MapReduce数据输入中InputFormat类源码解析

在MapReduce作业中的数据输入和输出必须使用到相关的InputFormat和OutputFormat类，来指定输入数据的格式，InputFormat类的功能是为map任务分割输入的数据。

InputFormat类中必须指定Map输入参数Key和Value的数据类型，以及对输入的数据如何进行分割。我们可以在Hadoop源码中看到InputFormat类提供的两个抽象方法：

```
/**
 * User: 过往记忆
 * Date: 15-07-11
 * Time: 下午10:24
 * bolg:
 * 本文地址：/archives/1407
 * 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
 * 过往记忆博客微信公共帐号：iteblog_hadoop
 */

public abstract class InputFormat<K, V> {

  public abstract
    List<InputSplit> getSplits(JobContext context
                ) throws IOException, InterruptedException;

  public abstract
    RecordReader<K,V> createRecordReader(InputSplit split,
```

```
        TaskAttemptContext context
        ) throws IOException, InterruptedException;
}
```

每一个InputFormat类的子类必须实现这两个方法，其中getSplits函数说明数据是怎么分割的，并将分割的数据存放到List中；而createRecordReader函数则根据不同的InputFormat实现创建不同的RecordReader，并读入相关的数据。

对于任何的InputFormat实现最重要的是确定如何来划分数据的文件，划分出来InputSplit将直接影响到map并行的数量，因为对于每一个分片MapReduce将会单独启动一个Map来处理。如果输入文件的划分不合理，那么启动的Map数据将变少，这样会直接影响到MapReduce作业的执行速度。

本文为了方便起见，主要介绍TextInputFormat的相关实现细节。在TextInputFormat类中仅仅实现了InputFormat类的createRecordReader函数，而getSplits的相关实现则由FileInputFormat类实现。FileInputFormat类是比较重要的类，它是所有基于文件InputFormat的父类，并提供了一些通用的方法。下面我们先来看看TextInputFormat类的关键实现代码：

```
public class TextInputFormat extends FileInputFormat<LongWritable, Text>
.....

 @Override
public RecordReader<LongWritable, Text>
   createRecordReader(InputSplit split,
                      TaskAttemptContext context) {
   String delimiter = context.getConfiguration().get(
      "textinputformat.record.delimiter");
   byte[] recordDelimiterBytes = null;
   if (null != delimiter)
     recordDelimiterBytes = delimiter.getBytes(Charsets.UTF_8);
   return new LineRecordReader(recordDelimiterBytes);
}
```

从上面的代码可以看到TextInputFormat类是FileInputFormat类的子类。TextInputFormat类是Key类型是LongWritable，其实它就是输入文本的偏移量；Value类型是Text，这就是文件的行内容。接下来比较重要的是createRecordReader函数的实现，首先会根据textinputformat.record.delimiter参数判断输入文件的行分隔符，默认情况下是₩n。然后根据行的分割符创建了一个LineRecordReader。关于LineRecordReader在后面的文章中再介绍。

TextInputFormat类中还有一个isSplitable函数的实现，它是用来判断输入的文件是否可分割

，实现如下：

```
/**
 * User: 过往记忆
 * Date: 15-07-11
 * Time: 下午10:24
 * bolg:
 * 本文地址：/archives/1407
 * 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
 * 过往记忆博客微信公共帐号：iteblog_hadoop
 */

@Override
protected boolean isSplitable(JobContext context, Path file) {
   final CompressionCodec codec =
     new CompressionCodecFactory(context.getConfiguration()).getCodec(file);
   if (null == codec) {
     return true;
   }
   return codec instanceof SplittableCompressionCodec;
}
```

如果输入文件不是压缩形式的，直接返回可分割(true)；如果输入文件是压缩的，那么判断这个压缩类是否是SplittableCompressionCodec接口的实现类（Hadoop内置的SplittableCompressionCodec类实现只有BZip2Codec）。

接下来我们再来看看FileInputFormat类中getSplits函数的实现，代码如下：

```
minSize = Math.max(getFormatMinSplitSize(), getMinSplitSize(job));
   long maxSize = getMaxSplitSize(job);

   // generate splits
   List<InputSplit> splits = new ArrayList<InputSplit>();
   List<FileStatus> files = listStatus(job);
   for (FileStatus file: files) {
     Path path = file.getPath();
     long length = file.getLen();
     if (length != 0) {
       BlockLocation[] blkLocations;
       if (file instanceof LocatedFileStatus) {
         blkLocations = ((LocatedFileStatus) file).getBlockLocations();
       } else {
         FileSystem fs = path.getFileSystem(job.getConfiguration());
```

```
        blkLocations = fs.getFileBlockLocations(file, 0, length);
      }
      if (isSplitable(job, path)) {
        long blockSize = file.getBlockSize();
        long splitSize = computeSplitSize(blockSize, minSize, maxSize);

        long bytesRemaining = length;
        while (((double) bytesRemaining)/splitSize > SPLIT_SLOP) {
          int blkIndex = getBlockIndex(blkLocations, length-bytesRemaining);
          splits.add(makeSplit(path, length-bytesRemaining, splitSize,
                      blkLocations[blkIndex].getHosts(),
                      blkLocations[blkIndex].getCachedHosts()));
          bytesRemaining -= splitSize;
        }

        if (bytesRemaining != 0) {
          int blkIndex = getBlockIndex(blkLocations, length-bytesRemaining);
          splits.add(makeSplit(path, length-bytesRemaining, bytesRemaining,
                      blkLocations[blkIndex].getHosts(),
                      blkLocations[blkIndex].getCachedHosts()));
        }
      } else { // not splitable
        splits.add(makeSplit(path, 0, length, blkLocations[0].getHosts(),
                    blkLocations[0].getCachedHosts()));
      }
    } else {
      //Create empty hosts array for zero length files
      splits.add(makeSplit(path, 0, length, new String[0]));
    }
  }
  // Save the number of input files for metrics/loadgen
  job.getConfiguration().setLong(NUM_INPUT_FILES, files.size());
  sw.stop();
  if (LOG.isDebugEnabled()) {
    LOG.debug("Total # of splits generated by getSplits: " + splits.size()
        + ", TimeTaken: " + sw.elapsedMillis());
  }
  return splits;
}
```

listStatus方法将获取MapReduce作业需要输入的所有文件。然后根据isSplitable函数来获取所有问及那的块，并存储到BlockLocation数组中。如果文件是可分的，那么根据long splitSize = computeSplitSize(blockSize, minSize,

maxSize);来计算每个块的大小，最后通过makeSplit函数来创建分块，并存放到List splits中。