CS5630 Programming Project-2

Due: Nov 18, 2022 midnight
Upload to ICON a single zip file containing all materials

Part-A. Understanding MapReduce and Hadoop **60 points**

The goal of this part is to get familiar with the fundamentals of MapReduce programming using the Hadoop framework. You will learn to work with large datasets, write data analytics tasks, and execute them on Hadoop ecosystem. This part is to be completed on a single server (e.g., your laptop or AWS VM).

Dataset: You are given a dataset of 3 million tweets that contain mentions of publicly traded companies. Please download the dataset here: https://www.kaggle.com/omermetinn/tweets-about-the-top-companies-from-2015-to-2020. We focus on the file ***Tweet.csv***, whose content is in the following format: *<tweet_id, writer, post_date, body, num_comments, num_retweets, num_likes>*. You are required to write two separate MapReduce programs to analyze this data.

MapReduce Program-1: The first program should compute the total number of times every company ticker has been mentioned. You compute this by analyzing the body of each tweet in the dataset, and extracting the company tickers mentioned there. Remember that a company ticker starts with a "$" sign, and is comprised of 1-5 alphabets (for example, $C and $GOOG are valid tickers whereas $12 is not). If a tweet mentions a given ticker multiple times, you should only count it once. Also, remember that a given tweet may mention multiple tickers (for instance, if a tweet says buy $GM over $TSLA, then you count it against both tickers). Your program's output should be a file that contains key-value pairs, one per line, such that each key is a ticker symbol, and the corresponding value is the total count of its mentions.

MapReduce Program-2: The second program measures the engagement metric of twitter users. We define the engagement metric of a tweet as its number of retweets, number of likes, and number of comments, all added together. Your program should iterate over all the tweets, computing the engagement metric of each tweet, and then aggregating such engagement metrics per user. Your program should ignore those tweets that have no engagement metrics. Your program's output should be a file that contains key-value pairs, one per line, such that each key is a twitter user id, and the corresponding value is their aggregate engagement metric.

Writing your programs: You will write the mapper and reducer functions in Python. Then, you will run them using the Hadoop Streaming utility (here are two sample tutorials to learn about it: https://hadoop.apache.org/docs/r1.2.1/streaming.html and https://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/). Amongst other things, Hadoop Streaming allows your mapper and reducer to simply read their inputs from STDIN and write their outputs to STDOUT.

Setting up Hadoop: Finally, install Hadoop to operate on a single-node by following instructions here: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html. *Please remember to use the Pseudo-distributed mode and not the Local (standalone) mode*. Download the input data into your machine and transfer it to HDFS. Then, execute your MapReduce programs (you could use the command below as a reference for that). Upon completion of the job, Hadoop will print a detailed report on its resource usage, runtime parameters as well as other useful information.

It is imperative that you write your own code (and not be tempted to copy-paste someone else's code). We will inspect your code both visually and run it through a plagiarism detector. Here is the command that we will use to check the correctness of your submitted programs:

```
hadoop jar /usr/lib/hadoop/hadoop-streaming.jar —files mapper.py,reducer.py —mapper
mapper.py —reducer reducer.py —input Tweet.csv —output results
```

Submission: You should submit the following items:

1. Mapper and reducer functions for both of the MapReduce programs. Submit these as four separate files. (40 points)
2. Output summary from your Hadoop runs. We don't want to see the entire log but only the final part that describes the summary of the run. (20 points)
3. Any references you used in performing these tasks (just a list of URLs would do). If you do not submit any references, we will assume that you did not use any resources beyond what is mentioned by us.

Part-B. Running Data Analytics on AWS EMR — 40 points

The goal of the second part is to familiarize yourself with running data analytics on PaaS cloud. You will use Amazon's Elastic MapReduce (EMR) to run the same two MapReduce programs that you developed earlier. However, in contrast to part-A, you will use a multi-node cluster here.

Setting up EMR: The benefit of using PaaS is that you do not have to install, maintain, administer, or scale the Hadoop infrastructure in order to run your MapReduce programs. Explore Amazon's offering and knowledge base here: https://aws.amazon.com/emr/. There are many guides that provide step-by-step instructions on how to set up an EMR cluster on top of EC2 instances and EBS storage (an example, https://levelup.gitconnected.com/map-reduce-with-python-hadoop-on-aws-emr-341bdd07b804). Please feel free to use a guide that suits you.

EMR Configuration: You will use the same AWS Academy credentials to create an EMR cluster. In the EMR configuration wizard, make sure you enable logging, choose a valid key pair, and select core-hadoop as the application. In the beginning, you will select an instance of type large (say m4.large) and add three nodes to the cluster (i.e., 1 master and 2 workers). Make sure to disable cluster-scaling and auto-termination. EMR takes several minutes to set up the cluster, so please be patient. Once it is ready, you can ssh into the master node to run your programs.

Benchmarking: First, you will establish a performance baseline for the initial cluster configuration (the two most important metrics are completion time and cost). To determine the program's runtime, use the Hadoop output report, and to determine the cost, refer to the pricing examples here (https://aws.amazon.com/emr/pricing/?nc=sn&loc=4). Next, you will benchmark your program's performance in two additional configurations. You could get to additional configurations either by scaling the cluster size up and down (2, 3, 4 nodes), or by changing the instance type (large, xlarge, 2xlarge), or by reconfiguring the number of reducers in Hadoop (1, 3, 6 reducers). You DO NOT need to do all three modifications, just pick one approach and observe how that impacts the program's baseline performance.

Submission: You should submit a report that contains the following three items:

1. ***Baseline comparison between single-node and cluster***: Compare the performance of your two programs from the single-node setup of part-A to the three-node cluster of part-B. Please include the VM specs such as CPU, RAM, and disks for both set up. Was it cheaper to run your programs on single-node or cluster? (20 points)
2. ***Benchmarking different EMR configurations***: Compare the performance of your two programs across the three configurations described in benchmarking. Make a table, where columns represent the three configs and rows represent the two programs. Be sure to compare both completion time and cost. Briefly explain your findings. (20 points)
3. List any references you used in performing these tasks (just a list of URLs would do). Pease remember that using any resources without citation constitutes plagiarism.

Note: We will not accept late submissions. However, you are allowed to submit partial solutions and will receive a grade proportional to your work.