EXP.NO:03 **DATE:**

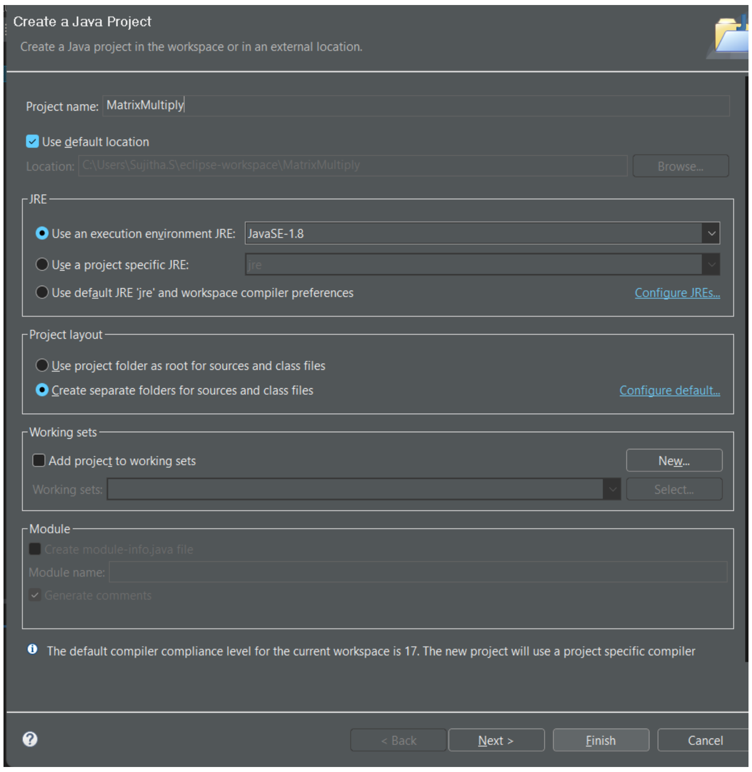
IMPLEMENTATION OF MATRIX MULTIPLICATION WITH HADOOP MAP REDUCE

AIM: To implement of Matrix Multiplication with Hadoop Map Reduce.

STEPS:

STEP 1: Run Eclipse for Java Developers

STEP 2: Create a new Java Project with name "MatrixMultiply "


Create a Java Project
Create a Java project in the workspace or in an external location.

Project name: MatrixMultiply

[✓] Use default location
Location: C:\Users\Sujitha.S\eclipse-workspace\MatrixMultiply

JRE
[●] Use an execution environment JRE: JavaSE-1.8
○ Use a project specific JRE: jre
○ Use default JRE 'jre' and workspace compiler preferences

Project layout
○ Use project folder as root for sources and class files
[●] Create separate folders for sources and class files

Working sets
☐ Add project to working sets
Working sets:

Module
☐ Create module-info.java file
Module name:
[✓] Generate comments

The default compiler compliance level for the current workspace is 17. The new project will use a project specific compiler

STEP 3: Set the Java Environment Version to your current version of Java (JRE : 1.8)

STEP 4: Add a Package with name "com.MapReduce.java" and Create three Classes in it.

STEP 5: Create a New Class With name Map.java.

STEP 6: Now write the below program in the "Map.java" Class

PROGRAM:

```
package com.MapReduce.wc;

import org.apache.hadoop.conf.*;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

//import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class Map extends org.apache.hadoop.mapreduce.Mapper<LongWritable, Text, Text, Text>

{

        @Override

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

throws IOException, InterruptedException {

                Configuration conf = context.getConfiguration();

                int m = Integer.parseInt(conf.get("m"));

                int p = Integer.parseInt(conf.get("p"));

String line = value.toString();

// (M, i, j, Mij);

String[] indicesAndValue = line.split(",");

Text outputKey = new Text();

Text outputValue = new Text();

if (indicesAndValue[0].equals("M")) {

for (int k = 0; k < p; k++) {

outputKey.set(indicesAndValue[1] + "," + k);
// outputKey.set(i,k);

outputValue.set(indicesAndValue[0] + "," + indicesAndValue[2]

+ "," + indicesAndValue[3]);

// outputValue.set(M,j,Mij);

context.write(outputKey, outputValue);

}

} else {

        // (N, j, k, Njk);

        for (int i = 0; i < m; i++) {

        outputKey.set(i + "," + indicesAndValue[2]); outputValue.set("N," + indicesAndValue[1] + ","

        + indicesAndValue[3]); context.write(outputKey, outputValue);

        }

        }

        }

}
```

STEP 7: Now Create another class with name "Reduce.java" and paste the below program in it.

