# BIG DATA ANALYTICS LAB

## EXPERIMENT_NO-05

**AIM:** Implement Matrix Multiplication with Hadoop Map Reduce.

**Description:**

1) Open Oracle VM VirtualBox->export cloudera->start

2) In eclipse->File->New->Java project->Project name "MatrixMultiplication"

   ->Finish

3) Create three classes.

   Right click on MatrixMultiplication-> New->class->Name "MatrixDriver"

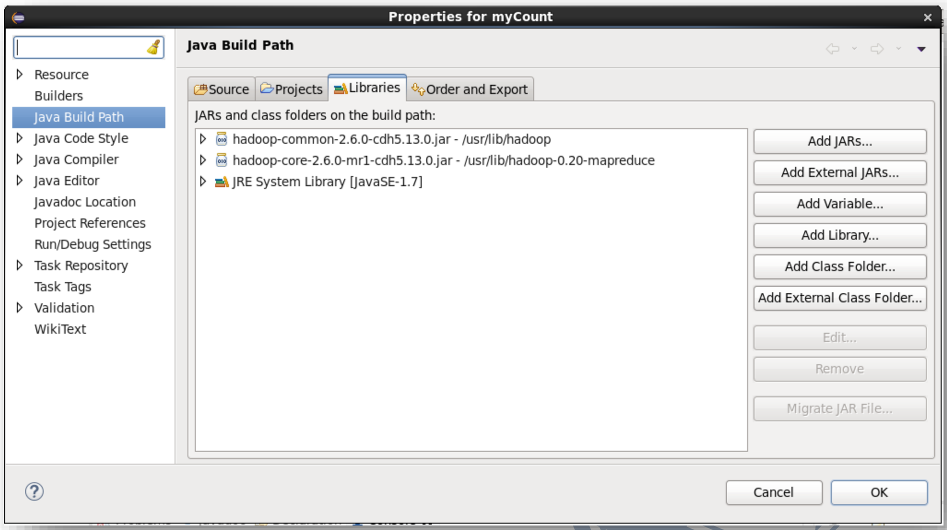
   Right click on MatrixMultiplication-> New->class->Name "MatrixMapper"

   Right click on MatrixMultiplication -> New->class->Name "MatrixReducer"

4) Add Hadoop libraries.

Right click on MatrixMultiplication->Build path->Configure Build path->Add external JARS. (usr\lib\hadoop\hadoop-common-2.6.0-cdh 5.13.0 jar,

usr\lib\hadoop\hadoop-core-2.6.0-cdh 5.13.0 jar)


Properties for myCount

Java Build Path

JARs and class folders on the build path:

hadoop-common-2.6.0-cdh5.13.0.jar - /usr/lib/hadoop
hadoop-core-2.6.0-mr1-cdh5.13.0.jar - /usr/lib/hadoop-0.20-mapreduce
JRE System Library [JavaSE-1.7]

## PROGRAM:

### MatrixDriver.java

```
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;

public class MatrixDriver

{

    public static void main(String[] args) throws Exception

    {
        Configuration conf = new Configuration();

        // M is an m-by-n matrix; N is an n-by-p matrix.

        conf.set("m", "2");

        conf.set("n", "2");

        conf.set("p", "2");

        Job job = Job.getInstance(conf, "MatrixMultiplication");

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

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

        job.setJarByClass(MatrixDriver.class);

        job.setOutputKeyClass(Text.class);

        job.setOutputValueClass(Text.class);

        job.setMapperClass(MatrixMapper.class);

        job.setReducerClass(MatrixReducer.class);

        job.setInputFormatClass(TextInputFormat.class);

        job.setOutputFormatClass(TextOutputFormat.class);

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


}

}
```

