Lab 9: Hadoop Programs on Non-Synthetic Datasets

Due date: June 5, (Friday), 11:59pm.

Lab Assignment

Assignment Preparation

This is a team lab. Partner assignment is done in class and is, rightly or wrongly, mostly random.

Overview

This is a research assignment. You are asked to implement three algorithms in Hadoop. As part of your reporting for this lab you will have to submit evidence that your programs correctly solve the problems.

For each problem, you are provided with one file representing a sample dataset. Your goal is to implement a MapReduce solution to each problem and compare its efficiency against a sequential solution (that you also need to implement). To make the comparison more robust, you will conduct a study of the run-time for each of your MapReduce programs vs. their sequential counterparts, with input size being the independent variable. For input sizes smaller than the size of the file provided to you, you can take the appropriate subset of records. For input sizes larger than the size of the file provided to you, you can duplicate rows, or synthesize additional data.

Problem 1

The first problem is a graph traversal problem. The input data provided to you contains a list of edges for a large directed graph.

Slashdot dataset is one of a series of social network datasets available on SNAP, the Stanford Network Analysis Package dataset collection of large

graphs/networks. The dataset we are using is the Slashdot social network, November 2008, available from this URL:

`http://snap.stanford.edu/data/soc-Slashdot0811.html`

The dataset consists of a single file, available on HDSF as:

```
-rw-r--r--   3 dekhtyar hdfs   10747200 2020-05-20 09:23 /data/soc-Slashdot0811.txt
```

Here are the first few lines of the data file:

```
# Directed graph (each unordered pair of nodes is saved once): Slashdot0811.txt
# Slashdot Zoo social network from Noveber 6 2008
# Nodes: 77360 Edges: 905468
# FromNodeId    ToNodeId
0       0
0       1
0       2
0       3
```

Your code shall ignore the lines that start with the `"#"` character. The remaining lines contain edges of the directed graph for the Slashdot Zoo social network. A directed edge $(first, second)$ represents an directed interaction from user $first$ to user $second$.

Note: For this dataset, you may want to use `NLineInputFormat` to process the data using multiple mapper instances (the size of the dataset is just under the 128Meg limit on a single split).

The Task

We are interested in the reachability relationship between the nodes in the Slashdot Zoo graph. Specifically, if we look at interactions between the users, we want to find out if the "Kevin Bacon" rule holds.

Write a Hadoop Java program which, for each node in the graph, finds all other nodes in the graph reachable from it in **six or fewer steps**. Your program then shall aggregate this information and report the following (as a single line of output):

- total number of nodes in the graph
- average number of nodes reachable from a node in the graph in six steps or fewer.

You will write a Hadoop implementation and a sequential implementation that reads the file in from a local directory path, and produces the same output. Name your Hadoop program `SlashdotReach.java`.

Problem 2

For this problem you will be working with the full Iowa Liquor Sales dataset, as it existed on May 1, 2020. The dataset contains over 18 million individual sales records and has exactly the same structure as the smaller JSON dataset you have used in the past. That is - the CSV file contains the same attributes and in the same order as the JSON data file with 10,000 records provided to you.

The full structure of the dataset is described on the Iowa data web page:

https://data.iowa.gov/Sales-Distribution/Iowa-Liquor-Sales/m3tr-qhgy

The first line of the data file contains the list of attributes, all other lines contain individual sales data.

The file is available in the `/data` directory on the hdfs:

```
-rw-r--r--   3 dekhtyar hdfs 4510906811 2020-05-20 09:52 /data/Iowa_Liquor_Sales.csv
```

The Task

Due to the large size of the dataset, we restrict ourselves to a fairly simple problem: for each year for which the sales data is available, report

- the total number of sales (invoices)
- the total volume of all purchases alcohol measured in liters (please make sure that you take into account the number of bottles in each invoice)
- the total amount of money that was paid for these purchases (this is the aggregation over the sales in dollars column).

As with other tasks, you will create a MapReduce (Hadoop) implementation and a sequential implementation. Name your Hadoop program `IowaLiquor.java`.

Problem 3

Your third program will work with the US county-by-county COVID-19 statistics data from the NY Times github dataset that you used in prior labs. As a reminder, each line of the county-by-county data file that you will be using for this program looks as follows:

```
date,county,state,fips,cases,deaths
```

The May 19, 2020 version of this dataset has been uploaded to hdfs as a file named `us-covid-counties.csv`:

```
-rw-r--r--   3 dekhtyar hdfs    6247037 2020-05-20 10:13 /data/us-covid-counties.csv
```

The Task

For this dataset, we are setting up a problem that uses some of the advanced features of Hadoop, such as the use of distributed cache and combiners.