PROGRAM:

```
package com.MapReduce.wc;

import org.apache.hadoop.io.Text;

// import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

import java.util.HashMap;

public class Reduce
```

```
extends org.apache.hadoop.mapreduce.Reducer<Text, Text, Text, Text> { @Override

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

String[] value;

//key=(i,k),

//Values = [(M/N,j,V/W),..]

HashMap<Integer, Float> hashA = new HashMap<Integer, Float>(); HashMap<Integer, Float> hashB =
new HashMap<Integer, Float>(); for (Text val : values) {

value = val.toString().split(",");

if (value[0].equals("M")) {

hashA.put(Integer.parseInt(value[1]), Float.parseFloat(value[2])); } else {

hashB.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));

}

}

int n = Integer.parseInt(context.getConfiguration().get("n"));

float result = 0.0f;

float m_ij;

float n_jk;

for (int j = 0; j < n; j++) {

m_ij = hashA.containsKey(j) ? hashA.get(j) : 0.0f; n_jk = hashB.containsKey(j) ? hashB.get(j) : 0.0f;
result += m_ij * n_jk;

}

if (result != 0.0f) {

context.write(null,
new Text(key.toString() + "," + Float.toString(result)));

}

}

}
```

STEP 8: Now, Create another class with name “MatrixMultiply.java” and paste the below program in it.

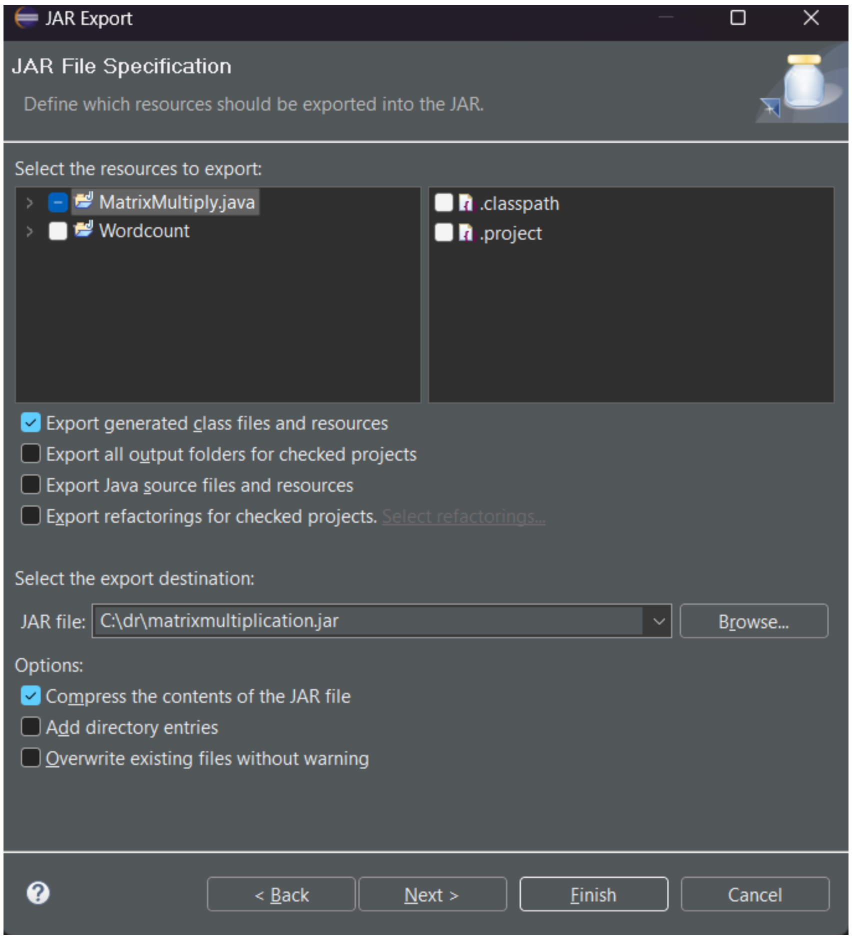
PROGRAM:

```
package com.MapReduce.wc;

import org.apache.hadoop.conf.*;

import org.apache.hadoop.fs.Path;

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;

public class MatrixMultiply {

public static void main(String[] args) throws Exception { if (args.length != 2) {

System.err.println("Usage: MatrixMultiply <in_dir> <out_dir>");

System.exit(2);

}

Configuration conf = new Configuration();

conf.set("m", "1000");

conf.set("n", "100");
conf.set("p", "1000");

@SuppressWarnings("deprecation")

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

job.setJarByClass(MatrixMultiply.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.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]));

job.waitForCompletion(true);

}

}
```

STEP 9: To resolve the errors in the programs we should add two External jar files to it.

