**AIM:** To write a map reduce program to implement marix multiplication.

**PROGRAM:**

**/\*MAPPER\*/**

**//Mapper for First input matrix**

import java.util.\*;

import java.io.\*;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.mapred.JobConf;

class MatrixMapper1 extends Mapper<LongWritable, Text, Text, Text>

{

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

{

Configuration conf = context.getConfiguration();

int iteration=Integer.parseInt(conf.get("colB"));

String ty=value.toString();

String[] gh=ty.split(" ");

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

{

context.write(new Text(gh[0]+","+j),new Text(value+" "+"0"));

}

}

}

**//Mapper for Second input marix**

import java.util.\*;

import java.io.\*;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.conf.Configuration;

class MatrixMapper2 extends Mapper<LongWritable, Text, Text, Text>

{

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

{

Configuration conf = context.getConfiguration();

int iteration=Integer.parseInt(conf.get("rowA"));

String ty=value.toString();

String[] gh=ty.split(" ");

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

{

context.write(new Text(j+","+gh[1]),new Text(value+" "+"1"));

}

}

}

**/\*REDUCER\*/**

import java.util.\*;

import java.io.\*;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.mapred.JobConf;

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

{

int sum,c1,c2,c,b,x,y,col;

int[][] a=new int[100][100];

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

{

Configuration conf = context.getConfiguration();

int iteration=Integer.parseInt(conf.get("colA"));

y=0;

sum=0;

String keys=key.toString();

String[] l=keys.split(",");

c1=Integer.parseInt(l[0]);

c2=Integer.parseInt(l[1]);

for(Text tre:values)

{

String valu=tre.toString();

String[] arr=valu.split(" ");

for(x=0;x<arr.length;x++)

{

a[y][x]=Integer.parseInt(arr[x]);

}

y++;

}

for(int k=0;k<iteration;k++)

{

c=search(a,c1,k,0);

b=search(a,k,c2,1);

sum=sum+c\*b;

}

context.write(new Text(key),new LongWritable(sum));

}

public int search(int[][] q,int w,int e,int u)

{

int g;

for(g=0;g<q[0].length;g++)

{

if((q[g][0]==w)&&(q[g][1]==e)&&(q[g][3]==u))

{

return q[g][2];

}

}

return 0;

}

}

**/\*DRIVER\*/**

import java.util.\*;

import java.io.\*;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;

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 extends Configured implements Tool

{

public int run(String args[]) throws Exception

{

Configuration conf = this.getConf();

conf.set("rowA",args[3]);

conf.set("colA",args[4]);

conf.set("rowB",args[5]);

conf.set("colB",args[6]);

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

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(LongWritable.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setMapperClass(MatrixMapper1.class);

job.setMapperClass(MatrixMapper2.class);

job.setReducerClass(MatrixReducer.class);

MultipleInputs.addInputPath(job, new Path(args[0]),TextInputFormat.class,MatrixMapper1.class);

MultipleInputs.addInputPath(job, new Path(args[1]),TextInputFormat.class,MatrixMapper2.class);

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

job.setJarByClass(MatrixDriver.class);

return job.waitForCompletion(true) ? 0 : 1;

}

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

{

int res= ToolRunner.run(new Configuration(), new MatrixDriver(), args);

System.exit(res);

