

Parallel And Distributive Computing

Project Report



05 December 2022

—

PDC

—

Dr. Nausheen Shoaib

PROJECT REPORT

Group Members:

1. Abdul Ahad Shaikh (20K-0319)
2. Ali Jodat (20K-0155)
3. Basil Ali Khan (20K-0477)

Problem Statement:

Calculating the Number of Products and Sales by Country using Java Program that utilizes Map-Reduce Framework

Task and Data:

The task is specifically focused on finding number of products and sum of sales per country given the input file data set SalesJan2009.csv.

	A	B	C	D	E	F	G	H	I	J	K	L
1	01/02/2009 6:17	Product1	1200	Mastercar	carolina	Basildon	England	United Kir	01/02/2009 6:00	01/02/2009 6:08	51.5	-1.11667
2	01/02/2009 4:53	Product1	1200	Visa	Betina	Parkville	MO	United Sta	01/02/2009 4:42	01/02/2009 7:49	39.195	-94.6819
3	01/02/2009 13:08	Product1	1200	Mastercar	Federica	Astoria	OR	United Sta	01/01/2009 16:21	01/03/2009 12:32	46.18806	-123.83
4	01/03/2009 14:44	Product1	1200	Visa	Gouya	Echuca	Victoria	Australia	9/25/05 21:13	01/03/2009 14:22	-36.1333	144.75
5	01/04/2009 12:56	Product2	3600	Visa	Gerd W	Cahaba He	AL	United Sta	11/15/08 15:47	01/04/2009 12:45	33.52056	-86.8025
6	01/04/2009 13:19	Product1	1200	Visa	LAURENCE	Mickleton	NJ	United Sta	9/24/08 15:19	01/04/2009 13:04	39.79	-75.2381
7	01/04/2009 20:11	Product1	1200	Mastercar	Fleur	Peoria	IL	United Sta	01/03/2009 9:38	01/04/2009 19:45	40.69361	-89.5889
8	01/02/2009 20:09	Product1	1200	Mastercar	adam	Martin	TN	United Sta	01/02/2009 17:43	01/04/2009 20:01	36.34333	-88.8503
9	01/04/2009 13:17	Product1	1200	Mastercar	Renee Elis	Tel Aviv	Tel Aviv	Israel	01/04/2009 13:03	01/04/2009 22:10	32.06667	34.76667
10	01/04/2009 14:11	Product1	1200	Visa	Aidan	Chatou	Ile-de-Fra	France	06/03/2008 4:22	01/05/2009 1:17	48.88333	2.15
11	01/05/2009 2:42	Product1	1200	Diners	Stacy	New York	NY	United Sta	01/05/2009 2:23	01/05/2009 4:59	40.71417	-74.0064
12	01/05/2009 5:39	Product1	1200	Amex	Heidi	Eindhoven	Noord-Br	Netherlan	01/05/2009 4:55	01/05/2009 8:15	51.45	5.466667
13	01/02/2009 9:16	Product1	1200	Mastercar	Sean	Shavano P	TX	United Sta	01/02/2009 8:32	01/05/2009 9:05	29.42389	-98.4933
14	01/05/2009 10:08	Product1	1200	Visa	Georgia	Eagle	ID	United Sta	11/11/2008 15:53	01/05/2009 10:05	43.69556	-116.353
15	01/02/2009 14:18	Product1	1200	Visa	Richard	Riverside	NJ	United Sta	12/09/2008 12:07	01/05/2009 11:01	40.03222	-74.9578
16	01/04/2009 1:05	Product1	1200	Diners	Leanne	Julianston	Meath	Ireland	01/04/2009 0:00	01/05/2009 13:36	53.67722	-6.31917

Data Set Description:

Column 1: Transaction on date

Column 2: Product

Column 3: Price

Column 4: Payment Type

Column 5: Name

Column 6: City

Column 7: State

Column 8: Country

Column 9: Account Created

Column 10: Last Login

Column 11: Latitude

Column 12: Longitude

Platform, Framework and Tools Description:

Hadoop

Hadoop is a distributed computing Framework developed and maintained by The Apache Software Foundation written in Java. Hadoop consists of HDFS and Map-Reduce and is generally deployed in a group of machines called cluster. Initially, GFS and Map-Reduce were built to empower Google Search. HDFS stands for Hadoop Distributed File System and is used to store data across multiple disks. Map-Reduce is a way to parallelize Data processing tasks.

Map-Reduce Algorithm

Map-Reduce Algorithm consists of Map() procedure that performs filtering and sorting of input data and Reduce() performs summary\aggregate function per (key, value) pair.

