Hadoop多文件输出：MultipleOutputFormat和MultipleOutputs深究(二)

由于本文比较长，考虑到篇幅问题，所以将本文拆分为二，请阅读本文之前先阅读本文的第一部分《Hadoop多文件输出：MultipleOutputFormat和MultipleOutputs深究(一)》。为你带来的不变，敬请谅解。

与MultipleOutputFormat类不一样的是，MultipleOutputs可以为不同的输出产生不同类型，到这里所说的MultipleOutputs类还是旧版本的功能，后面会提到新版本类库的强化版MultipleOutputs类，下面我们来用旧版本的MultipleOutputs类说明它是如何为不同的输出产生不同类型，MultipleOutputs类不是要求给每条记录请求文件名，而是创建多个OutputCollectors。每个OutputCollector可以有自己的OutputFormat和键值对类型，Mapreduce程序将决定如何向每个OutputCollector输出数据（看看上面的英文文档），说的你很晕吧，来看看代码吧！下面的代码将地理相关的信息存储在geo开头的文件中；而将时间相关的信息存储在chrono开头的文件中，具体的代码如下：

```
package com.wyp;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.mapred.lib.MultipleOutputs;
import org.apache.hadoop.util.GenericOptionsParser;

import java.io.IOException;

/**
 * User: /
 * Date: 13-11-27
 * Time: 下午10:32
 */
public class OldMulOutput {
    public static class MapClass
            extends MapReduceBase
            implements Mapper<LongWritable,
            Text, NullWritable, Text> {
        private MultipleOutputs mos;
        private OutputCollector<NullWritable, Text> collector;

        public void configure(JobConf conf) {
```

```
    mos = new MultipleOutputs(conf);
}

public void map(LongWritable key, Text value,
                OutputCollector<NullWritable, Text> output,
                Reporter reporter) throws IOException {
    String[] arr = value.toString().split(",", -1);
    String chrono = arr[1] + "," + arr[2];
    String geo = arr[4] + "," + arr[5];
    collector = mos.getCollector("chrono", reporter);
    collector.collect(NullWritable.get(), new Text(chrono));
    collector = mos.getCollector("geo", reporter);
    collector.collect(NullWritable.get(), new Text(geo));
}

public void close() throws IOException {
    mos.close();
}

public static void main(String[] args) throws IOException {
    Configuration conf = new Configuration();
    String[] remainingArgs =
     new GenericOptionsParser(conf, args).getRemainingArgs();

    if (remainingArgs.length != 2) {
        System.err.println("Error!");
        System.exit(1);
    }

    JobConf job = new JobConf(conf, OldMulOutput.class);
    Path in = new Path(remainingArgs[0]);
    Path out = new Path(remainingArgs[1]);
    FileInputFormat.setInputPaths(job, in);
    FileOutputFormat.setOutputPath(job, out);

    job.setJobName("MultiFile");
    job.setMapperClass(MapClass.class);
    job.setInputFormat(TextInputFormat.class);
    job.setOutputKeyClass(NullWritable.class);
    job.setOutputValueClass(Text.class);

    job.setNumReduceTasks(0);
    MultipleOutputs.addNamedOutput(job,
            "chrono",
            TextOutputFormat.class,
            NullWritable.class,
```

```
                Text.class);

        MultipleOutputs.addNamedOutput(job,
                "geo",
                TextOutputFormat.class,
                NullWritable.class,
                Text.class);
        JobClient.runJob(job);

    }
  }
}
```

上面程序来源《Hadoop in action》。同样将上面的程序打包成jar文件（具体怎么打包，也不说了），并在Hadoop2.2.0上面运行（测试数据请在这里下载：http://pan.baidu.com/s/1td8xN）：

```
/home/q/hadoop-2.2.0/bin/hadoop jar                    ₩
    /export1/tmp/wyp/OutputText.jar com.wyp.OldMulOutput ₩
    /home/wyp/apat63_99.txt                            ₩
    /home/wyp/out5
```

