Bigdata Assignment 1.7

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.

Solution -

**2.**

**Driver code -**

package mapreduce;

import org.apache.hadoop.fs.Path;

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

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;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

public class Task2 {

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

Configuration conf = new Configuration();

Job job = new Job(conf, "Task2");

job.setJarByClass(Task2.class);

//Key is text as it is the company\_name

job.setMapOutputKeyClass(Text.class);

//Key is LongWritable as it is the no of units

job.setMapOutputValueClass(LongWritable.class);

//Key is text as it is the company\_name

job.setOutputKeyClass(Text.class);

//Key is LongWritable as it is the no of units

job.setOutputValueClass(LongWritable.class);

job.setMapperClass(Task2mapper.class);

job.setReducerClass(Task2reducer.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

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

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

/\*

Path out=new Path(args[1]);

out.getFileSystem(conf).delete(out);

\*/

job.waitForCompletion(true);

}

}

**Mapper Code -**

package mapreduce;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

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

public class Task2mapper extends Mapper<LongWritable, Text, Text, LongWritable> {

@Override

public void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

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

//Checking if company name or product name must not equal to NA

if(!(lineArray[0].equals("NA")||lineArray[1].equals("NA")))

{

context.write(new Text(lineArray[0]), new LongWritable(1));

}

}

}

**Reducer Code** -

package mapreduce;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class Task2reducer extends Reducer<Text, LongWritable, Text, LongWritable>

{

@Override

public void reduce(Text key, Iterable<LongWritable> values,Context context) throws IOException, InterruptedException

{

//Summing the values that is the no of units for a particular key which is the company name.

long totalSales = 0;

for(LongWritable value:values)

{

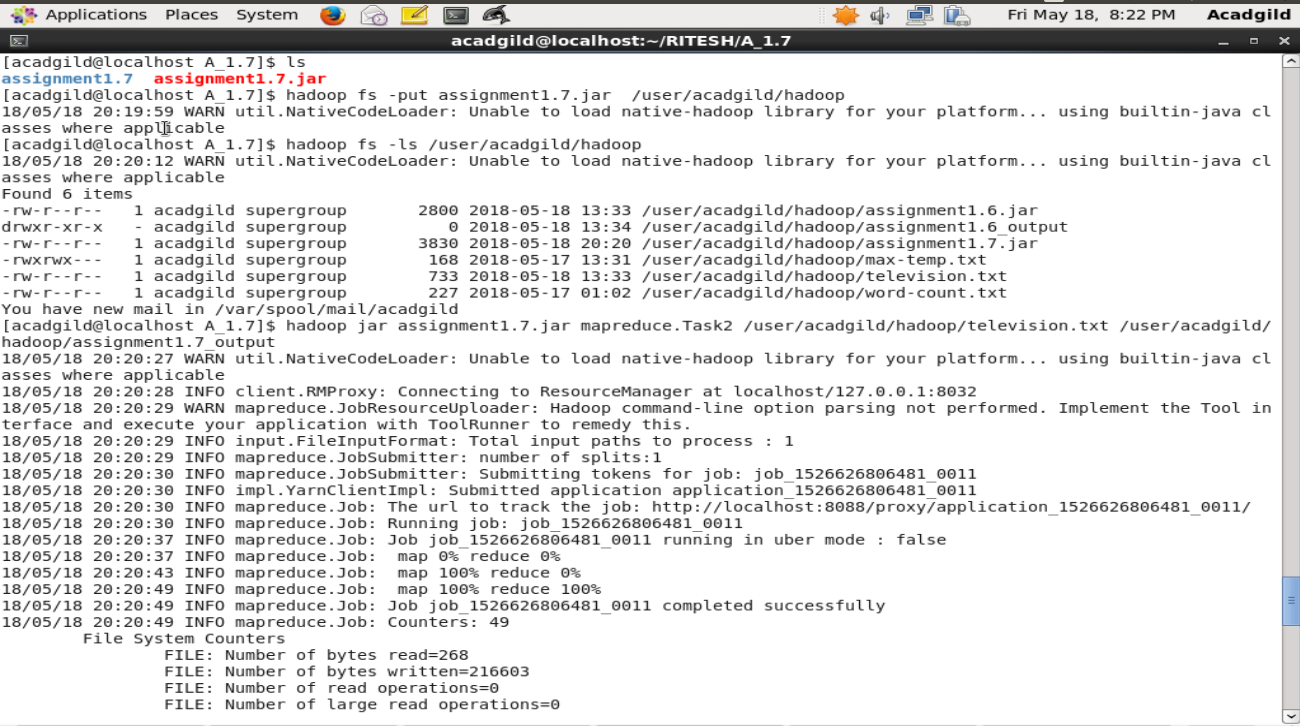
totalSales+= value.get();

