Sequential Programming

Writeup Submissions (/student/submissions/14/85)

Scoreboard (/student/scoreboard/14/85) Costs (/student/costs/14/85)

Show Submission Password

Module	Open	Deadline
Sequential Programming	01/18/2016 00:01 -0500	01/24/2016 23:59 -0500

- ✓ Course Rules and Dataset Details
- ✓ Data Filtering
- ✓ Data Analysis
- ✓ Success

Introduction

Learning Objectives

1. Process part of a large text dataset using sequential programs on a cloud
2. Understand the limitations of sequential methods when analyzing large volumes of data

Warnings

Working on the cloud may be a new experience for you. Please go through the necessary primers and set up your workstation to support easy and secure access to your cloud resources. Make sure the AMI we provide can run your code. You cannot `sudo apt-get` `install` anything on the instance.

General Details

The following table contains some general information about this project module (P1.1):

Applicable Languages	
• TAs can help you if you use Java/Python/Bash • Apart from those, you may use any language that your AWS instance supports. However, the TAs may not be able to help you resolve problems in other languages.	
Tasks	**Total Budget**
1	$5

Grading Penalties

Please keep in mind that this course has a preset amount of cloud resources that is being shared by all enrolled students. To make sure that we do not suffer from excess use, we have to employ budgets and penalties for each project module. For this project module, the budget is outlined below. Exceeding the budget for this project or not tagging your resources correctly will result in grade penalties. The following table outlines the violations of the project rules and their corresponding grade penalties for the first week of the course.

These rules apply for one week while you are working on P1.1.

Violation	Penalty of the project grade
Using more than $5 of AWS resources	10%
Using more than $10 of AWS resources	100%
Not tagging any of your resources	10%
Using any "Project" tags apart from "Project":"1.1"	10%

In this first project, we will get a feel for big data by diving head first into analyzing a large dataset. We will perform sequential analysis (no parallelism) on a single file, specifically one that contains hourly page view statistics from Wikipedia (http://dumps.wikimedia.org/other/pagecounts-raw/).

About the Dataset

Wikimedia maintains hourly page view statistics for all objects stored in Wikimedia servers as publicly accessible datasets. We will use these statistics to analyse page-view trends and derive the trending topics on wikipedia for a particular time range.

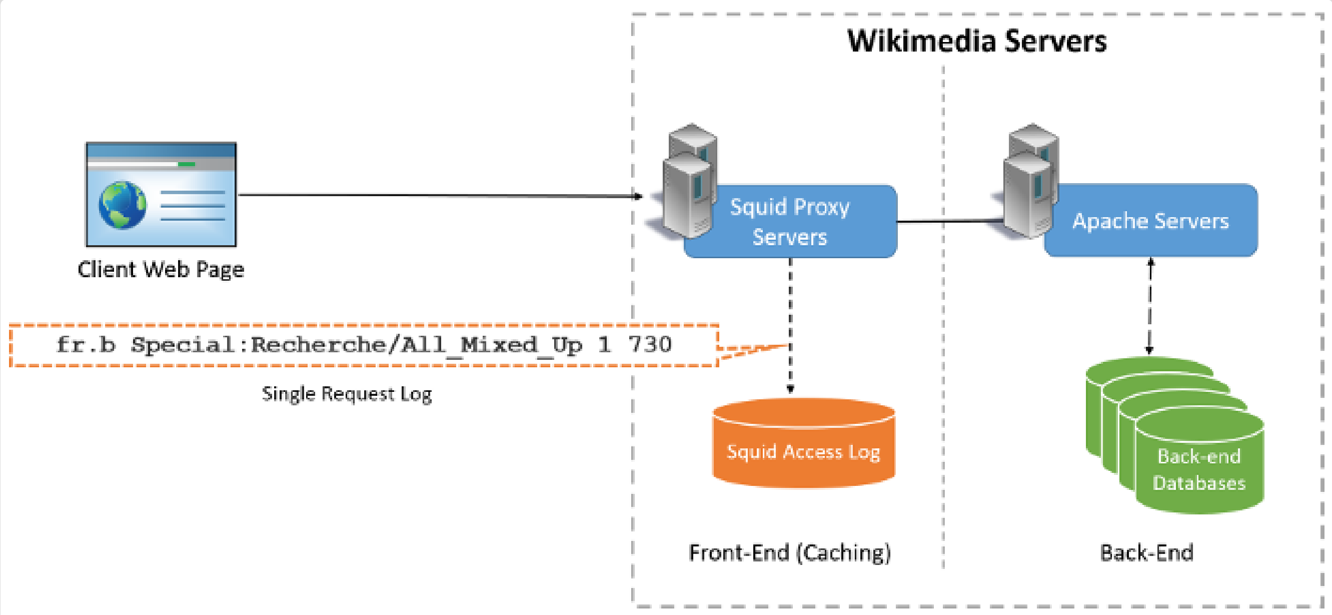

Figure 1: A simplified diagram of a page access from Wikimedia. More Information. (http://en.wikipedia.org/wiki/Wikimedia_Foundation#Hardware)

