Architecture document, Exercise 2

Author: Paul Durkin

Directory and file structure

There are some important files and directories under the GitHub project. The following table outlines them.

Directory	Description
extweetwordcount/	The streamparse structure for the application including the source code, topology, and scripts for setup and querying (details in this table)
screenshots/	3 screenshot images, required for submission in this exercise
Plot.PNG	A barchart of the top 20 words, required for submission
Readme.txt	Text document detailing the prerequisites, setup, running and use of this application
extweetwordcount/Scripts	The location for: **finalresults.py** – python script to query counts for one or all words **histogram.py** – python script to query words within a range of counts **setupdb.py** – set up the initial database
extweetwordcount/src/spouts/	Location of the python spout script **tweets.py** which is modified to connect to Twitter and pull in the tweets using tweepy
extweetwordcount/src/bolts/	Contains two python bolt scripts for this project: **parse.py** – python script to break the tweets into individual words **wordcount.py** – the modified python script that increments the count for every word found (adding new ones if they do not already exist).
extweetwordcount/topology/	Contains the topology file that matches the one required for exercise 2. It has • 3 spouts each with a connection to two parse-tweet-bolts • 3 parse-tweet-bolts, one with a connection from all three spouts, one with a connection from two and the last with a connection to one of them • 2 count-bolts, one that takes input from all three parse-tweet-bolts and one that take input from only two of them.

Table 1: Files and Directories

Application idea

The purpose of this application is to stream tweets from Twitter, parse them into individual words and keep a count of each word in a Postgres database.

Also, there are two simple tools are provided to query the database for information about the tweets. See the section on running the application for more detail on these.

Architecture description

At a high-level application takes a stream of tweets from Twitter, into a streamparse generated application that works on top of Apache Storm, and after processing writes the counts of the words found in a Postgres database, see Figure 1 High-Level Architecture

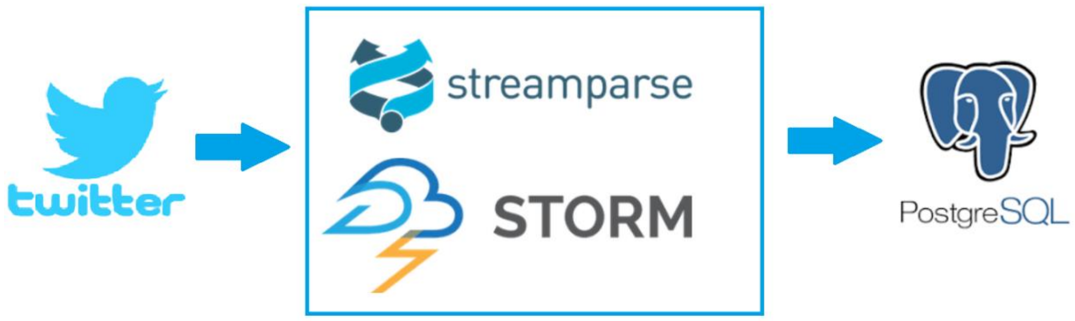

Figure 1 High-Level Architecture

Here data is ingested in real-time from Twitter, processed through an Apache Storm architecture to parse the data, and then persisted in a Postgres RDBMs.

The requested storm topology (from the exercise requirements) looks like the Figure 2: Detailed Apache Storm Architecture:

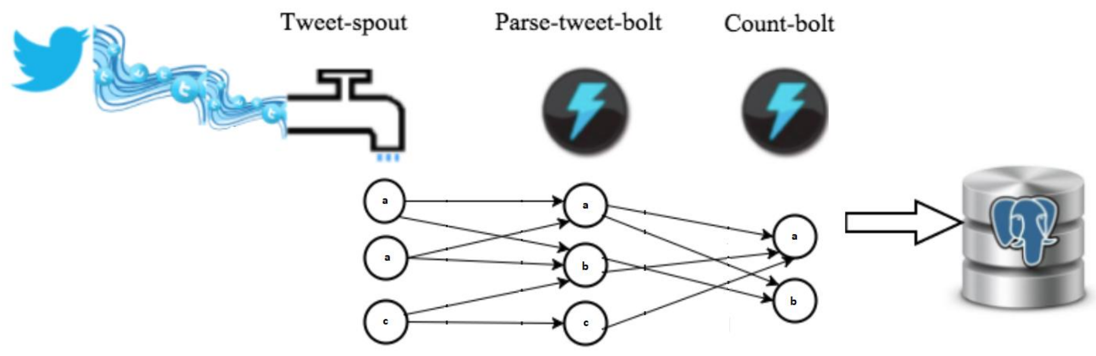

Figure 2: Detailed Apache Storm Architecture

This shows three tweet-spouts connected to twitter, streaming data to three Parse-tweet-bolt bolts in an unbalanced way. There are 3 connections into one bolt, 2 into the second and only 1 connection into the third. These three bolts stream their output into 2 Count-bolts, all three streaming to one of the two bolts and two of them streaming to the second bolt.

File dependencies

There are no extra file dependencies other than those imposed by using Streamparse and the prerequisites of the application. These are:

1) UCB MIDS W205 EX2-FULL image for EC2 instance
2) Postgres installed and running
3) A copy of the repository from directory exercise_2 and below
4) tweepy installed in python. To do this run the following as root:
   ```
   pip install tweepy
   ```
5) Before running for the first time set up the database. To do this go to the exercise_2/extweetwordcount/Scripts directory and run:
   ```
   python setupdb.py
   ```

Running the application

Once all the prerequisites from the section above on file dependencies have been provided then running the application is done using the following procedure:

1) Change directory into the exercise_2/extweetwordcount on the EC2 instance
2) Run:

   ```
   sparse run
   ```

This should start up the application and within about 30 seconds will start to pull in and process live tweets.

Running the utilities provided

There are two utility scripts provided:

1) finalresults.py: This script extracts either all words from the database with their counts or a specific word with its' count. It is a python script and if it is provided a single command line parameter, it searches for this word returning a count of zero if the word is not found or returning the count if it is found.

   The following is an example of its' usage and output:
   ```
   [w205@ip-172-31-89-53 Scripts]$ python finalresults.py The
   Total number of occurences of "The": 3627
   ```

2) histogram.py: This script returns all words that have count between two values. It takes two command line parameters, a lower count and an upper count.

The following is an example of its’ usage and output:

```
[w205@ip-172-31-89-53 Scripts]$ python histogram.py 1000 1100
her: 1002
up: 1041
more: 1043
don't: 1052
no: 1067
If: 1085
or: 1090
```