使用Hadoop的MapReduce算法实现矩阵相乘。

3112005816 计算机科学 2 班 何柏超

请简答：“数据输入、Map和Reduce”三大步骤中用到的编程接口，同时以<key,value>格式写出三大步骤各自的输入、输出数据。

答：

数据输入格式：

```
m,n,p       // 矩阵 A、B 的维度 （mxn 和 nxp）
A,0,0,5     // 表示 A 矩阵的（0，0） 值为 5
B,m-1,n-1,4 // 表示 B 矩阵的（m-1,n-1）值为 4
```

数据输出格式：

```
0,0 4  // 结果矩阵的（0, 0）值为 4
```

数据输入使用接口：

使用了 org.apache.hadoop.mapreduce.lib.input.FileInputFormat 来设定数据输入路径。

Map 使用接口：

通过继承 org.apache.hadoop.mapreduce.Mapper 和实现 map 方法来完成 map 操作。

Reduce 使用接口：

通过继承 org.apache.hadoop.mapreduce.Reducer 和实现 reduce 方法来完成 reduce 操作。

Map 过程输入输出数据格式：

输入：

```
<"", (矩阵名称,横坐标,纵坐标,元素值)>
```

输出：

```
<(结果矩阵横坐标,结果矩阵纵坐标), (矩阵名称,坐标分量,元素值)>
```

Reduce 过程输入输出数据格式：

输入：

```
<(结果矩阵横坐标,结果矩阵纵坐标), (矩阵名称,坐标分量,元素值)>
```

输出：

```
<(结果矩阵横坐标,结果矩阵纵坐标), 元素值>
```

程序实现：

程序通过多线程来模拟 map reduce 框架的结构来进行计算。
其中 map 步骤使用了 3 个线程， reduce 步骤使用了 10 个线程。

对一个 3x4 * 4x5 任务求值结果如下：

输入

```
3,4,5
A,0,0,5
A,0,1,2
A,0,2,1
A,0,3,2
A,1,0,5
A,1,1,5
A,1,2,5
A,1,3,1
A,2,0,3
A,2,1,3
A,2,2,1
A,2,3,3
B,0,0,3
B,0,1,2
B,0,2,1
B,0,3,3
B,0,4,3
B,1,0,2
B,1,1,3
B,1,2,4
B,1,3,5
B,1,4,2
B,2,0,2
B,2,1,2
B,2,2,3
B,2,3,3
B,2,4,2
B,3,0,3
B,3,1,2
B,3,2,4
B,3,3,4
B,3,4,3
```

输出


```
1,1 37.0
0,3 36.0
1,3 59.0
2,1 23.0
2,2 30.0
1,4 38.0
1,2 44.0
2,4 26.0
0,0 27.0
2,0 26.0
```

0,1 22.0
0,4 27.0
1,0 38.0
2,3 39.0
0,2 24.0

Mapper 代码:

```
protected void map(String key, String value, MapContext context) {
    // 每行输入形式：  A,i,j,A[i, j] / B,i,j,B[i, j]
    String[] unit = value.split(",");

    int m = Integer.parseInt(context.getConfigurationValue("m"));
    int p = Integer.parseInt(context.getConfigurationValue("p"));

    // map 输出
    String outputKey;
    String outputValue;

    if (unit[0].equals("A")) {
        for (int k = 0; k < p; k = k + 1) {
            // 使用 A 矩阵的 <i, k> 作为键
            outputKey = unit[1] + "," + k;
            // 使用 A 矩阵的 <j, A[i, j]> 作为值
            outputValue = unit[0] + "," + unit[2] + "," + unit[3];

            // emit
            context.write(outputKey, outputValue);
        }
    } else {
        for (int k = 0; k < m; k = k + 1) {
            // 使用 B 矩阵的 <k, j> 作为键
            outputKey = k + "," + unit[2];
            // 使用 B 矩阵的 <i, B[i, j]> 作为值
            outputValue = unit[0] + "," + unit[1] + "," + unit[3];

            // emit
            context.write(outputKey, outputValue);
        }
    }
}
```

Reducer 代码:

```
protected void reduce(String key, Iterable<String> values, ReduceContext context) {
    String[] value;

    // 记录 A / B 矩阵的值
    HashMap<Integer, Double> a = new HashMap<Integer, Double>();
    HashMap<Integer, Double> b = new HashMap<Integer, Double>();

    for (String val : values) {
        value = val.split(",");
        int k = Integer.parseInt(value[1]);
        double item = Double.parseDouble(value[2]);

        if (value[0].equals("A")) {
            a.put(k, item);
        } else {
            b.put(k, item);
        }
    }

    int n = Integer.parseInt(context.getConfigurationValue("n"));
    double rv = 0.0f;
    double a_ij;
```

```
    double b_jk;
    // 执行累加
    for (int j = 0; j < n; j = j + 1) {
        a_ij = a.containsKey(j) ? a.get(j) : 0.0f;
        b_jk = b.containsKey(j) ? b.get(j) : 0.0f;
        rv = rv + a_ij * b_jk;
    }

    // emit
    context.write(key, rv);
}
```

完整代码和测试用例见附件。