Exercise 2
w205, section 2
Ramsey Magaña
architecture.pdf

Twitter Application: Tweet Word Count

Application Description:

This application is designed to processes and stores streaming data from Twitter.com. It parses words from live tweets and aggregates counts of words, which are stored in postgres data base. Once data is stored in database, the database could be called from terminal API with psycopg package to use python scripts to call database. Additionally, some analytical scripts e.g. finalresults.py, histogram.py, are included.

Data Architecture:

1. TWITTER STREAMING DATA
2. STORM-PROCESSING LATER
   - 3 SPOUTS
   - 3 WORD PARSE BOLTS
   - 2 WORD COUNT BOLTS
3. POSTGRESQL DATASTORAGE (e.g. EC2 Instance)
4. psycopg API

How to Use:

1. Set up environment

1.1 The code for this exercise can be found in the main course repository, which you can clone from git@github.com:UC-Berkeley-I-School/w205-spring-17-labs-exercises.git.

1.2 Create an EC2 instance with Hadoop is automatically started and stopped as part of the init state scripts. Following community AMI could be used:

AMI Name: UCB MIDS W205 EX2-FULL

AMI ID: ami-d4dd4ec3

Also attach and mount the EBS volume at /data.

1.3 Select/create project directory and in said install sparse quick start

e.g. [user@ my_project_directory] $ sparse quickstart extweetwordcount

1.4 Clone github repository into your project folder (link: https://github.com/ramagana/w205_2017_fall)

1.4.1 delete existing spouts and bolts

e.g. ./my_project_directory/extweetwordcount/src/spouts/words.py

e.g. ./my_project_directory/extweetwordcount/src/bolts/wordcount.py

e.g. ./my_project_directory/extweetwordcount/src/bolts/parse.py

1.4.2 copy in existing spouts and bolts

Table 1: Table of files to copy from w205_2017_fall/exercise_2 filesystem to my_project_directory

Name of the program	Location
tweets.py	exercise_2/extweetwordcount/src/spouts/
parse.py	exercise_2/extweetwordcount/src/bolts/
wordcount.py	exercise_2/extweetwordcount/src/bolts
Twittercredentials.py	exercise_2/
hello-stream-twitter.py	exercise_2/
tweetwordcount.clj	exercise_2/extweetwordcount/topologies/
db_create.py	exercise_2/
setup_tweetwordcount_app.sh	exercise_2/
psycopg-sample.py	exercise_2/

Note: that destination folders should have the same taxonomy within my_project_directory as the have in the w205_2017_fall/exercise_2

see: screenshots/high_level_filesystem.png, screenshots/spouts_bolts_direcotry.png

2. Create Application

2.1 Acquire Application Credentials from Twitter
    2.1.1 Login to Twitter(https://www.twitter.com/).
    2.1.2 Visit https://apps.twitter.com/ and click on "Create New App".
    2.1.3 Agree to the terms,and click "Create your Twitter Application"
    2.1.4 Click on "Keys and Access Tokens" tab and "Create my access token"

2.2 Following information is will be used and needed to be updated in the Twittercredentials.py and tweets.py found in my_project_directory/ . See: screenshots/Twittercredentials_img.png & screenshots/tweets_img.png .

     A consumer key that identifies your application.
     A consumer secret that acts as a password for your application.
     An access token that identifies your authorized access.
     An access token secret that acts as a password for that authorized access.

2.3 Run the setup_tweetwordcount_app.sh bash script. This will install python packages need to connect to postgresql as well as create tcount database and tweetwordcount table to store words and counts emitted from the count-bolt

Note: if you run setup_tweetwordcount_app.sh multiple times the script will as if you wish to delete existing tweetwordcount table in existing tcount database. See: screenshots/setup_tweetwordcount_app_img.png to see set up code.

3. Run Application

3.1 Run application to connect to Twitter Storm with copied topology. See: screenshots/sparse_run_result.png & screenshots/topology_img.png
     e.g. sparse run

3.2 Use finalresults.py to get counts from the twitter stream for all words or a specific word by passing an argument. See: screenshots/finalresults_img.png

e.g. $ python finalresults.py hello
Total number of occurrences of of "hello": 10

e.g. $ python finalresults.py
$ (<word1>, 2), (<word2>, 8), (<word3>, 6), (<word4>, 1), ...

3.3 Use histogram.py to get the distribution of words between two inclusive limits, like 3 and 8 in the example below. See:

e.g.$ python histogram.py 3,8
<word2>: 8
<word3>: 6
<word1>: 3
See: screenshots/Plot.png