三种方法实现Hadoop(MapReduce)全局排序(2)

我在前面的文章介绍了MapReduce中两种全排序的方法及其实现。但是上面的两种方法都是有很大的局限性：

- 方法一在数据量很大的时候会出现OOM问题；
- 方法二虽然能够将数据分散到多个Reduce中，但是问题也很明显：我们必须手动地找到各个Reduce的分界点，尽量使得分散到每个Reduce的数据量均衡。而且每次修改Reduce的个数时，都得手动去找一次Key的分界点！非常不灵活。

本文这里介绍的第三种使用MapReduce全局排序的方法算是比较通用了，而且是内置的实现。

使用TotalOrderPartitioner进行全排序

我们都知道Hadoop内置有个 HashPartitioner
分区实现类，MapReduce默认就是使用它；但其实Hadoop内置还有个名为
TotalOrderPartitioner 的分区实现类，看名字就清楚它其实就是解决全排序的问题。如果你去看他的实现，其主要做的事实际上和我们上文介绍的 IteblogPartitioner 分区实现类很类似，也就是根据Key的分界点将不同的Key发送到相应的分区。问题是，上文用到的分界点是我们人为计算的；而这里用到的分界点是由程序解决的！

数据抽样

寻找合适的Key分割点需要我们对数据的分布有个大概的了解；如果数据量很大的话，我们不可能对所有的数据进行分析然后选出 N-1 （N代表Reduce的个数）个分割点，最适合的方式是对数据进行抽样，然后对抽样的数据进行分析并选出合适的分割点。Hadoop提供了三种抽样的方法：

- SplitSampler：从s个split中选取前n条记录取样
- RandomSampler：随机取样
- IntervalSampler：从s个split里面按照一定间隔取样，通常适用于有序数据

这三个抽样都实现了K[] getSample(InputFormat inf, Job job) throws IOException, InterruptedException; 方法；通过调用这个方法我们可以返回抽样到的Key数组，除了IntervalSampler 类返回的抽样Key是有序的，其他都无序。获取到采样好的Key数组之后，需要对其进行排序，然后选择好N-1 （N代表Reduce的个数）个分割点，最后将这些Key分割点存储到指定的HDFS文件中，存储的文件格式是SequenceFile，使用如下：

```
TotalOrderPartitioner.setPartitionFile(job.getConfiguration(), new Path(args[2]));
InputSampler.Sampler<Text, Text> sampler = new InputSampler.RandomSampler<>(0.01, 1000, 100);
InputSampler.writePartitionFile(job, sampler);
```

TotalOrderPartitioner

上面通过 InputSampler.writePartitionFile(job, sampler); 存储好了分割点，然后 TotalOrderPartitioner 类会在 setConf
函数中读取这个文件，并根据Key的类型分别创建不同的数据结构：

- 如果 Key 的类型是 BinaryComparable （ BytesWritable 和 Text ），并且 mapreduce.totalorderpartitioner.naturalorder 属性的指是 true ，则会构建trie 树，便于后面的查找；

  > 在计算机科学中，trie，又称前缀树或字典树，是一种有序树，用于保存关联数组，其中的键通常是字符串。与二叉查找树不同，键不是直接保存在节点中，而是由节点在树中的位置决定。一个节点的所有子孙都有相同的前缀，也就是这个节点对应的字符串，而根节点对应空字符串。一般情况下，不是所有的节点都有对应的
  > 值，只有
  > 叶子节点和部分内
  > 部节点所对应的键才有相关的值。（摘自：https://zh.wikipedia.org/wiki/Trie ）

- 其他情况会构建一个 BinarySearchNode，用二分查找

最后程序通过调用 getPartition 函数决定当前Key应该发送到哪个Reduce中：

```
public int getPartition(K key, V value, int numPartitions) {
   return partitions.findPartition(key);
}
```