- Hadoop_common :2.7.3.jar
- Hadoop_mapreduce:client:core:2.7.1.jar

STEP 10: Now export the project into a Jar file and name it as “MatrixMultiply.jar”


JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- [–] MatrixMultiply.java
- [ ] Wordcount
- [ ] .classpath
- [ ] .project

- [✓] Export generated class files and resources
- [ ] Export all output folders for checked projects
- [ ] Export Java source files and resources
- [ ] Export refactorings for checked projects. Select refactorings...

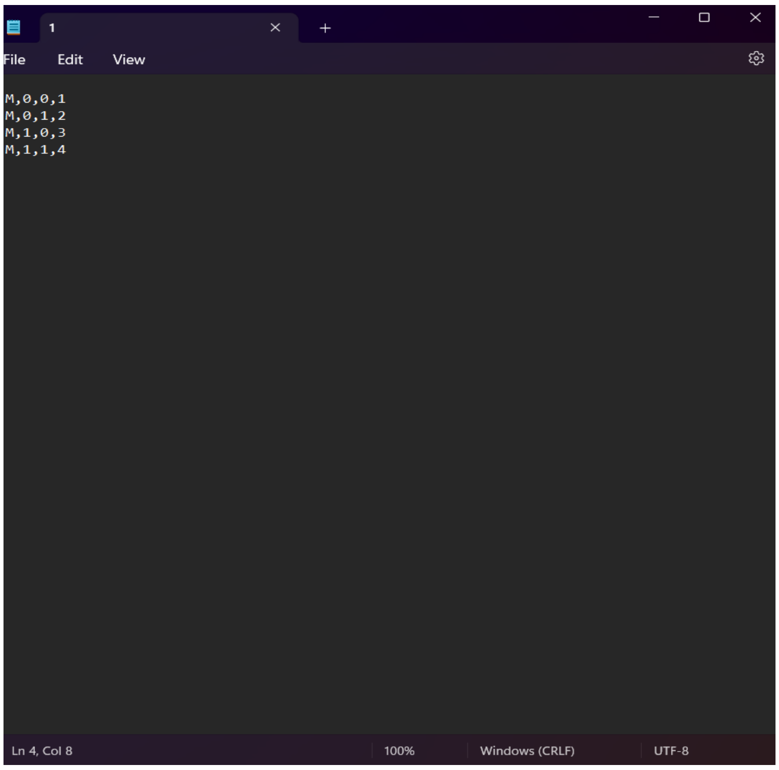
Select the export destination:

JAR file: C:\dr\matrixmultiplication.jar

Options:

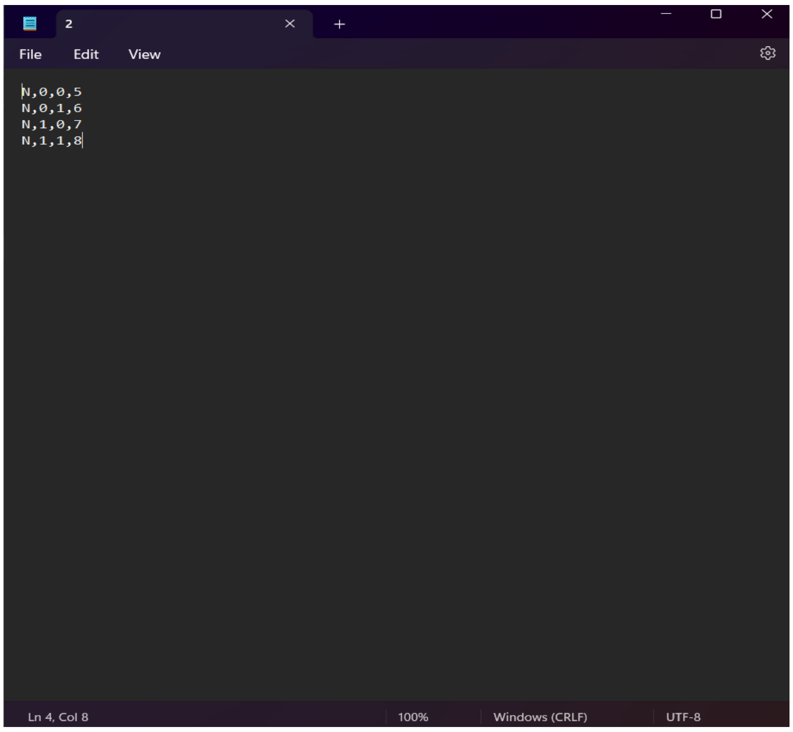
- [✓] Compress the contents of the JAR file
- [ ] Add directory entries
- [ ] Overwrite existing files without warning

STEP 11: Now create a Text file in Notepad and name it as “1.txt” and “2.txt. write some content inside the text file and save it.



