Assignment 3.3

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.

2. To calculate the total units sold for each, we need to do "the sort and shuffle" in the mapper class method. For this, we split the records, and we select the "key" as company and the value as "1" as integer, so for every record there's a 1 value. By the way, we use the method to filter out bad records.

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

    if(recordIsInvalid(value)==false){
            Text company = new Text();
            IntWritable unit = new IntWritable();
            company = new Text(value.toString().split("\\|")[0]);
            unit = new IntWritable(1);
        context.write(company, unit );
    }
```

Now we need to reduce the data in the Reducer class method. The objective is to sum every value for each key, with this; we will get the total units sold by each company. To achieve this we write the following code:

```
private IntWritable result = new IntWritable();

public void reduce (Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException{
        int sum = 0;
        for(IntWritable val: values){
                sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
}
```

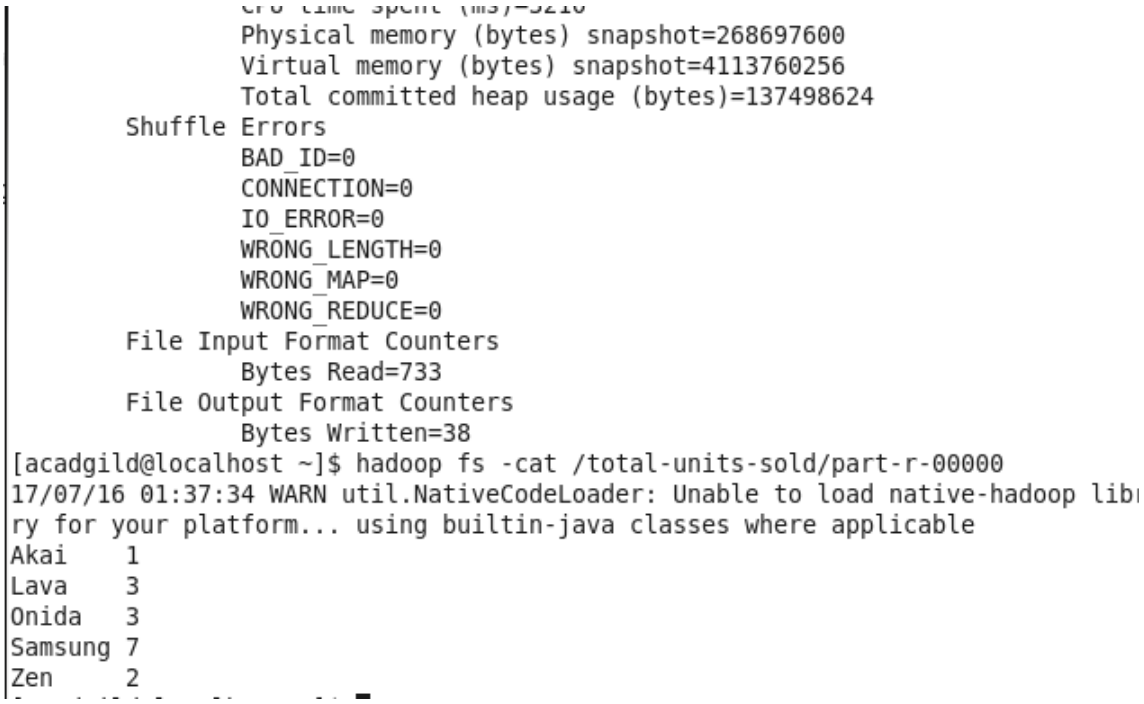
Then we need to export the .jar file from Eclipse and in the command shell we type:

hadoop jar /home/acadgild/workspace/TvSales.jar television.txt /total-units-sold

- **jar** is the command to execute the mapping and reducing
- **/home/acadgild/workspace/TvSales.jar** location on file system of the jar exported before
- **television.txt** is the file location on HDFS containing the records
- **/total-units-sold** is the destination directory on HDFS after reducing the records.

To check the results obtained on HDFS we can type

hadoop fs –cat /total-units-sold/part-r-00000



The complete code is in **total-units-sold-program.txt** and the results are in **total-units-sold.txt** obtained from HDFS using **hadoop fs –copyToLocal** command.

3. In this case, we will use the same reduce method than before, but we need to change the mapper method

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

    if(recordIsInvalid(value)==false & value.toString().split("\\|")[0].equals("Onida")){
        Text state = new Text();
        IntWritable unit = new IntWritable();
        state = new Text(value.toString().split("\\|")[3]);
        unit = new IntWritable(1);
        context.write(state, unit );
    }
```

So besides filtering out invalid records, we filter records by company. In this case, we will only accept records with "Onida" as a company. As as key, we will assing the state, so we can sum every record by state, and the value the same as before.

Then we need to export the .jar file from Eclipse and in the command shell we type:

hadoop jar /home/acadgild/workspace/TvSales.jar television.txt /total-units-onida

- **jar** is the command to execute the mapping and reducing
- **/home/acadgild/workspace/TvSales.jar** location on file system of the jar exported before
- **television.txt** is the file location on HDFS containing the records
- **/total-units-onida** is the destination directory on HDFS after reducing the records.

To check the results obtained on HDFS we can type

hadoop fs –cat /total-units-onida/part-r-00000

```
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=464
                CPU time spent (ms)=2590
                Physical memory (bytes) snapshot=267829248
                Virtual memory (bytes) snapshot=4113756160
                Total committed heap usage (bytes)=137498624
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=733
        File Output Format Counters
                Bytes Written=16
[acadgild@localhost ~]$ hadoop fs -cat /total-units-onida/part-r-00000
17/07/16 08:51:46 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Uttar Pradesh   3
[acadgild@localhost ~]$
```

We only get one state, because the other records are invalid

The complete code is in **total-units-onida-program.txt** and the results are in **total-units-onida.txt** obtained from HDFS using **hadoop fs –copyToLocal** command.