Java

Java is a popular programming language, created in 1995. It is owned by Oracle. It is a simple programming language. Java makes writing, compiling, and debugging programming easy. It helps to create reusable code and modular programs. Java is a class-based, object-oriented programming language and is designed to have as few implementation dependencies as possible. A general-purpose programming language made for developers to write once run anywhere that is compiled Java code can run on all platforms that support Java. Java applications are compiled to byte code that can run on any Java Virtual Machine. The syntax of Java is similar to C/C++.

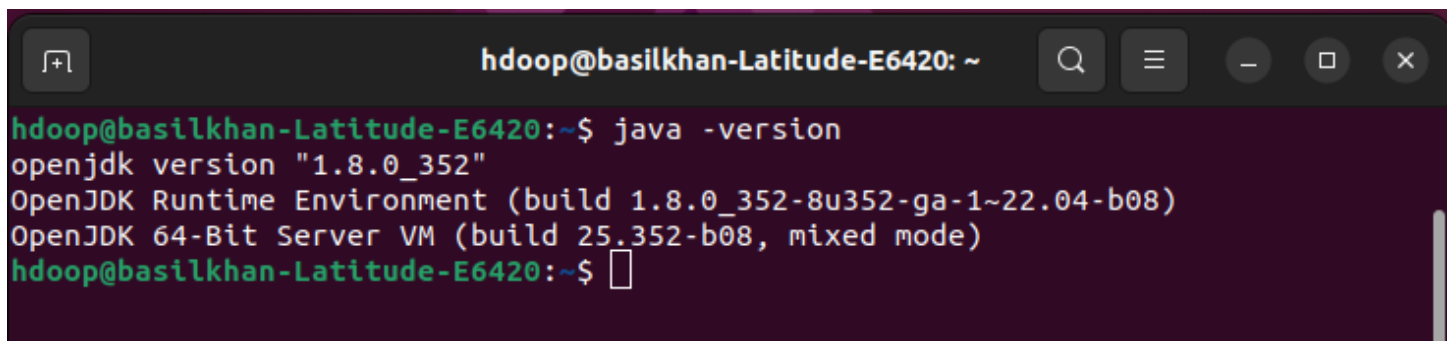
Methodology:

The main idea of this problem's solution is to use the same Key for every row with the same country name. In addition, the value used at each mapper will be the price (sales) of that row, which corresponds to the Key country.

In addition, since for this task we would like to output multiple values for each key, the code utilizes a custom made class that implements the Writable Interface. A custom Hadoop writable data type which needs to be used as value field in Map-Reduce programs must implement Writable interface.

Setup and Implementation:

❖ Java Version

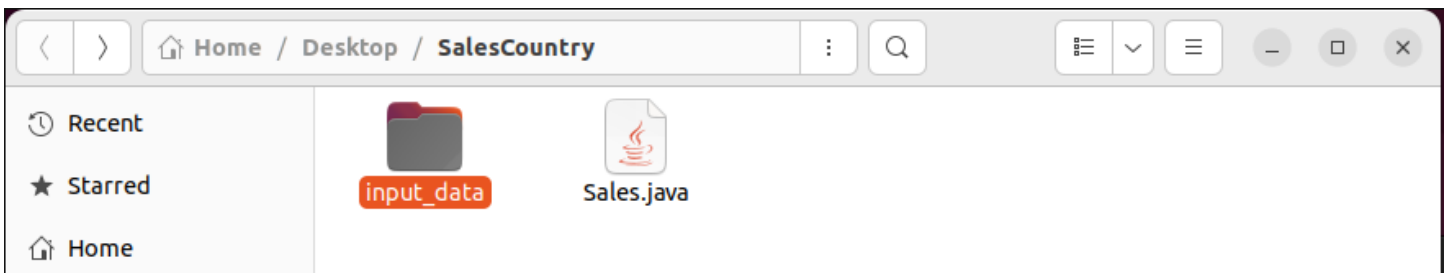
A screenshot of a terminal window with a dark background. The title bar at the top reads 'hdoop@basilkhan-Latitude-E6420: ~'. Below the title bar, the terminal shows the command 'java -version' being executed. The output is displayed in green text: 'openjdk version "1.8.0_352"', 'OpenJDK Runtime Environment (build 1.8.0_352-8u352-ga-1~22.04-b08)', and 'OpenJDK 64-Bit Server VM (build 25.352-b08, mixed mode)'. The prompt 'hdoop@basilkhan-Latitude-E6420:~\$' is followed by a cursor icon.

```
hdoop@basilkhan-Latitude-E6420: ~  
hdoop@basilkhan-Latitude-E6420:~$ java -version  
openjdk version "1.8.0_352"  
OpenJDK Runtime Environment (build 1.8.0_352-8u352-ga-1~22.04-b08)  
OpenJDK 64-Bit Server VM (build 25.352-b08, mixed mode)  
hdoop@basilkhan-Latitude-E6420:~$
```