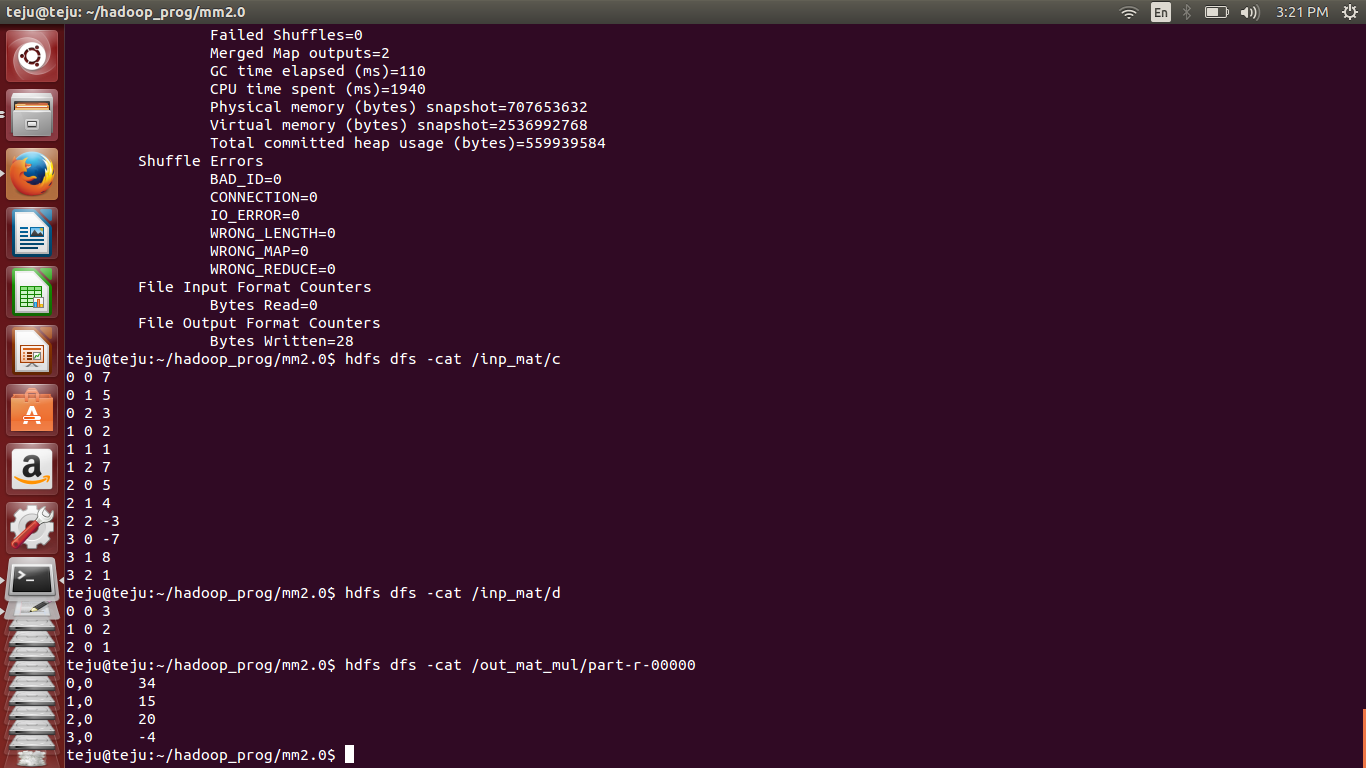
}

}

**OUTPUT 1:**

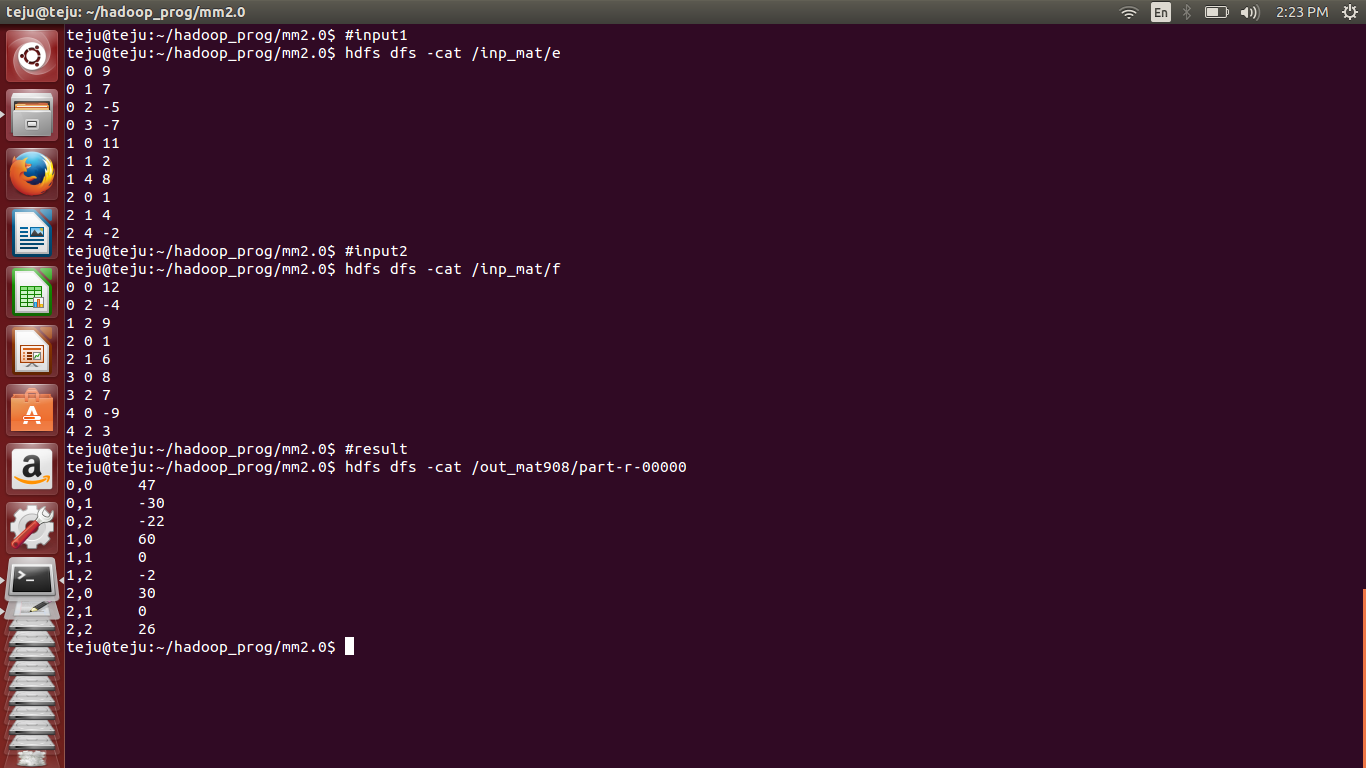
**matrixA of order 4X3**

**matrixB of order 3X1**

 **OUTPUT 2:**

