Exercise 2:
Subject 205: Real Time Data Processing Using Apache Storm

Overview

The goal of this project is to build an end-to-end pipeline that sources data from twitter, processes the data, stores it in a relational database and finally the serving layer extracts data from the relational database and presents a summary.

High level Architecture

For the implementation, python was used as a wrapper to code the storm wrapper that sources data from twitter stream and process it. Once data has been processed, the python wrapper connects to the relational database (postgres in this case) and inserts/updates records. In the final step, a python script connects to the postgres db and prepares a summary of the dataset to answer user questions.

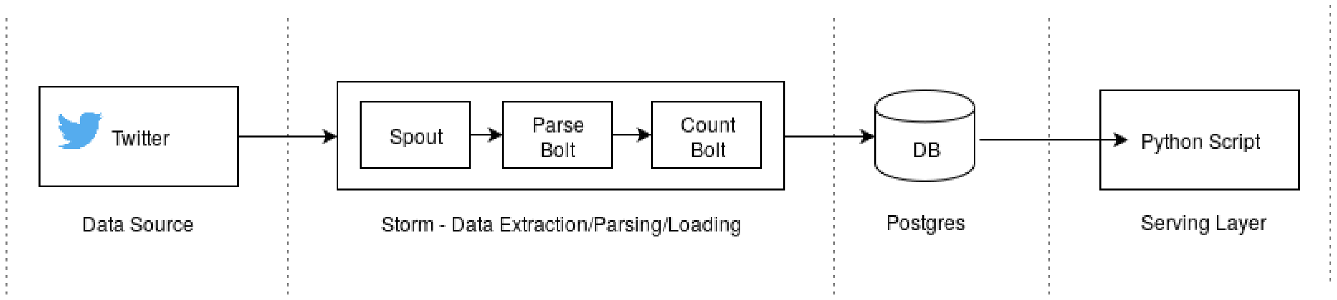


Figure 1

Dependencies

Following is a list of dependencies that need to be satisfied to setup and run this application.

a) Below python libraries were utilized and should be setup before running this application

- psycopg2 – to connect to postgres DB and run dml operations
- tweepy – to connect to twitter stream
- streamparse – to run python code against real-time stream of data via apache storm

b) Server instance setup – following AMI can be used to setup an EC2 instance that includes storm and postgres.

AMI Name: UCB MIDS W205 EX2-FULL
AMI ID: ami-d4dd4ec3

c) Postgres initialization – to initialize postgres, following commands need to be executed

service postgresql init # initialize the db
service postgresql start # start the instance

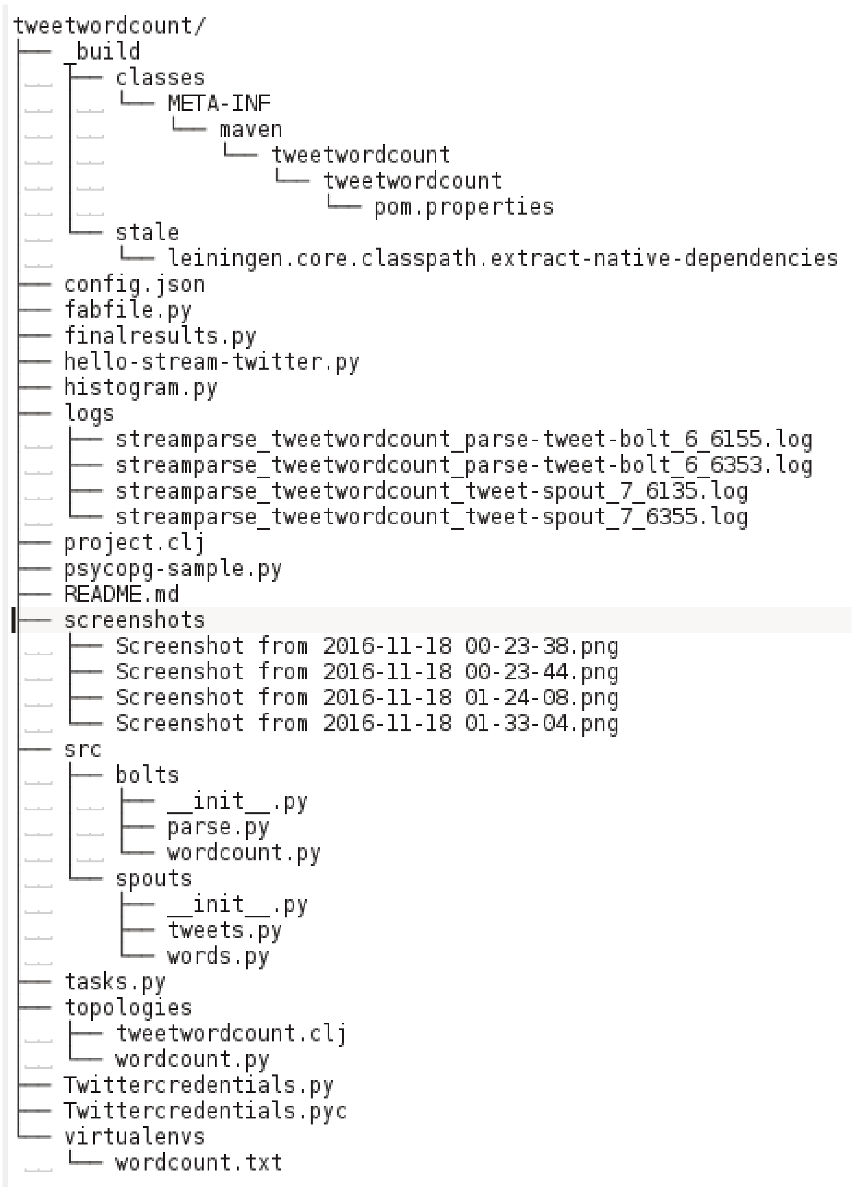
d) Twitter account – an application needs to be created on twitter using an active account. This would help create the credentials and access keys to connect to twitter.

