**Session 3: EXPLORING MAPREDUCE**

Assignment 3.3

Student Name: Abarajithan SA

Course: Big Data Hadoop & Spark Training

Start Date:  2017-09-09

End Date:  2017-11-26

**Assignment 3.3** – Write a Map Reduce java program to find out the total units sold for each company and for each state for Onida Company.

Contents

[Introduction: 2](#_Toc497238241)

[Associated Data Files 2](#_Toc497238242)

[Problem Statement 2](#_Toc497238243)

[Actual Data Set: 3](#_Toc497238244)

[1. Map Reduce Java Program: Write a Map Reduce program to calculate the total units sold for each Company 4](#_Toc497238245)

[Driver Code 4](#_Toc497238246)

[Mapper Code 5](#_Toc497238247)

[Reducer Code 6](#_Toc497238248)

[Command to run the jar 7](#_Toc497238250)

[Expected Output 7](#_Toc497238251)

[2. Write a Map Reduce program to calculate the total units sold in each state for Onida Company. 8](#_Toc497238252)

[Driver Code 8](#_Toc497238253)

[Mapper Code 9](#_Toc497238254)

[Reducer Code 10](#_Toc497238255)

[Command to run the jar 10](#_Toc497238256)

[Expected Output 11](#_Toc497238257)

# Introduction:

In this assignment, I’m going to write a **Map Reduce** java program to calculate the total units sold for each company and the total units sold in each state for Onida Company.

The operation occurs in 3 stages,

* Mapper Phase
* Shuffle Phase
* Reducer Phase

# Associated Data Files

<https://drive.google.com/file/d/0Bxr27gVaXO5sVjQ5QW0wQ3RCTUU/view?usp=sharing>

# Problem Statement

We have a dataset of sales of different TV sets across different locations.

Records look like:

Samsung|Optima|14|Madhya Pradesh|132401|14200

The fields are arranged like:

Company Name|Product Name|Size in inches|State|Pin Code|Price

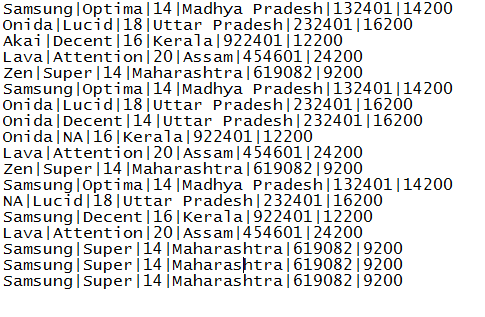
There are some invalid records which contain 'NA' in either Company Name or Product Name.

**2. Write a Map Reduce program to calculate the total units sold for each Company.**

**3. Write a Map Reduce program to calculate the total units sold in each state for Onida**

**Company.**

### Actual Data Set:



# Map Reduce Java Program: Write a Map Reduce program to calculate the total units sold for each Company

## Driver Code

**package** TotalUnitSale;

**import** org.apache.hadoop.conf.Configuration;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapreduce.Job;

**import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

**import** org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

**import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

**import** org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

**public** **class** TotalUnitSale

{

**public** **static** **void** main(String[] args) **throws** Exception

{

Configuration conf = **new** Configuration();

Job job = **new** Job(conf, "TV TotalUnitSale");// the job runs under this

job.setJarByClass(TotalUnitSale.**class**);

job.setMapOutputKeyClass(Text.**class**); //mapper key output

job.setMapOutputValueClass(IntWritable.**class**); //mapper output value

job.setOutputKeyClass(Text.**class**);// output key of the mapreduce

job.setOutputValueClass(IntWritable.**class**);//output value of the mapreduce

job.setMapperClass(TotalUnitSaleMapper.**class**);// Mapper class

job.setReducerClass(TotalUnitSaleReducer.**class**);//reducer class

job.setNumReduceTasks(2);

job.setInputFormatClass(TextInputFormat.**class**);

job.setOutputFormatClass(TextOutputFormat.**class**);

FileInputFormat.*addInputPath*(job, **new** Path(args[0]));

FileOutputFormat.*setOutputPath*(job, **new** Path(args[1]));

job.waitForCompletion(**true**);

}

}

## Mapper Code

**package** TotalUnitSale;

**import** java.io.IOException;

**import** java.util.StringTokenizer;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.LongWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapreduce.Mapper;

**public** **class** TotalUnitSaleMapper **extends** Mapper<LongWritable, Text, Text, IntWritable>

{

**private** **final** **static** IntWritable ***unit*** = **new** IntWritable(1); // declaring the Mapper value

**private** Text CompanyName = **new** Text(); //declaring the Mapper key

**public** **void** map(LongWritable key, Text value, Context context ) **throws** IOException, InterruptedException

{

String[] Linearray = value.toString().split("\\|");

StringTokenizer tokenizer=**new** StringTokenizer(Linearray[0]); //we have used the String Tokenizer class which takes array into single word/token.

