Assignment 4

Map-Reduce

Design Lab

1st February 2023

This assignment is on map-reduce, which is a distributed and scalable way of extracting/mining required information from multiple datasets stored on multiple servers. Follow the tutorial to understand how you can design mapper and reducer for specific queries/operations.

Tutorial on map-reduce

You can start with a simple word count problem. Say, we have a text file and we want to count the frequency of occurrence of each word. The tutorial below explains how to solve this problem using a map-reduce algorithm.

Tutorial References :

http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/

Next you can also look into the following tutorial for slightly harder query (tf-idf scores)
https://www.tutorialspoint.com/map_reduce/map_reduce_tutorial.pdf

Tasks

1. Study HDFS and MapReduce basics.
2. Write the necessary mapper and reducer routines in python to implement the queries mentioned in later sections.
3. Ensure that the reducer routine prints results of queries to a text file (as asked in the later sections).

Dataset

We will be using a dataset of an anonymized Web server log file from a public relations company. Please download the data from https://drive.google.com/file/d/1OKoKpWbyGxzil5uvFCkWQcdT4ogxQ02L/view?usp=sharing .

In the file "**access_log.txt**", each line represents a hit to the Web server. It includes the IP address which accessed the site, the date and time of the access, and the name of the page which was visited.

The logfile is in the Common Log Format:

10.223.157.186 - - [15/Jul/2009:15:50:35 -0700] "GET /assets/js/lowpro.js HTTP/1.1" 200 10469

%h %l %u %t \"%r\" %>s %b

where,

1. %h is the IP address of the client
2. %l is identity of the client, or "-" if it's unavailable
3. %u is username of the client, or "-" if it's unavailable
4. %t is the time that the server finished processing the request. The format is [day/month/year:hour:minute:second zone]
5. %r is the request line from the client is given (in double quotes). It contains the method, path, query-string, and protocol or the request.
6. %>s is the status code that the server sends back to the client. You will see see mostly status codes 200 (OK - The request has succeeded), 304 (Not Modified) and 404 (Not Found). For more info on status code - https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html
7. %b is the size of the object returned to the client, in bytes. It will be "-" in case of status code 304.

== Sample data for the file **access_log.txt** ==

```
10.223.157.186 - - [15/Jul/2009:14:58:59 -0700] "GET / HTTP/1.1" 403 202
10.223.157.186 - - [15/Jul/2009:15:50:35 -0700] "GET / HTTP/1.1" 200 9157
10.223.157.186 - - [15/Jul/2009:15:50:51 -0700] "GET / HTTP/1.1" 200 9157
10.223.157.186 - - [15/Jul/2009:20:50:29 -0700] "GET / HTTP/1.1" 200 9157
10.223.157.186 - - [15/Jul/2009:20:50:38 -0700] "GET / HTTP/1.1" 200 9157
10.223.157.186 - - [15/Jul/2009:21:04:42 -0700] "GET / HTTP/1.1" 200 524
10.223.157.186 - - [15/Jul/2009:21:24:10 -0700] "GET / HTTP/1.1" 200 7616
10.82.30.199 - - [29/Jul/2009:08:46:38 -0700] "GET / HTTP/1.1" 200 7616
```

==========

Queries

The queries are to be implemented in the mapper and reducer phases. Some of them may give empty results. You need to implement the following queries in this assignment. The queries have an imaginary backstory to help you find a real world perspective in this assignment.

Assume you are an engineer of the company. You want to conduct a quantitative analysis of the web server log of the company using the access log file available to you. Please use MapReduce to answer the following queries.

1. For every unique ip address, find the number of requests made to the server. [10]

   Note: The result will have lines such as: 10.223.157.186 10 which means that ip address 10.223.157.186 has made 10 requests to the server. The output file should contain one such count in each line. The ordering of the ip addresses does not matter.

2. Find the top-10 requested image files (png, jpg, gif, ico) by the clients. [20]

   Note: The result will have lines such as: /assets/img/loading.gif 12 which means that the resource file /assets/img/loading.gif was requested 12 times by the

clients. The ordering withing these top-10 resource files does not matter. The output file should contain 10 such lines.

3. For every 15 minute window, find the window with the maximum server load with respect to the size of the objects returned to the clients. You need to print the total size (bytes) of the returned objects by the server in the window with the maximum load. [30]

   Note: The output file should contain the size of the returned objects in bytes printed on a single line. In case of "-" in %b (size) consider size to be 0 bytes.

4. We define a bot as a client which have made atleast 10 requests to the server and all of the requested objects are of size less than 1000 bytes. Find the ip addresses of all the bots. [30]

   Note: The result will have lines such as: 10.223.157.186 which means that ip address 10.223.157.186 is a suspected bot. The output file should contain one such ip address in each line. The ordering of the ip addresses does not matter. In case of "-" in %b (size) consider size to be 0 bytes.

How to run and test your code:

Since we do not have access to a Hadoop cluster, we will be testing our codes on a Linux system as follows:

cat access_log.txt | python mapper.py | sort | python reducer.py > result.txt

Or just

python mapper.py | sort | python reducer.py > result.txt

Explanation on the above commands:

1. "cat" is a linux command to print the contents of a file on the console.
2. The pipe operator (|) directs the previous command's output to the next command.
3. "sort" is a linux command to sort the input lexicographically.
4. ">" can be used to save the standard output in a file.

Deliverables: 4 sets of python codes (mapper and reducer), 4 Makefiles, 4 result.txt, 4 readme files explaining your approach.

Evaluation Scheme: Results: 90 marks, Coding Style: 10 marks

Important Instructions

1. **The mapper routine can pass over the access_log.txt file only once. You can not use any intermediate file structure or data structure to save data. Python dictionaries should not be used in the mapper routine. However, you can use one dimensional arrays in the mapper or reducer, and dictionaries in the reducer only. Note that these must be limited to one dimension only. The reducer can't access the data file. So, the mapper should just go through the data and generate a series of key-value pairs which will then be passed to the suitable sort command. Finally the reducer should just go through the stream of tokens once to give the required output. You cannot perform any kind of sorting either in the mapper or the reducer. You can only sort the output of the mapper using the linux sort command before sending it to the reducer. Any violation will attract a penalty of upto 100% of the marks assigned.**

2. **Submission Rule:** All deliverables for the above functionalities must be compressed as **.zip** and named **"<RollNo>.zip"**. For each query, make a directory named **"Query<no.>"**. Your files must be inside the respective directories. Strictly adhere to this naming convention. Submissions not following the above guidelines will attract penalties.

3. **Makefile:** Each query folder **must** have it's own Makefile to execute the routines. You can find a relevant tutorial here (https://opensource.com/article/18/8/what-how-makefile).

   Note: "sort" behaves differently in different environments. Ensure that you use sort in a way which works properly on a linux machine.

4. **Code error:** If your code doesn't run or gives error while running, you will be awarded with zero mark. Your code must run correctly on **a linux machine**.