❖ Hadoop Version

```
hadoop@basilkhan-Latitude-E6420: ~  
hadoop@basilkhan-Latitude-E6420:~$ hadoop version  
Hadoop 3.2.3  
Source code repository https://github.com/apache/hadoop -r abe535814372008549861  
3d399be3bbf01e0f131  
Compiled by ubuntu on 2022-03-20T01:18Z  
Compiled with protoc 2.5.0  
From source with checksum 39bb14faec14b3aa25388a6d7c345fe8  
This command was run using /home/hadoop/hadoop-3.2.3/share/hadoop/common/hadoop-c  
ommon-3.2.3.jar  
hadoop@basilkhan-Latitude-E6420:~$
```

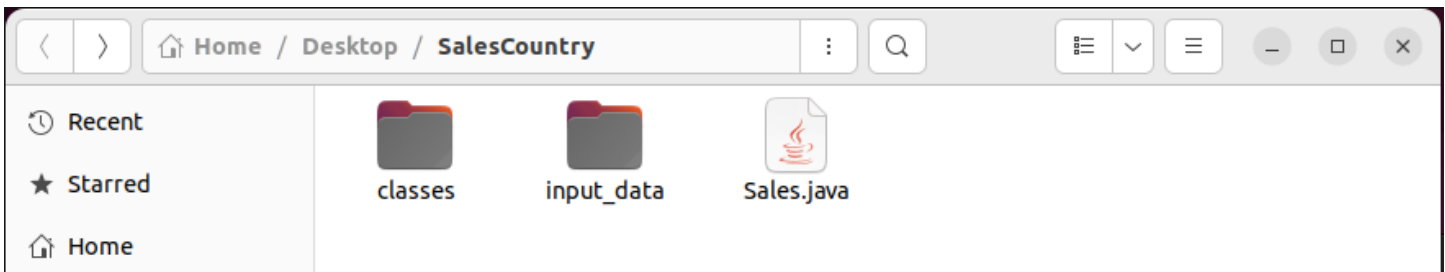
❖ Creating new folder for input data



❖ Copying data set .csv file in pervious created folder



❖ Creating folder to hold Java class files



❖ Starting start-dfs.sh and start-yarn.sh

start-dfs.sh - Starts the Hadoop DFS daemons, the name node and data node.

Start-yarn.sh- Starts the Hadoop YARN daemons, the resource and node managers.

```
hadoop@basilkhan-Latitude-E6420: ~  
hadoop@basilkhan-Latitude-E6420:~$ start-dfs.sh  
Starting namenodes on [localhost]  
Starting datanodes  
Starting secondary namenodes [basilkhan-Latitude-E6420]  
2022-11-29 14:39:53,556 WARN util.NativeCodeLoader: Unable to load native-hadoop  
library for your platform... using builtin-java classes where applicable  
hadoop@basilkhan-Latitude-E6420:~$ start-yarn.sh  
Starting resourcemanager  
Starting nodemanagers  
hadoop@basilkhan-Latitude-E6420:~$ jps  
8180 ResourceManager  
8676 Jps  
7606 NameNode  
7913 SecondaryNameNode  
8299 NodeManager  
7742 DataNode  
hadoop@basilkhan-Latitude-E6420:~$
```

❖ For checking services to start we go to: localhost:9870

Namenode information x +

localhost:9870/dfshealth.html#tab-overview




Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Overview 'localhost:9000' (active)




Started:	Tue Nov 29 14:39:47 +0500 2022
Version:	3.2.3, rabe5358143720085498613d399be3bbf01e0f131
Compiled:	Sun Mar 20 06:18:00 +0500 2022 by ubuntu from branch-3.2.3
Cluster ID:	CID-39fd5e80-9ee3-41ff-9ff1-f63415c981f6
Block Pool ID:	BP-50654569-127.0.1.1-1669304534308

Summary

Browse Directory

/   

Show entries Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	drwxr-xr-x	hdoop	supergroup	0 B	Nov 24 21:22	0	0 B	WordCountTutorial	
<input type="checkbox"/>	drwxr-xr-x	hdoop	supergroup	0 B	Nov 26 16:51	0	0 B	Youtube	
<input type="checkbox"/>	drwx-----	hdoop	supergroup	0 B	Nov 24 21:22	0	0 B	tmp	

Showing 1 to 3 of 3 entries

Hadoop, 2022.