**while**(tokenizer.hasMoreTokens()) // the while loop checks for the more tokens/words, if we have next token it will continue the loop

{

CompanyName.set(tokenizer.nextToken());

}

context.write(CompanyName, ***unit***); // output of the Mapper Key and Value

}

}

## Reducer Code

## 

**package** TotalUnitSale;

**import** java.io.IOException;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapreduce.Reducer;

**public** **class** TotalUnitSaleReducer **extends** Reducer<Text, IntWritable, Text, IntWritable>

{

**public** **void** reduce(Text CompanyName, Iterable<IntWritable> values, Context context) **throws** IOException, InterruptedException

{

**int** sum=0; // declaring a variable sum

**for**(IntWritable value:values) // the for loop get the iterable values and counting the values

{

sum+=value.get();

}

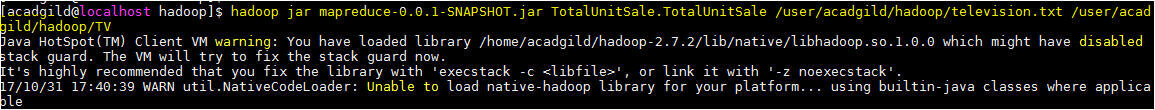
context.write(CompanyName, **new** IntWritable(sum)); // output of the the Key and value

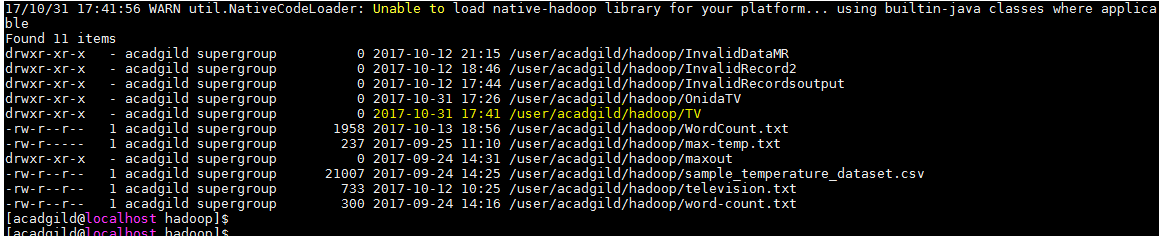
}

}

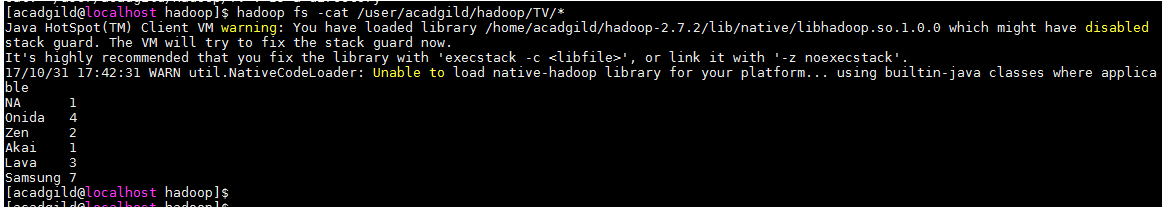
## Command to run the jar

***hadoop jar mapreduce-0.0.1-SNAPSHOT.jar TotalUnitSale.TotalUnitSale/user/acadgild/hadoop/television.txt /user/acadgild/hadoop/TV***





## Expected Output



# Write a Map Reduce program to calculate the total units sold in each state for Onida Company.

## Driver Code

**package** OnidaTotalUnit;

**import** org.apache.hadoop.conf.Configuration;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapreduce.Job;

**import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

**import** org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

**import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

**import** org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

**public** **class** OnidaTotalUnit

{

**public** **static** **void** main(String[] args) **throws** Exception

{

Configuration conf = **new** Configuration();