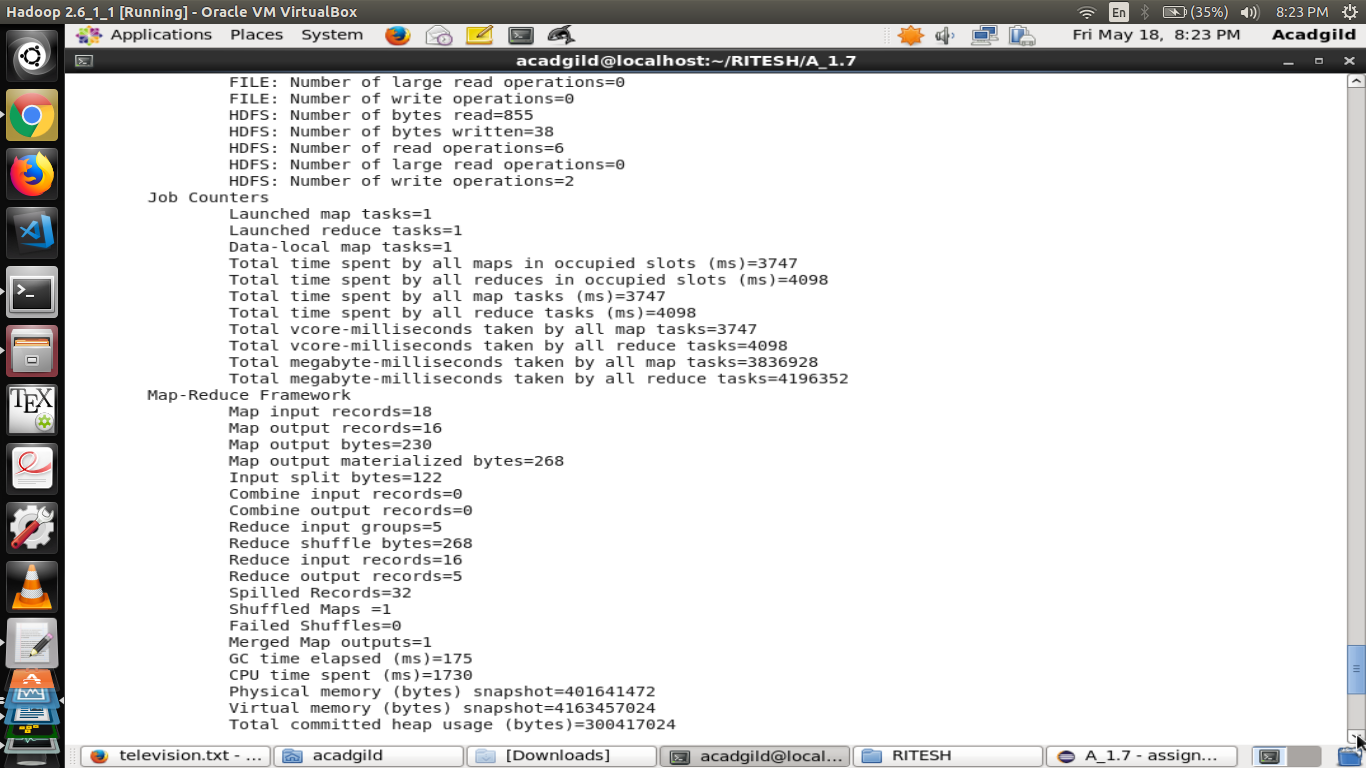
}

context.write(key,new LongWritable(totalSales));

}

}





**Execution Steps:-**

• television.txt was put to hdfs.

• After the driver code and mapping code , a jar file of was generated.

• assignmen1.7.jar file(mapping code) was put to hdfs.

• Mapping task was performed by the following command – **hadoop jar**

**assignmen1.7.jar mapreduce.Task2**

**/user/acadgild/hadoop/television.txt**

**/user/acadgild/hadoop/assignment1.7\_output**

• To see the content of the output following command was used -

**hadoop cat /user/acadgild//hadoop/assignment1.7\_output/part-r-00000**

****

**Output**

As in the above screen shot ,it is evident that the content of the output file shows the no of units sold by each company.

