Analysis on Sales Data Using Hadoop and MapReduce Algorithm

Ibraheem Sayeed Mohamed[1], Abdul Malik Sulaiman Said Al Jabri[1], Puttaswamy Malali Rajegowda[1#] and Jitendra Pandey[1#]

[1]Middle East College, Muscat, Oman
[#]Advisor

ABSTRACT

For this project, we developed an extremely flexible and efficient MapReduce method to analyze a big sales dataset. The program's objectives were to find interesting correlations plus tendencies within the sales statistics and provide a strong substitute for in-depth analysis of the data. We created Map and Reduce algorithms that processed the information in a networked & simultaneous manner using the MapReduce framework's distributed processing capabilities. We extended the application to efficiently process enormous datasets by deploying it on the Hadoop platform. The results of the investigation demonstrate the MapReduce framework's effectiveness at managing enormous volumes of information through a dispersed, simultaneous way. By employing the networked computing features of the framework, we were able to swiftly and efficiently analyze the sales data and get meaningful insights about it. The program's great scalability allowed us to quickly and effectively examine large amounts of data, making it a powerful tool enabling big information research.

Objectives

• Be familiar with the notion map reduce algorithm.
• Examine the supplied statistics.
• To examine the dataset, deploy Hadoop.
• Be familiar with the Linux operating system.
• Use programming to match the big data models to present time data.
• Review the results of the study.

Introduction

The term “big data” was not known 10 – 15 years ago yet, with the rapid growth of data generation sources that are many related to Internet of things or social media, the capacity and also the proccing capability of the traditional could not take-in the humongous rate of data and in that manner some new approaches did come up to handle this type of data which we name “big data” and as Oracle defines big data as “The definition of big data is data that contains greater variety, arriving in increasing volumes and with more velocity. This is also known as the three Vs.” (Oracle 2022)

In depth, this assignment we are given a large csv dataset that named as “sales data”, in it we can find sub-sets that represent months of sales records for the year 2019. If we take a look to one of these sub-sets we will find a record of order id, product, Quantity, price per item and the date of order and finally for more detail of meta-data for the date set it contains 72 columns, 52 strings, 4 integers, 4 date and time as column’s ID’s.

Moving on, we below shall see a diagram that that will illustrate how the map reduced algorithm do work and also discuss the various tools we shall utilize to implement the analysis on the large data set. Using Linux OS, in it we will put in use the Hadoop framework that supports big data analysis in which with the java programing language shall implement the map reduce algorithm and after that when the analysis phase is done, we shall move to the next phase that is analysis. In which we shall establish a reading upon the result produced by the implementation phase by comparing and understanding the results. Then we move to the testing phase that will involve the validation necessary to make sure that our process was successfully done or not. Finally, we shall give out a conclusion that will sum up all the process and its analysis but not only that but we also shall see some recommendation given based upon the analysis taken. As we aim for a clear and analysis.

"The pro y at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability." (GeeksforGeeks 2022)

Design and Implementation

Design of Map Algorithm

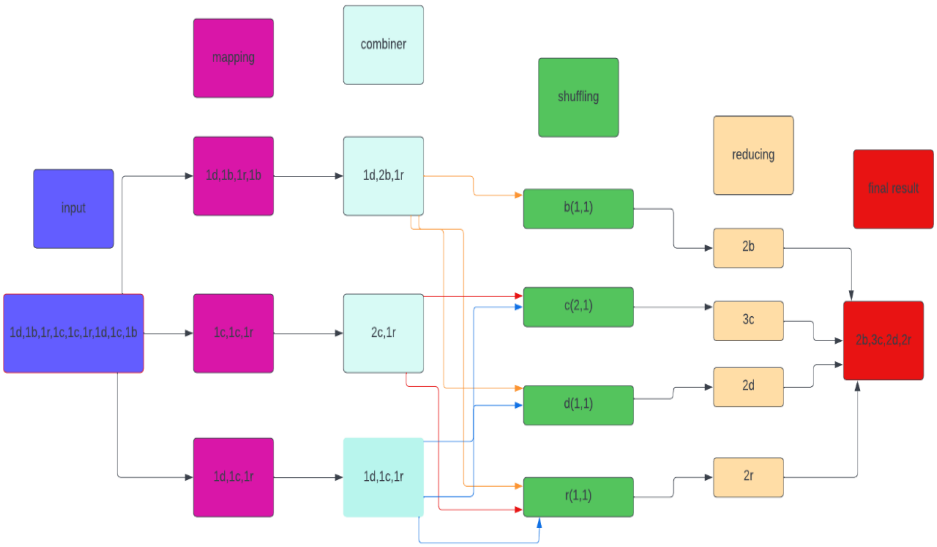


Figure 1. Designing the Map Algorithm

Via the utilization of lucid chart diagram creation tool, we did establish a diagram that represent the map algorithm concept that will be implemented below. as we will see some brief explanation for each part of this algorithm.