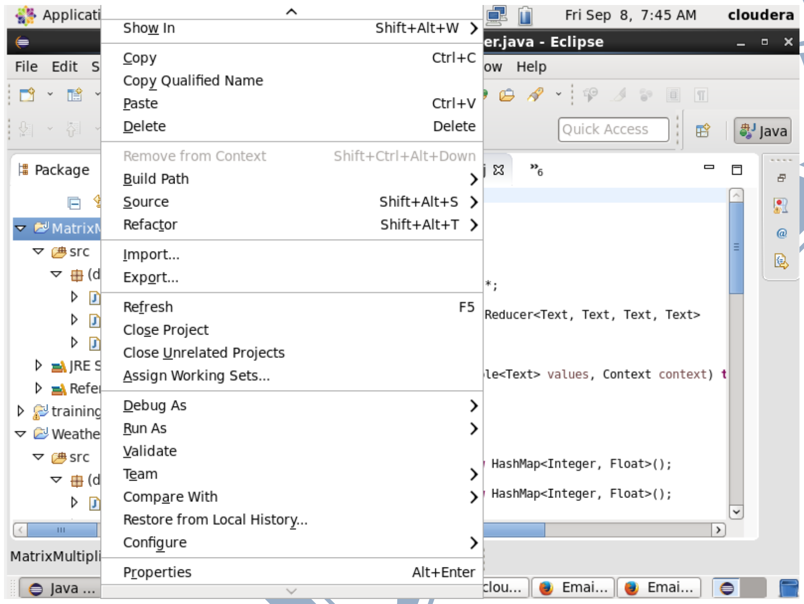
## MatrixMapper.java

```
import java.io.IOException;

import org.apache.hadoop.conf.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.*;

public class MatrixMapper extends Mapper <LongWritable, Text, Text, Text>

{

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();

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);

outputValue.set("M," + indicesAndValue[2] + "," + indicesAndValue[3]);
context.write(outputKey, outputValue);

}

}

else

{

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

{

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

outputValue.set("N," + indicesAndValue[1] + "," + indicesAndValue[3]);

context.write(outputKey, outputValue);

}

}

}

}
```

## MatrixReducer.java

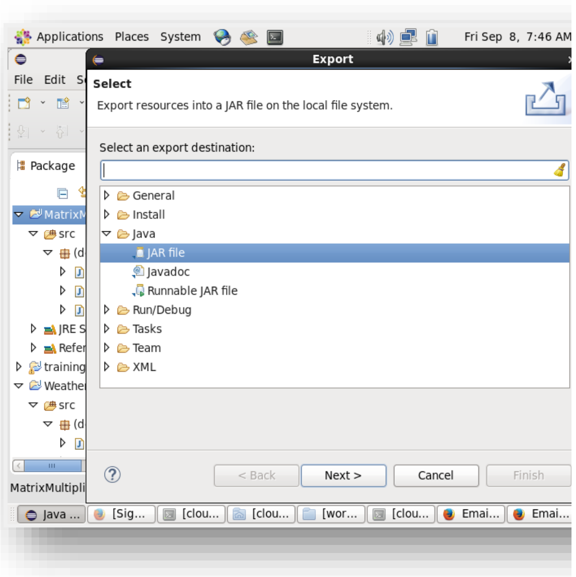
```
import java.io.IOException;

import java.util.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.*;

public class MatrixReducer extends Reducer<Text, Text, Text, Text>

{

public void reduce(Text key, Iterable<Text> values, Context
context) throws IOException, InterruptedException
```

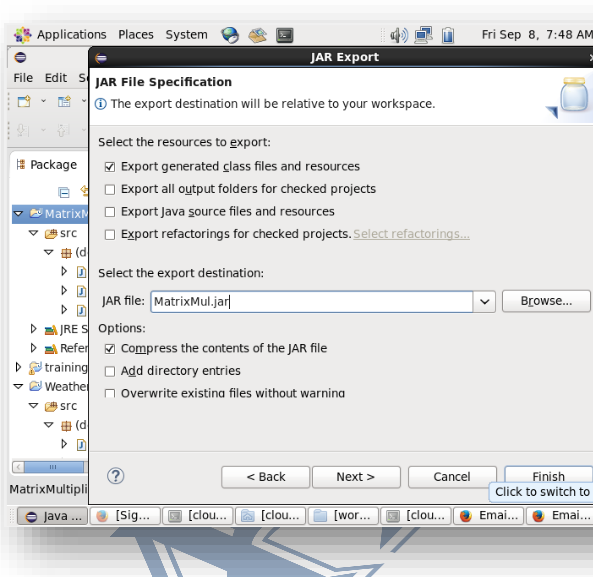
```
{
String[] value;

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 a_ij; float b_jk;

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

{

a_ij = hashA.containsKey(j) ? hashA.get(j) : 0.0f;

b_jk = hashB.containsKey(j) ? hashB.get(j) : 0.0f;

result += a_ij * b_jk;
}
if (result != 0.0f)
{
context.write(null, new Text(key.toString() + "," + Float.toString(result)));
}
}
}
```



```
*;
Reducer<Text, Text, Text, Text>
le<Text> values, Context context) t
HashMap<Integer, Float>();
HashMap<Integer, Float>();
```

