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 e	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	Eindhover	Noord-Bra	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 F	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	Julianstow	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

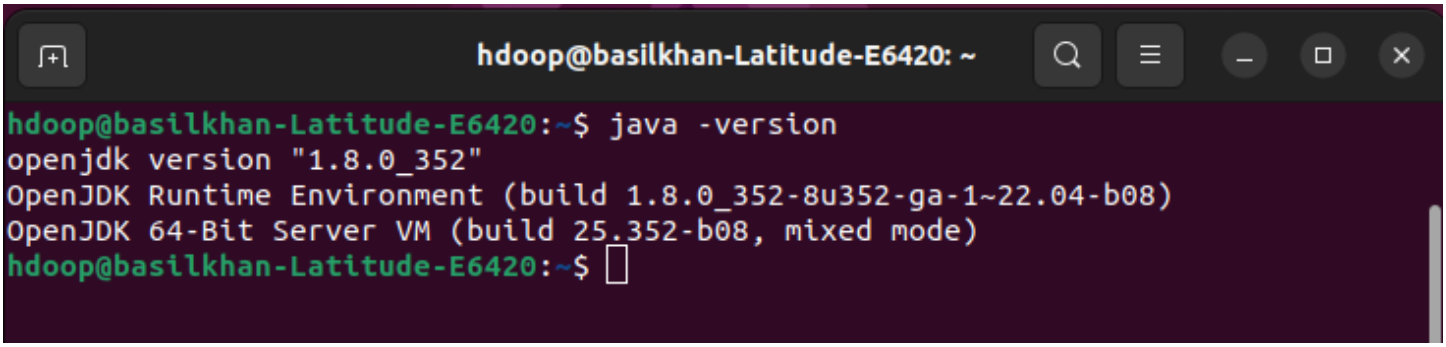
Java is a popular programming language, created in 1995. It is owned by Oracle. It is a simple programming language. Java makes writing, compiling, and debugging programming easy. It helps to create reusable code and modular programs. Java is a class-based, object-oriented programming language and is 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 Writeable 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:

