W205 – EXERCISE 2 – SRIRAM RAO - ARCHITECTURE

GitHub Repository

https://github.com/sriramrao1/exercise_2/

This is a public repository. So you should be able to make Pull requests.
I have provided 'DwMcclary' with access to contribute as well.

Data Source Parameters

Database:	tcount
Table:	tweetwordcount
DB Username used in the code:	w205
DB Password used in the code:	postgres

Repository File Structure:

File Name	Description
extweetwordcount	Name of the stream parse project. Folder contains all files to execute the twitter project
Screenshots	Folder has the screenshots of execution of the application

Finalresults.py	1. This script gets a word as an argument and returns the total number of word occurrences in the stream. For example:
	$ python finalresults.py and
	Count of records in tcount = 30
	word = and
	count = 30
	
	2. Running finalresults.py without an argument returns all the words in the stream and their total count of occurrences, sorted alphabetically in an ascending order, one word per line. For example:
	
	$ python finalresults.py
	Word = and
	Count = 30
	
	Word = to
	Count = 33

	…
Histogram.py	This script gets two integers k1,k2 and returns all the words that their total number of occurrences in the stream is more or equal than k1 and less or equal than k2. For example: $ python histogram.py 3,8 <word2>: 8 <word3>: 6 <word1>: 3
Twittercredentials	Stores the twitter credentials to run the app

Please note that the name of the project is NOT ex2tweetwordcount. During execution, the ex2tweetwordcount name errored out with the message that name of the project cannot be alphanumeric. Hence the change in the name.

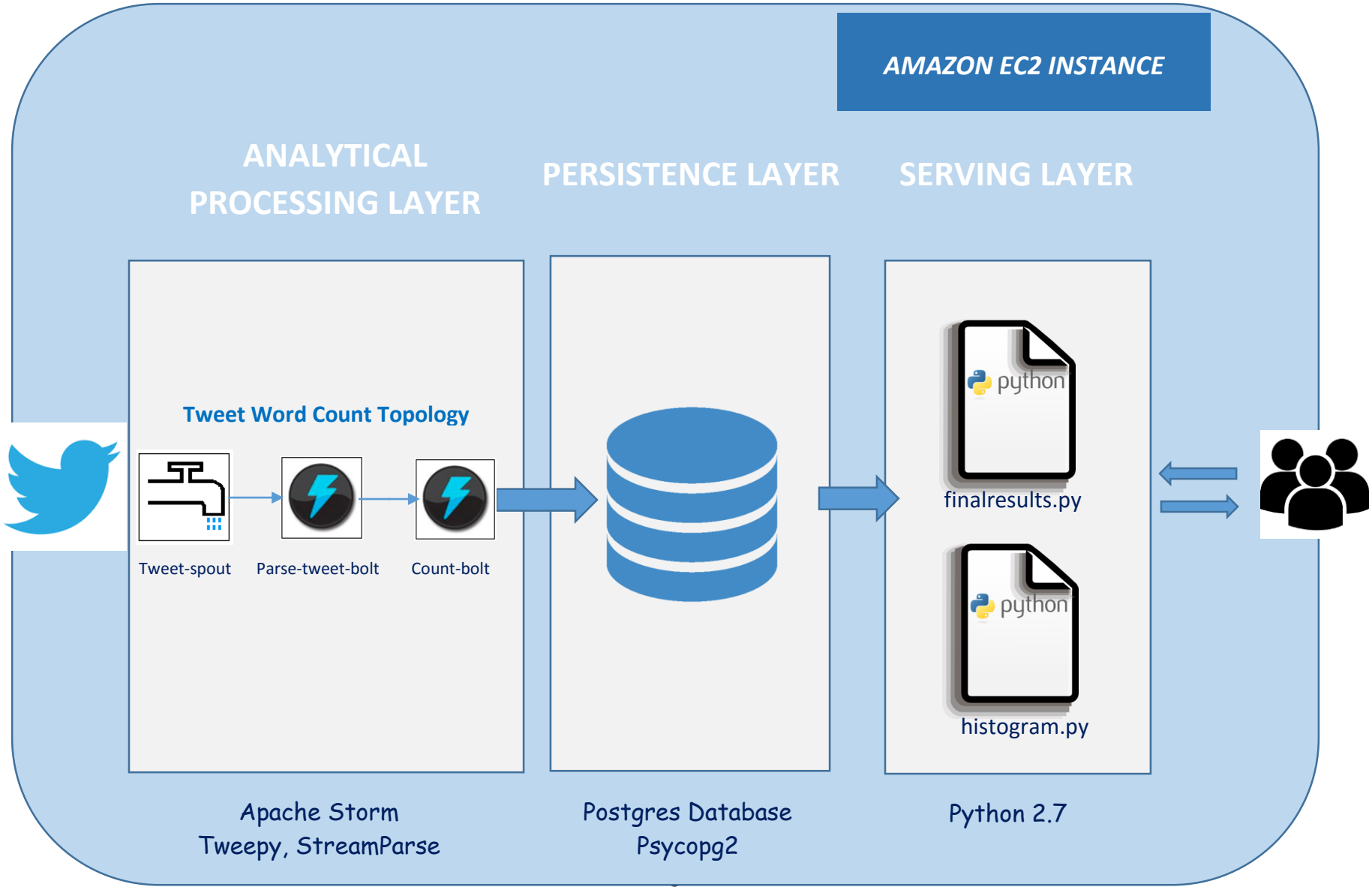
Architecture

The tweet count application is to count the list of words from a real time data stream source. The tweet count application consists of a three layered architecture that consists of:

1. Processing Layer
2. Persistence Layer
3. Serving Layer

This layered architecture allows for separation of responsibilities and selection of technologies that are most appropriate for that layer to meet the business purpose of the application. The application is hosted and executed within Amazon's EC2 instance.

W205 – Exercise 2- Architecture



Processing Layer:

In this layer, tweets from the twitter stream are collected, parsed, and counted using Apache Storm, streamparse and tweepy. In our application, the only analytical processing that is performed is the counting of words.

Apache Storm is a "distributed real time computation system" that allows reliable processing of streams of real time data such as the one from Twitter.

Streamparse allows running Python code against real-time streams of data via Apache Storm. With streamparse you can create Storm bolts and spouts in Python without having to write a single line of Java. It also provides handy CLI utilities for managing Storm clusters and projects.

Tweepy is a python library that allows processing of tweets from the Twitter API. It handles the authentication, session management, message routing.

In order to process the twitter data using Apache Storm, we use the topology represented in the diagram above:

- "tweet-spout": collects the tweets from the Twitter API
- "parse-tweet-bolt": parses the tweets from the "tweet-spout" in to words and emits it to the "count-bolt"
- "count-bolt": counts the words emitted from the "parse-tweet-bolt"

Persistence Layer:

The responsibility of this layer is to materialize the words and the count of words emitted. We could typically store this content in a Hive database, a text file, or in a dataframe. For this exercise, we choose to store the words and their counts in a **Postgres** database. We use the psycopg2 driver to connect to the Postgres database from our python application.

The database 'tcount' is created in the EC2 instance ahead of time. The 'tweetwordcount' table (schema consisting of word TEXT, count INT) stores the words and their counts.

Serving Layer:

The responsibility of this layer is to query the 'tcount' database for the words that are of interest to the user and provide their count. We have 2 python scripts:

- Finalresults.py – This script gets a word as an argument and returns the total number of word occurrences in the stream
- Histogram.py – This script gets two integers k1,k2 and returns all the words that their total number of occurrences in the stream is more or equal than k1 and less or equal than k2