

# Group B

## Group B

### Assignment No. 1

**AIM:** Hadoop Programming: Word Count MapReduce Program Using Eclipse

**TITLE:** Write a code in JAVA for a simple Word Count application that counts the number of occurrences of each word in a given input set using the Hadoop Map-Reduce framework on local-standalone set-up.

**COURSE OBJECTIVES:** To gain practical, hands-on experience with statistics programming languages and Big Data tools

**COURSE OUTCOMES:** Students will be able to use cutting edge tools and technologies to analyze Bid Data.

**REQUIREMENTS:**  
Hadoop and MapReduce

#### **THEORY:**

In Hadoop, MapReduce is a computation that decomposes large manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers. The results of tasks can be joined together to compute final results.

MapReduce consists of 2 steps:

- **Map Function** – It takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (Key-Value pair).

**Example** – (Map function in Word Count)

<b>Input</b>	Set of data	Bus, Car, bus, car, train, car, bus, car, train, bus, TRAIN,BUS, buS, caR, CAR, car, BUS, TRAIN
<b>Output</b>	Convert into another set of data (Key,Value)	(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1), (TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1), (car,1), (BUS,1), (TRAIN,1)

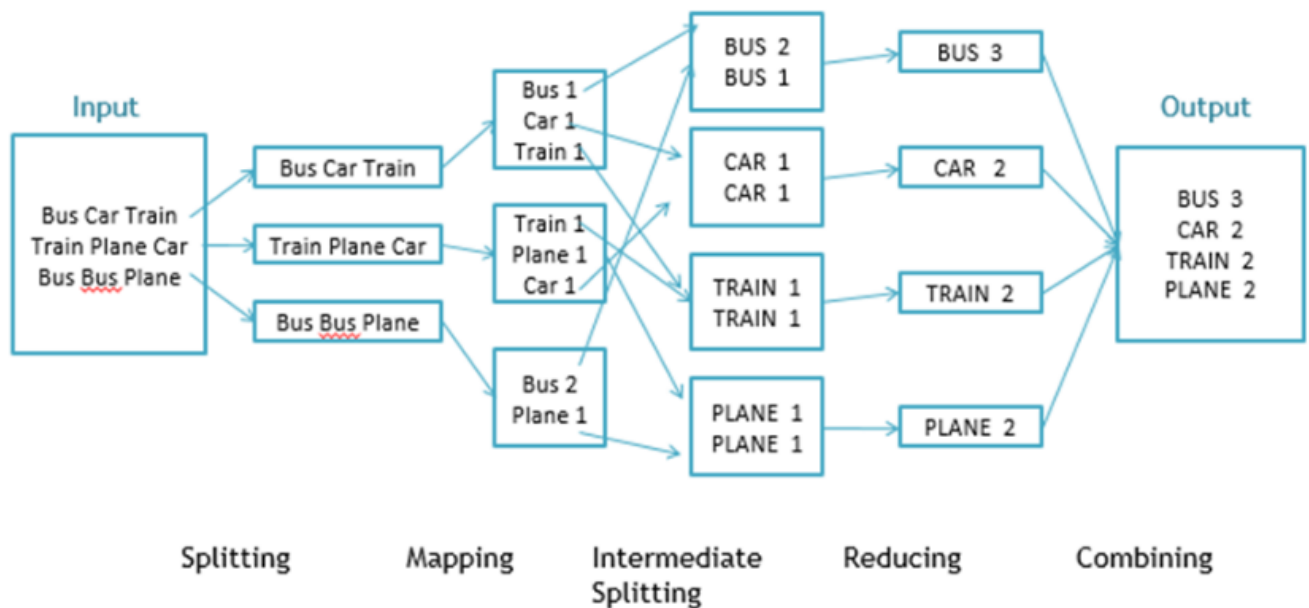
- **Reduce Function** – Takes the output from Map as an input and combines those data

tuples into a smaller set of tuples.

**Example** – (Reduce function in Word Count)

<b>Input</b> (output of Map function)	Set of Tuples	(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1), (TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1), (car,1), (BUS,1), (TRAIN,1)
<b>Output</b>	Converts into smaller set of tuples	(BUS,7), (CAR,7), (TRAIN,4)

### Work Flow of the Program



Workflow of MapReduce consists of 5 steps:

1. **Splitting** – The splitting parameter can be anything, e.g. splitting by space, comma, semicolon, or even by a new line ('\n').
2. **Mapping** – as explained above.
3. **Intermediate splitting** – the entire process in parallel on different clusters. In order to group them in “Reduce Phase” the similar KEY data should be on the same cluster.
4. **Reduce** – it is nothing but mostly group by phase.
5. **Combining** – The last phase where all the data (individual result set from each cluster) is combined together to form a result.

Counting the number of words in any language is a piece of cake like in C, C++, Python, Java, etc.