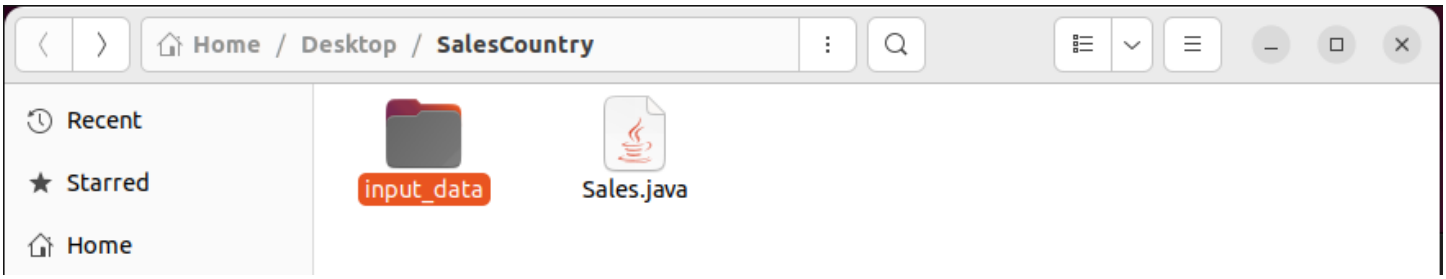
- *Java Version*



❖ *Hadoop Version*

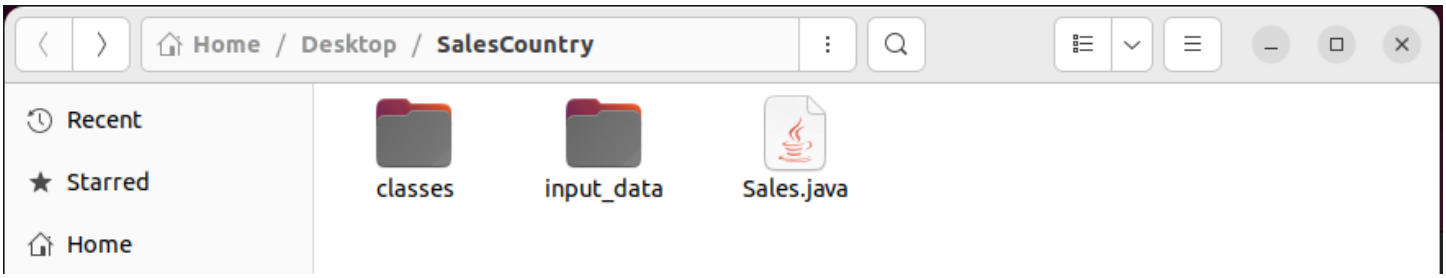
```
hdoop@basilkhan-Latitude-E6420: ~
hdoop@basilkhan-Latitude-E6420:~$ hadoop version
Hadoop 3.2.3
Source code repository https://github.com/apache/hadoop -r abe5358143720085498613d399be3bbf01e0f131
Compiled by ubuntu on 2022-03-20T01:18Z
Compiled with protoc 2.5.0
From source with checksum 39bb14faec14b3aa25388a6d7c345fe8
This command was run using /home/hdoop/hadoop-3.2.3/share/hadoop/common/hadoop-common-3.2.3.jar