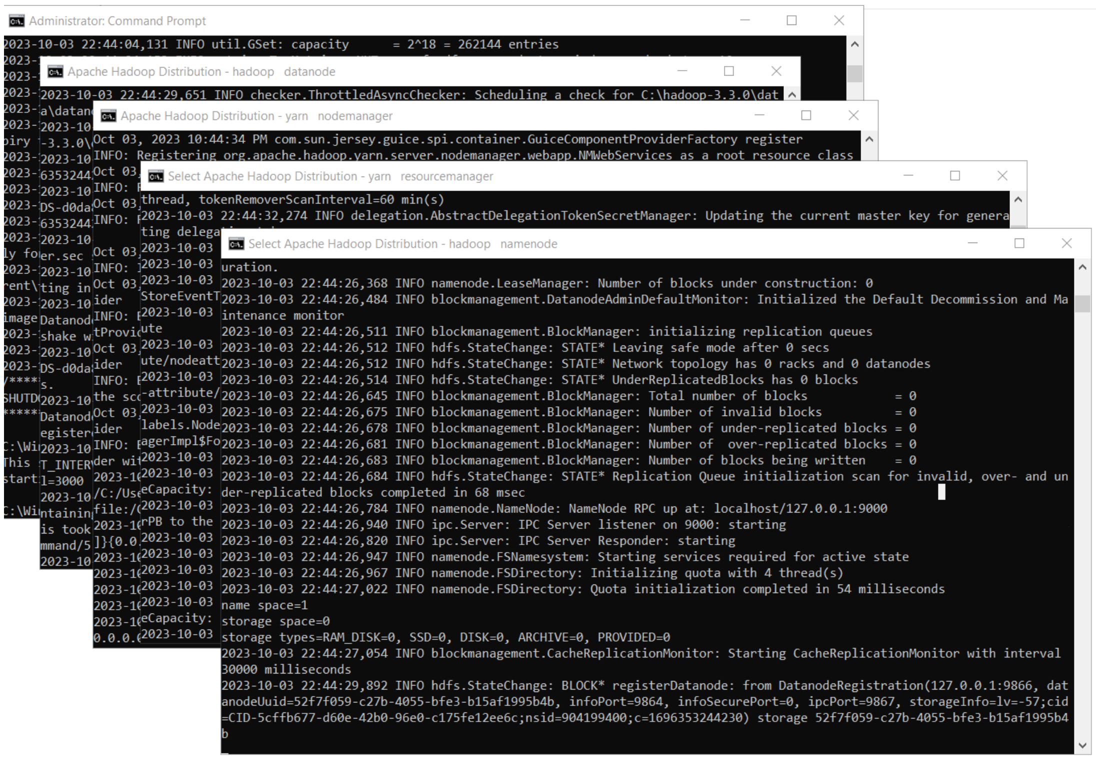
```
M,0,0,1
M,0,1,2
M,1,0,3
M,1,1,4
```

Ln 4, Col 8   100%   Windows (CRLF)   UTF-8



```
N,0,0,5
N,0,1,6
N,1,0,7
N,1,1,8
```