Let us compare the spread of COVID in three locations in the state of California: Central Coast the Bay Area, and Los Angeles area. For this exercise, we assume the following:

- Central Coast is San Luis Obispo, Santa Barbara, and Monterey counties.
- The Bay Area is the city/county of San Francisco, and San Mateo, Santa Clara, Alameda, and Contra Costa counties.
- The LA area is Los Angeles county and Orange county (we could do more with the greater LA going into the Inland Empire, but LA and Orange are enough).

We want to know how similar the infection trends are. We use Pearson correlation (see Lab 6) as our similarity measure for a pair of vectors. Our vectors will be the vectors of daily new cases for each of the three geographic areas in California. Note that daily new case information is NOT available in the input file, you need to compute daily new case numbers from consecutive total cases numbers that are available to you.

So, your full task is as follows: your code shall filter out all data except for the counties belonging to one of the geographic areas in California described above. You then shall create, for each geographic area a vector of daily new cases. Finally, you will output Pearson correlations for LA Area - Bay Area, LA Area - Central Coast, and Central Coast - Bay Area comparisions of these vectors.

In working on this problem you should be able to repurpose some of the Lab 6 Pearson correlation coefficient computation code.

As before, you will compare a Hadoop implementation, which you will call `CovidCompare.java` with a sequential implementation.

Experimental Design

For each program, your goal is to answer the same question - at what point is it more advantegeous to solve the underlying problem using a Hadoop program rather than the sequential program. To properly answer this question, your programs have to run on input files of different sizes. Creating input files of different sizes for this program is left up to you. We have enough space both on the cluster and on hdfs for your experiments.

Essentially, you want to find an input size at which the overhead of running Hadoop makes it less effective than the sequential program. Then you want to keep on increasing the size of the input until you read the inflection point,

at which Hadoop solutions overtake sequential solutions. Finally, you want to demonstrate how bad it can get for the sequential solution by comparing the run time on the full dataset - or, on input data of even larger size if necessary.

The bulk of organizing this work is left to you. We will keep other assignments in the upcoming weeks sufficiently light to enable you to devote necessary time to this problem.

All your programs need to be instrumented with the code collecting timing information. When running your experiments, you need to make sure that the timing you get is not a fluke. You can do it, for example, by running the same program on the same input multiple times and averaging the runtime.

Note: To make your life slightly easier, copies of all data files for this assignment are also stored on local file system in the `/usr/data` directory:

```
dekhtyar@ambari-head:~$ cd /usr/data
dekhtyar@ambari-head:/usr/data$ ls -al
total 4421796
drwxr-xr-x  2 dekhtyar dekhtyar       4096 May 20 12:12 .
drwxr-xr-x 14 root     root           4096 May 20 12:11 ..
-rw-r--r--  1 dekhtyar dekhtyar 4510906811 May 20 12:12 Iowa_Liquor_Sales.csv
-rw-r--r--  1 dekhtyar dekhtyar   10747200 May 20 12:12 soc-Slashdot0811.txt
-rw-r--r--  1 dekhtyar dekhtyar    6247037 May 20 12:12 us-covid-counties.csv
```

Deliverables

The main thing you need to submit for this assignment is a written report documenting your implementation of the Hadoop and sequential programs, and the experiments you ran with them. Your report, for each problem shall include a graph showing the change in runtime as the size of the data increases. You can measure the size of the data as number of records, or as overall file size, although number of records is probably better.

The report shall be word-processed, and submitted in PDF format. It shall contain a title, list of authors with their email addresses, a brief introduction, and a section detailing each of the three problems. Additionally, the report shall contain your thoughts on the observed behavior of Hadoop vs. the sequential implementation. Is Hadoop worth using?

In addition, you shall submit all code necessary to run each of your three Hadoop implementations. You DO NOT need to submit any utlity code to build timing graphs, and you do not need to submit your sequential programs.

Submission

Submit your Java programs and all other files necessary to run your code. Submit README file describing how to compile and run all your programs. Submit Makefiles or compile-and-run bash scripts to compile and run your programs is necessary. Submit your report.

All submitted files must contain your names on them.

Submit all your code in a single archive (zip or tar.gz).

Use `handin` on the CSL unix servers to submit as follows:

```
$ handin dekhtyar lab07 <FILES>
```

Good Luck!