hdoop@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*

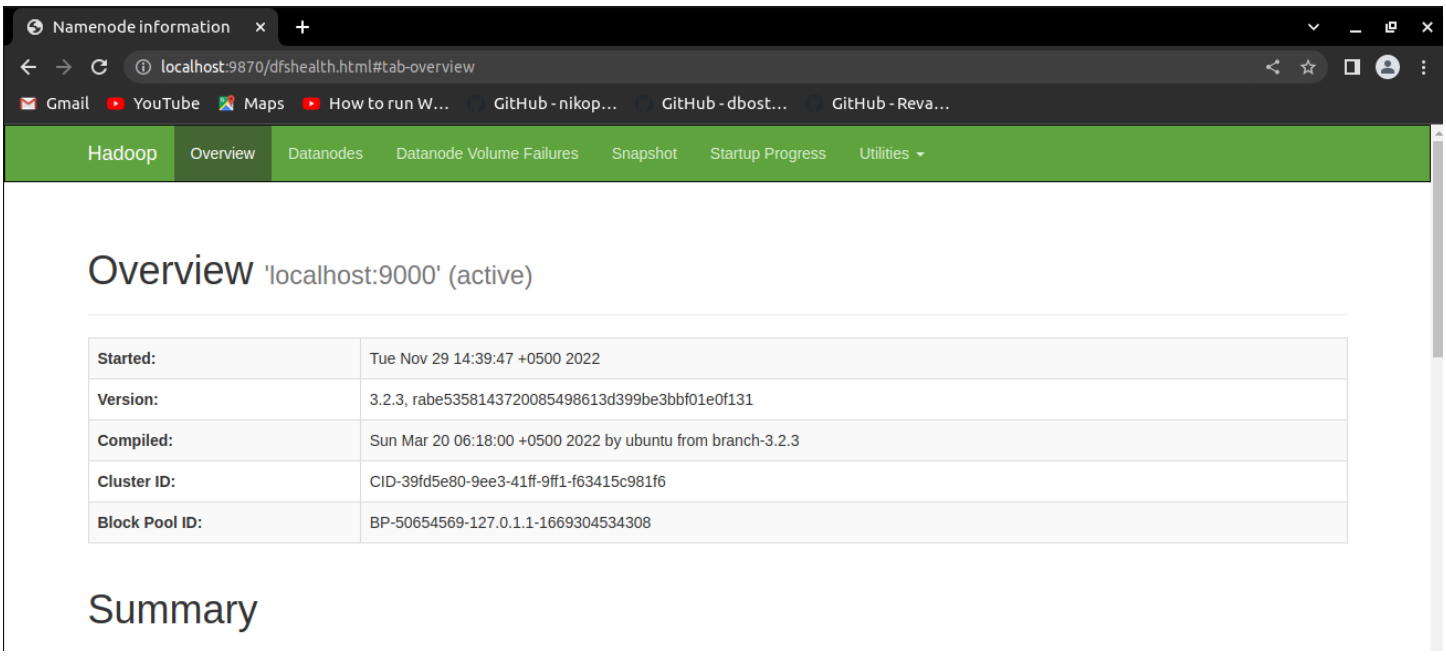


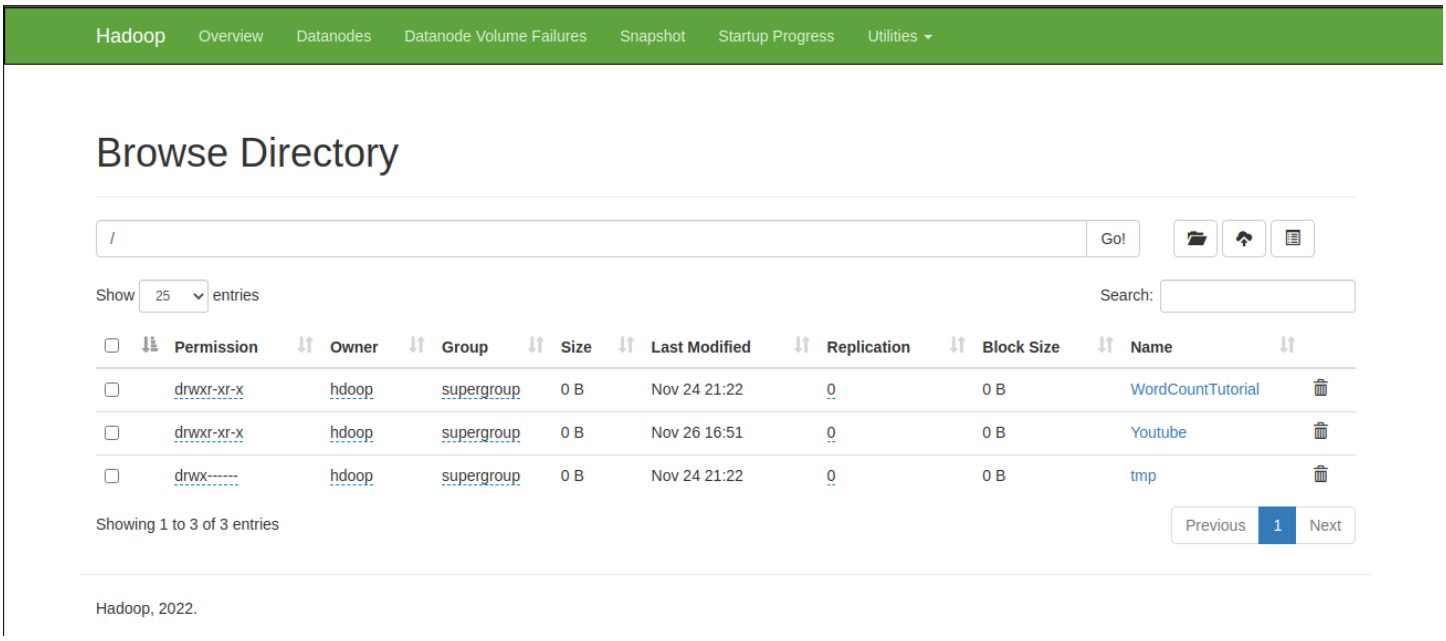
❖ *Starting start-dfs.sh and start-yarn.sh*

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.

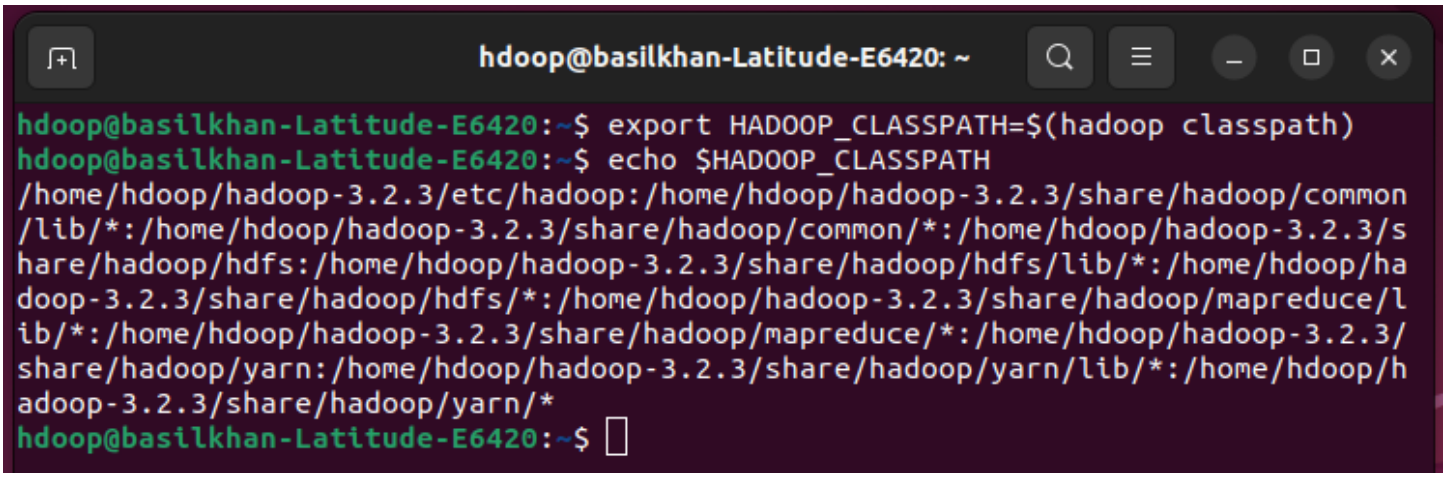
```
hdoop@basilkhan-Latitude-E6420: ~
hdoop@basilkhan-Latitude-E6420:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [basilkhan-Latitude-E6420]
2022-11-29 14:39:53,556 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform...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