3.

Driver Code-

package mapreduce;

import org.apache.hadoop.fs.Path;

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

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;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

public class Task3 {

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

Configuration conf = new Configuration();

Job job = new Job(conf, "Task3");

job.setJarByClass(Task2.class);

//Key is text as it is the state name

job.setMapOutputKeyClass(Text.class);

//Key is LongWritable as it is the no of units of Onida

job.setMapOutputValueClass(LongWritable.class);

//Key is text as it is the state name

job.setOutputKeyClass(Text.class);

//Key is LongWritable as it is the no of units of Onida

job.setOutputValueClass(LongWritable.class);

job.setMapperClass(Task3mapper.class);

job.setReducerClass(Task3reducer.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

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

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

/\*

Path out=new Path(args[1]);

out.getFileSystem(conf).delete(out);

\*/

job.waitForCompletion(true);

}

}

**Mapper Code-**

package mapreduce;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

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

public class Task3mapper extends Mapper<LongWritable, Text, Text, LongWritable> {

@Override

public void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

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

//Checking if company name is Ondia or product name must not equal to NA

if((lineArray[0].equals("Onida")&& !lineArray[1].equals("NA")))

{

context.write(new Text(lineArray[3]), new LongWritable(1));

}

}

}

**Reducer Code -**

package mapreduce;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class Task3reducer extends Reducer<Text, LongWritable, Text, LongWritable>

{

@Override

public void reduce(Text key, Iterable<LongWritable> values,Context context) throws IOException, InterruptedException

{

//Summing the values no of units for a particular key which is the state name.

long totalSales = 0;

for(LongWritable value:values)

{

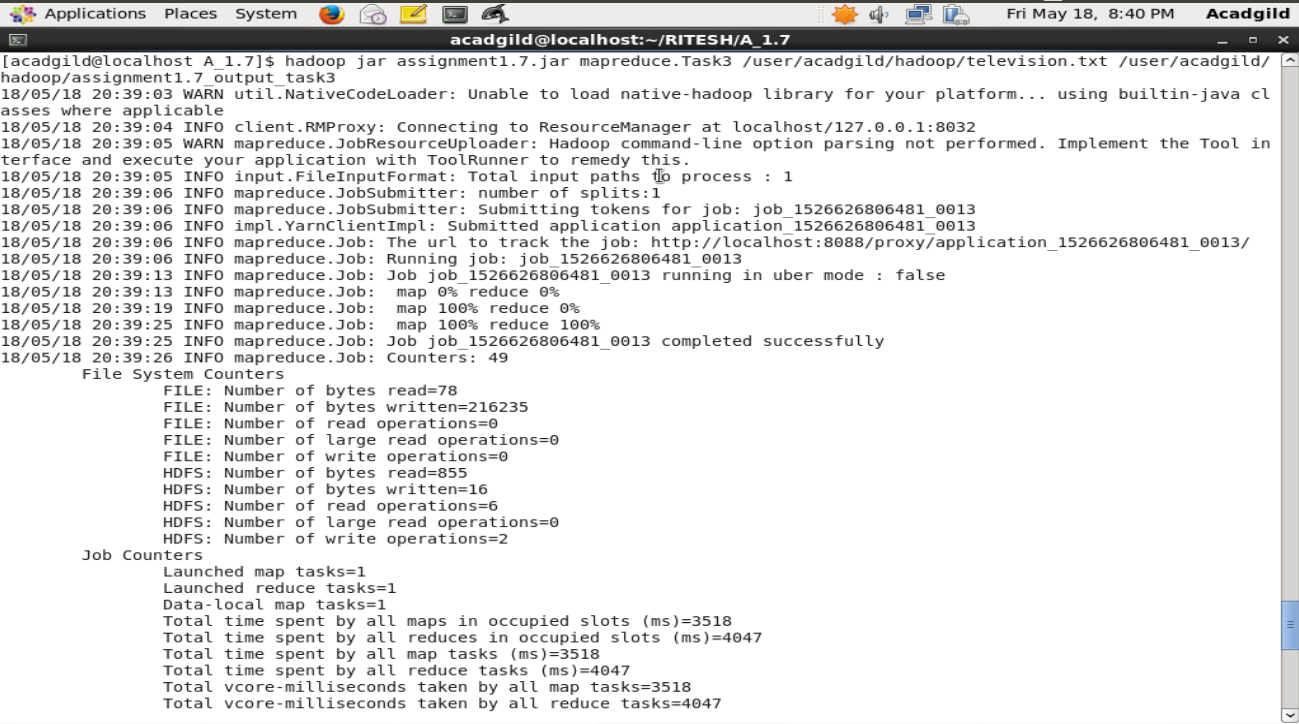
totalSales+= value.get();

}

context.write(key,new LongWritable(totalSales));

}

}



**Execution Steps:-**

• television.txt was put to hdfs.

