Sungtai (Max) Shen
W205

Exercise 2: Real Time Data Processing Using Apache Storm

Key files directory

<u>EX2Tweetwordcount</u>

- All Exercise 2 materials are inside EX2Tweetwordcount.
- The directories follow a structure similar to the one when using the command $sparse quickstart wordcount.

EX2Tweetwordcount/writeup - architecture.pdf, screenshots, readme.txt, Plot.png

EX2Tweetwordcount/topologies/tweetwordcount.clj - topologies file for Storm.

EX2Tweetwordcount/src/spouts/tweets.py - file for spout with Twitter API login credentials.

EX2Tweetwordcount/src/bolts/parse.py - first part of the bolts that parse Tweets.

EX2Tweetwordcount/src/bolts/wordcount.py - second part of the bolts which receives the parse Tweets from parse.py then performed wordcount as well as updating the pre-created Postgre table tweetwordcount in database tcount.

EX2Tweetwordcount/finalresults.py - this is the Python program that performs the analysis which brings back count based on an input word.

EX2Tweetwordcount/histogram.py - this is the Python program that performs the histogram analysis (finding words with count >= number 1 and <= number 2).

Design Principal

Storm Process
As mentioned briefly above, the EX2Tweetwordcount follows similar structure provided from the class example. The program tweets.py setup the Twitter credientials then starts the streaming process.

The first bolt prase.py parses the streaming tweets then passes them to wordcount.py to perform wordcount.

Wordcount.py is modified to not only peforming wordcounts, but also updating a pre-created table tweetwordcount in a Postgre database called tcount.

Bolts - wordcount.py and Postgre update
The wordcount.py recevies streaming words from parse.py then uses a counter program (self.counts = Countr(), self.counts[word] += 1). For each of the incoming word, wordcount.py perform a SELECT statement to retreive all the words currently exist in Postgre table tweetwordcount. If the word already exists, then an UPDATE to the Postgre table for this word is performed. If the word doesn't exist, then an INSERT to the Postgre table for this word is performed.

For example, if (word, count) = (dog, 5) is the current word count coming out the Storm. If table tweetwordcount has (word, count) = (dog, 1), an UPDATE will result in (dog, 6) in the table tweetwordcount. Existing count in tweetwordcount + current count = new updated wordcount. Suppose the word "dog" is not in tweetwordcount table, then an INSERT will result in (dog, 5).

Also, because self.counts[word] is cumulative, at the end of each update process, self.counts[word] will be reset to zero. I took this approach because the current setup for the Storm wordcount program is a streaming process that requires manual shut down. By setting the counter to zero, my goal is to update counts in words already exist in tweetwordcount instead of overwriting the count with new stream data. (For example, the first time the program is launched, the count for the word "MIDS" is 3. The next time the count is 2 when the program stopped. Instead of replace 3 with 2, my program adds count 2 to get a count of 5.

Postgre SQL
Before launching the wordcount process with Storm. A database called tcount was created. Inside tcount is a table called tweetwordcount with two columns word of type varchar(100) and count with type integer.

The design principle for the database is the database only store unique word with a cumulative count from every time a user launch the Storm wordcount program. For example, the word "cat" with count 120 means the word "cat" has 120 counts from past streaming process and the word "cat" is unique among all the records in column "word".

Query results:
The two programs finalresults.py and histogram.py simply takes in the system inputs then do a query on the Postgre table tweetwordcount. The finalresults.py takes in a key word from system input, then do a SELECT statement with word equals to the input. If there is no input, the logic default to a SELECT * which brings back all records.

The histogram.py takes in two system inputs, both are numbers. The program also do a SELECT statement with a clause of COUNT BETWEEN the two numbers.