程序实现

下面是使用 TotalOrderPartitioner 类进行全局排序的完整代码：

```
package com.iteblog.mapreduce.sort;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
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.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.KeyValueTextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.partition.InputSampler;
import org.apache.hadoop.mapreduce.lib.partition.TotalOrderPartitioner;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

import java.io.IOException;

public class TotalSortV3 extends Configured implements Tool {
  static class SimpleMapper extends Mapper<Text, Text, Text, IntWritable> {
    @Override
    protected void map(Text key, Text value,
                       Context context) throws IOException, InterruptedException {
      IntWritable intWritable = new IntWritable(Integer.parseInt(key.toString()));
      context.write(key, intWritable);
    }
  }

  static class SimpleReducer extends Reducer<Text, IntWritable, IntWritable, NullWritable> {
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values,
                          Context context) throws IOException, InterruptedException {
      for (IntWritable value : values)
        context.write(value, NullWritable.get());
    }
  }

  public static class KeyComparator extends WritableComparator {
    protected KeyComparator() {
      super(Text.class, true);
    }

    @Override
    public int compare(WritableComparable w1, WritableComparable w2) {
      int v1 = Integer.parseInt(w1.toString());
      int v2 = Integer.parseInt(w2.toString());

      return v1 - v2;
    }
  }

  @Override
  public int run(String[] args) throws Exception {
```

```
        Configuration conf = getConf();
        Job job = Job.getInstance(conf, "Total Order Sorting");
        job.setJarByClass(TotalSortV3.class);
        job.setInputFormatClass(KeyValueTextInputFormat.class);
        job.setSortComparatorClass(KeyComparator.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.setNumReduceTasks(3);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);
        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(NullWritable.class);

        TotalOrderPartitioner.setPartitionFile(job.getConfiguration(), new Path(args[2]));
        InputSampler.Sampler<Text, Text> sampler = new InputSampler.RandomSampler<>(0.01,
 1000, 100);
        InputSampler.writePartitionFile(job, sampler);

        job.setPartitionerClass(TotalOrderPartitioner.class);
        job.setMapperClass(SimpleMapper.class);
        job.setReducerClass(SimpleReducer.class);

        job.setJobName("iteblog");

        return job.waitForCompletion(true) ? 0 : 1;
    }

    public static void main(String[] args) throws Exception {
        int exitCode = ToolRunner.run(new TotalSortV3(), args);
        System.exit(exitCode);
    }
}
```

运行程序

```
[iteblog@www.iteblog.com /home/iteblog]$ hadoop jar total-sort-0.1.jar com.iteblog.mapredu
ce.sort.TotalSortV3 /user/iteblog/input /user/iteblog/output /user/iteblog/partitions
```

##生成的 Key 分割点

```
[iteblog@www.iteblog.com ~]$ hadoop fs -text /user/iteblog/partitions
10978 (null)
21611 (null)
```

```
[iteblog@www.iteblog.com ~]$ hadoop fs -ls /user/iteblog/output/
Found 4 items
-rw-r--r--   3 iteblog supergroup          0 2017-05-09 16:56 /user/iteblog/output/_SUCCESS
-rw-r--r--   3 iteblog supergroup     335923 2017-05-09 16:56 /user/iteblog/output/part-r-00000
-rw-r--r--   3 iteblog supergroup     388362 2017-05-09 16:56 /user/iteblog/output/part-r-00001
-rw-r--r--   3 iteblog supergroup     407472 2017-05-09 16:56 /user/iteblog/output/part-r-00002

[iteblog@www.iteblog.com ~]$ hadoop fs -text /user/iteblog/output/part-r-00000 | head -n 10
0
0
0
1
1
1
1
1
1

[iteblog@www.iteblog.com ~]$ hadoop fs -text /user/iteblog/output/part-r-00000 | tail -n 10
10976
10976
10976
10977
10977
10977
10977
10977
10977
10977

[iteblog@www.iteblog.com ~]$ hadoop fs -text /user/iteblog/output/part-r-00001 | head -n 10
10978
10978
10978
10978
10978
10978
10978
10978
10978
10978

[iteblog@www.iteblog.com ~]$ hadoop fs -text /user/iteblog/output/part-r-00001 | tail -n 10
21609
21609
```

```
21609
21610
21610
21610
21610
21610
21610
21610

[iteblog@www.iteblog.com ~]$ hadoop fs -text /user/iteblog/output/part-r-00002 | head -n 10
21611
21611
21611
21611
21611
21611
21611
21611
21611
21611

[iteblog@www.iteblog.com ~]$ hadoop fs -text /user/iteblog/output/part-r-00002 | tail -n 10
32766
32766
32766
32766
32767
32767
32767
32767
32767
32767
```

注意事项

1、我们这里使用的 InputFormat 类是 KeyValueTextInputFormat ，而不是 TextInputFormat 。因为采样是对Key进行的，而 TextInputFormat 的 Key 是偏移量，这样的采样结果是无意义的；而如果使用 KeyValueTextInputFormat 作为输入类型，则可以将数据存放在 Key 中，从而得到正确的采样结果。

2、我们 map 输出 Key 的类型是 Text ，这是没办法的，因为 InputSampler.writePartitionFile 函数实现的原因，必须要求 map 输入和输出 Key 的类型一致，否则会出现如下的异常：

```
Exception in thread "main" java.io.IOException: wrong key class: org.apache.hadoop.io.Text is not class org.apache.hadoop.io.LongWritable
  at org.apache.hadoop.io.SequenceFile$RecordCompressWriter.append(SequenceFile.java:1380)
  at com.iteblog.mapreduce.sort.TotalSortV3.writePartitionFile(TotalSortV3.java:106)
  at com.iteblog.mapreduce.sort.TotalSortV3.run(TotalSortV3.java:47)
  at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:70)
  at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:84)
  at com.iteblog.mapreduce.sort.TotalSortV3.main(TotalSortV3.java:73)
  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
  at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
  at java.lang.reflect.Method.invoke(Method.java:606)
  at org.apache.hadoop.util.RunJar.main(RunJar.java:212)
```