MapReduce also uses Java but it is very easy if you know the syntax on how to write it. It is the basic

of MapReduce. You will first learn how to execute this code similar to “Hello World” program in

other languages. So here are the steps which show how to write a MapReduce code for Word Count.

Steps:

1. Open Eclipse> File > New > Java Project >( Name it – MRProgramsDemo)  
> Finish.

Create Three Java Classes into the project. Name them WCDriver(having the main function), WCMapper, WCDReducer.

You have to include two Reference Libraries for that:

Right Click on Project -&gt; then select Build Path-&gt; Click on Configure Build Path

In the above figure, you can see the Add External JARs option on the Right Hand Side. Click on it

and add the below mention files. You can find these files in /usr/lib/

1. /usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.13.0.jar
2. /usr/lib/hadoop/hadoop-common-2.6.0-cdh5.13.0.jar

## Copy mapper code

Mapper code:

```
// Importing libraries
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WCMapper extends MapReduceBase implements Mapper<LongWritable,
Text, Text, IntWritable> {
// Map function
public void map(LongWritable key, Text value, OutputCollector<Text,
IntWritable> output, Reporter rep) throws IOException
{
String line = value.toString();
// Splitting the line on spaces
for (String word : line.split(" "))
{
if (word.length() > 0)
{
output.collect(new Text(word), new IntWritable(1));
}
}
}
}
```

## Copy reducer code

Reducer code:

```
// Importing libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
public class WCReducer extends MapReduceBase implements Reducer<Text,
IntWritable, Text, IntWritable>
```

```

{

// Reduce function
public void reduce(Text key, Iterator<IntWritable> value,
OutputCollector<Text, IntWritable> output,
Reporter rep) throws IOException
{
int count = 0;
// Counting the frequency of each words
while (value.hasNext())
{
IntWritable i = value.next();
count += i.get();
}
output.collect(key, new IntWritable(count));
}
}

```

Copy driver code

```

// Importing libraries
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;

```

Conclusion – Thus we created word count program using Hadoop map-reduce.

## FAQ

1. What is Hadoop? What are different components of Hadoop Ecosystem?
2. List different advantages and disadvantages of Hadoop.
3. What is HDFS? What is purpose of it?
4. What are the different Hadoop configuration files?
5. What are the differences between regular FileSystem and HDFS?

## **Group- B**

### **Assignment 2**

**TITLE: Design of Distributed Application -Processing Log file using Map-Reduce Framework**

**PROBLEM STATEMENT:**

Design a distributed application using Map-Reduce which processes a log file of a system.

**COURSE OBJECTIVES:** To gain practical, hands-on experience with Big Data tools

**COURSE OUTCOMES:** Students will be able to use cutting edge tools and technologies to analyze Bid Data.

**THEORY:**

What is MapReduce?

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

#### The Algorithm

MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

- Map stage – The map or mapper’s job is to process the input data. Generally, the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
- Reduce stage – This stage is the combination of the Shuffle stage and the Reduce stage. The Reducer’s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.

## Inputs and Outputs (Java Perspective)

The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework. Input and Output types of a MapReduce job – (Input) <k1, v1> → map → <k2, v2> → reduce → <k3, v3>(Output).

The input for our program is txt\_file .

### 1.Steps for Compilation & Execution of Program:

```
#sudo mkdir analyzelogs
ls
#sudo chmod -R 777 analyzelogs/
cd
ls
cd ..
pwd ls cd
pwd
#sudo chown -R hduser analyzelogs/
cd ls
#cd analyzelogs/
ls cd ..
```

### 2. Copy the Files (Mapper.java, Reduce.java, and Driver.java to Analyzelogs Folder)

```
#sudo cp /home/isha/Desktop/count_logged_users/* -/analyzelogs/
Start HADOOP
#start-dfs.sh
#start-yarn.sh
#jps
```



### 3. Compile Java Files

```
# javac -d .  
  DsbdaMapper.java DsbdaReducer.java DsbdaDriver.java ls  
#cd Dsbda/  
ls cd .. #sudo gedit Manifest.txt  
#jar -cfm analyzelogs.jar Manifest.txt Dsbda/*.class  
ls cd jps  
#cd analyzelogs/
```

#### 1. Create Directory on Hadoop #sudo mkdir ~/input2000

```
ls pwd #sudo cp access_log_short.csv ~/input2000/  
# $HADOOP_HOME/bin/hdfs dfs -put ~/input2000 /  
# $HADOOP_HOME/bin/hadoop jar analyzelogs.jar /input2000 /output2000  
# $HADOOP_HOME/bin/hdfs dfs -cat /output2000/part-00000  
# stop-all.sh  
# jps
```

**Conclusion:** Thus, we have learnt how to design a distributed application using MapReduce and process a log file of a system

## **Group B**

### **Assignment No. 3**

**TITLE: Weather Data Forecasting – Map Reduce**

**PROBLEM STATEMENT:** Locate dataset (e.g., sample\_weather.txt) for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed.

**COURSE OBJECTIVES:**

To gain practical, hands-on experience with statistics programming languages and Big Data tools.