Directory structure

Below screen-shot shows the directory structure of the application that was setup. Here's a summary of the key directories/files

- hello-stream-twitter.py – this program can be used to test connectivity to twitter.
- Twittercredentials.py – this file stores the credentials to runt the test application

- src/spouts/tweets.py – this is the spout that connect to twitter stream to pass on the data to the parsing bolt
- src/bolts/parse.py – this program parses words from the twitter stream
- src/bolts/wordcount.py – this program connects to the postgres db to insert/update counts
- topologies/tweetwordcount.clj – this config file defines the topology of the storm app setup
- histogram.py – this program connects to the postgres db to extract data for serving layer



```
tweetwordcount/
├── _build
│   ├── classes
│   │   └── META-INF
│   │       └── maven
│   │           └── tweetwordcount
│   │               └── tweetwordcount
│   │                   └── pom.properties
│   └── stale
│       └── leiningen.core.classpath.extract-native-dependencies
├── config.json
├── fabfile.py
├── finalresults.py
├── hello-stream-twitter.py
├── histogram.py
├── logs
│   ├── streamparse_tweetwordcount_parse-tweet-bolt_6_6155.log
│   ├── streamparse_tweetwordcount_parse-tweet-bolt_6_6353.log
│   ├── streamparse_tweetwordcount_tweet-spout_7_6135.log
│   └── streamparse_tweetwordcount_tweet-spout_7_6355.log
├── project.clj
├── psycopg-sample.py
├── README.md
├── screenshots
│   ├── Screenshot from 2016-11-18 00-23-38.png
│   ├── Screenshot from 2016-11-18 00-23-44.png
│   ├── Screenshot from 2016-11-18 01-24-08.png
│   └── Screenshot from 2016-11-18 01-33-04.png
├── src
│   ├── bolts
│   │   ├── __init__.py
│   │   ├── parse.py
│   │   └── wordcount.py
│   └── spouts
│       ├── __init__.py
│       ├── tweets.py
│       └── words.py
├── tasks.py
├── topologies
│   ├── tweetwordcount.clj
│   └── wordcount.py
├── Twittercredentials.py
├── Twittercredentials.pyc
└── virtualenvs
    └── wordcount.txt
```

Running the application

Once twitter credentials have been updated in Twittercredentials.py, hello-stream-twitter.py can be executed to test connectivity to twitter
Command: python hello-stream-twitter.py

To execute the data sourcing part (this would connect to twitter, parse the data and update the postgres db), twitter credentials need to be updated in src/spouts/tweets.py before executing the application.
Command: cd tweetwordcount; sparse run

For serving layer, finalresults.py can be executed to connect to the db and extract data out.
Command: python finalresults.py

Note: the database parameters (schema name, tablename, id, password, port # have been set to default in the python files where connection is established, so depending on your setup, the connection params may need to be updated).