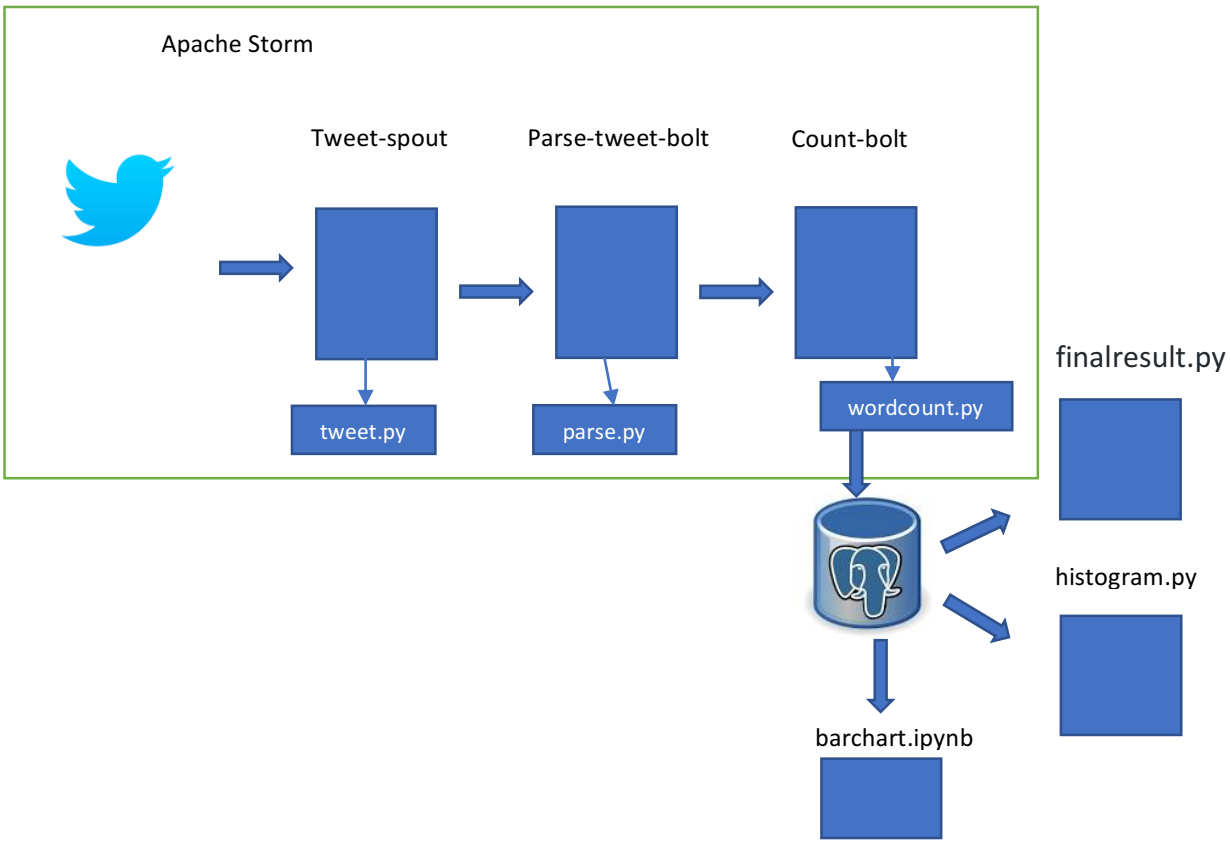
1. Architecture of the application



Processing flow of the application

- Tweet stream is captured in tweet.py managed by the tweet-spout.
- Parse-tweet bolt is used to parse words from tweet stream
- Count-bolt count the words and store the word into postgresDB. In wordcount.py where the detail logic is implemented, a check into the DB table is implemented to test if the word is already present before deciding if an insert or an update is needed
- The finalresult.py application is used to display count of word passed as argument to this python program
- Histogram.py is used to query words that meet requirement of the number of counts on lower and upper limits passed as parameters
- Barchart program is used to produce the bar chart of top 20 most common words

Topology.

The topology of the application is stored in tweetwordcount.clj

```
(ns tweetwordcount
  (:use     [streamparse.specs])
  (:gen-class))

(defn tweetwordcount [options]
   [
    ;; spout configuration
    {"tweet-spout" (python-spout-spec
          options
          "spouts.tweets.Tweets"
          ["tweet"]
          :p 1
          )
    }
```

```
    ;; bolt configuration
    {"parse-tweet-bolt" (python-bolt-spec
          options
          {"tweet-spout" :shuffle}
          "bolts.parse.ParseTweet"
          ["word"]
          :p 2
          )
     "count-bolt" (python-bolt-spec
          options
          {"parse-tweet-bolt" ["word"]}
          "bolts.wordcount.WordCounter"
          ["word" "count"]
          :p 2
          )
    }
  ]
)
```

The application is run using

```
sparse run --name=tweetwordcount.clj
```

this is because there're other configurations and tweetwordcount.clj is the configuration for this application.