Input first the user will input and to give data to the system that we can see in the diagram is in a random manner with no order or categorization, this input data will be fed to the algorithm to further on be categorized and shuffled as we will see in the next points.

Mapping in the second phase which is mapping that is taking in the data and separating them into fragments which will make it easier to process them rather than processing a whole data at once to avoid any crashes or miss validations.

Combiner moving to the third phase, from the name we can see data with the similar form will be combined and added together as seen in the diagram above various variables which has the same ending are merged together.

Shuffling in the fourth phase we can see the data is shuffled and categorized according to their variable as seen in the diagram we have b ,c ,d and r that are visible in the shuffling process.

Reducing here we can see after the shuffling process the data is it used to be smaller in size in which in the diagram it's illustrated that 2b, 3c, 2d, and 2r are reduced from the shuffling process.

Final result at the end after the process is done the data processed and analyzed will be given to the end user.

Process of methodology diagram

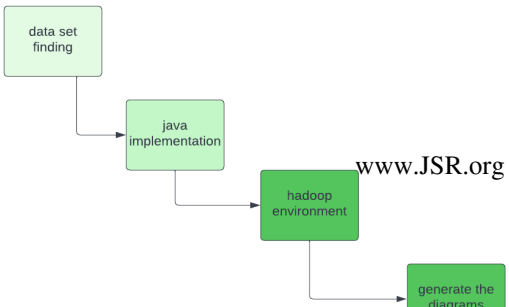


Figure 2. Process Methodology Diagram

As it's seen in the diagram, we will follow the path of the sequence displayed above. First, is to collect the data thus we did search on secondary data sources, and we did find a sales data set that would be perfect for our implementation. moving to the next phase, establishing mapper, reducer and combiner classes via Java that will help us in creating a jar file which then will be used in the Hadoop environment. as Hadoop phase who will generate

Charts that give us the illustration of the data analyzed from the second resources. which will take us to the next phase, that we have to analyze the charts produced by comparing different variables within the data set. finally in the last phase we shall give a feedback and impression through the documentation for the data set analyzed by creating pinpoint on what was wrong or what was right during the process of this data set creation as you will see further down below in this group project.

Implementation

Employing the right computing tools, an individual may combine whatever large amount of knowledge which is being given to our sake as real-time content with something to offer.

Our team uses Apache Hadoop for handling such vast amount of data.

Before anything else, let's define Apache Hadoop:

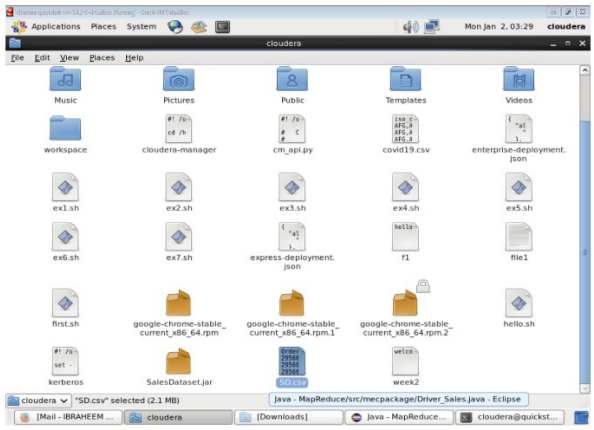
Cloudera offers a digital edition of Apache Hadoop, a software for dealing with enormous volumes of data. Hadoop, an open source framework, is said to gather and process enormous amounts of data on clusters of low-cost technologies. This provides sharable collecting as well as calculation of extremely large content volumes using simple computing techniques, making it easy to develop interpret data at a large scale.

Cloudera's Hadoop package provides the Hadoop basic frameworks along with a variety of extra solutions as well as techniques for organizing data, interpreting, and research. Among these instruments & techniques are solutions for information intake, storing data, handling information, management, as well as information privacy (O'Rielly Media, n.d.).

The big data that we have is a dataset based on Sales and we have chosen the dataset for the month December 2019.

Step 1: Turn on Cloudera Hadoop Virtual Machine

Step 2: Once the software is turned on, go to the browser and download the dataset and save it in the directory /home/cloudera/'filename'. The dataset is named as 'SD.csv'.



› directory, our next step is to open Eclipse and create a new project and package.

We created a project called ‘MapReduce’ and created a new package ‘mecpackage’.

Inside this package we create three classes named Mapper_Sales, Reducer_Sales and Driver_Sales.

