The Course Project

The course project includes 3 parts. The first part is to develop a Mapper and Reducer application to retrieve Year and Temperature from original NCDC records (i.e., the dataset we are using for this class) and then write the Year and Temperature data into a text file. The second part is to load the text file into Pig and get the highest and lowest temperatures for each year. The third part is to load the text file into Hive and get the average temperature for each year.

You need to turn in:

**1)** a. *if you are using JAVA to develop the Mapper and Reducer applications:* the three java files (mapper, reducer and main);

b. *if you are using Hadoop streaming jar and developing two python programs (mapper python file and reducer python file)*: the two python files (mapper and reducer);

c. *if you are using mrjob library and developing one python program with two functions:* the python file (with the mapper and reducer functions);

**2)** the commands from converting java files into a Jar file to running the Jar file in Hadoop, or the commands to execute the python files in Hadoop;

**3)** the text file including Year and Temperature data created by you;

**4)** the screenshot of the text file being created;

**5)** the Pig commands and the screenshot of the final Pig output showing the year and the highest and lowest temperatures;

**6)** the Hive commands and the screenshot of the final Hive output showing the year and average temperature.

The original dataset for this project is available on Blackboard.

**Submitted By**

Prathamesh Bhople (yv5392)

**PART I**

Develop Mapper and Reducer application to retrieve Year and Temperature from original NCDC records and writing data into text file.

A) Here, we will be using the **Hadoop streaming jar** and develop two python programs (retrieve\_temperature\_map.py & retrieve\_temperature\_reduce.py) for retrieval of Year and Temperature.

B) First, gunzipping and taking all the required 50 NCDC records from the local side to server side as shown below,

Graphical user interface, application

Description automatically generated

C) Below are the two developed Mapper and Reducer python files to retrieve the Year and Temperature, also these files are loaded onto the server side,

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

D) Now that the Mapper and Reducer are developed, connecting to **University Hadoop System,** and changing the permissions of two python files,

Graphical user interface, text

Description automatically generated

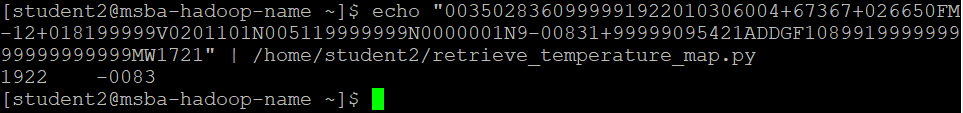
Also, gunzipping all the 50 files using below command,

A screenshot of a computer

Description automatically generated with medium confidence

E) Testing the two python files locally using below commands before running then using Hadoop, here testing files on any random record and file from 50 NCDC record files,

**Command 1:** From file *028360-99999-1922.gz*, testing mapper python file, as shown below, Year and Temperature is getting fetched for this record using the mapper file.



**Command 2:** From file *029350-99999-1925.gz*, testing both mapper and reducer files, as shown below, Year and Temperature is fetched. Reducer file is correctly fetching the integer values of the temperature.

Text

Description automatically generated

**Command 3:** Now, testing *014030-99999-1930* file using below command. Below screenshot shows sample output.

Text

Description automatically generated

**Command 4:** Using sort -k1,1 we can sort the output of mapper file based on temperature, below is the sample screenshot of sorted output.

Text

Description automatically generated

**Command 5:** Using the below command, passing the sorted output to the Reducer file to fetch the sorted Year and Temperature, below is he sample screenshot.

Text

Description automatically generated

F) Based on the above testing, it can be concluded that the Mapper and Reducer python files seems to be working correctly to fetch Year and Temperature. Therefore, Copying all the records to HDFS for further processing.

hdfs dfs -copyFromLocal /home/student2/\*-99999-\* /home/2student2/inputforpython

A picture containing text

Description automatically generated

G) Using hadoop streaming jar file (hadoop-streaming-2.7.3.jar) and two python files (Mapper and Reducer) to retrieve the Year and Temperature, and loading the output into /project folder,

hadoop jar hadoop-streaming-2.7.3.jar -file /home/student2/retrieve\_temperature\_map.py -mapper /home/student2/retrieve\_temperature\_map.py -file /home/student2/retrieve\_temperature\_reduce.py -reducer /home/student2/retrieve\_temperature\_reduce.py -input /home/2student2/inputforpython/\*-99999-\* -output /home/2student2/project

