Map/Reduce Implementation

Submitted by: Team 20 – Nhat Dao (1001030170)

**Overview**

The project involves implementation of a simple map/reduce paradigm using which students have to analyze the weather datasets of the cities given and generate <Key, Value> pairs indication the Station Ids having the similar weather. The project scope is to implement 2 Map/Reduce jobs each of which is used to arrive at the desired result.

MapReduce has 2 main parts:

• Map: processes a key value pair to generate a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key and passes them to the Reduce function. • Reduce: accepts an intermediate key and a set of values for that key. It merges together these values to form a possibly smaller set of values.

1. Mapper 1 – Key should be <Year + State + Gender>

Value should be <Weight>

1. Reducer 1 – Generates averages of Weight

**Overall Status**

The Project has been **successfully completed**. Two mappers and two reducers are being implemented.

* Successful installation and setup of Hadoop 2.7.2.
* Successful implementation of Mappers and Reducers and job chaining
* Successful deployment and testing.

**Configuration**:

1. Hadoop 2.7.2
2. **OS** : MacOs El Capitan
3. **Java** : Java 1.8.0\_66

**Implementation Details:**

**Setup:**

* After going through the project material, I got a fair idea about the setup so I started looking for other documents or steps for installation. The used the below link for the following installation
* <https://getblueshift.com/setting-up-hadoop-2-4-and-pig-0-12-on-osx-locally/>
* I installed Hadoop with Homebrew, copy all required HADOOP\_PATHS to shell path
* **Start the Nodes:**

*start-dfs*

*start-yarn*

**HDFS file system:**

* HDFS file system is what Hadoop uses for the mechanism it provides for map/reduce.
* The input file has to be loaded into the HDFS file system so that Hadoop is able to understand the input and carry on with its process.
* The input gets uploaded as block of 64MB and Hadoop uses it for input splitting and assigning to mappers.
* The input can be loaded to the HDFS filesystem using :

*hadoop dfs -copyFromLocal input inputhd*

**Compile and generate jar:**

I put in a sh file run.sh

#!/bin/sh

#Compile java files

javac -classpath ${HADOOP\_CLASSPATH} -d WeightTrend/ WeightTrend.java –Xlint

jar -cvf WeightTrend.jar -C WeightTrend/ .

**Run Map/Reduce:**

hadoop jar WeightTrend.jar WeightTrend wtInput/data.csv wtOutput/

**Division of Labor**

The project was completed single handedly as is a one student team.

I was able to get Hadoop running on a VM but due to my limited experience and my Macbook’s configuration, I cant find a good way to share files between host and guest. Thus, I couldn’t test how my MapReduce code run until recently when I installed Hadoop on my local machine (Mac El calpitan)

Total took me 45 hours and 40 hours were just to fixing Hadoop’s version, deprecations, configuration on local and virtual machine.

**Logical Errors**

* **Input Datatype mismatch** – I encountered this problem where I had designed with different data types of inputs for each mapper and reducer. I was using **Text** for the first mapper and when the second Mapper was getting invoked, there was a datatype mismatch. Hence I went ahead with Text as the datatype for both map/reduce jobs.
* **Hadoop setup**: I could set up on a VM but still couldn’t get the dataset in there so I had to set it up locally on a Mac which was not very nice.
* **Data column**: I mistook a column for State and Sex, so the results didn’t show as expected but then I fixed them.

**Conclusion**

The project gave the insight of the Map/Reduce paradigm. A good knowledge to have for the current trend of problems in the software world.