Project Report

On

Big Data Analytics: Average weight of people

According to Year, Gender and Location

*By: Supervisor:*

Kumar Jayram Gayatri Dr. Upendranath Chakravarthy

Ankit Anil Shingavi

*Project submitted in fulfilment of the requirements for the course*

*DBMS Models and Implementation (CSE- 5331)*

Department of Computer Science & Engineering

University Of Texas Arlington

May 5th, 2016

### Table of Contents

1. Introduction
2. Function Implementation
   1. Mapper
   2. Reducer
   3. Combiner
3. Overall Status
4. Division of Labour
5. Report
6. Performance Analysis
7. Logical Errors
8. References

**Introduction**

RDBMS is not solution for all the application of data, so invention of NoSQLs such Hadoop, CouchBase, Cassandra has taken place to address the horizontal scalability, high performance, reducing join costs and cloud computing framework compatible framework. One of the advantages of cloud computing framework and the use of Map/Reduce paradigm is its ability to deal with very large data sets and one-time computations with a reasonable response time. This is done by using as many processors as needed. Typically, the map/reduce paradigm is used for these types of problems in contrast to the RDBMS approach which is better suited for storing, managing, and manipulating this data. Hadoop is a widely used open source map/reduce platform. Hadoop Map/Reduce is a software framework for writing applications which process vast amounts of data in parallel on large clusters. In this project, you will use the US census dataset and develop a program to compute a number of demographic statistics using Hadoop map/reduce paradigm. Please use the following links for a better understanding of Hadoop and Map/Reduce.

We have implemented Map/Reduce paradigm to find average age of the people in a year among males and females over different location of the United States. The averages are found over the years 2009-2013, in seven states of U.S. California/CA, Colorado/CO, Florida/FL, Pennsylvania/PA, Texas/TX, Vermont/VT, and Washington/WA.

**Problem Statement:**

We need to compute the weight trend over a few years for different states along genders using the given data. We need to develop a map/reduce solution. For each of the following states, calculate average weight of males and females for each year and plot on a graph with year as the X-axis and avg. weight as the Y-axis. Note that this can be done for all states as well as in other ways.

For this problem, map/reduce can be beneficially used. It will be easy to adjust the number of mapper and reducer nodes based on the data size and the Hadoop framework will do most of the work in partitioning (sharding) the data and passing intermediate output from mappers to reducers. It will also take care of failures of nodes etc. We need to design and develop a map program (including a combiner if needed) and a reduce program to solve the above problem. The most important aspects of this design will be to identify the key value pairs to be output by the mapper and worked on by the reducer. If we want to do ii) above to understand and appreciate the power of this paradigm and the ease of scaling using this paradigm, we will run the same data set:

i. On multiple mappers and to see how parallelism works

ii. Measure response time for different configurations and see how scalability is accomplished.

The input data is typically partitioned (sharded) into 64MB splits as default. The number of splits can also be configured by the user.

**Function Implementation**

Following are the major components of the buffer manager system:

## Mapper

Mapper takes the sharded input from inpuTextFormat and maps each value with the count or some intended value to the key. In our problem the key is the combination of Year, Gender, State of the individual separated by a ‘|’, and the value is the Weight of the individual. In a map number of occurrences of a key can be more than once and the value also can be same. It does not aggregate any values, even if the keys are same. For example,

Input:

….,2011,Female,Washington/WA,70,…

…..,2011,Female,Washington/WA,60,…

….,2012,Female,Washington/WA, 70,…

…..,2012,Female,Washington/WA, 60,…

Key:

2011|Female|Washington/WA, 70

2011|Female|Washington/WA, 60

2012|Female|Washington/WA, 70

2012|Female|Washington/WA, 60

## Reducer method:

Reducer takes the input from the mapper and combines the result for of each key as the user has written the code. Input for the reducer is a key and a collection of values. In our example reducer gets the input from the mapper/combiner and input is the key which is combination of year, gender and state separated by a ‘|’. Value is the collection of the weights of the individual who are from same year, gender and the state. It gets the sum of the weights and count of the people. Calculates the average and writes it to the output.

Input:

2011|Female|Washington/WA, <70, 60>

2012|Female|Washington/WA, <70, 60>

Output:

2011|Female|Washington/WA, 65

2012|Female|Washington/WA, 65

## Combiner method:

Combiner is a kind of mini reducer which is run on a mapper to decrease the number items in a collection of the reducer, so that it increases the performance of the map/reduce framework. Combiner takes the intermediate result of a mapper and combines it like a reducer. However, sometimes combiner may not as similar to the reducer, like in our example finding an average at the combiner is not suitable, so we find the count and sum in the combiner and send this result to the Reducer.

