W205 Exercise 2: Tweet Word Count

Erika Lawrence

Introduction

The goal of this exercise is to create an end to end streaming application which takes twitter data in through a storm spout, parses the words, and stores the count of each word occurrence in a postgres table. Two python scripts can be used to query the results: one for the count of a word supplied in a parameter, and another for the words with counts between numbers supplied in parameters.

Directory and File Structure

Files under the root/extweetwordcount directory are stored as follows:

- Storm Files (all required)
 - topologies:
 - tweetwordcount.clj file directing spouts and bolts processing
 - src/spouts:
 - "tweets.py": pulls tweets in a dynamic stream
 - src/bolts:
 - parse.py: tokenizes all tweet words
 - wordcount.py: keeps a running count of tokens and stores results in the postgres table tweetwordcount
- Result Scripts
 - finalresults.py: displays the count of a parameter token word, or all word counts if no parameter is provided
- spouts tweets.py 🛀 __init__py arse.py a wordcount.py 🚵 __init___py logs _resources topologies tweetwordcount.clj virtualenvs _build config.json READVE.md histogram.py project.clj tasks.py finalresults.py 🐒 fabfile.py

Fig 1: extweetwordcount file structure

histogram.py: displays words with counts between parameters given

Application Overview

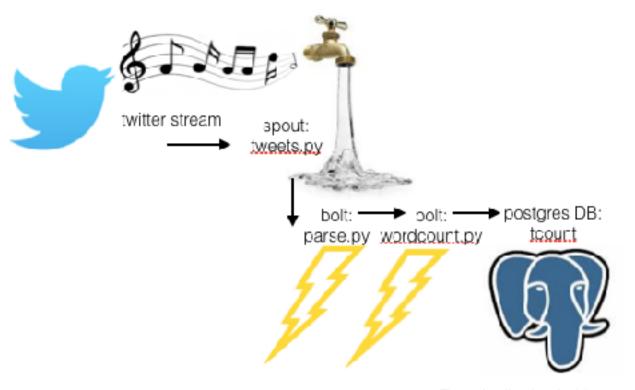


Fig 2: Application Architecture

The Tweet Word Count application consists of

1. a storm component which streams tweets through a spout into a word parser and counter bolt:

```
📵 🧶 🌔 🔃 edmin — root@ip-172-31-14-239://root/extweetwordcount — ssh -i ucb_aws_1.pem root@ec2-54-157-255-13.compute-.
62857 [Thread-39] INFO backtype.storm.task.ShellBolt - ShellLog pid:3547, name:count-bolt 1: 261
62858 [Thread-39] INFO backtype.storm.task.ShellBolt - ShellLog pid:3547, name:count-bolt DIO: 1
62050 |Thread_30| INFO
                           backtype.storm.task.ShellBelt - ShellLeg pid:3547, name:count-bolt NOT: 2
                          backtype.storm.task.ShellBolt - ShellLog pid:3547, name:count-balt UP: 3
62854 [Thread-38] INFO
62465 [Thread-80] INPO backtype.storm.task.ShellHolt - Shelltog pid:3547, name:count-bul: ARD: 5
62866 [Thread-39] INFO
                           backtype.storm.task.ShellEglt - ShellLog pid:3547, name:count-bolt LEAVE: 1
62068 |Thread-301 INFO | backtype.storm.task.ShellBolt - ShellLog pid:3547, name:count-bolt looking: 8 62075 |Thread-301 INFO | backtype.storm.task.ShellBolt - ShellLog pid:3547, name:count-bolt I: 252
62977 [Thread-38] INFO
                          backtype.storm.task.ShellBolt - ShellLog pid:3547, name:count-balt to: 253
62078 [Thread-80] INFO
                          backtype.storm.task.ShellHolt - Shelltog pid:3547, name:count-bolt DID: 2
62879 [Thread-39] INFO backtype.storm.task.ShellBolt - ShellLog pid:3547, name:count-bolt give: 9
62031 [Thread-30] INFO | backtype.storm.task.ShellBolt - ShellLog pid:3547, name:count-bolt NOT: 3
62034 |Thread_30| INFO
                          backtype.storm.task.ShellBelt - ShellLeg pid:3547, name:count-bolt RUN: 1
62935 [Thread-38] INFO
                          backtype.storm.task.ShellBolt - ShellLog pid:3547, name:count-bolt to: 254
62488 [Thread-88] INFO backtype.storm.task.ShellHolt - Shelltog pid:3542, name:count-bolt I: 263
```

Fig 3: extweetwordcount streaming

2. a postgres database easily queried for result sets

```
[root@ip-172-31-14-239 extweetwordcount]# python finalresults.py Lorax
  Total number of occurences of 'Lorax':1
  [root@ip-172-31-14-239 extweetwordcount]# python finalresults.py puppies
  Total number of occurences of 'puppies':1
  [root@ip-172-31-14-239 extweetwordcount]# python finalresults.py was
  Total number of occurences of 'was':11
  [root@ip-172-31-14-239 extweetwordcount]#
                                  Fig 4: final results.py script results with parameter
[[root@ip-172-31-14-239 extweetwordcount]# python finalresults.py
('!', 2)
('!!', 1)
('#CleanPowerPlan', 1)
('$0.00', 1)
('%^*!', 1)
('&amp', 18)
('*coughs*', 2)
('-', 7)
('--', 1)
('-Wife', 1)
('/', 1)
('//forgets', 1)
('1', 1)
('10', 2)
('1000', 1)
('12', 1)
                                 Fig 5: final results.py script results without parameter
[[root@ip-172-31-14-239 extweetwordcount]# python histogram.pv 8 9
[("don't", 9)
('When', 9)
('get', 9)
('right', 9)
('who', 9)
('think', 9)
('what', 8)
('know', 8)
('as', 9)
('some', 8)
('or', 9)
[root@ip-172-31-14-239 extweetwordcount]#
```

Fig 6: histogram.py script results

Run the Application

- Copy the extweetwordcount directory to an AWS instance using UCBMIDSW205EX2-FULL
- 2. Replace the existing twitter application credentials with a new set.
- 3. Attach a disk to the instance that includes postgres.
- 4. Create a postgres database and table:

- 5. From the extweetwordcount directory, type 'sparse run' and hit enter. Use Ctrl-C to break when ready
- 6. Use finalResults.py and histogram.py to review the results.

Results

