**[MapReduce生成HFile入库到HBase](http://www.cnblogs.com/shitouer/archive/2013/02/20/hbase-hfile-bulk-load.html)**

个人小站,正在持续整理中,欢迎访问:[http://shitouer.**cn**](http://shitouer.cn/)

小站博文地址：[MapReduce生成HFile入库到HBase](http://shitouer.cn/2013/02/hbase-hfile-bulk-load/" \o "Permalink to MapReduce生成HFile入库到HBase" \t "_blank)

**一、这种方式有很多的优点：**

1. 如果我们一次性入库hbase巨量数据，处理速度慢不说，还特别占用Region资源， 一个比较高效便捷的方法就是使用 “Bulk Loading”方法，即HBase提供的HFileOutputFormat类。

2. 它是利用hbase的数据信息按照特定格式存储在hdfs内这一原理，直接生成这种hdfs内存储的数据格式文件，然后上传至合适位置，即完成巨量数据快速入库的办法。配合mapreduce完成，高效便捷，而且不占用region资源，增添负载。

**二、这种方式也有很大的限制：**

1. 仅适合初次数据导入，即表内数据为空，或者每次入库表内都无数据的情况。

2. HBase集群与Hadoop集群为同一集群，即HBase所基于的HDFS为生成HFile的MR的集群(额，咋表述~~~)

**三、接下来一个demo，简单介绍整个过程。**

**1. 生成HFile部分**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68 | package zl.hbase.mr;    import java.io.IOException;    import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.Path;  import org.apache.hadoop.hbase.KeyValue;  import org.apache.hadoop.hbase.io.ImmutableBytesWritable;  import org.apache.hadoop.hbase.mapreduce.HFileOutputFormat;  import org.apache.hadoop.hbase.mapreduce.KeyValueSortReducer;  import org.apache.hadoop.hbase.mapreduce.SimpleTotalOrderPartitioner;  import org.apache.hadoop.hbase.util.Bytes;  import org.apache.hadoop.io.LongWritable;  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.output.FileOutputFormat;  import org.apache.hadoop.util.GenericOptionsParser;    import zl.hbase.util.ConnectionUtil;    public class HFileGenerator {        public static class HFileMapper extends              Mapper<LongWritable, Text, ImmutableBytesWritable, KeyValue> {          @Override          protected void map(LongWritable key, Text value, Context context)                  throws IOException, InterruptedException {              String line = value.toString();              String[] items = line.split(",", -1);              ImmutableBytesWritable rowkey = new ImmutableBytesWritable(                      items[0].getBytes());                KeyValue kv = new KeyValue(Bytes.toBytes(items[0]),                      Bytes.toBytes(items[1]), Bytes.toBytes(items[2]),                      System.currentTimeMillis(), Bytes.toBytes(items[3]));              if (null != kv) {                  context.write(rowkey, kv);              }          }      }        public static void main(String[] args) throws IOException,              InterruptedException, ClassNotFoundException {          Configuration conf = new Configuration();          String[] dfsArgs = new GenericOptionsParser(conf, args)                  .getRemainingArgs();            Job job = new Job(conf, "HFile bulk load test");          job.setJarByClass(HFileGenerator.class);            job.setMapperClass(HFileMapper.class);          job.setReducerClass(KeyValueSortReducer.class);            job.setMapOutputKeyClass(ImmutableBytesWritable.class);          job.setMapOutputValueClass(Text.class);            job.setPartitionerClass(SimpleTotalOrderPartitioner.class);            FileInputFormat.addInputPath(job, new Path(dfsArgs[0]));          FileOutputFormat.setOutputPath(job, new Path(dfsArgs[1]));            HFileOutputFormat.configureIncrementalLoad(job,                  ConnectionUtil.getTable());          System.exit(job.waitForCompletion(true) ? 0 : 1);      }  } |

生成HFile程序说明：

①. 最终输出结果，无论是map还是reduce，输出部分key和value的类型必须是： < ImmutableBytesWritable, KeyValue>或者< ImmutableBytesWritable, Put>。

②. 最终输出部分，Value类型是KeyValue 或Put，对应的Sorter分别是KeyValueSortReducer或PutSortReducer。

③. MR例子中job.setOutputFormatClass(HFileOutputFormat.class); HFileOutputFormat只适合一次对单列族组织成HFile文件。

④. MR例子中HFileOutputFormat.configureIncrementalLoad(job, table);自动对job进行配置。SimpleTotalOrderPartitioner是需要先对key进行整体排序，然后划分到每个reduce中，保证每一个reducer中的的key最小最大值区间范围，是不会有交集的。因为入库到HBase的时候，作为一个整体的Region，key是绝对有序的。

⑤. MR例子中最后生成HFile存储在HDFS上，输出路径下的子目录是各个列族。如果对HFile进行入库HBase，相当于move HFile到HBase的Region中，HFile子目录的列族内容没有了。

**2. HFile入库到HBase**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | package zl.hbase.bulkload;    import org.apache.hadoop.fs.Path;  import org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles;  import org.apache.hadoop.util.GenericOptionsParser;    import zl.hbase.util.ConnectionUtil;    public class HFileLoader {        public static void main(String[] args) throws Exception {          String[] dfsArgs = new GenericOptionsParser(                  ConnectionUtil.getConfiguration(), args).getRemainingArgs();          LoadIncrementalHFiles loader = new LoadIncrementalHFiles(                  ConnectionUtil.getConfiguration());          loader.doBulkLoad(new Path(dfsArgs[0]), ConnectionUtil.getTable());      }    } |

通过HBase中 LoadIncrementalHFiles的doBulkLoad方法，对生成的HFile文件入库