**Overall Status**

We have implemented the map reduce for the intended problem and we have succeeded in getting the result on Single node cluster. However, we tried setting up of the AWS EC2 cluster for the map reduce paradigm to check the different results. Even though, we have successfully done the implementation we could not run map reduce for the data set given for the problem due to the RAM size problem.

To improve the performance, we have implemented the combiner for the map/reduce. This improvement really reduced the time of execution almost 40%.

**Division of Labour**

This project is implemented by a team of two members. The both team members have equally contributed for the successful completion of the project. The distribution of project work is as following:

1. Kumar Jayram Gayatri:

Kumar has tried the implementation of Hadoop in AWS EC2 cluster and written the combiner for the mapper.

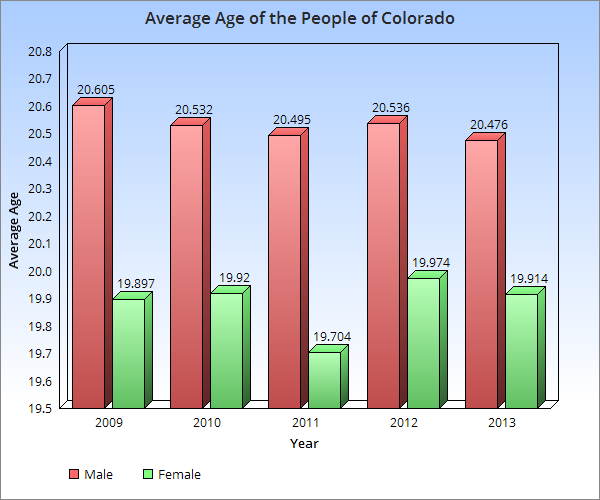
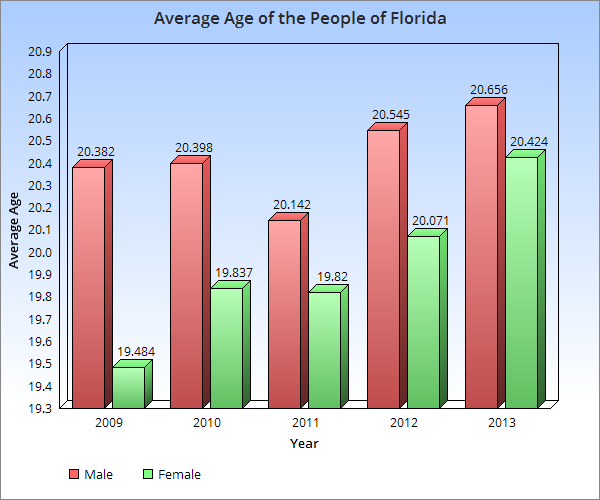
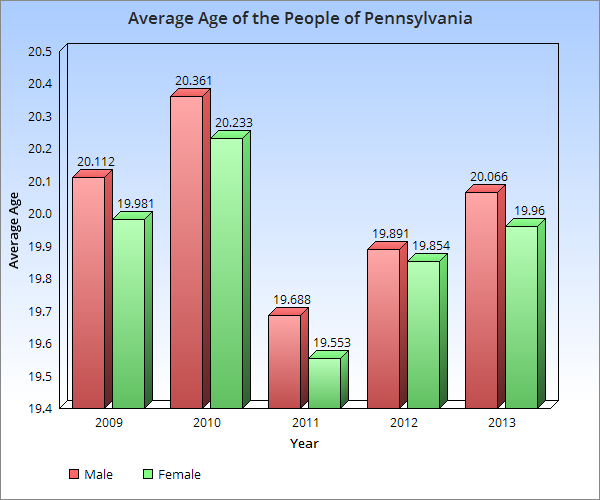
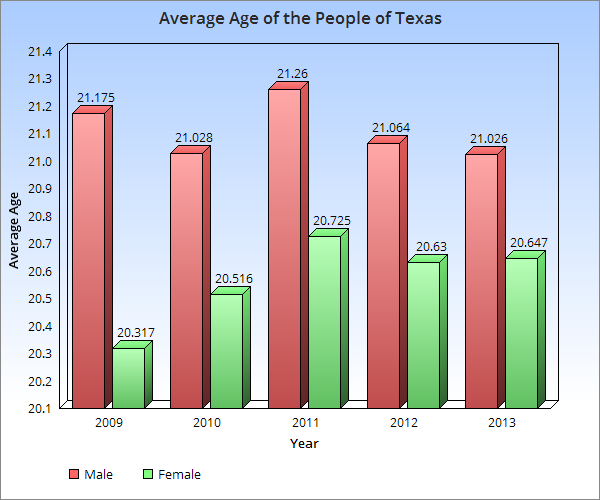
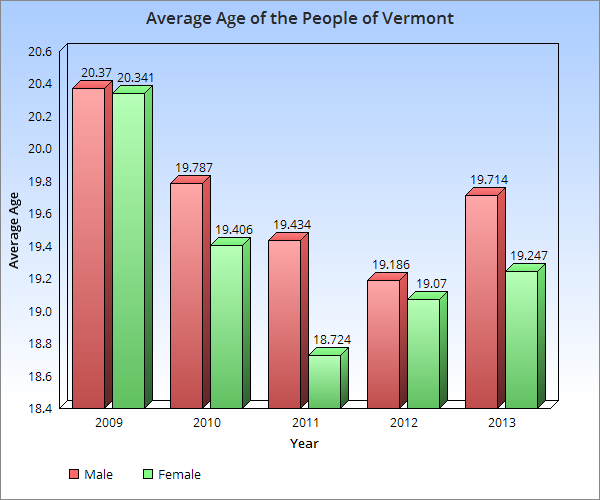
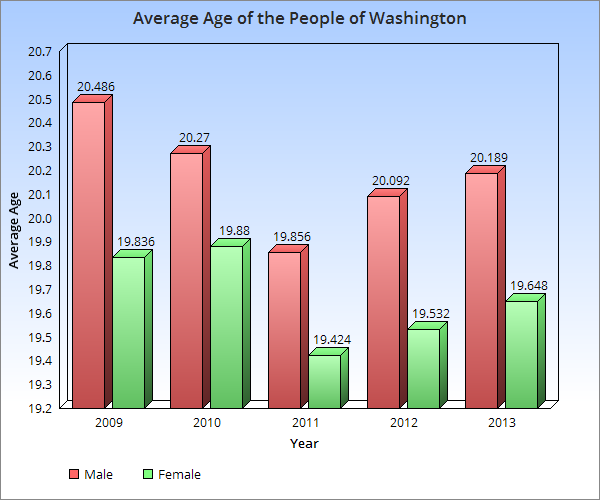
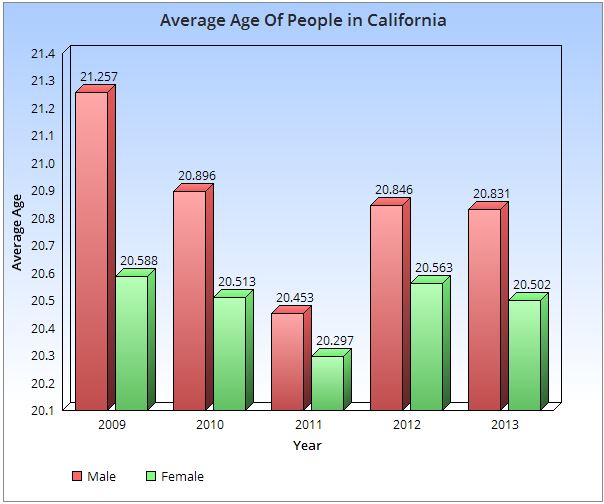
2. Ankit Anil Shingavi:

Ankit has implemented the Map/Reduce for the given data set, and he has also installed the single node cluster setup of Hadoop in his centOS system.

We did a great team work by reviewing both of our work each other, and we had great brainstorming sessions to understand the requirements of Map/reduce and increasing the performance of the map/reduce.

**Report**

We have considered the 7 states, and the result is plotted on graphs for each state, gender and year versus the average age.



**Performance Analysis**

Initially, with single node ubuntu cluster (1 mapper, 1 reducer), we have run the map reduce without any combiners which resulted in response time of 438s. After the first attempt, we tried to run map reduce with combiner which sis not show much improvements in the performance as we had only one mapper and reducer.

Further, we could improve the performance with multiple mappers and reducers with cluster node. We tried installing cluster in AWS node with (5 nodes, 1GB RAM, 8GB memory each EC2 cluster). However, the set up did not work due to I/O and RAM memory shortage. This would have increased the performance by response time < 406/4 as there will 4 mapper and reducers.

**Logical Errors**

There were some logical errors that we have faced during installation of the Hadoop. Nonetheless, we had minimal error in implementing the map/reduce such as restricting results only for 7 states and exception handling.

**REFERENCES**

1. Database Management System 3rd Edition book by Raghu Ramakrishnan and Johannes Gehrke.

2. https://wweb.uta.edu/faculty/sharmac/javadocs/index.html

3. <https://elearn.uta.edu/bbcswebdav/pid-4624294-dt-content-rid-39372361_2/courses/2162-DBMS-MODELS-AND-IMPLEMENTATION-33973-001/TxMgr-proj2.pdf>

Thanks to Jay Bodra.