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

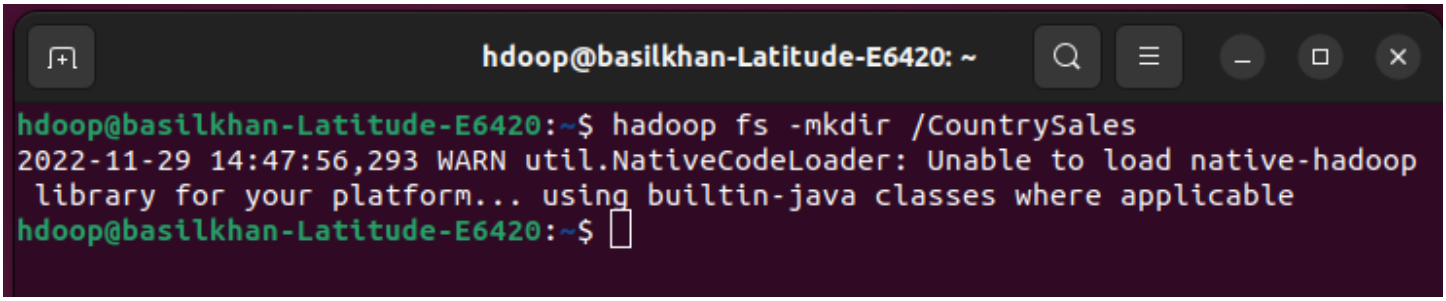




- *Setting HADOOP_CLASSPATH environment variable and making it sure that it is set correctly*



- *Creating a directory on Hadoop file system*



Browse Directory

/ Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hdoop	supergroup	0 B	Nov 29 14:47	0	0 B	CountrySales

❖ *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:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hdoop	supergroup	0 B	Nov 29 14:49	0	0 B	Input

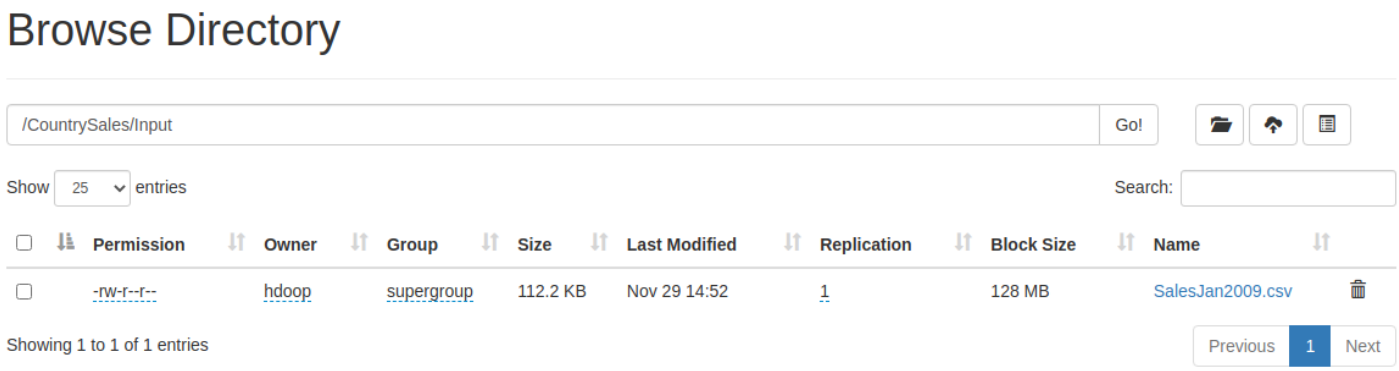
Showing 1 to 1 of 1 entries Previous 1 Next

