Assignment-2 : Matrix Multiplication

`MTCS-202(P) - Hadoop Assignment`

`Aryan Sai Arvapelly, Regd. No - 23352, I MTech CS`

The **Hadoop Map-Reduce** program to calculate the product of two matrices, A(m x n) and B(n x p), is as follows:

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
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.FileInputForma
t;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFor
mat;

public class MatrixMul {

    public static class MatrixMapper extends Mapper<Object,
Text, Text, Text> {
        public void map(Object key, Text value, Context con
text) throws IOException, InterruptedException {
            String[] tokens = value.toString().split(",");
            String matrix = tokens[0];
            int row = Integer.parseInt(tokens[1]);
            int col = Integer.parseInt(tokens[2]);
            int val = Integer.parseInt(tokens[3]);
```

```
            if (matrix.equals("A")) {
                for (int k = 0; k < context.getConfiguratio
n().getInt("p", 1); k++) {
                    context.write(new Text(row + "," + k),
new Text(matrix + "," + col + "," + val));
                }
            } else if (matrix.equals("B")) {
                for (int i = 0; i < context.getConfiguratio
n().getInt("m", 1); i++) {
                    context.write(new Text(i + "," + col),
new Text(matrix + "," + row + "," + val));
                }
            }
        }
    }

    public static class MatrixReducer extends Reducer<Text,
Text, Text, IntWritable> {
        public void reduce(Text key, Iterable<Text> values,
Context context) throws IOException, InterruptedException {
            int[] rowA = new int[context.getConfiguration
().getInt("n", 1)];
            int[] colB = new int[context.getConfiguration
().getInt("n", 1)];

            for (Text val : values) {
                String[] tokens = val.toString().split
(",");
                String matrix = tokens[0];
                int index = Integer.parseInt(tokens[1]);
                int value = Integer.parseInt(tokens[2]);

                if (matrix.equals("A")) {
                    rowA[index] = value;
                } else if (matrix.equals("B")) {
                    colB[index] = value;
                }
            }
```

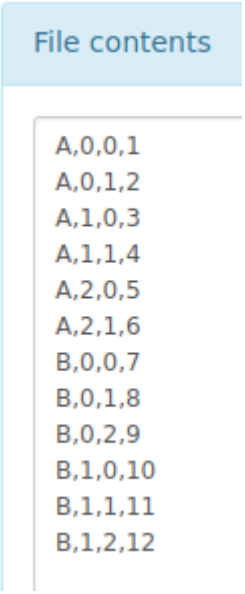
```
            int result = 0;
            for (int i = 0; i < context.getConfiguration().
getInt("n", 1); i++) {
                result += rowA[i] * colB[i];
            }
            context.write(key, new IntWritable(result));
        }
    }

    public static void main(String[] args) throws Exception
{
        Configuration conf = new Configuration();
        conf.setInt("m", Integer.parseInt(args[2]));
        conf.setInt("n", Integer.parseInt(args[3]));
        conf.setInt("p", Integer.parseInt(args[4]));
        Job job = Job.getInstance(conf, "matrix multiplicat
ion");
        job.setJarByClass(MatrixMul.class);
        job.setMapperClass(MatrixMapper.class);
        job.setReducerClass(MatrixReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        FileInputFormat.addInputPath(job, new Path(args
[0]));
        FileOutputFormat.setOutputPath(job, new Path(args
[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Input

The program takes the input in the format: `matrix,row,column,value`

To run the program, we need to pass the input and output paths along with the dimensions of the two matrices: **m, n, p.**

```
hadoop jar MatrixMul.jar MatrixMul /matmul/input.txt /outpu
t m n p
```



Mapper

The mapper receives input lines representing elements of Matrix A and Matrix B. For each input line, the mapper emits key-value pairs in the below way:

1. **For Matrix A:**
    - **Intuition:** The value `val` at row `row` and column `col` in Matrix A needs to be multiplied with the value at row `row` and column `k` in Matrix B.
    - The mapper receives an input value from Matrix A. It iterates over all columns of Matrix B for the given row of Matrix A.
    - For each column `k`, it emits a key-value pair where the key is of the format `"row,k"` and the value is of the format `"A,col,val"`.
2. **For Matrix B:**
    - **Intuition:** The value `val` at row `row` and column `col` in Matrix B needs to be multiplied with the value at row `i` and column `col` in Matrix A.
    - The mapper receives an input value from Matrix B. It iterates over all rows of Matrix A for the given column of Matrix B.

- For each row `i`, it emits a key-value pair where the key is of the format `"i,col"` and the value is of the format `"B,row,val"`.

Reducer

1. **Array Initialization:**
    - Two integer arrays `rowA` and `colB` are initialized with the length equal to the number of columns in the matrices(**n**).
2. **Adding elements to the arrays:**
    - For each received value, it parses the string representation into matrix identifier, index, and value.
    - Based on the matrix identifier, the value is stored in the appropriate array (`rowA` for Matrix A and `colB` for Matrix B) at the corresponding index.
3. **Matrix Multiplication:**
    - Perform the matrix multiplication by iterating over the arrays `rowA` and `colB`.
    - For each index `i`, it computes the product of corresponding elements from `rowA` and `colB`, and accumulates the result.
    - The accumulated result represents the value of the resulting matrix at the key (row, column) pair.

Output

- The reducer emits the key-value pair where the key is the (row, column) pair and the value is the computed result of matrix multiplication at that position.

File contents

```
0,0  27
0,1  30
0,2  33
1,0  61
1,1  68
1,2  75
2,0  95
2,1  106
2,2  117
```