**<u>SOURCE CODE:</u>**

**<u>MatrixMultiply.java:</u>**

```
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import java.util.ArrayList;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.*;
import org.apache.hadoop.mapreduce.lib.output.*;
import org.apache.hadoop.util.ReflectionUtils;

class Element implements Writable {
		int tag;
		int index;
		double value;

		Element() {
				tag = 0;
				index = 0;
				value = 0.0;
		}

		Element(int tag, int index, double value) {
				this.tag = tag;
				this.index = index;
				this.value = value;
		}

		@Override
		public void readFields(DataInput input) throws IOException {
				tag = input.readInt();
				index = input.readInt();
				value = input.readDouble();
		}

		@Override
		public void write(DataOutput output) throws IOException {
				output.writeInt(tag);
				output.writeInt(index);
				output.writeDouble(value);
```

```
	}
}

class Pair implements WritableComparable<Pair> {

	int i;
	int j;

	Pair() {
		i = 0;
		j = 0;
	}

	Pair(int i, int j) {
		this.i = i;
		this.j = j;
	}

	@Override
	public void readFields(DataInput input) throws IOException {
		i = input.readInt();
		j = input.readInt();
	}

	@Override
	public void write(DataOutput output) throws IOException {
		output.writeInt(i);
		output.writeInt(j);
	}

	@Override
	public int compareTo(Pair compare) {

		if (i > compare.i) {
			return 1;
		} else if ( i < compare.i) {
			return -1;
		} else {
			if(j > compare.j) {
				return 1;
			} else if (j < compare.j) {
				return -1;
			}
		}
		return 0;
	}

	public String toString() {
		return i + " " + j + " ";
	}
```

```
}

public class MatrixMultiply {

        public static class MatriceMapperM extends Mapper<Object,Text,IntWritable,Element> {

                @Override
                public void map(Object key, Text value, Context context)
                                throws IOException, InterruptedException {
                        String readLine = value.toString();
                        String[] stringTokens = readLine.split(",");

                        int index = Integer.parseInt(stringTokens[0]);
                        double elementValue = Double.parseDouble(stringTokens[2]);

                        Element e = new Element(0, index, elementValue);

                        IntWritable keyValue = new IntWritable(Integer.parseInt(stringTokens[1]));
                        context.write(keyValue, e);
                }
        }

        public static class MatriceMapperN extends Mapper<Object,Text,IntWritable,Element> {

                @Override
                public void map(Object key, Text value, Context context)
                                throws IOException, InterruptedException {
                        String readLine = value.toString();
                        String[] stringTokens = readLine.split(",");

                        int index = Integer.parseInt(stringTokens[1]);
                        double elementValue = Double.parseDouble(stringTokens[2]);

                        Element e = new Element(1,index, elementValue);

                        IntWritable keyValue = new IntWritable(Integer.parseInt(stringTokens[0]));
                        context.write(keyValue, e);
                }
        }

        public static class ReducerMxN extends Reducer<IntWritable,Element, Pair, DoubleWritable>
{

        @Override
        public void reduce(IntWritable key, Iterable<Element> values, Context context)
                        throws IOException, InterruptedException {

                ArrayList<Element> M = new ArrayList<Element>();
                ArrayList<Element> N = new ArrayList<Element>();

                Configuration conf = context.getConfiguration();
```

```
			for(Element element : values) {

				Element tempElement = ReflectionUtils.newInstance(Element.class, conf);
				ReflectionUtils.copy(conf, element, tempElement);

				if (tempElement.tag == 0) {
					M.add(tempElement);
				} else if(tempElement.tag == 1) {
					N.add(tempElement);
				}
			}

			for(int i=0;i<M.size();i++) {
				for(int j=0;j<N.size();j++) {

					Pair p = new Pair(M.get(i).index,N.get(j).index);
					double multiplyOutput = M.get(i).value * N.get(j).value;

					context.write(p, new DoubleWritable(multiplyOutput));
					}
				}
			}
		}

		public static class MapMxN extends Mapper<Object, Text, Pair, DoubleWritable> {
			@Override
			public void map(Object key, Text value, Context context)
					throws IOException, InterruptedException {

				String readLine = value.toString();
				String[] pairValue = readLine.split(" ");

				Pair p = new
	Pair(Integer.parseInt(pairValue[0]),Integer.parseInt(pairValue[1]));
				DoubleWritable val = new
	DoubleWritable(Double.parseDouble(pairValue[2]));

				context.write(p, val);
			}
		}

		public static class ReduceMxN extends Reducer<Pair, DoubleWritable, Pair, DoubleWritable>
	{
			@Override
			public void reduce(Pair key, Iterable<DoubleWritable> values, Context context)
			throws IOException, InterruptedException {

				double sum = 0.0;
				for(DoubleWritable value : values) {
					sum += value.get();
```

```
                    }

                    context.write(key, new DoubleWritable(sum));
            }
    }

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

            Job job = Job.getInstance();
            job.setJobName("MapIntermediate");
            job.setJarByClass(MatrixMultiply.class);

            MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class,
MatriceMapperM.class);
            MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class,
MatriceMapperN.class);
            job.setReducerClass(ReducerMxN.class);

            job.setMapOutputKeyClass(IntWritable.class);
            job.setMapOutputValueClass(Element.class);

            job.setOutputKeyClass(Pair.class);
            job.setOutputValueClass(DoubleWritable.class);

            job.setOutputFormatClass(TextOutputFormat.class);

            FileOutputFormat.setOutputPath(job, new Path(args[2]));

            job.waitForCompletion(true);



            Job job2 = Job.getInstance();
            job2.setJobName("MapFinalOutput");
            job2.setJarByClass(MatrixMultiply.class);

            job2.setMapperClass(MapMxN.class);
            job2.setReducerClass(ReduceMxN.class);

            job2.setMapOutputKeyClass(Pair.class);
            job2.setMapOutputValueClass(DoubleWritable.class);

            job2.setOutputKeyClass(Pair.class);
            job2.setOutputValueClass(DoubleWritable.class);

            job2.setInputFormatClass(TextInputFormat.class);
            job2.setOutputFormatClass(TextOutputFormat.class);

            FileInputFormat.setInputPaths(job2, new Path(args[2]));
            FileOutputFormat.setOutputPath(job2, new Path(args[3]));
```

```
                job2.waitForCompletion(true);
        }
}
```

```
m-matrix
1,1,-2.0
0,0,5.0
2,2,6.0
0,1,-3.0
3,2,7.0
0,2,-1.0
1,0,3.0
1,2,4.0
2,0,1.0
3,0,-4.0
3,1,2.0
```

```
n-matrix
1,0,3.0
0,0,5.0
1,2,-2.0
2,0,9.0
0,1,-3.0
0,2,-1.0
1,1,8.0
2,1,4.0
```

to run

**Create m-matrix n-matrix**
**upload in root folder of hdfs**
**hdfs dfs -put m-matrix**
**hdfs dfs -put n-matrix**

To compile java files into class files located in src folder:

```
javac *.java -cp $(hadoop classpath)
```

To move compiled class files into jar named awlBMH.jar:

```
jar cvf mm.jar *.class
```

To submit jar file to Hadoop cluster. Input directoryoutput directory

```
hadoop jar mm.jar MatrixMultiply m-matrix n-matrix interim output
```

To look file names in newly created hadoop directory:

```
hdfs dfs -ls output
```

To look at output file text in hadoop directory:

```
hadoop fs -cat output/part-r-00000 ( or whatever your file name is)
```