- ❖ Setting `HADOOP_CLASSPATH` environment variable and making it sure that it is set correctly

```
hdoop@basilkhan-Latitude-E6420: ~
hdoop@basilkhan-Latitude-E6420:~$ export HADOOP_CLASSPATH=$(hadoop classpath)
hdoop@basilkhan-Latitude-E6420:~$ echo $HADOOP_CLASSPATH
/home/hdoop/hadoop-3.2.3/etc/hadoop:/home/hdoop/hadoop-3.2.3/share/hadoop/common
/lib/*:/home/hdoop/hadoop-3.2.3/share/hadoop/common/*:/home/hdoop/hadoop-3.2.3/s
hare/hadoop/hdfs:/home/hdoop/hadoop-3.2.3/share/hadoop/hdfs/lib/*:/home/hdoop/ha
dooop-3.2.3/share/hadoop/hdfs/*:/home/hdoop/hadoop-3.2.3/share/hadoop/mapreduce/l
ib/*:/home/hdoop/hadoop-3.2.3/share/hadoop/mapreduce/*:/home/hdoop/hadoop-3.2.3/
share/hadoop/yarn:/home/hdoop/hadoop-3.2.3/share/hadoop/yarn/lib/*:/home/hdoop/h
adoop-3.2.3/share/hadoop/yarn/*
hdoop@basilkhan-Latitude-E6420:~$
```

- ❖ Creating a directory on Hadoop file system

```
hdoop@basilkhan-Latitude-E6420: ~
hdoop@basilkhan-Latitude-E6420:~$ hadoop fs -mkdir /CountrySales
2022-11-29 14:47:56,293 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
hdoop@basilkhan-Latitude-E6420:~$
```

Browse Directory

/

Go!

Show

25

▼

entries

Search:

<input type="checkbox"/>	<div><div>⌵</div>Permission</div>	<div><div>⌵</div>Owner</div>	<div><div>⌵</div>Group</div>	<div><div>⌵</div>Size</div>	<div><div>⌵</div>Last Modified</div>	<div><div>⌵</div>Replication</div>	<div><div>⌵</div>Block Size</div>	<div><div>⌵</div>Name</div>	<div><div>⌵</div></div>
<input type="checkbox"/>	drwxr-xr-x	hdoop	supergroup	0 B	Nov 29 14:47	0	0 B	CountrySales	<div><div></div></div>

❖ Creating a directory inside it for input data set

```
hdoop@basilkhan-Latitude-E6420: ~  
hdoop@basilkhan-Latitude-E6420:~$ hadoop fs -mkdir /CountrySales/Input  
2022-11-29 14:49:36,589 WARN util.NativeCodeLoader: Unable to load native-hadoop  
library for your platform... using builtin-java classes where applicable  
hdoop@basilkhan-Latitude-E6420:~$
```

Browse Directory

/CountrySales

Go!

Show

25

▼

entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	drwxr-xr-x	hdoop	supergroup	0 B	Nov 29 14:49	0	0 B	Input	

Showing 1 to 1 of 1 entries

Previous

1

Next

❖ Uploading the input file to that HDFS directory

Home / Desktop / SalesCountry / input_data

```
hdoop@basilkhan-Latitude-E6420: ~  
hdoop@basilkhan-Latitude-E6420:~$ hadoop fs -put '/home/hdoop/Desktop/SalesCountry/input_data/SalesJan2009.csv' /CountrySales/Input  
2022-11-29 14:52:02,359 WARN util.NativeCodeLoader: Unable to load native-hadoop  
library for your platform... using builtin-java classes where applicable  
hdoop@basilkhan-Latitude-E6420:~$
```

Browse Directory

/CountrySales/Input

Go!

Show

25

entries

Search:

☐

Permission

Owner

Group

Size

Last Modified

Replication

Block Size

Name

☐

-rw-r--r--

hdoop

supergroup

112.2 KB

Nov 29 14:52

1

128 MB

SalesJan2009.csv

Showing 1 to 1 of 1 entries

Previous

1

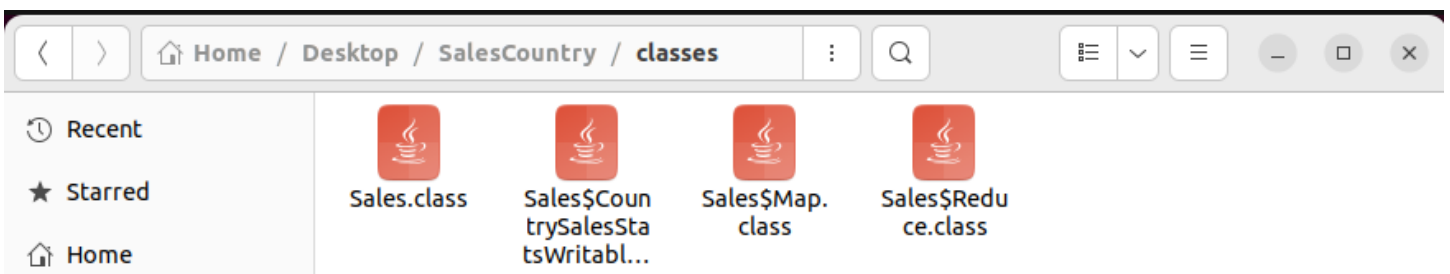
Next

❖ Changing current directory to local machine directory

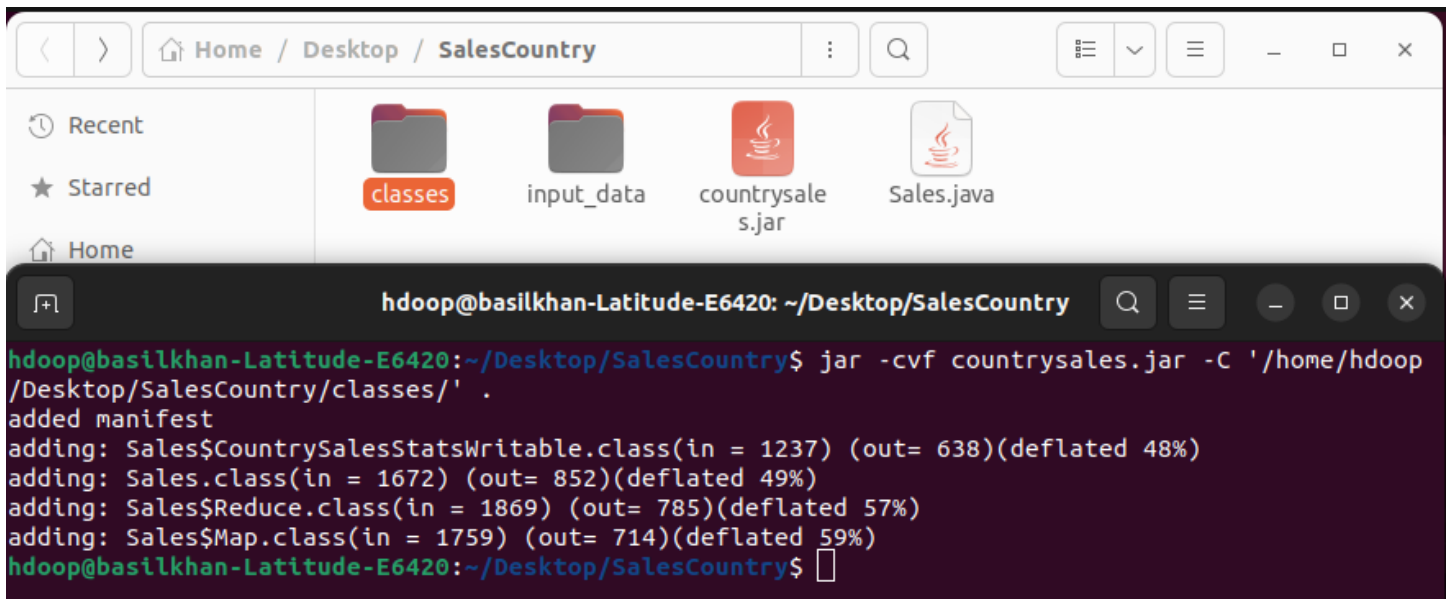
```
hdoop@basilkhan-Latitude-E6420: ~/Desktop/SalesCountry
hdoop@basilkhan-Latitude-E6420:~$ cd '/home/hdoop/Desktop/SalesCountry'
hdoop@basilkhan-Latitude-E6420:~/Desktop/SalesCountry$
```

❖ Compiling the Java code and checking files

```
hdoop@basilkhan-Latitude-E6420: ~/Desktop/SalesCountry
hdoop@basilkhan-Latitude-E6420:~/Desktop/SalesCountry$ javac -classpath ${HADOOP_CLASSPATH} -d '/home/hdoop/Desktop/SalesCountry/classes' '/home/hdoop/Desktop/SalesCountry/Sales.java'
hdoop@basilkhan-Latitude-E6420:~/Desktop/SalesCountry$
```