Job job = **new** Job(conf, "Onida Total Unit");// the job runs under this

job.setJarByClass(OnidaTotalUnit.**class**);

job.setMapOutputKeyClass(Text.**class**); //mapper key output

job.setMapOutputValueClass(IntWritable.**class**); //mapper output value

job.setOutputKeyClass(Text.**class**);//output key of the mapreduce

job.setOutputValueClass(IntWritable.**class**); //output value of the mapreduce

job.setMapperClass(OnidaMapper.**class**); // mapper class

job.setReducerClass(OnidaReducer.**class**);// reducer class

job.setNumReduceTasks(2);

job.setInputFormatClass(TextInputFormat.**class**);

job.setOutputFormatClass(TextOutputFormat.**class**);

FileInputFormat.*addInputPath*(job, **new** Path(args[0]));

FileOutputFormat.*setOutputPath*(job, **new** Path(args[1]));

job.waitForCompletion(**true**);

}

}

## Mapper Code

**package** OnidaTotalUnit;

**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.mapreduce.Mapper;

**public** **class** OnidaMapper **extends** Mapper<LongWritable, Text, Text, IntWritable>

{

**public** **void** map(LongWritable key, Text value, Context context) **throws** IOException, InterruptedException

{

String[] Linearray = value.toString().split("\\|"); //the array is split into string value and stored in Linearray

**if**(Linearray[0].equals("Onida")) // checking the word Onida in the linearray[0], if it is Onida print the state name in linearray[3]and unit value

{

Text State = **new** Text(Linearray[3]);

IntWritable unit= **new** IntWritable(1);

context.write(State, unit);

}

}

}

## Reducer Code

**package** OnidaTotalUnit;

**import** java.io.IOException;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapreduce.Reducer;

**public** **class** OnidaReducer **extends** Reducer<Text, IntWritable, Text, IntWritable>

{

**public** **void** reduce(Text State, Iterable<IntWritable> values, Context context) **throws** IOException, InterruptedException

{

**int** sum = 0; // declaring the variable sum

**for**(IntWritable value:values) // the for loop get the iterable values and counting the values

{

sum+= value.get();

}

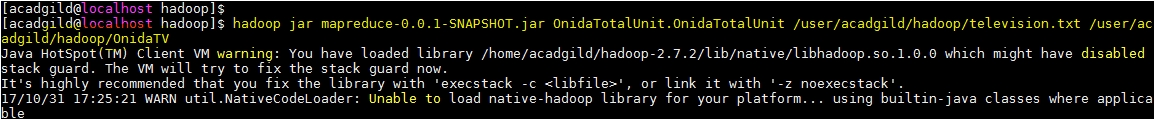
context.write(State, **new** IntWritable(sum)); // print the state name which is the key and the number of units stored in the sum

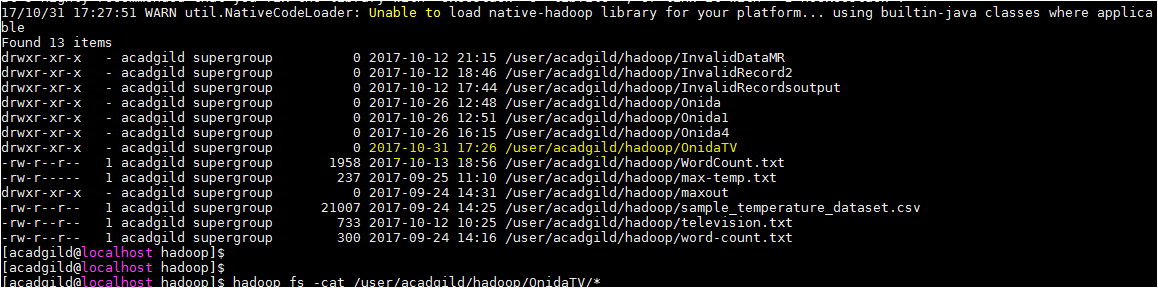
}

}

## Command to run the jar

**hadoop jar mapreduce-0.0.1-SNAPSHOT.jar OnidaTotalUnit.OnidaTotalUnit/user/acadgild/hadoop/television.txt/user/acadgild/hadoop/OnidaTV**





## Expected Output