STEP 12: Now run all the deamons in Hadoop.



```
2023-10-03 22:44:04,131 INFO util.GSet: capacity      = 2^18 = 262144 entries
```

```
2023-10-03 22:44:29,651 INFO checker.ThrottledAsyncChecker: Scheduling a check for C:\hadoop-3.3.0\dat
```

```
Oct 03, 2023 10:44:34 PM com.sun.jersey.guice.spi.container.GuiceComponentProviderFactory register
INFO: Registering org.apache.hadoop.yarn.server.nodemanager.webapp.NMWebServices as a root resource class
```

```
thread, tokenRemoverScanInterval=60 min(s)
2023-10-03 22:44:32,274 INFO delegation.AbstractDelegationTokenSecretManager: Updating the current master key for genera
```

```
uration.
2023-10-03 22:44:26,368 INFO namenode.LeaseManager: Number of blocks under construction: 0
2023-10-03 22:44:26,484 INFO blockmanagement.DatanodeAdminDefaultMonitor: Initialized the Default Decommission and Ma
intenance monitor
2023-10-03 22:44:26,511 INFO blockmanagement.BlockManager: initializing replication queues
2023-10-03 22:44:26,512 INFO hdfs.StateChange: STATE* Leaving safe mode after 0 secs
2023-10-03 22:44:26,512 INFO hdfs.StateChange: STATE* Network topology has 0 racks and 0 datanodes
2023-10-03 22:44:26,514 INFO hdfs.StateChange: STATE* UnderReplicatedBlocks has 0 blocks
2023-10-03 22:44:26,645 INFO blockmanagement.BlockManager: Total number of blocks            = 0
2023-10-03 22:44:26,675 INFO blockmanagement.BlockManager: Number of invalid blocks          = 0
2023-10-03 22:44:26,678 INFO blockmanagement.BlockManager: Number of under-replicated blocks = 0
