Matrices Sum Map Reducer

Matrix sum is the operation of adding matrices by adding corresponding entries together.

Entrywise sum
The sum of two m × n matrices A and B, denoted by A + B, is again an m × n matrix computed by adding corresponding elements

$$\mathbf{A}+\mathbf{B} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{bmatrix}$$

1. Mapper emits the row index as key and entire row as a value. If you have different types of matrices then create separate mappers to process/ filters the matrix.

```
class MatrixSumMapper extends
Mapper&lt;LongWritable, Text, LongWritable, Text&gt; {
String fName = null;
char keySeprator;
@Override
protected void setup(
Mapper&lt;LongWritable, Text, LongWritable, Text&gt;.Context context)
throws IOException, InterruptedException {
fName = ((FileSplit)context.getInputSplit()).getPath().getName();
keySeprator=(char)context.getConfiguration().getInt("matrix.key.separator",0x001
}
@Override
protected void map(LongWritable key, Text value,
Mapper&lt;LongWritable, Text, LongWritable, Text&gt;.Context context)
throws IOException, InterruptedException {
LongWritable keyM = new LongWritable(Long.parseLong(value.toString().split(Strin
Text val = new Text(value.toString().split(String.format("%c",keySeprator))[1]);
context.write(keyM, val);
}
}
```

2. Reducer gets the row key as (m). Next split each value to generate (n) then add the values index and position wise.

```
class MatrixSumReducer extends
Reducer&lt;LongWritable, Text, LongWritable, Text&gt; {
String vseparator;
String fseparator;
private static Logger logger = Logger.getLogger(MatrixSumMapper.class);
@Override
protected void setup(
Reducer&lt;LongWritable, Text, LongWritable, Text&gt;.Context context)
throws IOException, InterruptedException {
logger.debug("reducer- setup: begin");
vseparator = context.getConfiguration().get("matrix.element.separator");
logger.debug("reducer- setup: end");
}
@Override
protected void reduce(LongWritable key, Iterable value,
Reducer&lt;LongWritable, Text, LongWritable, Text&gt;.Context ctxt)
throws IOException, InterruptedException {
List&lt;Integer[]&gt; matrixValues = new ArrayList&lt;Integer[]&gt;();
List val = new ArrayList();
for (Text t : value) {
prepareKeyValueMap(t, matrixValues);
}
for (Integer[] i : matrixValues) {
for (int j = 0; j &lt; i.length; j++) {
try{
int sum = val.remove(j);
val.add(j,sum+i[j]);
}catch(Exception e){
System.out.println("throwing exception");
val.add(i[j]);
}
}
}
ctxt.write(key, new Text(val.toString()));
}
private void prepareKeyValueMap(Text value, List&lt;Integer[]&gt; valuesLst) {
String[] valuesStr = value.toString().split(vseparator);
Integer[] valuesInt = new Integer[valuesStr.length];
for (int i = 0; i &lt; valuesStr.length; i++) {
try {
valuesInt[i] = Integer.parseInt(valuesStr[i]);
} catch (NumberFormatException e) {
logger.error(e.getMessage());
}
}
valuesLst.add(valuesInt);
}
}
```

3. Driver Code

```java
public class Driver extends Configured implements Tool {
private static Logger logger = Logger.getLogger(Driver.class);
private boolean deleteDirectory(Path path) throws IOException {
FileSystem fs = FileSystem.get(getConf());
return fs.delete(path, true);
}
public int run(String[] args) throws Exception {
logger.info("job Matrix Sum Driver Begin");
Configuration conf = getConf();
conf.setInt("matrix.key.separator", 0x001);
conf.set("matrix.element.separator",",");
Job job = new Job(conf, "Matrix Sum");
job.setJarByClass(Driver.class);
Path input1 = new Path(args[0]);
Path input2 = new Path(args[1]);
Path output = new Path(args[2]);
deleteDirectory(output);
job.setMapOutputKeyClass(LongWritable.class);
job.setMapOutputValueClass(Text.class);
job.setMapperClass(MatrixSumMapper.class);
job.setReducerClass(MatrixSumReducer.class);
job.setNumReduceTasks(1);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
logger.info("deleting output directory: " + deleteDirectory(output));
FileInputFormat.setInputPaths(job, input1, input2);
FileOutputFormat.setOutputPath(job, output);
return job.waitForCompletion(true) ? 0 : 1;
}
public static void main(String[] args) throws Exception {
for (String str : args)
System.out.println(str);
Configuration config = new Configuration();
System.exit(ToolRunner.run(config, new Driver(), args));
}
}
```