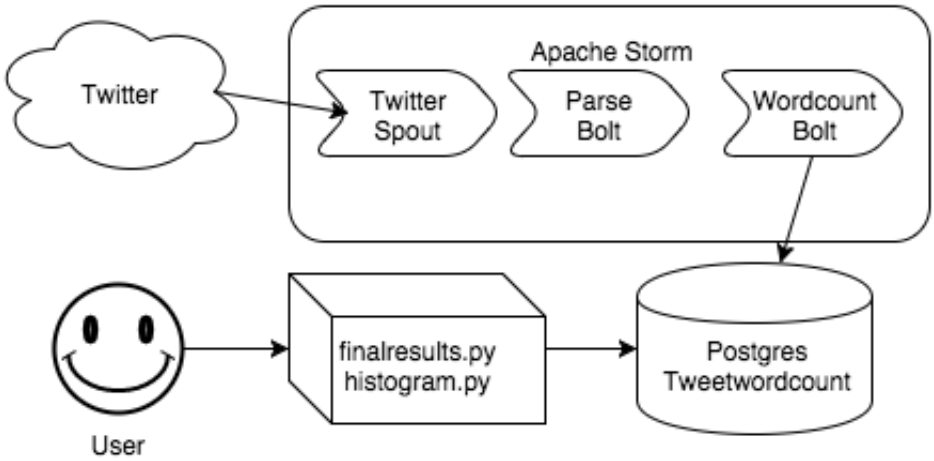
W205 Exercise 2 Architecture/Design

John A. Bocharov

Top-Level Architecture

The high-level architecture of the system consists of the following components:

- Twitter's public REST API
- An Apache Storm cluster with the `Twitter Spout`, `Parse Bolt`, and `Wordcount Bolt`
- A Postgres database with the key `Tweetwordcount` table
- A set of user-facing scripts (`finalresults.py`, `histogram.py`) for consuming the results



Directory Layout

The exercise started with a provided reference implementation for some parts of the project. The layout below focuses on files I created or modified.

All the files for the submissions are contained under the exercise2/ folder (omitted below for brevity).

```
exercise2/
```

Under `exercise2`, the following files and directories have notable new code and changes:

```
tweetwordcount/ <- the deployable sparse application
               src/
                   spouts/
                          tweets.py <- Added working credentials
                   bolts/
                         wordcount.py <- Added write to Postgres

finalresults.py <- script for reading all or specific word results
histogram.py <- script for reading words results in a range of counts

Plot.png <- bar chart of top 20 (21 due to ties) words and counts
Plot.out <- data behind Plot.png
Plot.ipynb <- IPython 2 notebook used to generate Plot.png
Plot.sh <- shell command used to generate Plot.out

Twittercredentials.py <- Working credentials to exercise Twitter

Architecture.pdf <- this document, in PDF format
Architecture.md <- this document, in source Markdown format
Architecture.png <- high-level architectural diagram above

screenshots/ <- screenshots of the process

Readme.txt <- text version of instructions to run the solution
Readme.md <- native Markdown version of intructions
```

Dependencies

This solution does not introduce new dependencies beyond those already existing in the starting-point reference implementation. IPython notebook 2 was used to generate the plots but does required to view them (additionally, GitHub acts as an effective viewer for ipynb).

https://github.com/jbocharov-mids/w205-labs-exercises/blob/feature/exercise2/exercise_2/Plot.ipynb

An overview of existing dependencies.

Twitter REST API

The solution depends on access to the Twitter REST API's streaming methods using the OAuth 4-tuple of (`consumer key` , `consumer secret` , `access token` , and `access token secret`).

Apache Spark

The Spark layer requires the Apache Spark runtime, streamparse, and additionally the `tweepy` and `psycopg2` Python packages.

Postgres

Postgres needs to be configured and running at (relative to the Spark cluster) `localhost:5432` with the user `postgres` identified by the password `pass` having full rights to the `Tcount` database.

Once the database is created using

```
createdb Tcount
```

An administrative shell

```
psql -s Tcount
```

can be used to create the requisite user

```
CREATE USER postgres PASSWORD 'pass';
CREATE DATABASE Tcount;
GRANT ALL PRIVILEGES ON DATABASE Tcount TO postgres;
```

Finally, the `Tcount` database needs to have a single `Tweetwordcount` table with `word` character string column and `count` integer. It can be created with the following DDL:

```
CREATE TABLE Tweetwordcount
       (word TEXT PRIMARY KEY     NOT NULL,
       count INT     NOT NULL);
```

Python

All examples are designed to run on Python 2.7.

Application Idea

One of my W205 classmates runs a small, nimble content marketing company. He can use this solution to understand what words are popular and trending in near-real-time, to better target which videos to release when, and what words to use in describing them for maximum exposure.