, the final step for deployment is to update the Storm tweet spout with Twitter API credentials. The spout file is located in EXTwoTweetwordcount/src/tweets.py. Once credentials have been provided, running ‘sparse run’ in /root/ EXTwoTweetwordcount will begin stream processing.

Architecture

The general structure of the Storm project is provided by the topology, which articulates at the highest level the number of spout and bolt objects as well as their relationships to each other. This application implements the topology EX2tweetwordcount.clj which creates one spout, and one of two different types of bolts, a ‘parse-tweet’ bolt and a ‘count’ bolt. The spout uses the Python package Tweepy to initialize a tweet-stream listener and ‘Tweets’ class with a method ‘next_tuple’ which emits a single tweet when called. The output of the tweet spout is passed into the ‘parse-tweet’ bolt which removes all non ASCII characters and omits words beginning with commonly used special characters on Twitter like ‘@’ or ‘#’ before passing its output to the ‘count’ bolt. The count bolt logs a running count of words the stream has processed to the console and uses an update-else-insert strategy with the Postgres database to increment a particular word’s running count.

Serving Scripts

Two serving scripts 'finalresults.py' and 'histogram.py' can be used to explore results stored in Postgres. Passing a single word to finalresults will return the total wordcount for that term; execution without an argument will return all wordcount results. The histgram script requires two integer arguments and will return all words with counts between the two arguments.