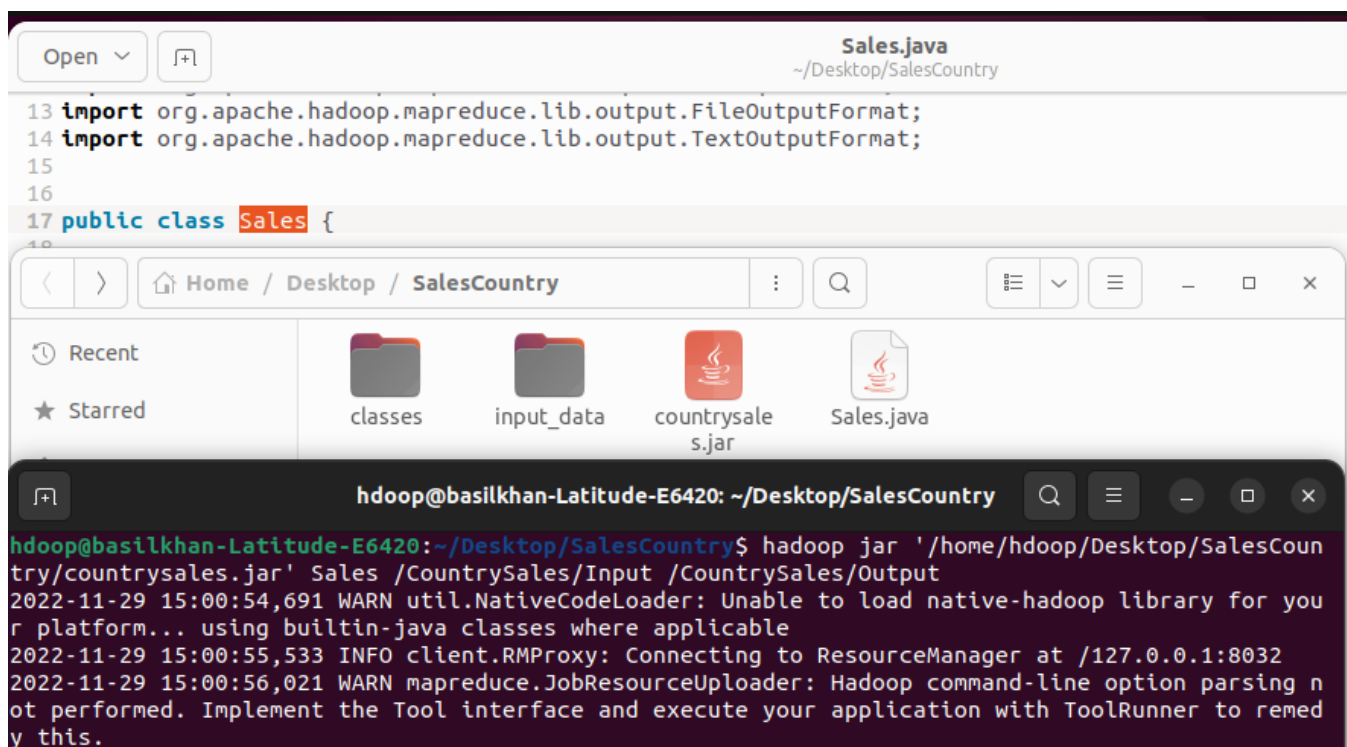
❖ Putting output files in one JAR file



The screenshot shows a file manager window titled 'SalesCountry' with files 'classes', 'input_data', 'countrysales.jar', and 'Sales.java'. Below it, a terminal window shows the command to create the JAR file:

```
hadoop@basilkhan-Latitude-E6420: ~/Desktop/SalesCountry
hadoop@basilkhan-Latitude-E6420:~/Desktop/SalesCountry$ jar -cvf countrysales.jar -C '/home/hadoop/Desktop/SalesCountry/classes/' .
added manifest
adding: Sales$CountrySalesStatsWritable.class(in = 1237) (out= 638)(deflated 48%)
adding: Sales.class(in = 1672) (out= 852)(deflated 49%)
adding: Sales$Reduce.class(in = 1869) (out= 785)(deflated 57%)
adding: Sales$Map.class(in = 1759) (out= 714)(deflated 59%)
hadoop@basilkhan-Latitude-E6420:~/Desktop/SalesCountry$
```