2023-10-03 22:44:26,681 INFO blockmanagement.BlockManager: Number of  over-replicated blocks = 0
2023-10-03 22:44:26,683 INFO blockmanagement.BlockManager: Number of blocks being written    = 0
2023-10-03 22:44:26,684 INFO hdfs.StateChange: STATE* Replication Queue initialization scan for invalid, over- and un
der-replicated blocks completed in 68 msec
2023-10-03 22:44:26,784 INFO namenode.NameNode: NameNode RPC up at: localhost/127.0.0.1:9000
2023-10-03 22:44:26,940 INFO ipc.Server: IPC Server listener on 9000: starting
2023-10-03 22:44:26,820 INFO ipc.Server: IPC Server Responder: starting
2023-10-03 22:44:26,947 INFO namenode.FSNamesystem: Starting services required for active state
2023-10-03 22:44:26,967 INFO namenode.FSDirectory: Initializing quota with 4 thread(s)
2023-10-03 22:44:27,022 INFO namenode.FSDirectory: Quota initialization completed in 54 milliseconds
name space=1
storage space=0
storage types=RAM_DISK=0, SSD=0, DISK=0, ARCHIVE=0, PROVIDED=0
2023-10-03 22:44:27,054 INFO blockmanagement.CacheReplicationMonitor: Starting CacheReplicationMonitor with interval
30000 milliseconds
2023-10-03 22:44:29,892 INFO hdfs.StateChange: BLOCK* registerDatanode: from DatanodeRegistration(127.0.0.1:9866, dat
anodeUuid=52f7f059-c27b-4055-bfe3-b15af1995b4b, infoPort=9864, infoSecurePort=0, ipcPort=9867, storageInfo=lv=-57;cid
=CID-5cffb677-d60e-42b0-96e0-c175fe12ee6c;nsid=904199400;c=1696353244230) storage 52f7f059-c27b-4055-bfe3-b15af1995b4
b
```

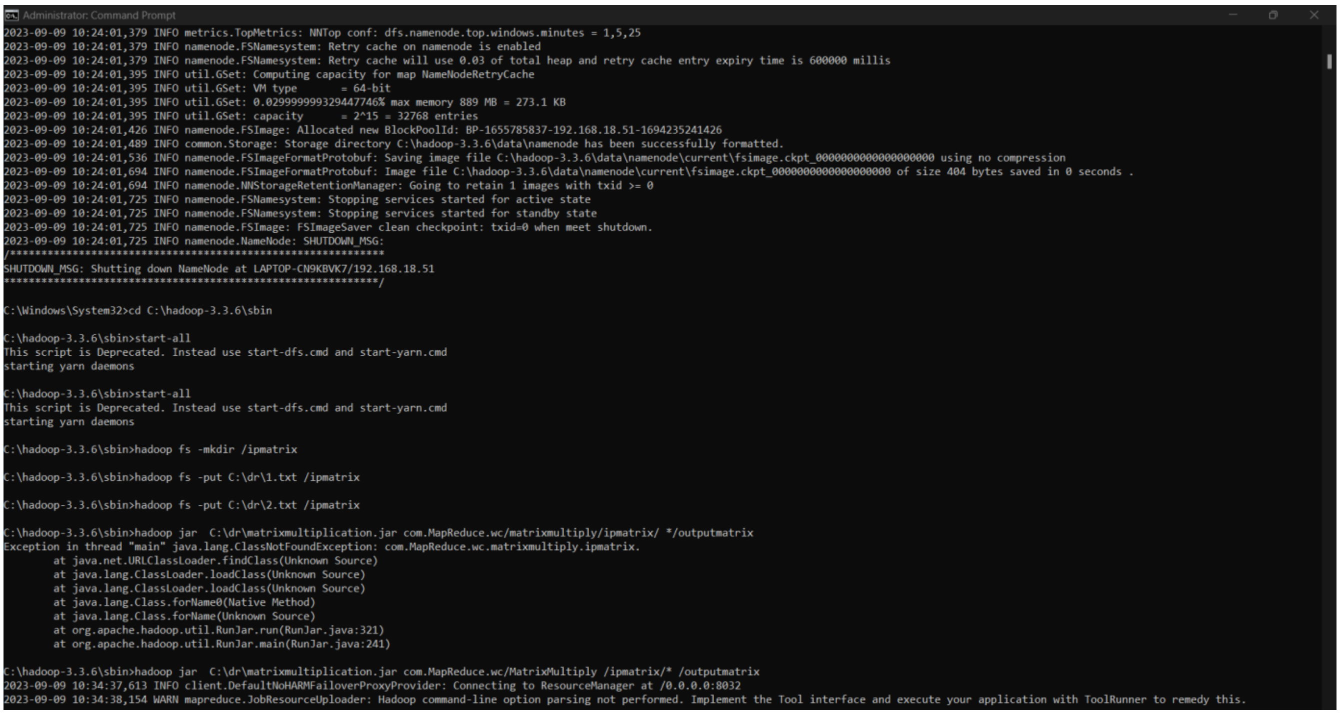
STEP 13: Create a new input directory named as "ipmatrix".

