Author: Matthew Mills
Project: Bikeshare Research
Technical Proposal
Date: 03/12/2019

Data Archival System

Overview:

This document is a technical proposal for the software system to be used in gathering and archiving data for the Bikeshare research project. It will describe a set of software technologies whose integration and implementation should achieve the following goal:

- Data from feeds in the General Bikeshare Feed Specification shall be extracted and processed to remove redundant records. This data will then be stored in a database, ready to be used in a multi-variable regression model.

Each software component is described in detail, as well as the coding standards for this project and the intended operation of this future system.

Cloud Computing Instance:

One constraint in this project is that data needs to be collected from the web in real-time every day. The best way to achieve this goal is with a cloud instance, which can be configured to run scraping programs remotely and continuously. There are several cloud computing options to choose from: Amazon Web Services, Microsoft Azure, Oracle Cloud, IBM Cloud, Rackspace, VMWare, etc. Below is a cost analysis of each service:

VM Type US Linux	AWS 1Y RI Annual	Google 1Y CUD Annual	Azure 1Y RI Annual	IBM Monthly + 30% off Annual	AWS 1Y RI Annual /GB RAM	Google 1Y CUD Annual /GB RAM	Azure 1Y RI Annual /GB RAM	IBM Monthly + 30% off Annual /GB RAM
Standard 2 vCPU w Local SSD	$867	$884	$508	$764	$116	$118	$64	$95
Standard 2 vCPU no Local disk	$622	$524	$508	$624	$78	$70	$64	$78
Highmem 2 vCPU w Local SSD	$946	$1,013	$683	$998	$63	$78	$43	$62
Highmem 2 vCPU no Local disk	$850	$653	$683	$998	$56	$50	$43	$62
Highcpu 2 vCPU w Local SSD	$666	$751	$543	$418	$178	$417	$136	$209
Highcpu 2 vCPU no Local disk	$543	$391	$543	$418	$136	$217	$136	$209

According to this table, Microsoft Azure and Amazon Web Services (AWS) are the cheapest cloud computing options. Both services offer discounts for the first year of use. More specific information about each may be found at the following links:

- AWS:
  - https://aws.amazon.com/ec2/pricing/on-demand/
  - https://aws.amazon.com/ebs/pricing/

- Microsoft Azure:
  - https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/
  - https://azure.microsoft.com/en-us/pricing/details/managed-disks/

In cloud computing infrastructures, customers are billed separately for the virtual server that runs their applications, and the secondary SSD or HDD memory storage used by that server. From the above links, it is evident that Azure and AWS have comparable prices for virtual servers. However, Azure charges a flat rate for secondary storage, while AWS charges on-demand, so AWS will likely be the cheapest of the two options, since the amount of memory storage used by this project is unlikely to amount to multiples of the smallest unit provided by Azure. Thus, this document is proposing the use of AWS for the project's cloud computing services.

Assuming that AWS is selected as this project's cloud computing services, a remote virtual server from the AWS EC2 (Elastic Cloud Compute) services will be setup. For this to happen, it is requisite that a new AWS free-tier account be created with appropriate roles and access permissions for those working on the project. Once an account is created and an EC2 instance is setup, an AWS EBS (Elastic Block Storage) drive will be created and mounted to the EC2 instance. This EC2 instance is the virtual machine that will run the python and database applications for the archival process. It can be accessed remotely using SSH and a public IP address. Those working on this project shall be obligated to report the computing bill weekly, to prevent an egregiously large cost incurred by neglecting to check the rate at which expensive services are used long term.

Database:

The database system that will be used by this project shall be PostgreSQL implemented in AWS RDS (Relational Database Service). The postgres database shall be the central location which stores all large volumes of data pertaining to the project. It shall have one schema called "Master" and this schema shall have the following tables and respective schemas:

- Links (id int, company CHAR(15), city CHAR(15), url CHAR(100))
- Archive (id int, timestamp CHAR(20), company CHAR(15), city CHAR(15), bike_id CHAR(15), is_reserved int, is_disabled int, lat double precision, lon double precision, cluster int)

The table called Links shall store the urls for the "free_bike_status" (see GBFS Summary) data feeds from various cities for various bikeshare rental companies. It is from this table that the archival system shall ascertain a list of url's to scrape on a daily basis.

The table called Archive shall store the geographic rental trip information after it has been processed to remove redundant and noisy records.

Datapipeline:

AWS provides an automation service called "Datapipeline" that allows users to automatically run certain tasks and programs on their EC2 instances at given times and schedules. These tasks can include running python and shell scripts essential to the system's workflow. These scripts may also be set to run at specific times, which in the case of this project will likely be the beginning and end of business hours.

Data Archival System Workflow:

The following process shall occur upon deployment of this system:

1. A python script called scrape.py will do the following:
    a. Start a screen for each url from the Links table of the database
    b. Scrape the data from each url into the Archive table of the database called bikedata once a minute.
2. Every 24 hours, an AWS Datapipeline will call a program called consolidate.py to query the database and remove duplicate rows and nonessential records within the clusters.

The database will now have all usable records from that day from each city and company which contain the latitude and longitude information for each bike's trips. This data may be used at a later stage of the project to discover when bikes enter and leave geographic zones, and it is from this information that the regression model will be formed.

Folder Structure on EC2 Instance

The code and data stored on the EC2 instance for the Archival system must abide by the following standard:

I. Home (/home/ubuntu/)

A. RESEARCH

1. Scrape.py
2. Consolidate.py
3. OBSELETE
4. CSV_REPOSITORY

Coding Standards

All code written for this project shall abide by the following standards:

1. All loops, functions, if statements, and any other logically nested block of code shall be indented with 4 spaces, regardless of the programming language being used.

2. Ample whitespace shall be used to promote human-readability of code.

3. All variable names must be meaningful and relevant words. Single or double letter variable names are disallowed.

4. All function names and variable names shall be lowercase. If it makes sense to use more than one word in a single name, each word shall be lowercase and separated from the next word by an underscore.

5. All class names shall begin with uppercase letters. If it makes sense to use more than one word in the name, each word shall be uppercase, and not be separated from the next word, but all words shall occur consecutively with no spaces as one phrase.

6. All files shall be strictly lowercase.

7. All database names, schemas, and tables shall be strictly lowercase.