**CSE 5331-Fall 2016**

**DBMS Models and Implementation**

**Project 3: Map/Reduce Implementation**

Submitted by: Parth Shah (1001246682)

Shivesh Ranjan Singh (1001231391)

Department of Computer Science,

University of Texas at Arlington

**Overall Status**

We have completed both parts of the project. There are two folders in the zipped file. One contains the Custom partitioner class and the other contains the Temperature sort using sampling of data.

The custom partitioner uses the temperature ranges 0-200, 200-400,400-600,600-800,800-999.99

First we use a mapper function to output key as temperature and the whole tuple as the values. Then a partitioner class is used which decides which reducer the data goes to. Its simply using comparison statements to decide this.

The Temperature sort using sampling of data uses a built in sorting partitioner known as the TotalOrderPartitioner. This total order partioner uses RandomSampling() method samples our input into equal partition. So here we have created two jobs, one which uses our mapper class and passes the data to the TotalOrderPartitioner. The output of this then is passed on to another mapper and reducer which finally prints out output.

**Performance Measure**

When we used our own custom partitioner we had files which were divided into

Part-0 1.1 Gb Temp -> 0-200

Part-1 425.6Kb Temp -> 200-400

Part-2 404.8Kb Temp -> 400-600

Part-3 424.1kb Temp -> 600-800

Part-4 703.7 Kb Temp -> 800-999.9

Because we divided temperature range into 5 equal partitions we used 5 reducers to complete this job

Here we can see that most of the work done is done by one reducer and the other reducers are not doing equal amount of work.

The run time of a Total Order Partitioner

When we used our own random partitioner we had files which were divided into the following parts and temperature ranges.

Part-0 218.8Mb Temp -> -4.27 – 54.12

Part-1 211.1Mb Temp -> 54.13 – 66.57

Part-2 211.9Mb Temp -> 66.58 – 75.19

Part-3 191.3Mb Temp - > 75.20- 81.9

Part-4 230.7Mb Temp -> 81.9-999.9

Because the random partitioner is used the data is partitioned into equal splits and then passed out to reducers therefore the work done is almost equally split into partition files.

We have set the number of reducers to 5.

The run time of a Total order partitioner actually took about 3 minutes and 35 seconds.

**Where I encountered difficulty**

This project has been very challenging. Kindly find few of the challenges as below:-

Understanding the Hadoop/Map reduce framework was quite difficult as it was quite cumbersome. Also setting it up was a challenge as we were new to it and we faced numerous challenge in setting up Hadoop on our computer.

We got exposure to Mapper, reducer and partitioner class in our classes which was quite helpful, but actual implementation of it proved to be quite difficult.

**File Descriptions**

We have two folders inside the the zip file. One implements the CustomPartitioner and the other uses sampling and implements TotalOrderPartitioner.

Both of them contain the file Temperature.java This is our main file which contains all other functions such as the mapper and the reducer.

**CustomPartitioner**

The mapper splits the input to give us the temperature and we pass this as the key. The rest of the tuple is then used as the value which is passed to the partitioner class. The partitioner class then decides based on the temperature ranges which reducer to pass the data to. This reducer then outputs the final result in the output file.

The input to be given to run this is – Hadoop jar –jarname input output

**TotalOrderPartitioner**

In this we have a main java file called Temperature.java containing all the class files.

We are configuring 2 Jobs to do achieve this.

The first Job uses our mapper class. The mapper splits the input to give us the temperature and we pass this as the key to our Partioner class. The partitioner class then creates random partitions and sends data to another mapper.

This is the second Job. This job uses a built in mapper class and then sends it to our reducder class which writes text to the output file.

**Division of Labor**

Me and Parth worked together as a team to complete this project. The steps for installation and other instructions are very well explained in the document provided and it was very helpful in setting up the environment and developing the logic of the code. The project was given to us in first week of Nov and we had 4 weeks to complete the project. We started with getting an understanding of how to setup the cloud in our system. Understanding it was not a cakewalk and we had to devote a week time in getting complete understanding of it. Next challenge was to help understand the framework of Hadoop .We devoted 2-3hours religiously in 2nd and 3rd week for it. Once we got the understanding of the framework and different classes used its implementation was not that time consuming. All in all we have spent roughly 25-30 hrs in completing the project.

**Logical Errors**

Hadoop Installation- Initially we faced issues in setting up environment variable for hduser. Due to user permission associated with Ubuntu we were unable locate and edit the file. We went through the installation guide again and followed the instructions religiously to solve this problem’

Input Datatype mismatch – There were different data types of each mapper and reducer. For example data type was Text in first mapper and when it was invoked by second mapper then it was of different type which resulted in datatype mismatch. To resolve it we used same type data set for both the mappersuser in Ubuntu. After carefully reading the Hadoop installation guide, we got the proper commands to resolve this issue.

Compile Time Error-

We used Eclipse Ide to compile our code, therefore setting up maven and then setting the pom.xml file was a tedious task and required quite a bit of research. The dependencies took time to match.

Run Time Error-

As we were compiling on Eclipse many times there would be no errors at compile time. But when we would actually run it on Hadoop it would give us weird errors. The importing of files was a major issue as we were mixing the old Hadoop version which required “mapred” instead of “mapreduce” while importing libraries.