By using the command: hadoop fs -mkdir /ipmatrix

STEP 14: Now put the “1.txt” and 2.txt file to the ipmatrix directory.

By using these commands: hadoop fs -put C:\dr\1.txt   /ipmatrix

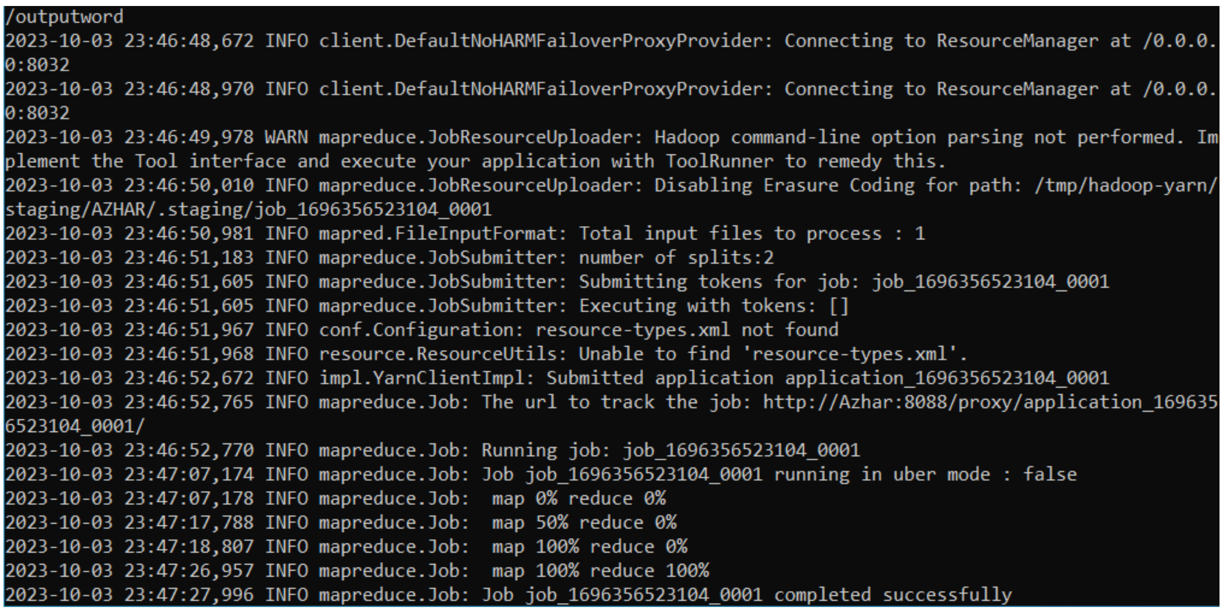
hadoop fs -put C:\dr\2.txt   /ipmatrix



```
Administrator: Command Prompt
2023-09-09 10:24:01,379 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.windows.minutes = 1,5,25
2023-09-09 10:24:01,379 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
2023-09-09 10:24:01,379 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry time is 600000 millis
2023-09-09 10:24:01,395 INFO util.GSet: Computing capacity for map NameNodeRetryCache
2023-09-09 10:24:01,395 INFO util.GSet: VM type       = 64-bit
2023-09-09 10:24:01,395 INFO util.GSet: 0.029999999329447746% max memory 889 MB = 273.1 KB
2023-09-09 10:24:01,395 INFO util.GSet: capacity      = 2^15 = 32768 entries
2023-09-09 10:24:01,426 INFO namenode.FSImage: Allocated new BlockPoolId: BP-1655785837-192.168.18.51-1694235241426
2023-09-09 10:24:01,489 INFO common.Storage: Storage directory C:\hadoop-3.3.6\data\namenode has been successfully formatted.
2023-09-09 10:24:01,536 INFO namenode.FSImageFormatProtobuf: Saving image file C:\hadoop-3.3.6\data\namenode\current\fsimage.ckpt_0000000000000000000 using no compression
2023-09-09 10:24:01,694 INFO namenode.FSImageFormatProtobuf: Image file C:\hadoop-3.3.6\data\namenode\current\fsimage.ckpt_0000000000000000000 of size 404 bytes saved in 0 seconds .
2023-09-09 10:24:01,694 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2023-09-09 10:24:01,725 INFO namenode.FSNamesystem: Stopping services started for active state
2023-09-09 10:24:01,725 INFO namenode.FSNamesystem: Stopping services started for standby state
2023-09-09 10:24:01,725 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2023-09-09 10:24:01,725 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at LAPTOP-CN9KBVK7/192.168.18.51
************************************************************/

C:\Windows\System32>cd C:\hadoop-3.3.6\sbin

C:\hadoop-3.3.6\sbin>start-all
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\hadoop-3.3.6\sbin>start-all
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\hadoop-3.3.6\sbin>hadoop fs -mkdir /ipmatrix

C:\hadoop-3.3.6\sbin>hadoop fs -put C:\dr\1.txt /ipmatrix

C:\hadoop-3.3.6\sbin>hadoop fs -put C:\dr\2.txt /ipmatrix

C:\hadoop-3.3.6\sbin>hadoop jar  C:\dr\matrixmultiplication.jar com.MapReduce.wc/matrixmultiply/ipmatrix/ */outputmatrix
Exception in thread "main" java.lang.ClassNotFoundException: com.MapReduce.wc.matrixmultiply.ipmatrix.
        at java.net.URLClassLoader.findClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        at java.lang.Class.forName0(Native Method)
        at java.lang.Class.forName(Unknown Source)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:321)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:241)

C:\hadoop-3.3.6\sbin>hadoop jar  C:\dr\matrixmultiplication.jar com.MapReduce.wc/MatrixMultiply /ipmatrix/* /outputmatrix
2023-09-09 10:34:37,613 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2023-09-09 10:34:38,154 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
```