❖ Running JAR file on Hadoop



The screenshot shows a code editor with the 'Sales.java' file. Below it, a terminal window shows the command to run the JAR file on Hadoop:

```
hadoop@basilkhan-Latitude-E6420:~/Desktop/SalesCountry$ hadoop jar '/home/hadoop/Desktop/SalesCountry/countrysales.jar' Sales /CountrySales/Input /CountrySales/Output
2022-11-29 15:00:54,691 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2022-11-29 15:00:55,533 INFO client.RMPProxy: Connecting to ResourceManager at /127.0.0.1:8032
2022-11-29 15:00:56,021 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
```

```
Use
2022-11-29 15:01:06,765 INFO mapreduce.Job: map 0% reduce 0%
2022-11-29 15:01:11,903 INFO mapreduce.Job: map 100% reduce 0%
2022-11-29 15:01:16,959 INFO mapreduce.Job: map 100% reduce 100%
2022-11-29 15:01:17,998 INFO mapreduce.Job: Job job_1669714808938_0001 completed successfully
2022-11-29 15:01:18,161 INFO mapreduce.Job: Counters: 54
```

```

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=114888
File Output Format Counters
    Bytes Written=948

```

Browse Directory

Show entries
 Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rw-r--r--	hadoop	supergroup	0 B	Nov 29 15:01	1	128 MB	_SUCCESS	
<input type="checkbox"/>	-rw-r--r--	hadoop	supergroup	948 B	Nov 29 15:01	1	128 MB	part-r-00000	

Showing 1 to 2 of 2 entries

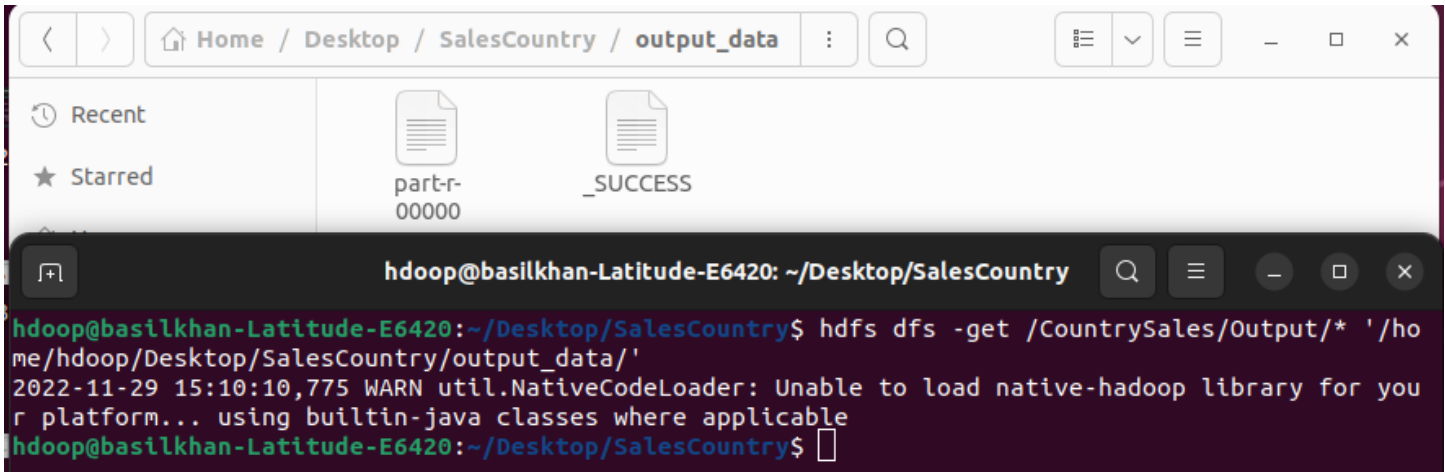
❖ Output

```

hadoop@basilkhan-Latitude-E6420: ~/Desktop/SalesCountry
hadoop@basilkhan-Latitude-E6420:~/Desktop/SalesCountry$ hadoop fs -cat /CountrySales/Output/*
2022-11-29 15:06:05,907 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Argentina      1 1200
Australia      38 64800
Austria 7      10800
Bahrain 1      1200
Belgium 8      12000
Bermuda 1      1200
Brazil 5       12300
Bulgaria       1 1200
Canada 76     124800
Cayman Isls    1 1200
China 1        1200
Costa Rica     1 1200
Czech Republic 3 6000
Denmark 15     18000
Dominican Republic 1 1200
Finland 2      2400
France 27     53100
Germany 25     42000
Greece 1       1200
Guatemala      1 1200
Hong Kong      1 1200
Hungary 3      3600
Iceland 1      1200
India 2        2400
Ireland 49     69900
Israel 1       1200
Italy 15      37800
Japan 2        2400
Jersey 1       1200
Kuwait 1       1200
Latvia 1       1200
Luxembourg     1 1200
Malaysia       1 1200
Malta 2        4800
Mauritius      1 3600

```

❖ Copying output files from HDFS environment to local machine directory



The image shows a text editor window with the title `part-r-00000` and the path `~/Desktop/SalesCountry/output_data`. The editor contains a list of countries and their corresponding values, numbered 1 to 37.