❖ *Uploading the input file to that HDFS directory*

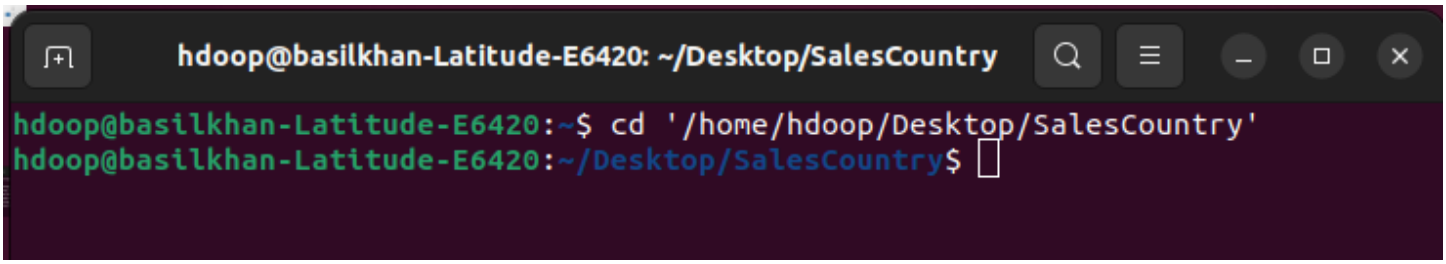
Home / Desktop / SalesCountry / input_data