运行完程序之后，可以去/home/wyp/out5目录看下运行结果：

```
[wyp@l-datalogm1.data.cn1 bin]$ /home/q/hadoop-2.2.0/bin/hadoop fs  ₩
                    -ls /home/wyp/out5
Found 7 items
-rw-r--r-- 3 wyp sg     0 2013-11-26 14:57 /home/wyp/out5/_SUCCESS
-rw-r--r-- 3 wyp sg 31243 2013-11-26 15:57 /home/wyp/out5/chrono-m-00000
-rw-r--r-- 3 wyp sg 22719 2013-11-26 15:57 /home/wyp/out5/chrono-m-00001
-rw-r--r-- 3 wyp sg 29922 2013-11-26 15:57 /home/wyp/out5/geo-m-00000
-rw-r--r-- 3 wyp sg 20429 2013-11-26 15:57 /home/wyp/out5/geo-m-00001
-rw-r--r-- 3 wyp sg     0 2013-11-26 14:57 /home/wyp/out5/part-m-00000
-rw-r--r-- 3 wyp sg     0 2013-11-26 14:57 /home/wyp/out5/part-m-00001
```

大家可以看到在输出的文件中还存在以part开头的文件，但是里面没有信息，这是程序默认的输出文件，输出的收集器的名称是不能为part的，这是因为它已经被使用为默认的值了。

从上面的程序可以看出，旧版本的MultipleOutputs可以将文件基于列来进行分割，但是如果你想进行基于行的分割，这就要求你自己去实现代码了，恨不方便，针对这个问题，新版本的MultipleOutputs已经将旧版本的MultipleOutputs和MultipleOutputFormat的功能合并了，也就是说新版本的MultipleOutputs类具有旧版本的MultipleOutputs功能和MultipleOutputFormat功能；同时，在新版本的类库中已经不存在MultipleOutputFormat类了，因为MultipleOutputs都有它的功能了，还要她干嘛呢？看看官方文档是怎么说的：

The MultipleOutputs class simplifies writing output data to multiple outputs
Case one: writing to additional outputs other than the job default output. Each additional output, or named output, may be configured with its own OutputFormat, with its own key class and with its own value class.
Case two: to write data to different files provided by user

再看看下面摘自Hadoop：The.Definitive.Guide(3rd,Early.Release)P251，它是怎么说的：

In the old MapReduce API there are two classes for producing multiple outputs: MultipleOutputFormat and MultipleOutputs. In a nutshell, MultipleOutputs is more fully featured, but MultipleOutputFormat has more control over the output directory structure and file naming. MultipleOutputs in the new API combines the best features of the two multiple output classes in the old API.

也就是说MultipleOutputs合并了旧版本的MultipleOutputs功能和MultipleOutputFormat功能，新api都是用mapreduce包。好了，刚刚也说了新版本的MultipleOutputs有了旧版本的MultipleOutputFormat功能，那么我该怎么在新版的MultipleOutputs中实现旧版本MultipleOutputFormat的多文件输出呢？也就是上面第一个程序。看看下面的代码吧。

```
package com.wyp;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
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.LazyOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.MultipleOutputs;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

import java.io.IOException;
```

```
/**
 * User: /
 * Date: 13-11-26
 * Time: 下午2:27
 */
public class MulOutput {
  public static class MapClass
      extends Mapper<LongWritable, Text, NullWritable, Text> {
    private MultipleOutputs mos;
    @Override
    protected void setup(Context context)
        throws IOException, InterruptedException {
      super.setup(context);
      mos = new MultipleOutputs(context);
    }

    @Override
    protected void map(LongWritable key,
                       Text value,
                       Context context)
        throws IOException, InterruptedException {
      mos.write(NullWritable.get(), value,
          generateFileName(value));
    }

    private String generateFileName(Text value) {
      String[] split = value.toString().split(",", -1);
      String country = split[4].substring(1, 3);
      return country + "/";
    }

    @Override
    protected void cleanup(Context context)
        throws IOException, InterruptedException {
      super.cleanup(context);
      mos.close();
    }
  }

  public static void main(String[] args)
      throws IOException, ClassNotFoundException,
      InterruptedException {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "MulOutput");
    String[] remainingArgs =
```

```
            new GenericOptionsParser(conf, args)
                .getRemainingArgs();

        if (remainingArgs.length != 2) {
            System.err.println("Error!");
            System.exit(1);
        }
        Path in = new Path(remainingArgs[0]);
        Path out = new Path(remainingArgs[1]);

        FileInputFormat.setInputPaths(job, in);
        FileOutputFormat.setOutputPath(job, out);

        job.setJarByClass(MulOutput.class);
        job.setMapperClass(MapClass.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputKeyClass(NullWritable.class);
        job.setOutputValueClass(Text.class);
        job.setNumReduceTasks(0);

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

上面的程序通过setup(Context context)来初始化MultipleOutputs对象，并在mapper函数中调用MultipleOutputs的write方法将数据输出到根据value的值不同的文件夹中（通过调用generateFileName函数来处理）。MultipleOutputs类有多个不同版本的write方法，它们的函数原型如下：