**matrixA of order 3X5**

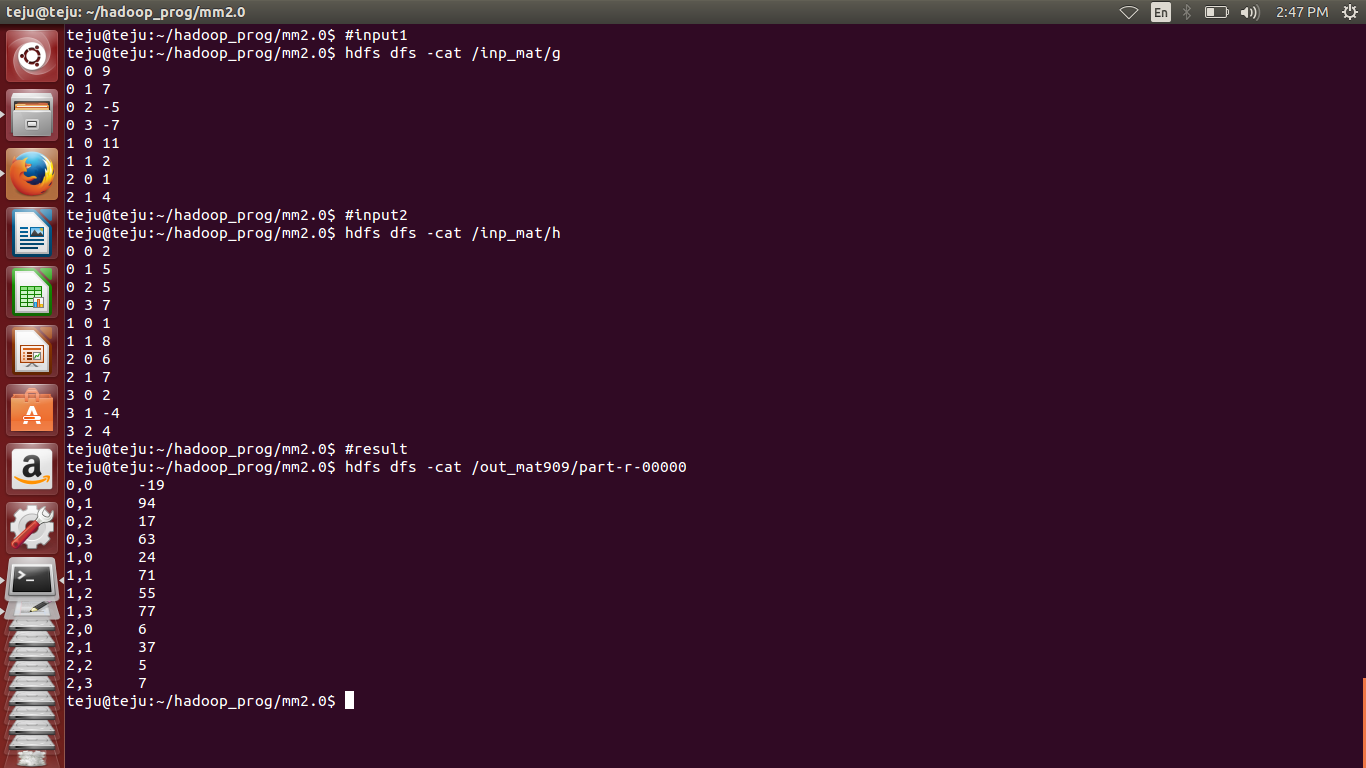
**matrixB of order 5X3**

****

**OUTPUT 3:**

**matrixA of order 3X4**

**matrixB of order 4X4**

****