SalesJan2009.csv

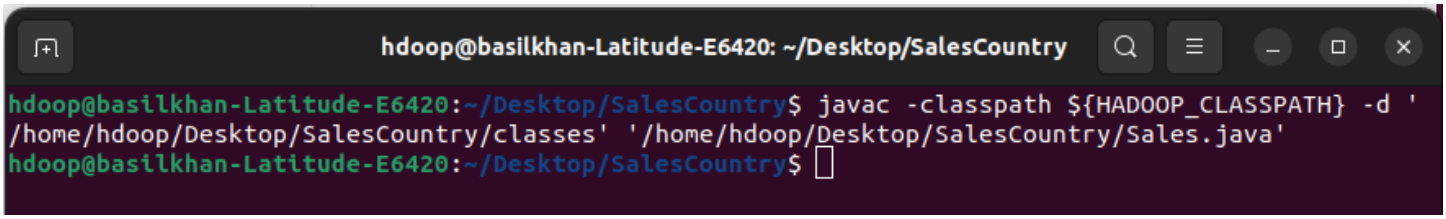
```
hdoop@basilkhan-Latitude-E6420: ~
hdoop@basilkhan-Latitude-E6420:~$ hadoop fs -put '/home/hdoop/Desktop/SalesCount
ry/input_data/SalesJan2009.csv' /CountrySales/Input
2022-11-29 14:52:02,359 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java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

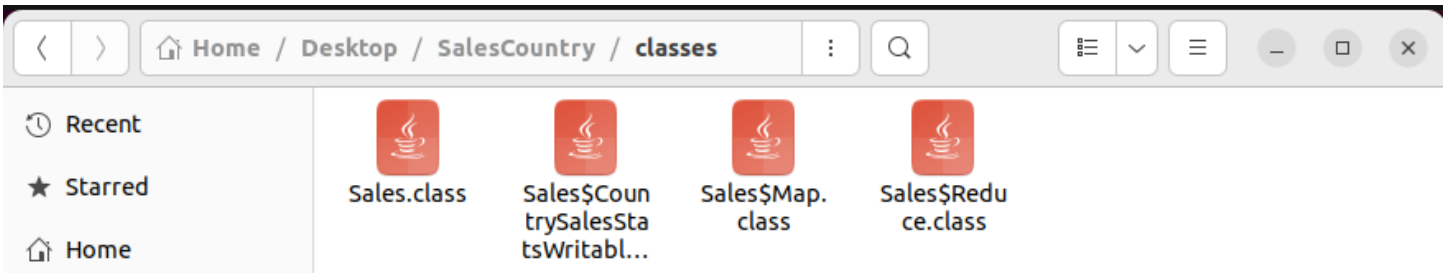


❖ *Changing current directory to local machine directory*

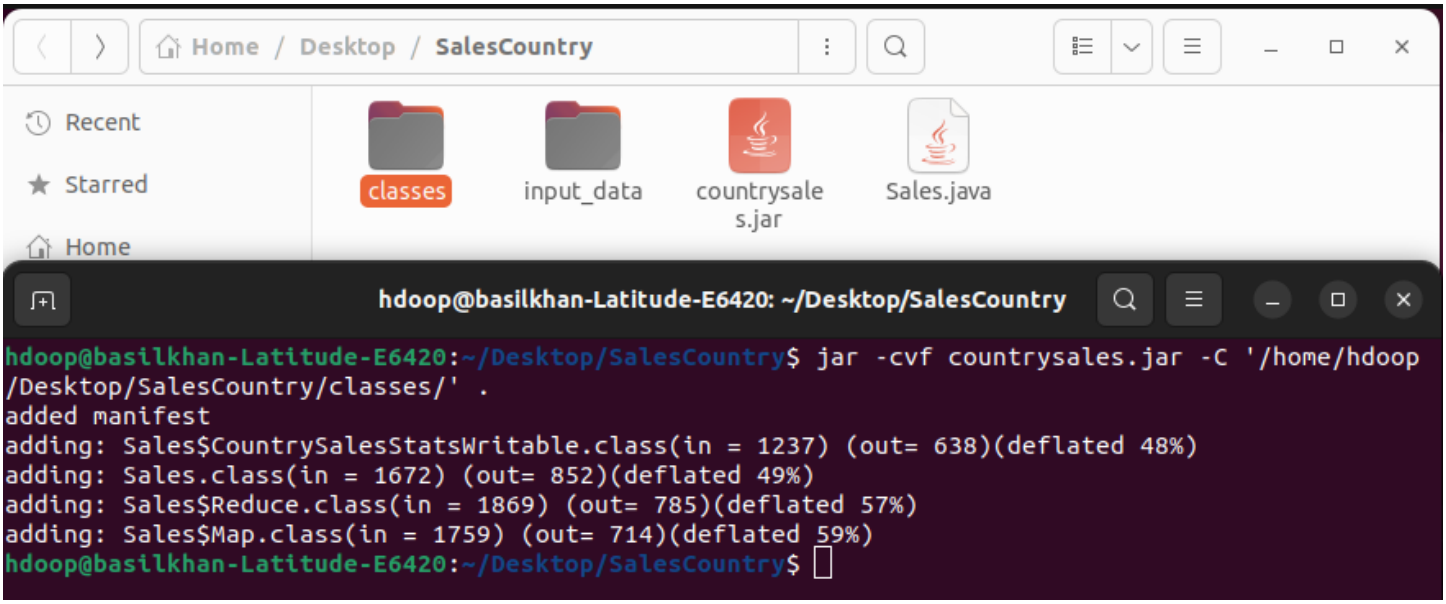


❖ *Compiling the Java code and checking files*

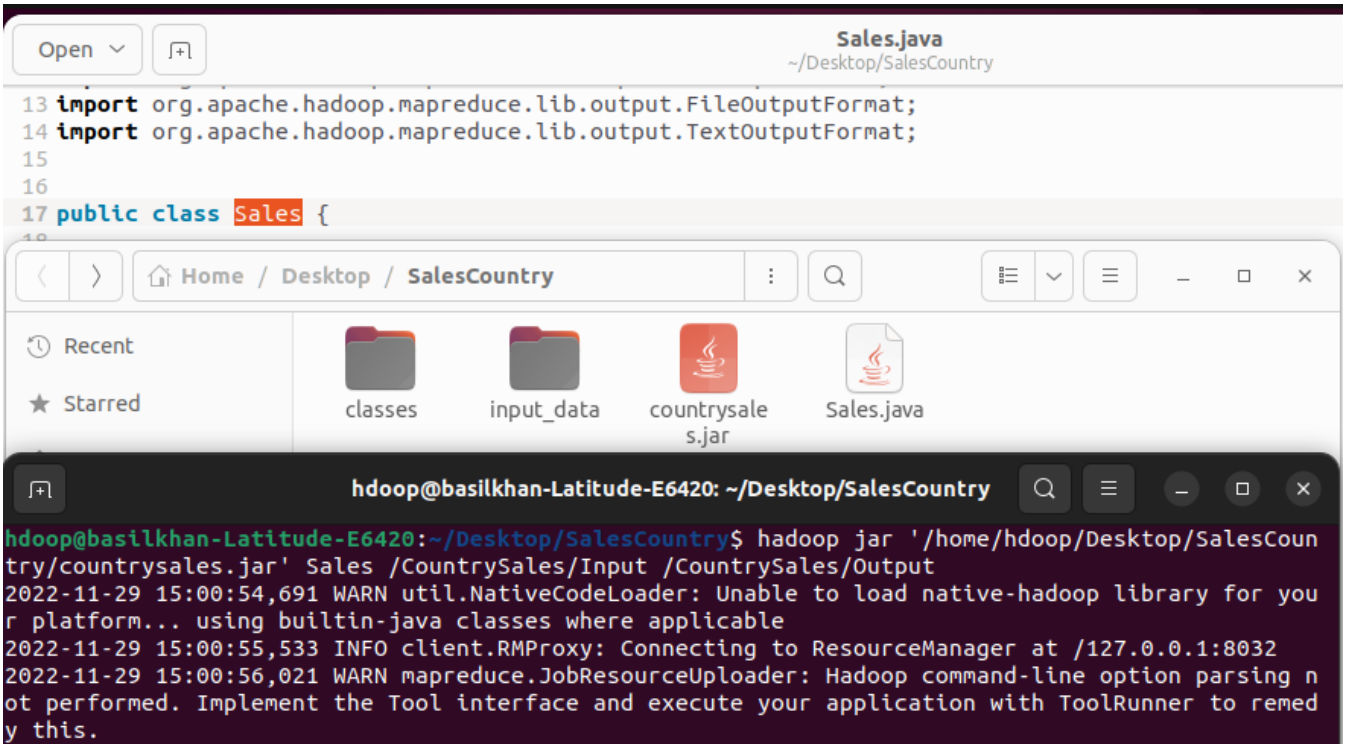


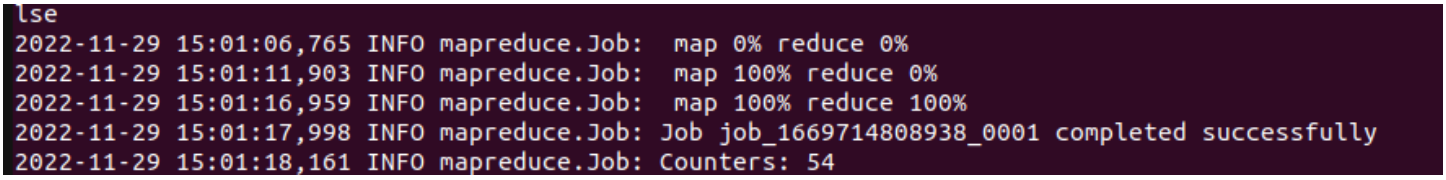


❖ *Putting output files in one JAR file*



❖ *Running JAR file on Hadoop*