Note that Driver_Sales is our main class.

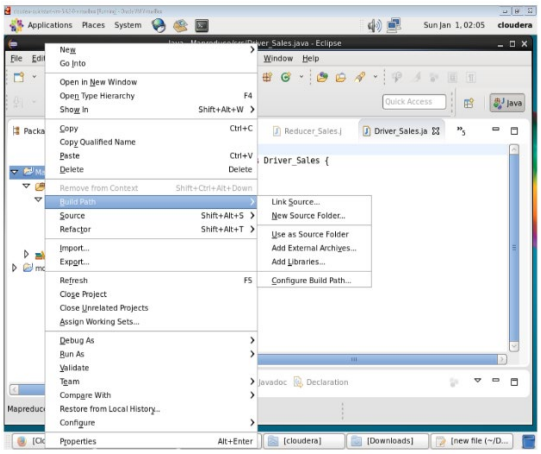
Once we are done doing all this we follow up with the next step.

Step 4: In order to properly program the code on Eclipse, we are required to add two external JAR files.

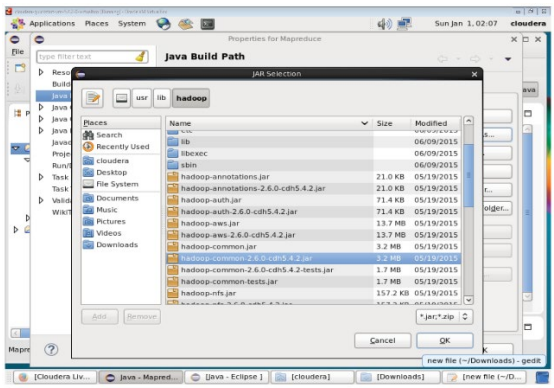
A JAR (Java Archive) is a package file format typically used to aggregate many Java class files and associated metadata and resources (text, images, etc.) into one file to distribute application software or libraries on the Java platform. JAR files are similar to ZIP files, but they have a manifest file that describes the contents of the JAR file and how to use it(Oracle, n.d.).

Here’s how we add JAR files.

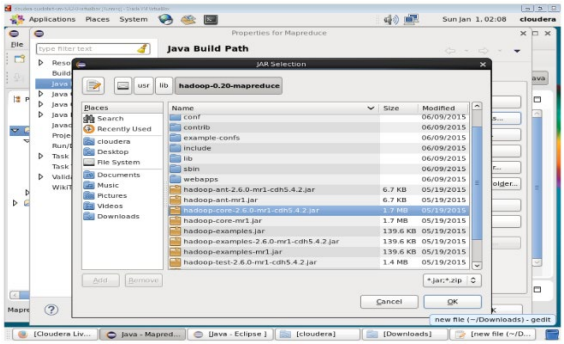
Step 5: Click on file > Build Path> Configure Build Path.



Step 6: Click on lib> Hadoop and select the file called ‘hadoop-common-2.6.0-cdh5.4.2.jar’



Step 7: Now, we go to lib> Hadoop-0.20-mapreduce and select the file named ‘hadoop-core-2.6.0-mr1-cdh5.4.2.jar’



Once the external JAR files are added, we move on to the next step and that is to code down the three classes we created.

three classes as mentioned. The three classes are Mapper_Sales, Reducer_Sales and Driver_Sales.

Firstly let's get an idea about what is a Mapper class:

Within the framework of the Hadoop MapReduce computing approach, a mapper class is an element that conducts the mapping process in a MapReduce job. The mapper class in a MapReduce job accepts an input data then turns its contents into an array of subsequent key-value combinations. The reducer class then collects and sums up the information based on these intermediary key-value relationships (Mapper Class, n.d.).

The code for the Mapper class is pasted below:

```
package mecpackage;
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.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class Mapper_Sales extends MapReduceBase implements Mapper<LongWritable,
Text, Text, IntWritable>
 {
          // Map function
   public void map(LongWritable key, Text value, OutputCollector<Text,
            IntWritable> output, Reporter rep) throws IOException
   {
      String line = value.toString();
      // Space lines being separated
      for (String expression : line.split(" "))
      {
         if (expression.length() > 0)
         {
            output.collect(new Text(expression), new IntWritable(1));
         }
      }
   }
}
```

Next, we code down the Reducer class,

Inside the context of the Hadoop MapReduce software framework, a reducer class is an element which conducts the reducing process in a MapReduce job. The reducer class in a MapReduce job examines a collection of preliminary key-value combinations created by the mapper class and generates an assortment of final key-value combinations(Reducer Class, n.d.).