STEP 15: Run the Jar file created from the project

Using the command: hadoop jar   C:\dr\matrixmultiplication.jar com.MapReduce.wc/MatrixMultiply /ipmatrix/ */outputmatrix

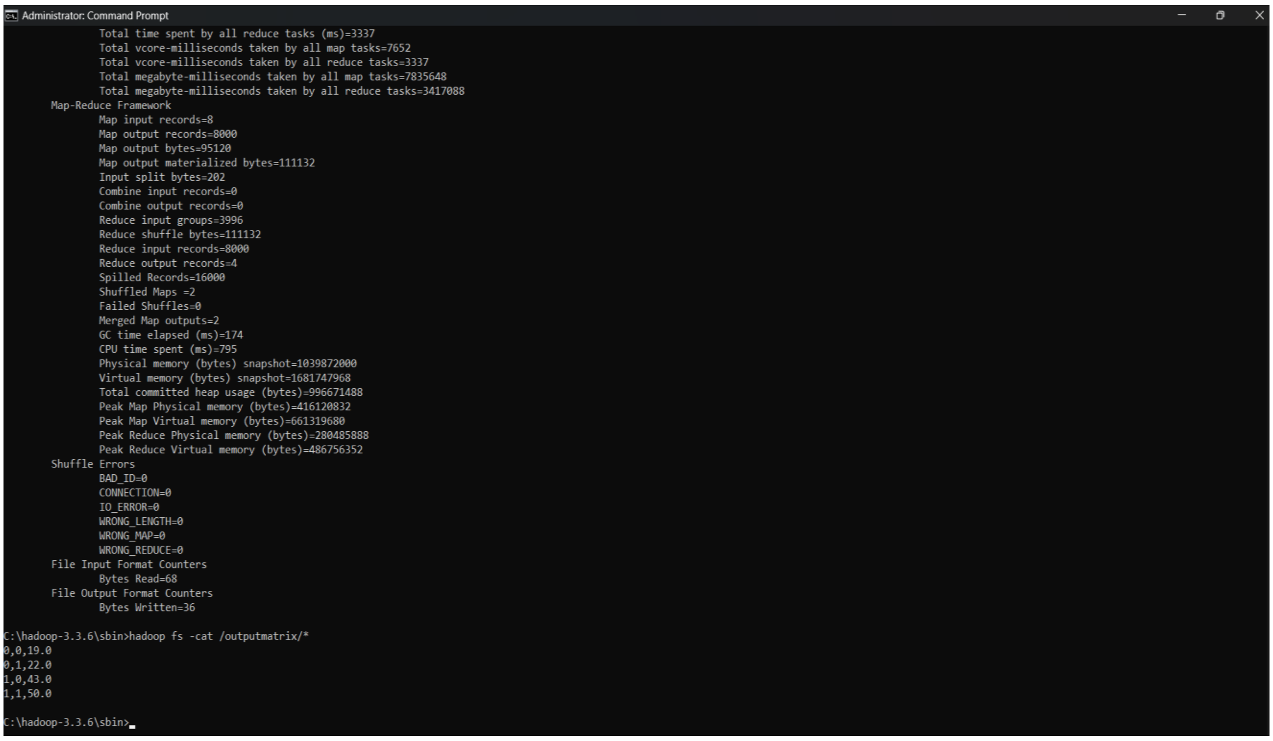


```
/outputword
2023-10-03 23:46:48,672 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.
0:8032
2023-10-03 23:46:48,970 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.
0:8032
2023-10-03 23:46:49,978 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Im
plement the Tool interface and execute your application with ToolRunner to remedy this.
2023-10-03 23:46:50,010 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/
staging/AZHAR/.staging/job_1696356523104_0001
2023-10-03 23:46:50,981 INFO mapred.FileInputFormat: Total input files to process : 1
2023-10-03 23:46:51,183 INFO mapreduce.JobSubmitter: number of splits:2
2023-10-03 23:46:51,605 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1696356523104_0001
2023-10-03 23:46:51,605 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-10-03 23:46:51,967 INFO conf.Configuration: resource-types.xml not found
2023-10-03 23:46:51,968 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2023-10-03 23:46:52,672 INFO impl.YarnClientImpl: Submitted application application_1696356523104_0001
2023-10-03 23:46:52,765 INFO mapreduce.Job: The url to track the job: http://Azhar:8088/proxy/application_169635
6523104_0001/
2023-10-03 23:46:52,770 INFO mapreduce.Job: Running job: job_1696356523104_0001
2023-10-03 23:47:07,174 INFO mapreduce.Job: Job job_1696356523104_0001 running in uber mode : false
2023-10-03 23:47:07,178 INFO mapreduce.Job:  map 0% reduce 0%
2023-10-03 23:47:17,788 INFO mapreduce.Job:  map 50% reduce 0%
2023-10-03 23:47:18,807 INFO mapreduce.Job:  map 100% reduce 0%
2023-10-03 23:47:26,957 INFO mapreduce.Job:  map 100% reduce 100%
2023-10-03 23:47:27,996 INFO mapreduce.Job: Job job_1696356523104_0001 completed successfully
```

STEP 16: At last Print your output for the MatrixMultiply text file.

Using the Command : hadoop fs -cat /outputmatrix/*

OUTPUT :



Administrator: Command Prompt

```
                Total time spent by all reduce tasks (ms)=3337
                Total vcore-milliseconds taken by all map tasks=7652
                Total vcore-milliseconds taken by all reduce tasks=3337
                Total megabyte-milliseconds taken by all map tasks=7835648
                Total megabyte-milliseconds taken by all reduce tasks=3417088
        Map-Reduce Framework
                Map input records=8
                Map output records=8000
                Map output bytes=95120
                Map output materialized bytes=111132
                Input split bytes=202
                Combine input records=0
                Combine output records=0
                Reduce input groups=3996
                Reduce shuffle bytes=111132
                Reduce input records=8000
                Reduce output records=4
                Spilled Records=16000
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=174
                CPU time spent (ms)=795
                Physical memory (bytes) snapshot=1039872000
                Virtual memory (bytes) snapshot=1681747968
                Total committed heap usage (bytes)=996671488
                Peak Map Physical memory (bytes)=416120832
                Peak Map Virtual memory (bytes)=661319680
                Peak Reduce Physical memory (bytes)=280485888
                Peak Reduce Virtual memory (bytes)=486756352
        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=68
        File Output Format Counters
                Bytes Written=36

C:\hadoop-3.3.6\sbin>hadoop fs -cat /outputmatrix/*
0,0,19.0
0,1,22.0
1,0,43.0
1,1,50.0

C:\hadoop-3.3.6\sbin>
```

RESULT : Thus the program to run a basic wordcount mapreduce program to understand mapreduce is excecuted and output is verified successfully.