**COURSE OUTCOMES:** Students will be able to use cutting edge tools and technologies to analyze Bid Data.

**THEORY:**

**Introduction to Map- Reduce:**

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner. MapReduce is a processing technique and a program model for distributed computing based on java.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes

Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model

**The Algorithm**

MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

**Mapstage:** The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

**Reduce stage:** This stage is the combination of the Shuffle stage and the Reduce stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

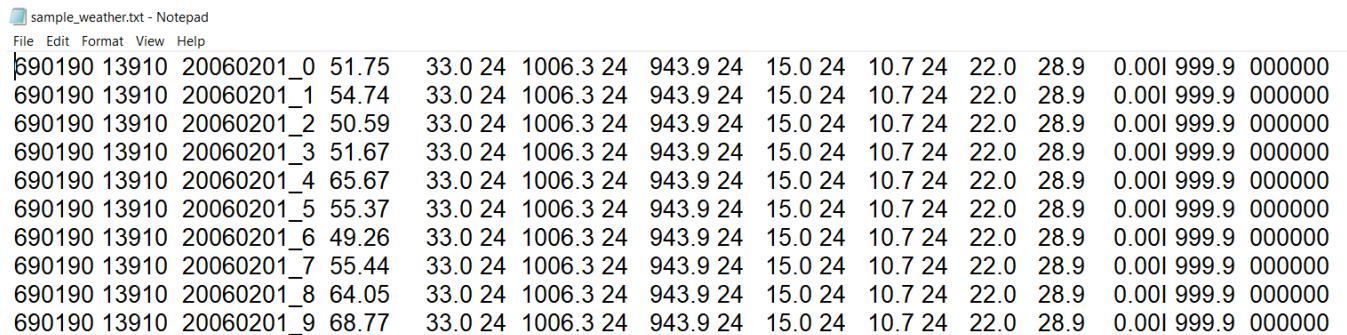
## Inserting Data into HDFS:

- The MapReduce framework operates on pairs, that is, the framework views the input to the job as a set of pairs and produces a set of pairs as the output of the job, conceivably of different types.

- The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable -Comparable interface to facilitate sorting by the framework.

- Input and Output types of a MapReduce job: (Input) <k1,v1>-> map -><k1,v2> -> reduce -><k1,v3> (Output).

1. To start with Map reduce Project, please install Apache Hadoop , Refer the below site for more details <https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>  
sample\_weather.txt file will look like this



```
sample_weather.txt - Notepad
File Edit Format View Help
690190 13910 20060201_0 51.75 33.0 24 1006.3 24 943.9 24 15.0 24 10.7 24 22.0 28.9 0.001 999.9 000000
690190 13910 20060201_1 54.74 33.0 24 1006.3 24 943.9 24 15.0 24 10.7 24 22.0 28.9 0.001 999.9 000000
690190 13910 20060201_2 50.59 33.0 24 1006.3 24 943.9 24 15.0 24 10.7 24 22.0 28.9 0.001 999.9 000000
690190 13910 20060201_3 51.67 33.0 24 1006.3 24 943.9 24 15.0 24 10.7 24 22.0 28.9 0.001 999.9 000000
690190 13910 20060201_4 65.67 33.0 24 1006.3 24 943.9 24 15.0 24 10.7 24 22.0 28.9 0.001 999.9 000000
690190 13910 20060201_5 55.37 33.0 24 1006.3 24 943.9 24 15.0 24 10.7 24 22.0 28.9 0.001 999.9 000000
690190 13910 20060201_6 49.26 33.0 24 1006.3 24 943.9 24 15.0 24 10.7 24 22.0 28.9 0.001 999.9 000000
690190 13910 20060201_7 55.44 33.0 24 1006.3 24 943.9 24 15.0 24 10.7 24 22.0 28.9 0.001 999.9 000000
690190 13910 20060201_8 64.05 33.0 24 1006.3 24 943.9 24 15.0 24 10.7 24 22.0 28.9 0.001 999.9 000000
690190 13910 20060201_9 68.77 33.0 24 1006.3 24 943.9 24 15.0 24 10.7 24 22.0 28.9 0.001 999.9 000000
```

2. Install maven dependencies/plugins required for Hadoop

```
$ mvn install
```

3. Add the sample input file given in the project to the hdfs

```
hdfs dfs -put sample_weather.txt
```

4. Then, Start the Map reduce application to submit the job to Hadoop

```
To run: bin/hadoop jar target/weather-1.0.jar [-m <i>maps</i>] [-r<i>reduces</i>] <i>in-  
dir for job 1</i> <i>out-dir for job 1</i> <i>out-dir for job 2</i>
```

## OUTPUT:

**CONCLUSION:** Thus, we have learnt how to process input data using map reduce framework.

1. What is map-reduce?
2. Explain in how many stages the map-reduce program executes with example?
3. How to insert data into HDFS?
4. How to Interact with MapReduce Jobs?