The code for the Reducer class is as follows:

```
package mecpackage;
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapr0065d.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class Reducer_Sales extends MapReduceBase implements Reducer<Text,
IntWritable, Text, IntWritable> {

// Reduce function
public void reduce(Text key, Iterator<IntWritable> value,
OutputCollector<Text, IntWritable> output,
Reporter rep) throws IOException
{

   int count = 0;

   // frequency of each expression is being counted
   while (value.hasNext())
   {
      IntWritable i = value.next();
      count += i.get();
   }

   output.collect(key, new IntWritable(count));
}
}
```

Lastly, we code down the Driver class,

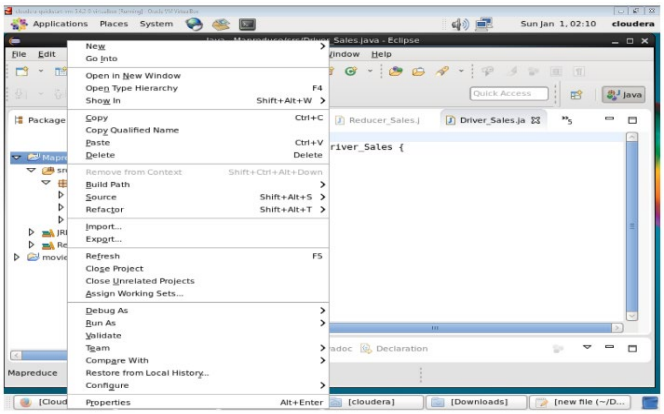
Using the Apache Hadoop MapReduce computing framework, a driver class is accountable in configuring and performing a MapReduce task. The driver class provides the job's intake & result destinations, as well as the mapper and reducer classes that will be utilized(Driver Class, n.d.).

The code is as follows:

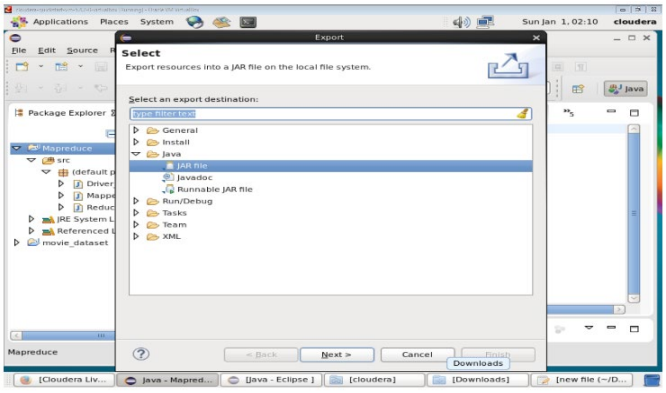
```
kage;
)Exception;
he.hadoop.conf.Configured;
he.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class Driver_Sales extends Configured implements Tool {

   public int run(String args[]) throws IOException
   {
      if (args.length < 2)
      {
         System.out.println("Please give valid inputs");
         return -1;
      }
      JobConf conf = new JobConf(Driver_Sales.class);
      FileInputFormat.setInputPaths(conf, new Path(args[0]));
      FileOutputFormat.setOutputPath(conf, new Path(args[1]));
      conf.setMapperClass(Mapper_Sales.class);
      conf.setReducerClass(Reducer_Sales.class);
      conf.setMapOutputKeyClass(Text.class);
      conf.setMapOutputValueClass(IntWritable.class);
      conf.setOutputKeyClass(Text.class);
      conf.setOutputValueClass(IntWritable.class);
      JobClient.runJob(conf);
      return 0;
   }
// Method to call the main class
   public static void main(String args[]) throws Exception
   {
      int exitCode = ToolRunner.run(new Driver_Sales(), args);
      System.out.println(exitCode);
   }
}
```

Once we code down all the classes, we move on to the next step:

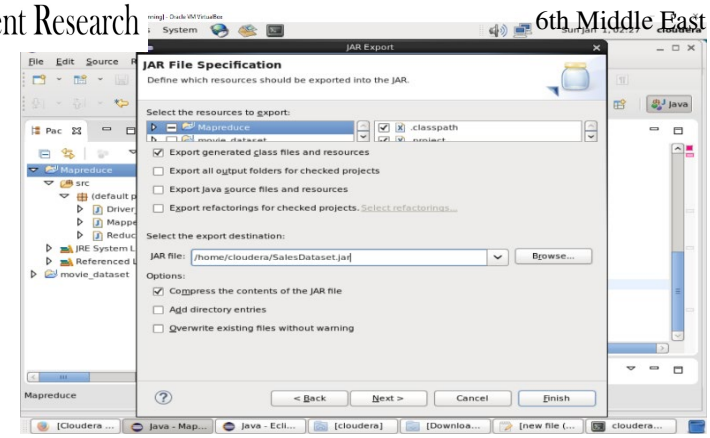
Step 9: As seen in step 8, we coded down all the three classes. Once this is done, we create a JAR file. In order to create a JAR file, we go to file> and click on Export.



