Assignment #2

Large-Scale Computing for the Social Sciences
MACS 30123, Autumn 2020
Due Wednesday, November 11th at 11:59pm (CST)

For this assignment, you will submit a README file with your answers to the questions below, along with the code you used to produce your answers. You should commit these items to your Assignment 2 repository on GitHub and submit a link to your repository on Canvas for Assignment 2.

1. **Parallel Web Scraping** (8 Points). Your research group scrapes online bookstore web listings every 24 hours to identify and record current information about books that are listed. Currently, the scraping process is done serially on a single machine and it takes a long time for all of this information to be scraped at any scale. Your PI found out that you're learning about large-scale computing methods and has tasked you with accelerating the group's current scraping code.

   Currently, the workflow is as follows: Set up a SQLite relational database (.db file) to store book data and update changing book data, scrape general book listing pages for links to individual book product description pages (adding each book id and date last updated to a table in the database), and, finally, scrape individual book product description pages (adding individual information for the books to a separate table in the database). All of the database operations are performed using the Python `dataset` package. The serial code for this workflow is available here, along with the url information for the (fake) bookstore website that you should scrape data from as a proof of concept. This code is drawn from Baesens and vanden Broucke (2018).

   (a) (5 Points) You have the idea to speed the process up with PyWren, parallelizing the scraping process across many AWS Lambda functions. For this prompt, you should write this PyWren parallel scraping code. When you have a working parallel version of the code, time how long it takes for your parallel solution to run. What is the parallel speed-up in comparison to the original serial solution? What are some bottlenecks in your code that you can't effectively parallelize with PyWren? Two hints:

      i. Have each AWS Lambda instance scrape a batch of books (you should test to see which batch size is optimal), not an individual book.

      ii. AWS Lambda makes it fairly challenging to import and use non-native Python packages (importing these packages involves zipping up dependencies and loading them as Lambda Layers, which are beyond the scope of this course). For this reason, you should only use PyWren to perform the scraping component of your research workflow (and not your database upsertion/insertion steps).

(b) (3 Points) What is the logic behind using a relational database to store data in this scraping solution? Under what conditions might your research group want to scale up their single-server relational books database to one of the large-scale database solutions on AWS (or another cloud provider)? Consider the strengths and limitations of these different approaches to data storage and management.

2. **Identifying the Most-Used Words** (4 Points). Use `mrjob` to write code that identifies the overall top ten words used in the book descriptions that you scraped in (1). Don't worry about pre-processing the text data; just pass the raw text data into your mapper.

   You are welcome to input data directly from the SQLite database via SQL queries, following the guidance in the `mrjob` documentation. Alternatively, you can write all of the book descriptions in the `books.db` file into another text format and use that as the input data for your MapReduce job, employing the methods demonstrated in the lab.

   Note that if you choose to input data directly from the SQLite database, you will need to add a dummy `.txt` file with a line of (arbitrary) text in it in order to suppress `mrjob`'s call for an input file (which is not stated explicitly in the `mrjob` documentation). So, to run such a job from your terminal, you would write:

```
python mrjob_top10.py dummy.txt --database=books.db
```

3. **Streaming Stock Data** (5 Points). For this prompt, you will:

   (a) (2 Points) Set up an EC2 instance that uses `Boto3` to write real-time stock data (randomly generated using the code below) into a Kinesis stream (a producer). Optional: You are welcome to plug in code to stream real stock data from the API of your choice and set your own price target in 3(b).

```
import random
import datetime
import json

def getReferrer():
    data = {}
    now = datetime.datetime.now()
    str_now = now.isoformat()
    data['EVENT_TIME'] = str_now
    data['TICKER'] = 'AAPL'
    price = random.random() * 100 # Assume price is in USD
    data['PRICE'] = round(price, 2)
    return data
```

```
while True:
        data = json.dumps(getReferrer())
```

You should then set up another EC2 instance that uses `Boto3` to read the real-time stock data from your Kinesis stream (a consumer).

(b) (3 Points) If the stock price goes below $3, you should have SNS send you an email informing you of the current price and the exact time that the stock hit that price (by publishing the alert to a "Price_Alert" SNS topic using `Boto3`). As soon as you receive a price alert, you should terminate your EC2 instances and delete your Kinesis Stream, as well as your SNS topic. Screenshot the price alert in your email inbox and include the screenshot in your Assignment 2 README.

4. **Propose a Final Project Topic** (3 Points). In ∼250 words, propose a Final Project topic for this class (see Canvas site for more details on the requirements). Explain why your project idea helps to solve a social science research problem using large-scale computing methods and outline a schedule for completing the project by the deadline. You are welcome to meet with course staff and discuss your ideas with us before submitting your answer to this prompt.

References

Baesens, B. and S. vanden Broucke (2018). *Practical Web Scraping for Data Science: Best Practices and Examples with Python*. New York: Apress.