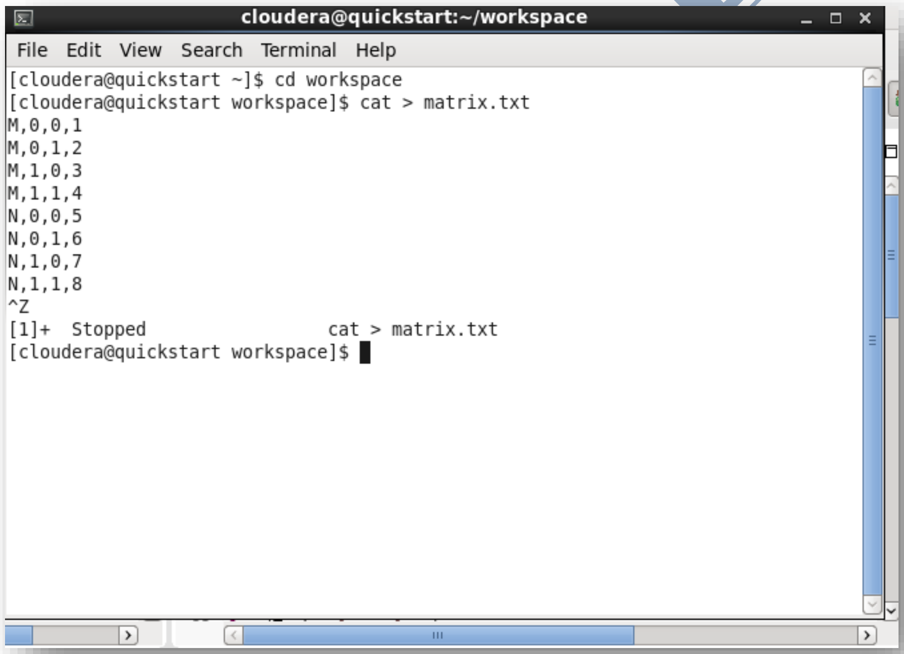
Right click on MatrixMultiplication->Export->java->jar file->JAR file:”**MatrixMul**”->Finish



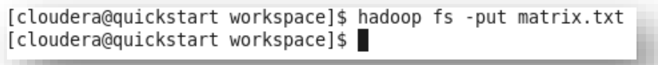

JAR Export
JAR File Specification
The export destination will be relative to your workspace.
Select the resources to export:
Export generated class files and resources
Export all output folders for checked projects
Export Java source files and resources
Export refactorings for checked projects. Select refactorings...
Select the export destination:
JAR file: MatrixMul.jar
Browse...
Options:
Compress the contents of the JAR file
Add directory entries
Overwrite existing files without warning
< Back
Next >
Cancel
Finish

## OUTPUT:

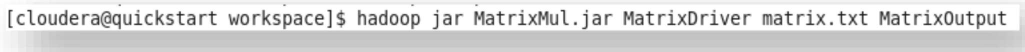
In Terminal



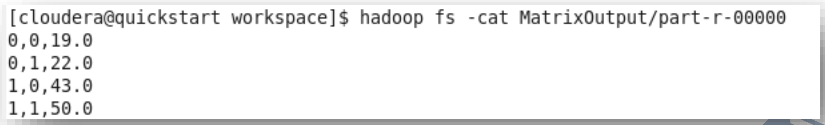
```
[cloudera@quickstart ~]$ cd workspace
[cloudera@quickstart workspace]$ cat > matrix.txt
M,0,0,1
M,0,1,2
M,1,0,3
M,1,1,4
N,0,0,5
N,0,1,6
N,1,0,7
N,1,1,8
^Z
[1]+  Stopped                 cat > matrix.txt
[cloudera@quickstart workspace]$
```



```
[cloudera@quickstart workspace]$ hadoop fs -put matrix.txt
[cloudera@quickstart workspace]$ █
```



```
[cloudera@quickstart workspace]$ hadoop jar MatrixMul.jar MatrixDriver matrix.txt MatrixOutput
```



```
[cloudera@quickstart workspace]$ hadoop fs -cat MatrixOutput/part-r-00000
0,0,19.0
0,1,22.0
1,0,43.0
1,1,50.0
```