Once clicked on Export, click on 'JAR file' and click Next.

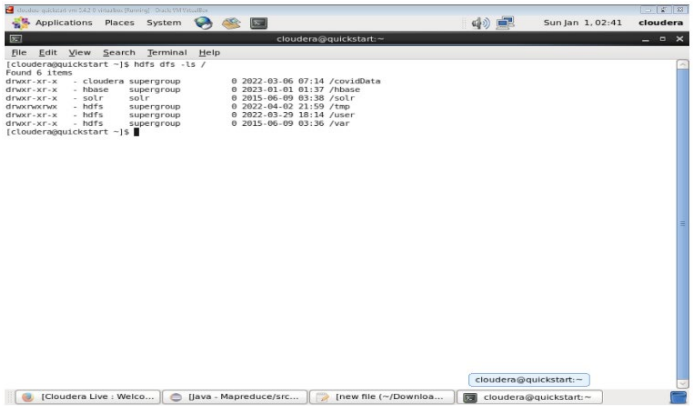


Once clicked on Next, it will open a JAR file specification page where you will need to choose your project where your classes exist. Once selected, make sure your export destination is in /home/clouder/'nameofJARfile'.jar and click Finish.

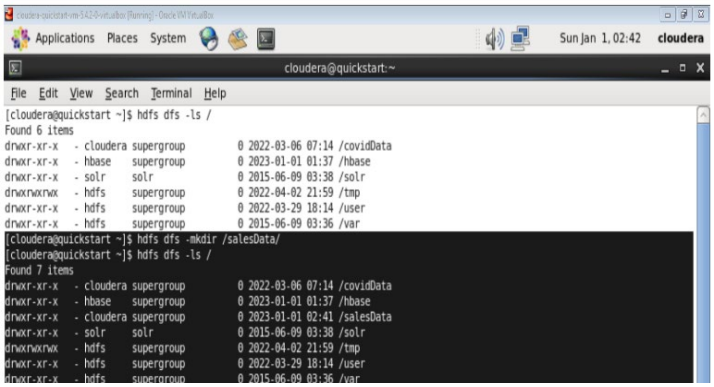


Now, we go to the cloudera quickstart where we enter the commands to exeute the program:

Step 10: Open cloudera@quickstart and write the command 'hdfs dfs -ls /' to check the files and directories.



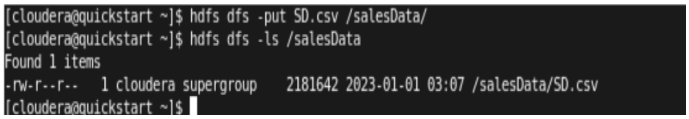
Step 11: Enter the command 'hdfs dfs -mkdir /salesData/' to create a new files called salesData. To check if the file exists repeat the command from step 10.



Step 12: Once it is confirmed that the file exists, we enter the following command to bring in the dataset we downloaded and saved it in cloudera home.

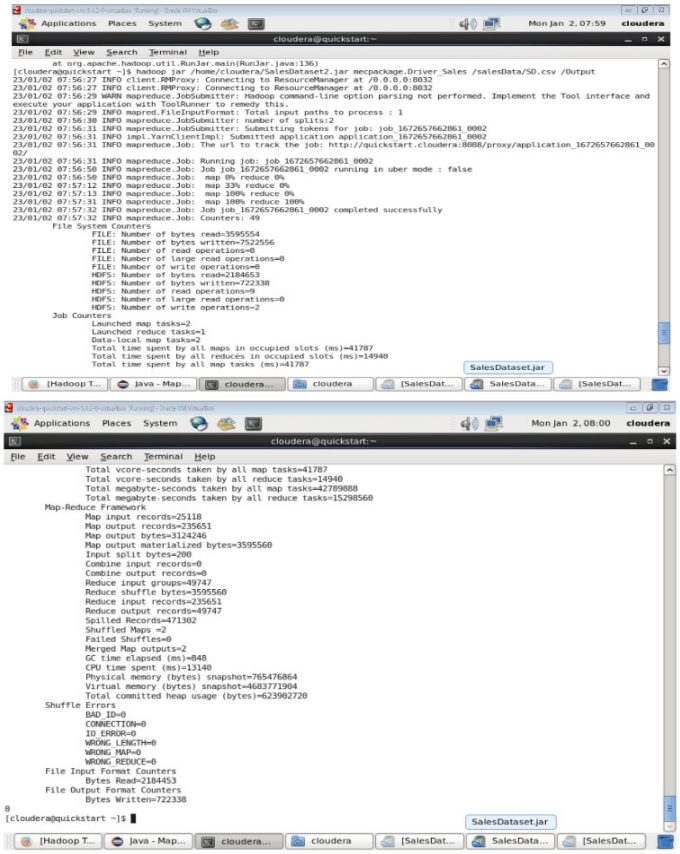
The command is 'hdfs dfs -put SD.csv /salesData/', this puts the dataset into the file we created.