**Note:** *hadoop-streaming-2.7.3.jar* is present on the server side.

A picture containing graphical user interface

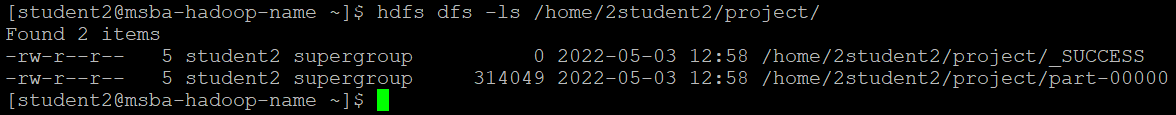
Description automatically generated

Graphical user interface, text

Description automatically generated

As shown above, the MapReduce job is executed successfully.

H) As shown below, the output file is generated in the /project folder at HDFS,



I) Taking this output file back from HDFS to server side using -copyToLocal command, renaming the file to *YearTemp.txt*



J) It can be observed that, all the records from 50 NCDC records are retrieved in the output file *YearTemp.txt* file, it contains 36262 records in total

Graphical user interface, application, table

Description automatically generated

**PART II**

Load the *YearTemp.txt* text file into Pig and get the highest and lowest temperatures for each year.

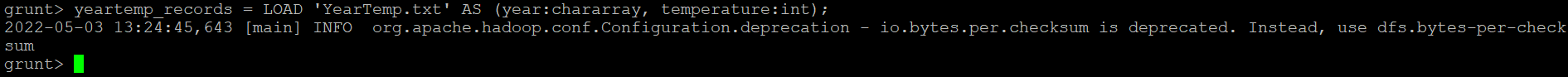
A) Connecting to local pig session,

Text

Description automatically generated with low confidence

B) Using LOAD command to load the contents of *YearTemp.txt* file into Pig. Year is defined as **chararray** and temperature as **int**.

yeartemp\_records = LOAD 'YearTemp.txt' AS (year:chararray, temperature:int);



C) Using DUMP command to load the records.

DUMP yeartemp\_records;

Shape

Description automatically generated with medium confidence

D) Using DESCRIBE command to show the column datatypes. Chararray and Int used to define the Year and Temperature respectively.

DESCRIBE yeartemp\_records;

Text

Description automatically generated

E) Filtering records that are not equal to 9999, for cleaning the data.

filtered\_yeartemp\_records = FILTER yeartemp\_records BY temperature != 9999;

DUMP filtered\_yeartemp\_records;

E) Now to find out highest and lowest temperatures by each year, we will group the records by **Year**, using GROUP command.

grouped\_yeartemp\_records = GROUP filtered\_yeartemp\_records BY year;

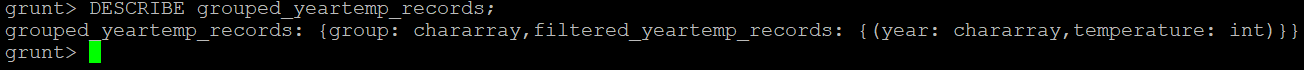
DUMP grouped\_yeartemp\_records;

A picture containing night sky

Description automatically generated

Above screenshot is the sample output of grouped\_yeartemp\_records for the year 1930.

DESCRIBE grouped\_yeartemp\_records;



This is how the records are group based on the year.

F) Below is the screenshot of the output with **highest and lowest temperature for each year**.

yeartemp = FOREACH grouped\_yeartemp\_records GENERATE group, MAX(filtered\_yeartemp\_records.temperature), MIN(filtered\_yeartemp\_records.temperature);

DUMP yeartemp;

A screenshot of a computer

Description automatically generated

**PART III**

Load the *YearTemp.txt* file into Hive and get the **average temperature** for each year.

A) After connecting to the Hive, using below command to check if the table already exists in the hive or not. If present, then dropping the table,

Text

Description automatically generated

B) Creating the table using CREATE TABLE command,

CREATE TABLE YearTemperature\_std2 (year STRING, temperature INT)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t';

Graphical user interface, text

Description automatically generated

C) Now loading the data from ‘*YearTemp.txt*’ file into the above created table in hive using below command,

Text

Description automatically generated

D) Now that the data is in hive, we run the below query to calculate the **average** temperature for each year,

Text

Description automatically generated

E) Below screenshot is the output of the above query with average temperature for each year,

Text

Description automatically generated