Every request made to Wikipedia's servers is serviced by a squid cache proxy (http://en.wikipedia.org/wiki/Squid_%28software%29), which also logs the request. These logs are shared publically every hour in flat text files. Each line of this file corresponds to a single access from the Wikimedia servers in the following format:

```
[project name]  [page title]    [number of accesses]    [total data returned ir
```

`[project name]` has two parts, a language identifier and a subproject suffix. The following abbreviations are used in the subproject suffix:

- (no suffix) : wikipedia
- .b : wikibooks
- .d : wiktionary
- .m : wikimedia
- .mw : wikipedia mobile
- .n : wikinews
- .q : wikiquote
- .s : wikisource
- .v : wikiversity
- .w : mediawiki

For Example, the following line:

```
fr.b Special:Recherche/All_Mixed_Up 1 730
```

denotes that from the French Wikibooks page, the page `Special:Recherche/All_Mixed_Up` was accessed once and 730 bytes were transferred in total. The article's title in this line is `Special:Recherche/All_Mixed_Up`.

In this project, we will focus on analyzing the page view logs from December 2015. For this week, we will focus on the first hour of Dec. 1st 2015. The data has been uploaded to the s3 location: `s3://cmucc-datasets/wikipediatraf/201512/pagecounts-20151201-000000.gz`. You can use `aws-cli`, `s3cmd` or S3 Browser to explore the location. Alternatively, you can `wget https://cmucc-datasets.s3.amazonaws.com/wikipediatraf/201512/pagecounts-20151201-000000.gz`

Refer to the page Amazon S3 (https://theproject.zone/student/writeup/14/78) in the Project Primer for details on accessing S3.

Accessing 15-319/619 Resources

In this module of Project 1, we require you to SSH into ubuntu VMs. Your login will be blocked if any of the following cases occur:

1. You did not update your AWS email address in the profile (https://theproject.zone/website/profile/).
2. You did not update your 12-digit AWS account ID in the profile (https://theproject.zone/website/profile/).
3. You did not click on "Accept" in the consolidated billing request from AWS sent to the email ID in Step 1.
4. You modified your existing linked account number or AWS email ID in the profile.
5. Your linked AWS account got unlinked from us for any reason (check here (https://console.aws.amazon.com/billing/home?region=us-east-1#/consolidatedbilling) and confirm that Carnegie Mellon University appears under "Account Name".
6. This week, you can only launch a t1.micro instance.

The course account is a limited and precious resource shared by **ALL** students. The budget and tagging requirements are strictly enforced and are meant to simulate a real-world environment. You cannot use personal AWS resources for the course. When detected, severe penalties will ensue.

Data Filtering

Wikipedia is often a great reflector of current events. Pages that are being accessed with increasing frequency often indicate an event related to the topic of the page. Births, deaths, Google Doodles, wardrobe malfunctions, or even something mundane as a technical conference can often trigger a spurt of interest in a topic. Wikipedia serves as a fairly unbiased source of reporting of the news, and sometimes even reveals hidden patterns. Over the next two weeks, we will see if we can find what the data tells us.

This week we will start to analyze the data with a single instance running small scripts on an hour's worth of Wikipedia traffic log. We will focus exclusively on English Wikipedia and ignore the rest of the Wikimedia project, to cater to the interest of students. To be able to do this, we will write a basic filter this week, test it on the first hour of the logs and then use it on the remaining 743 hours of the month next week. This form of sanity testing your code on a subset of the data will be useful in all your future data wrangling tasks.

Task to Complete:

Provision a `t1.micro` instance using the AMI ID: **`ami-95e9cdff`** , which can be found in the AWS community AMIs. If you have trouble finding the AMI, ensure that you have configured EC2 to work in the US-East-1 (N. Virginia) region (https://console.aws.amazon.com/ec2/v2/home?region=which-region#LaunchInstanceWizard:). Allow incoming SSH (TCP port 22) and HTTP connections (TCP port 80) through your security group. The username to SSH in is `ubuntu` .

You must use the tag {"Key":"Project","Value":"1.1"} for your instance. Not doing so will lead to a 10% penalty.