To check if the dataset exists in the file we created we enter the command 'hdfs dfs -ls /salesData'.



Step 13: Now, we enter the command 'hadoop jar /home/cloudera/SalesDataset.jar mecpackage.Driver_Sales /salesData/SD.csv /Output' to run a Hadoop MapReduce job from the command line. This operation accepts numerous contentions, namely the location

where to Reduce script we wish to execute, the primary component containing the MapReduce driver, along with the MapReduce job's intake as well as end result routes.

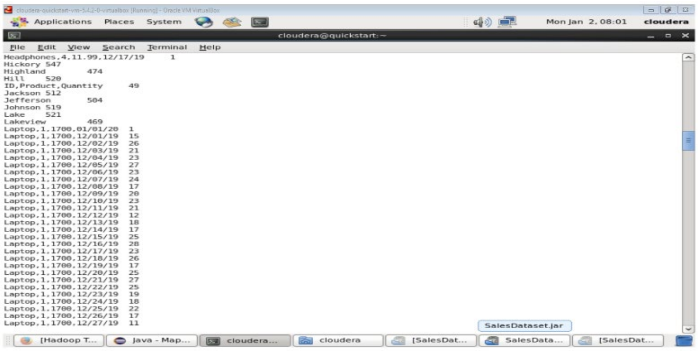


Step 14: Here, we enter the command ‘hadoop fs -cat /Output/part-00000’.

The data inside of a directory within the file system of Hadoop may be seen with this particular query. Instead of needing to locate the file to your personal computer first, the -cat argument enables users to read what's inside of the file directly from the terminal or command prompt..

```
[cloudera@quickstart ~]$ hadoop fs -cat /Output/part-00000
```

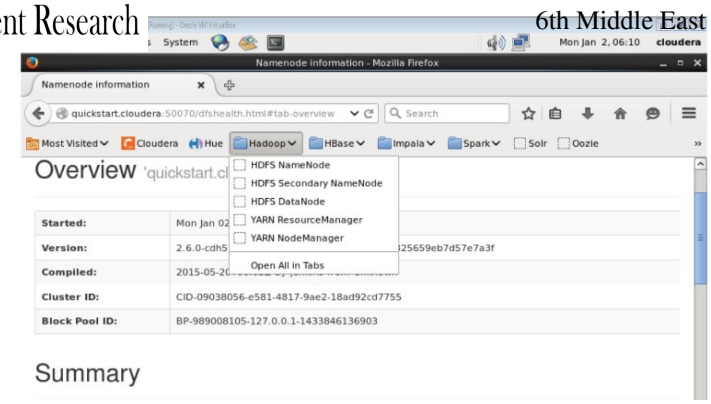
The output for the command is as follows:



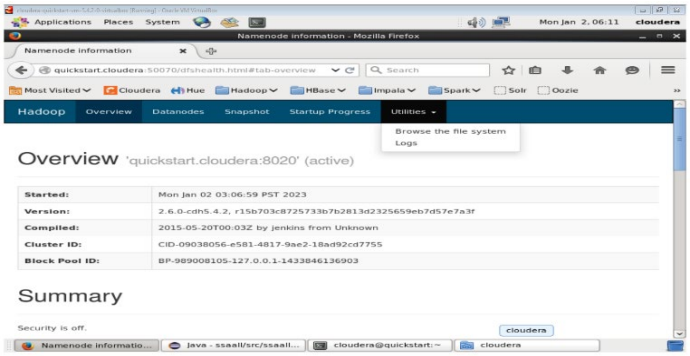
Now, we open the cloudera browser and follow the given steps:

Step 15: Now that the browser is open, locate the hadoop option below the searchbar and click on it and it will provide to you a list of HDFS and YARN options.