```
public <K, V> void write(String namedOutput, K key, V value)
    throws IOException, InterruptedException

public <K, V> void write(String namedOutput, K key, V value,
    String baseOutputPath) throws IOException, InterruptedException

public void write(KEYOUT key, VALUEOUT value, String baseOutputPath)
    throws IOException, InterruptedException
```

我们可以根据不同的需求调用不同的write方法。
好了，大家来看看上面程序运行的结果吧：

```
/home/q/hadoop-2.2.0/bin/hadoop jar                    ₩
    /export1/tmp/wyp/OutputText.jar com.wyp.MulOutput    ₩
    /home/wyp/apat63_99.txt                          ₩
    /home/wyp/out11
```

测试数据还是上面给的地址。看下/home/wyp/out11文件中有什么吧：

```
[wyp@l-datalogm1.data.cn1 bin]$ /home/q/hadoop-2.2.0/bin/hadoop fs ₩
                                -ls /home/wyp/out11
.............................这里省略了很多.....................................
drwxr-xr-x - wyp  supergroup  0 2013-11-26 19:42 /home/wyp/out11/VN
drwxr-xr-x - wyp  supergroup  0 2013-11-26 19:41 /home/wyp/out11/VU
drwxr-xr-x - wyp  supergroup  0 2013-11-26 19:42 /home/wyp/out11/YE
drwxr-xr-x - wyp  supergroup  0 2013-11-26 19:42 /home/wyp/out11/YU
drwxr-xr-x - wyp  supergroup  0 2013-11-26 19:42 /home/wyp/out11/ZA
.............................这里省略了很多.....................................
-rw-r--r-- 3 wyp  supergroup  0 2013-11-26 19:42 /home/wyp/out11/_SUCCESS
-rw-r--r-- 3 wyp  supergroup  0 2013-11-26 19:42 /home/wyp/out11/part-m-00000
-rw-r--r-- 3 wyp  supergroup  0 2013-11-26 19:42 /home/wyp/out11/part-m-00001
```

现在输出的结果和用旧版本的MultipleOutputFormat输出的结果很类似了；但是在输出的结果中还有两个以part开头的文件夹，而且里面什么都没有，这是怎么回事？和第二个测试程序一样，这也是程序默认的输出文件名。那么我们可以在程序输出的结果中不输出两个文件夹吗？当然可以了，呵呵。如何实现呢？其实很简单，在上面的代码的main函数中加入以下一行代码：

```
LazyOutputFormat.setOutputFormatClass(job,
            TextOutputFormat.class);
```

如果加入了上面的一行代码，请同时注释掉你代码中下面一行代码（如果有）

```
job.setOutputFormatClass(TextOutputFormat.class);
```

再去看下输出结果吧：

```
[wyp@l-datalogm1.data.cn1 bin]$ /home/q/hadoop-2.2.0/bin/hadoop fs ₩
                                    -ls /home/wyp/out12
.............................这里省略了很多.....................................
drwxr-xr-x - wyp supergroup    0 2013-11-26 19:44 /home/wyp/out12/VU
drwxr-xr-x - wyp supergroup    0 2013-11-26 19:44 /home/wyp/out12/YE
drwxr-xr-x - wyp supergroup    0 2013-11-26 19:44 /home/wyp/out12/YU
drwxr-xr-x - wyp supergroup    0 2013-11-26 19:44 /home/wyp/out12/ZA
drwxr-xr-x - wyp supergroup    0 2013-11-26 19:44 /home/wyp/out12/ZM
drwxr-xr-x - wyp supergroup    0 2013-11-26 19:44 /home/wyp/out12/ZW
.............................这里省略了很多.....................................
-rw-r--r-- 3 wyp supergroup    0 2013-11-26 19:44 /home/wyp/out12/_SUCCESS
```

结果完全和例子一一样。（本文完）