```
        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

/CountrySales/Output Go!

Show 25 entries Search:

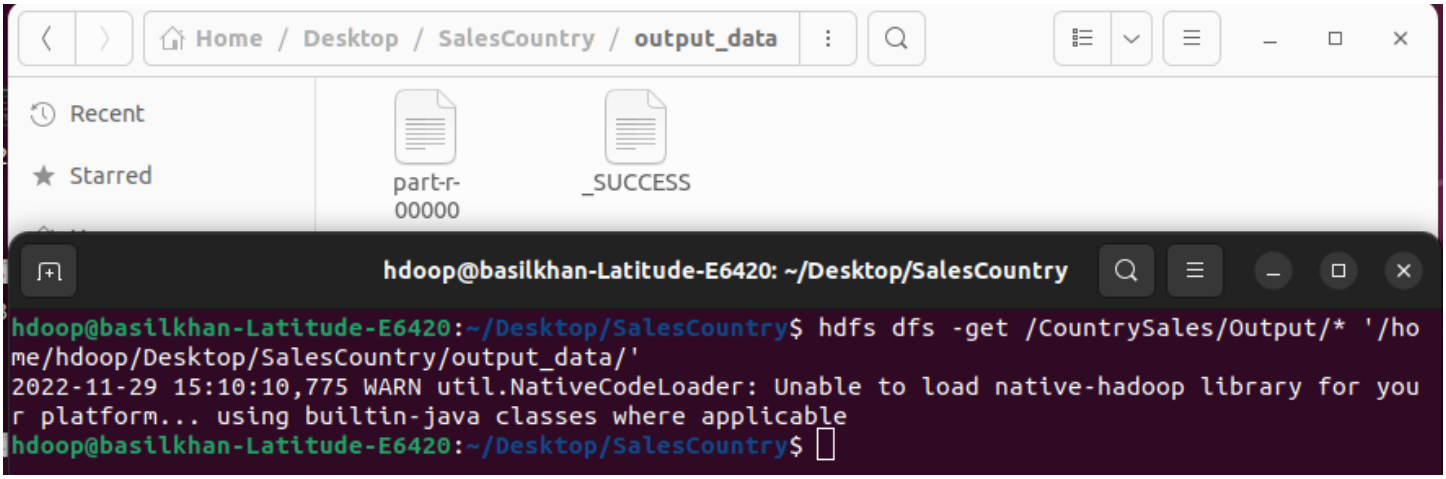
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hdoop	supergroup	0 B	Nov 29 15:01	1	128 MB	_SUCCESS
-rw-r--r--	hdoop	supergroup	948 B	Nov 29 15:01	1	128 MB	part-r-00000