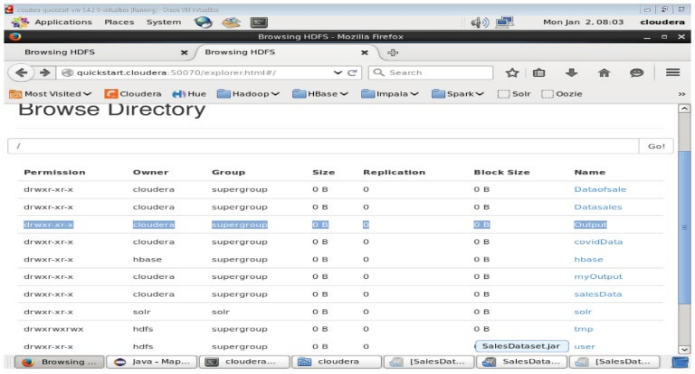
Select HDFS NameNode



Step 16: Once the HDFS NameNode tab opens, click on the Utilities options and select 'Browse the file system'.

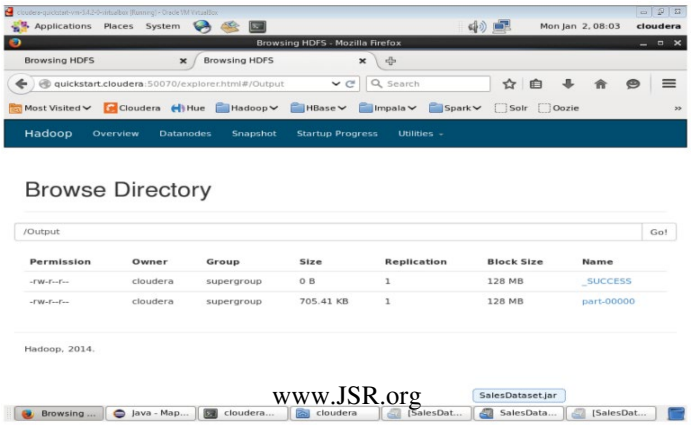


Once Clicked on it, it will redirect you to a page something like this:



This page browses all directories in the file system. Going back to step 13, we saved the data in a directory called 'Output'. As seen in the picture above the file directory highlighted is the one we created.

Click on it and you will be redirected to a page like the one pasted below:



As one could observe, we are working with a file with the name "part-0000," which is an element file containing the end results of a MapReduce operation. The output of a MapReduce task is often split up into several component files, each of which contains a piece of the whole result. The dimensions of every component file and the total amount of part files depend on the MapReduce job's parameters as well as the volume of the incoming data.

The initial part file, 'part-00000', normally includes the initial part of the output. The output directory designated for the MapReduce operation houses the 'part-00000' file as well as the other part filesUsing 'part-00000' root alongside using remaining component directories in the Hadoop file system may be seen by using the 'hadoop fs -cat' command.With this, the programming comes to an end.

Next, we move on towards the analysis part.

Analysis

Our dataset is based upon a sales data for the year 2019. The dataset we have chosen is for the month December 2019.
The dataset includes the following data titles:

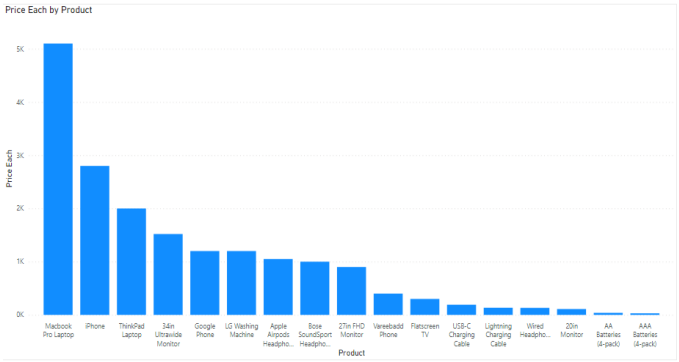
- Order Date
- Order ID
- Price Each
- Product
- Purchase address
- Quantity Ordered

In order to analyze the data we have created visual representations. This will give us a much more clear understanding of our data.

Note: to create the visual representation we have just selected the first 100 values in the dataset since the dataset is very large which has over 20,000 values in it.

A stackable bar chart is the initial visual representation we have produced. Under particular scenario, a bar chart on cost per commodity would display the items on the x-axis and the averaged cost for every commodity on the y-axis. Every bar would show the typical cost of a certain item.

Analyzing the costs of several goods with this bar chart will help you spot any price anomalies. By way of an instance, if you see a bar that is noticeably longer compared to the others, it can be a hint that the accompanying item is substantially more expensive than the other items. This might be caused by a number of things, including the price of rival items, the cost of raw materials, or the amount of need for the commodity.