• After the driver code and mapping code , a jar file of was generated.

• assignmen1.7.jar file(mapping code) was put to hdfs.

• Mapping task was performed by the following command – **hadoop jar**

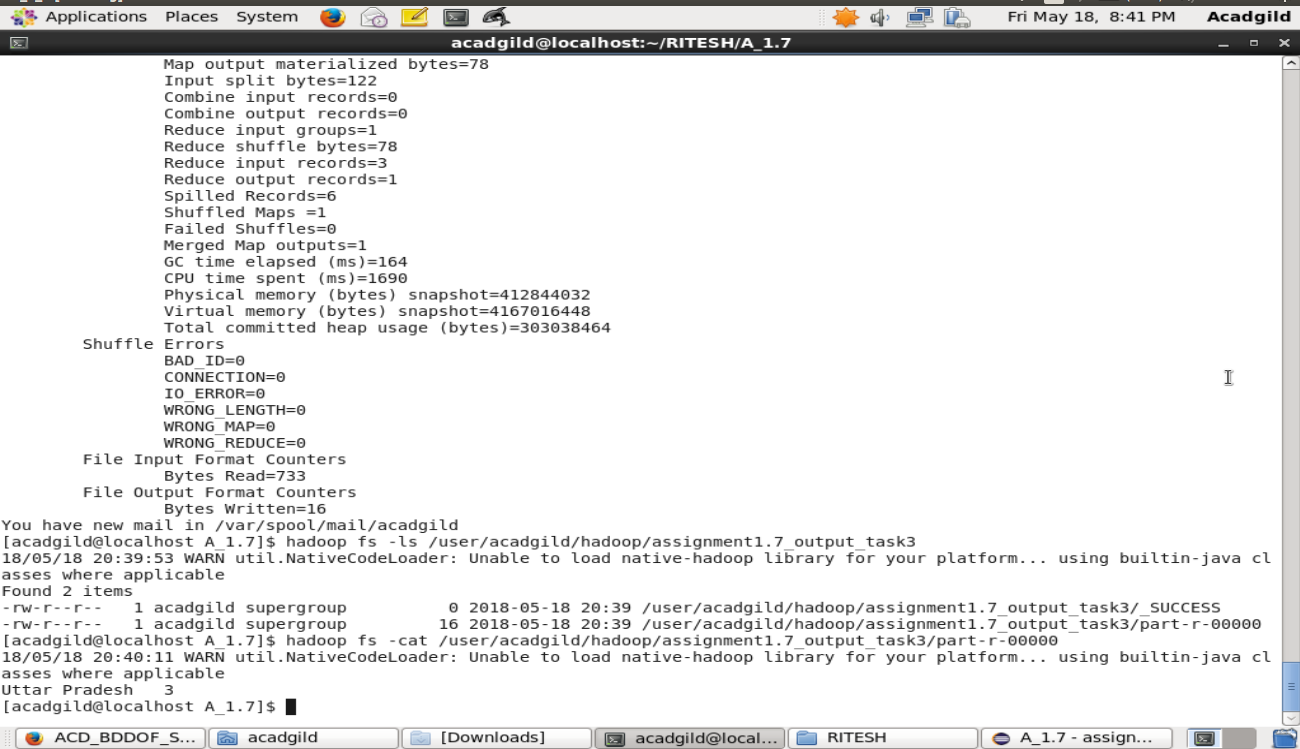
**assignmen1.7.jar mapreduce.Task3**

**/user/acadgild/hadoop/television.txt**

**/user/acadgild/hadoop/assignment1.7\_output\_task3**

• To see the content of the output following command was used -

**hadoop cat /user/acadgild//hadoop/assignment1.7\_output\_task3/part-r-00000**

****

**Output**

As in the above screen shot ,it is evident that the content of the output file shows the no of units sold by company Onida in each state.