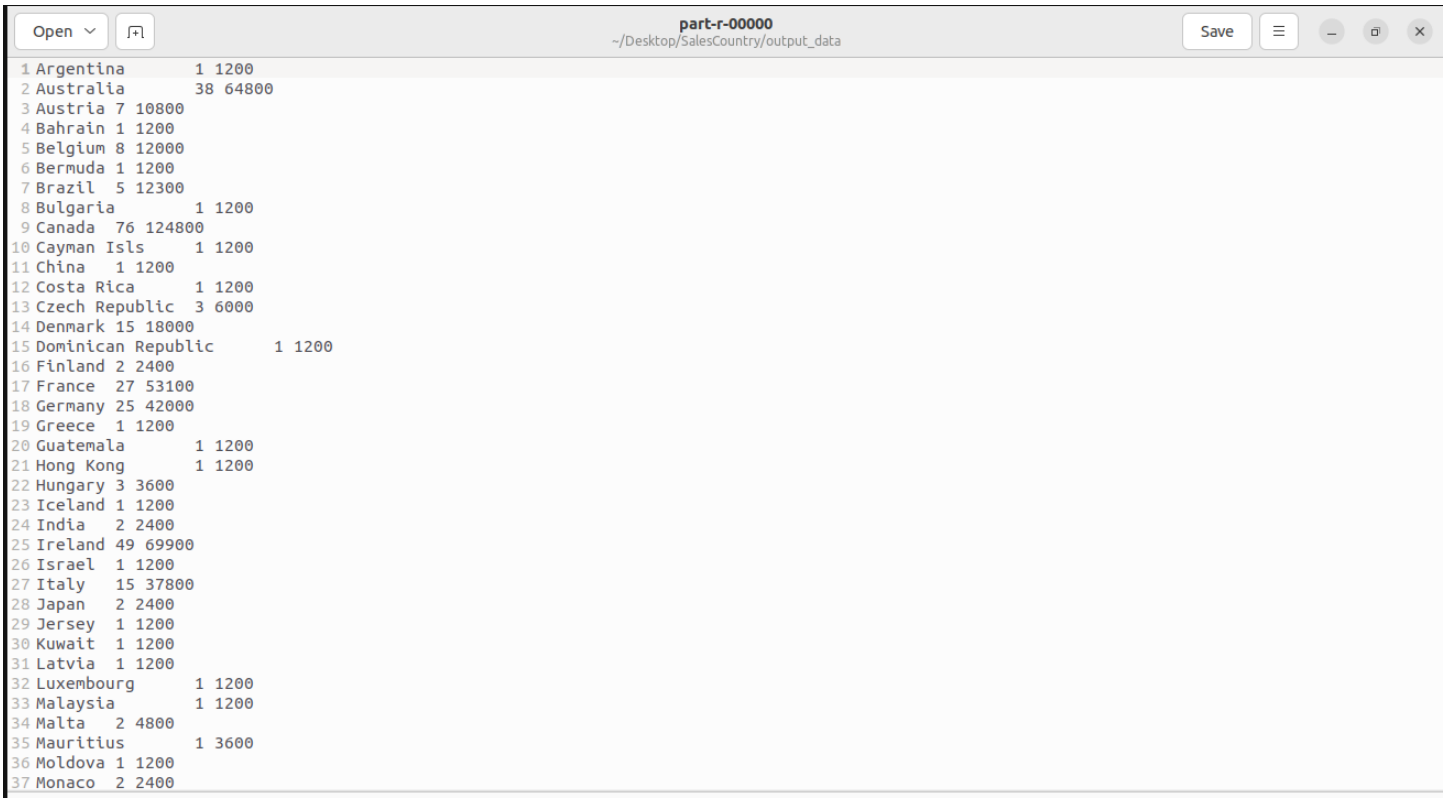
Showing 1 to 2 of 2 entries Previous 1 Next

❖ *Output*

```
hdoop@basilkhan-Latitude-E6420: ~/Desktop/SalesCountry
hdoop@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*





Libraries and Packages:

```
import java.io.IOException;
port 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/