1	Argentina	1	1200
2	Australia	38	64800
3	Austria	7	10800
4	Bahrain	1	1200
5	Belgium	8	12000
6	Bermuda	1	1200
7	Brazil	5	12300
8	Bulgaria	1	1200
9	Canada	76	124800
10	Cayman Isls	1	1200
11	China	1	1200
12	Costa Rica	1	1200
13	Czech Republic	3	6000
14	Denmark	15	18000
15	Dominican Republic	1	1200
16	Finland	2	2400
17	France	27	53100
18	Germany	25	42000
19	Greece	1	1200
20	Guatemala	1	1200
21	Hong Kong	1	1200
22	Hungary	3	3600
23	Iceland	1	1200
24	India	2	2400
25	Ireland	49	69900
26	Israel	1	1200
27	Italy	15	37800
28	Japan	2	2400
29	Jersey	1	1200
30	Kuwait	1	1200
31	Latvia	1	1200
32	Luxembourg	1	1200
33	Malaysia	1	1200
34	Malta	2	4800
35	Mauritius	1	3600
36	Moldova	1	1200
37	Monaco	2	2400

Libraries and Packages:

```
import java.io.IOException;
import java.io.DataInput;
import java.io.DataOutput;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
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;
```

Code Snippets and Explanation:

- ❖ The objects of this class will be utilized, since for this task we would like to output multiple values for each key. A custom Hadoop writable data type which needs to be used as value field in Map-Reduce programs must implement Writable interface

```
public class Sales {
    public static class CountrySalesStatsWritable implements Writable {
        private IntWritable productCount;
        private LongWritable priceSum;
        // Default Constructor
        public CountrySalesStatsWritable() {
            this.productCount = new IntWritable();
            this.priceSum = new LongWritable();
        }
        // Custom Constructor
        public CountrySalesStatsWritable(IntWritable productCount, LongWritable priceSum) {
            this.productCount = productCount;
            this.priceSum = priceSum;
        }
        @Override
        // Overriding default readFields method.
        public void readFields(DataInput in) throws IOException {
            productCount.readFields(in);
            priceSum.readFields(in);
        }
        @Override
        // Overriding default write method.
        public void write(DataOutput out) throws IOException {
            productCount.write(out);
            priceSum.write(out);
        }
        @Override
        // Overriding default toString method.
        public String toString() {
            return productCount.toString() + " " + priceSum.toString();
        }
    }
}
```

- ❖ Input types to Mapper are: <LongWritable, V:Text>
Output types from Mapper are: <Text, V:LongWritable>

```
public static class Map extends Mapper<LongWritable, Text, Text, LongWritable> {
    private Text country = new Text();
    private LongWritable price;

    // Overwritten method of Mapper
    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

        String line = value.toString(); // Transform the value (input line) from Text object to String
        String[] columns = line.split(regex: ",");

        country.set(columns[7]); // Country name column
        price = new LongWritable(Long.parseLong(columns[2]));
        context.write(country, price);
    }
}
```

- ❖ Input types to Reducer are: <Text, V:list(LongWritable)>
Output types from Reducer are: <Text, V:CountrySalesStatsWritable>

```
public static class Reduce extends Reducer<Text, LongWritable, Text, CountrySalesStatsWritable> {

    // Overwritten method of Reducer
    public void reduce(Text key, Iterable<LongWritable> values, Context context)
        throws IOException, InterruptedException {
        int productCount = 0;
        long priceSum = 0;

        for (LongWritable price : values) {
            productCount++;
            priceSum += price.get();
        }

        CountrySalesStatsWritable countrySalesStatsWritable = new CountrySalesStatsWritable(
            new IntWritable(productCount), new LongWritable(priceSum));
        context.write(key, countrySalesStatsWritable);
    }
}
```


❖ Main function

```
Run | Debug
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "sales"); // The the name of the job is just for seeing which program-job is
                                           // running
    job.setJarByClass(Sales.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(LongWritable.class);
    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

Conclusion:

This project implements Hadoop Map-Reduce algorithm on the Country Sales data and display the results in output file in Hadoop file system.

Data Set and Reference Links:

- ❖ <https://www.cs.ucy.ac.cy/courses/DSC511/data/SalesJan2009.csv>
- ❖ <https://medium.com/edureka/mapreduce-tutorial-3d9535ddbe7c>
- ❖ <http://hadooptutorial.info/creating-custom-hadoop-writable-data-type/>