To be able to solve Project 1.1, you should download (and unzip) a single hour of Wikimedia traffic logs into the following folder.

```
/home/ubuntu/Project1_1/
```

The file contains all of the pages from all Wikimedia projects. Our aim is to identify trending topics from the English Wikipedia articles. In order to do this, develop a script or write a program in any language to:

0. Some lines have only three (or fewer) elements. You should filter out these lines, for example:

   ```
   en  1282 10636194
   ```

1. Filter out all pages that are not English Wikipedia. (This means that the log lines should start with `en` (case sensitive), without any suffix attached).
2. There are many special pages in Wikipedia that do not need to be considered when trying to find trending topics. Exclude any pages whose title starts with the following strings (case sensitive):

```
Media:
Special:
Talk:
User:
User_talk:
Project:
Project_talk:
File:
File_talk:
MediaWiki:
MediaWiki_talk:
Template:
Template_talk:
Help:
Help_talk:
Category:
Category_talk:
Portal:
Wikipedia:
Wikipedia_talk:
```

3. Wikipedia policy states that all English articles must start with an uppercase character. Filter out all page titles that start with lowercase English characters. You may notice that some page titles start with non-English characters, you should choose to retain them in the analysis.
4. You may also get results which refer to image files, exclude any page title that ends with the following extensions (Keep all other extensions intact). (.jpg, .gif, .png, .JPG, .GIF, .PNG, .txt, .ico). Do not use case-insensitive matching, remove exactly those file extensions.
5. Finally, there are some boilerplate page titles, which should be excluded as well. Page titles that are exactly (case sensitive) any of the following strings should be excluded:

   ```
   404_error/
   Main_Page
   Hypertext_Transfer_Protocol
   Search
   ```

6. Once the filtering is done, output the remaining articles in the following format:

   ```
   [page title]\t[number of accesses]
   ```

7. At this stage, you may wish to test your code. To do so, please add the commands to compile and run your filtering script within the `Q0 section` of the `/home/ubuntu/Project1_1/runner.sh` file.

   You may want to run this file using the command:

   ```
   ./runner.sh
   ```

You should be able to see your answers printed out.

8. If you are satisfied with your answers, you may choose to send your code and the file to the grading server. To do this, please run:

```
./submitter -a andrewID -l language <python/java/bash>
```

Example:

```
./submitter -a msakr -l python
```

Notes:

- You may notice that the data set consists of files beginning with languages `en` , `En` and `EN` . Apply the filter to only `en` (case sensitive).
- `\t` stands for the tab character.
- **The output should be sorted in the numeric descending order of the number of accesses.**

Data Analysis

After filtering the data, you are expected to analyze your results and answer nine increasingly challenging questions, which are present in the file `/home/ubuntu/Project1_1/runner.sh` on your instance. To complete this module, do the following:

1. Go to the auto-grader folder located at `/home/ubuntu/Project1_1`
2. The auto-grader consists of three files, `runner.sh` , `submitter` and `references` . You have permissions to edit `runner.sh` and `references` files.
3. Edit the script `runner.sh` to include the commands/code used to answer the checkpoint questions. Using bash scripting is recommended, but not required. Do not move any of the provided files. If you are using any external scripts, ensure that you are calling the correct scripts from `runner.sh` . Please ensure that you are placing all your code in the same folder and also assume that the dataset is present in the current folder. When you need to access the dataset in your code assume that it is present in the working directory (i.e do not use any absolute or relative paths for accessing the dataset -- we will autograde it later in a single jailed environment).
4. Edit the text file `references` to include all the links that you referred to for completing this project. Also include the Andrew IDs of all the other students who you might have discussed general ideas with when working on this project in the same file. This is extremely important. We analyze many aspects of your AWS and TPZ usage, as well as two different automated

code similarity detection tools. Being honest about your collaboration does not excuse cheating. **You are not allowed to look at, or discuss code with another person. Similarly, you are not allowed to use any code snippet from anyone or anywhere on the Internet.** When in doubt, please post privately on Piazza.

5. You can run the following to check your answers by typing `./runner.sh` from the Project1_1 folder. Running this script should print out the answers to all the questions. Please ensure that the answers are printing correctly before you submit your answers. Do not print hardcoded answers, we will run your code on other log files for verification and grading.

6. Once you have completed all the questions, you can submit the answers to the evaluation system using the `submitter` executable. Run the executable using the command `./submitter -a andrewId -l language` from the autograder folder. Remember that your submission password can be found by clicking on the button on the top of this page. **Make sure you open port 80 for incoming HTTP traffic before you proceed.**

7. After running the `./submitter` command, a website will be created with answers and values which are specific to your submission. Once your details are validated, these pages will be read by our load generators within a few minutes, at which point, you will see the feedback and scores in your submission table. The load generators also read your code and save them to our repository, where they will be strictly analyzed for similarities with submissions from other students in the past and present, as well as with code from the internet. There is no limit on the number of submissions allowed before the project deadline. However, submissions must be separated by at least 60 seconds.

Notes:

When submitting, please make sure the following source code is in your Project1_1 folder:

- Code to convert `pagecounts-20151201-000000 to output`.
- Code to use the `output` file to generate the answers for Q1 to Q9.
- Make sure that source code is present (.py, .Java) and not just executables (.jar). If your code is complicated, leave a README.
- **Remember, output should be sorted in the numeric descending order of the number of accesses.** (you may want to write a local verifier for this).

If you have completed all your submissions, congratulations and welcome to cloud computing.