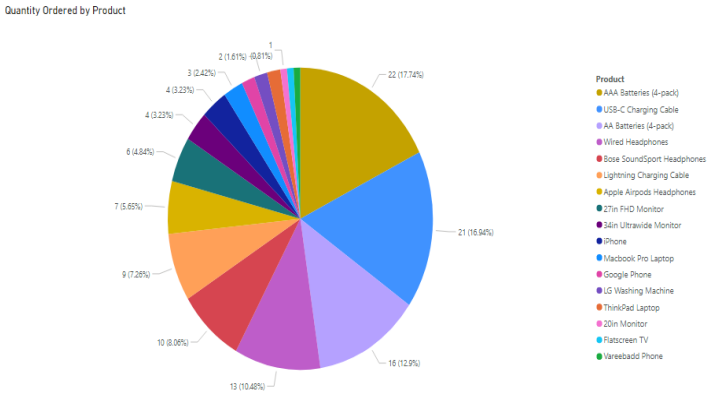


The second visual represenatation we have created is a Pie chart. In this case, a pie chart for quantity ordered for every product would show the quantities of each product as a percentage of the total quantity ordered.

One must initially determine the total number of goods that were instructed in order to build the pie chart. The percentage of the overall amount which every commodity represents would then be determined. Last but not least, you would make a pie chart alongside every goods depicted like a portion in a pie, having the dimensions of the slice proportionate to the percentage of the overall amount which the item indicates.

Determining the corresponding amounts of the various items getting requested might be helped by looking at this pie chart. In particular, while a single good makes up a sizable portion of the overall pie, it may mean that it is a favored item which is being purchased

in huge a m has a tiny share of the market, it can mean the item is not as well-liked or that fewer orders are being placed for it.



These visual representations of data are a powerful tool for understanding and analyzing complex datasets. The sales dataset analyzed in this project is no exception - the various charts and graphs provide valuable insights into the data and allow us to identify trends, patterns, and opportunities for improvement.

Whether it is a bar graph comparing the price of every product, or a pie chart breaking down quantity ordered by product, visual representations of data offer a quick and easy way to gain insights into the data. By leveraging these tools, we can more effectively understand the performance of the business and make informed decisions that drive growth and success.

Conclusion

In a nutshell the MapReduce software created for this research offered a thorough and effective means of evaluating a sizable sales dataset. We were able to find similarities as well as tendencies within the collected information and get important perspectives into the information about sales through the implementation of Map and Reduce operations and making use of the networked computing characteristics using the MapReduce framework. The application illustrated the value of Hadoop as an architecture for executing MapReduce operations as well as the efficiency of the MapReduce technique for handling massive amounts of data. The initiative was successful overall, and the study' findings will help company decision-makers make wise choices and spot areas where they can improve.

Future Enhancements

Incorporating machine learning methods into this MapReduce software might be one future improvement that makes it possible to analyze sales statistics in a deeper and more comprehensive manner. To find correlations as well as anomalies within the information, for instance, that would not be instantly obvious employing conventional quantitative research techniques, researchers can utilize machine learning techniques. In order to get more precise forecasts for future sales, this might entail training a forecasting strategy using data from sales, including a regression or categorization model.

Instantaneous data processing skills might be incorporated within the application as additional possible improvement. At present, the software runs in batch mode, which means it works with an established dataset as well as generates a single output. One might make it possible the software to continually analyze updated information as it becomes accessible through implementing immediate computing skills, allowing us to get more rapid conclusions about the information related to sales. The application might be integrated with a streaming data system, like Apache Kafka, to provide real-time data import and processing in order to achieve this. In general, by giving more sophisticated and accurate analysis of the sales data, such improvements could empower the MapReduce engine to add greater value. One can grasp the sales data better and generate more effective company choices by utilizing artificial intelligence approaches as well as real-time data analysis capacities.

References

What Is Big Data?. (2022). Retrieved 11 December 2022, from https://www.oracle.com/big-data/what-is-big-data/

Oracle. (n.d.). What is a JAR file? Retrieved January 3, 2023, from https://www.oracle.com/java/technologies/what-is-a-jar-file.html

Mapper class. (n.d.). In Hadoop MapReduce Tutorial. Retrieved January 3, 2023, from https://hadoopmapreducetutorial.com/mapper-class/

Reducer class. (n.d.). In Hadoop MapReduce Tutorial. Retrieved January 3, 2023, from https://hadoopmapreducetutorial.com/reducer-class/

Driver class. (n.d.). In Hadoop MapReduce Tutorial. Retrieved January 3, 2023, from https://hadoopmapreducetutorial.com/driver-class/

Software Testing | Basics - GeeksforGeeks. (2017). Retrieved 11 December 2022, from https://www.geeksforgeeks.org/software-testing-basics/

O'Reilly Media. (n.d.). What is Apache Hadoop? Retrieved January 3, 2023, from https://www.oreilly.com/library/view/hadoop-the-definitive/9781449327891/ch01.html