**10. Implement Matrix Multiplication with Hadoop Map Reduce**

**Step 1. Download the hadoop jar files with these links.**

Download Hadoop Common Jar files: https://goo.gl/G4MyHp

$ wget https://goo.gl/G4MyHp -O hadoop-common-2.2.0.jar

Download Hadoop Mapreduce Jar File: https://goo.gl/KT8yfB

$ wget https://goo.gl/KT8yfB -O hadoop-mapreduce-client-core-2.7.1.jar

**Step 2. Creating Mapper file for Matrix Multiplication.**

package www**.**ehadoopinfo**.**com**;**  
**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**);**  
                       **}**  
               **}**  
       **}**  
**}**  
**program ends here**

**Step 3. Creating Reducer.java file for Matrix Multiplication.**

package www**.**ehadoopinfo**.**com**;**  
  
**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 4. Creating MatrixMultiply.java file for**

package www**.**ehadoopinfo**.**com**;**

**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**();**  
       // M is an m-by-n matrix; N is an n-by-p matrix.  
       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 5. Uploading the M, N file which contains the matrix multiplication data to HDFS.**

$ cat M

M,0,0,1

M,0,1,2

M,1,0,3

M,1,1,4

$ cat N

N,0,0,5

N,0,1,6

N,1,0,7

N,1,1,8

$ hadoop fs -mkdir Matrix/

$ hadoop fs -copyFromLocal M Matrix/

$ hadoop fs -copyFromLocal N Matrix/

**Step 6. Executing the jar file using hadoop command and thus how fetching record from HDFS and storing output in HDFS.**

**Step 7. Getting Output from part-r-00000 that was generated after the execution of the hadoop command.**

$ hadoop fs -cat result/part-r-00000

0,0,19.0

0,1,22.0

1,0,43.0

1,1,50.0