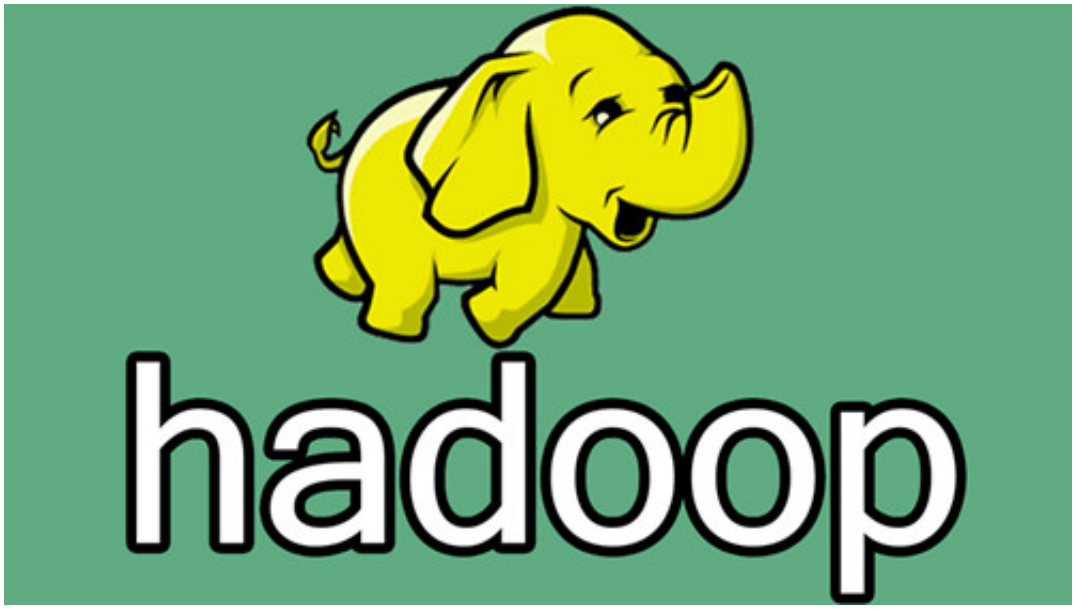
使用CombineFileInputFormat来优化Hadoop小文件

我们都知道，HDFS设计是用来存储海量数据的，特别适合存储TB、PB量级别的数据。但是随着时间的推移，HDFS上可能会存在大量的小文件，这里说的小文件指的是文件大小远远小于一个HDFS块（128MB）的大小；HDFS上存在大量的小文件至少会产生以下影响：

- 消耗NameNode大量的内存
- 延长MapReduce作业的总运行时间



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

本文将介绍如何在MapReduce作业层面上将大量的小文件合并，以此减少运行作业的Map Task的数量；关于如何在HDFS上合并这些小文件，请参见《Hadoop小文件优化》。

Hadoop内置提供了一个 CombineFileInputFormat
类来专门处理小文件，其核心思想是：根据一定的规则，将HDFS上多个小文件合并到一个InputSplit中，然后会启用一个Map来处理这里面的文件，以此减少MR整体作业的运行时间。CombineFileInputFormat类继承自FileInputFormat，主要重写了List getSplits(JobContext job)方法；这个方法会根据数据的分布，mapreduce.input.fileinputformat.split.minsize.per.node、mapreduce.input.fileinputformat.split.minsize.per.rack以及mapreduce.input.fileinputformat.split.maxsize 参数的设置来合并小文件，并生成List。其中mapreduce.input.fileinputformat.split.maxsize参数至关重要：

- 如果用户没有设置这个参数（默认就是没设置），那么同一个机架上的所有小文件将组成一个InputSplit，最终由一个Map Task来处理；
- 如果用户设置了这个参数，那么同一个节点（node）上的文件将会组成一个InputSplit。

同一个 InputSplit 包含了多个HDFS块文件，这些信息存储在 CombineFileSplit 类中，它主要包含以下信息：

```
private Path[] paths;
private long[] startoffset;
private long[] lengths;
private String[] locations;
private long totLength;
```

从上面的定义可以看出，CombineFileSplit类包含了每个块文件的路径、起始偏移量、相对于原始偏移量的大小以及这个文件的存储节点，因为一个CombineFileSplit包含了多个小文件，所以需要使用数组来存储这些信息。

CombineFileInputFormat是抽象类，如果我们要使用它，需要实现createRecordReader方法，告诉MR程序如何读取组合的InputSplit。内置实现了两种用于解析组合InputSplit的类：org.apache.hadoop.mapreduce.lib.input.CombineTextInputFormat 和 org.apache.hadoop.mapreduce.lib.input.CombineSequenceFileInputFormat，我们可以把这两个类理解是 TextInputFormat 和 SequenceFileInputFormat。为了简便，这里主要来介绍CombineTextInputFormat。

在 CombineTextInputFormat 中创建了 org.apache.hadoop.mapreduce.lib.input.CombineFileRecordReader，具体如何解析CombineFileSplit中的文件主要在CombineFileRecordReader中实现。

CombineFileRecordReader类中其实封装了TextInputFormat的RecordReader，并对CombineFileSplit中的多个文件循环遍历并读取其中的内容，初始化每个文件的RecordReader主要在initNextRecordReader里面实现；每次初始化新文件的RecordReader都会设置mapreduce.map.input.file、mapreduce.map.input.length以及mapreduce.map.input.start参数，这样我们可以在Map程序里面获取到当前正在处理哪个文件。

现在我们就来看看如何使用CombineTextInputFormat类，如下：

```
package com.iteblog.hadoop.examples;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.InputSplit;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.MRJobConfig;
```

```
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.CombineTextInputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

import java.io.IOException;
import java.util.List;

////////////////////////////////////////////////////////////////////
 User: 过往记忆
 Date: 2017-04-25
 Time: 22:59
 bolg: https://www.iteblog.com
 本文地址：https://www.iteblog.com/archives/2139
 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 class HadoopTest extends Configured implements Tool {
    private static final Log LOG = LogFactory.getLog(HadoopTest.class);
    private static final long ONE_MB = 1024 * 1024L;

    static class TextFileMapper extends Mapper<longwritable , Text, Text, Text> {

        @Override
        protected void map(LongWritable key, Text value, Context context)
                throws IOException, InterruptedException {
            Configuration configuration = context.getConfiguration();
            LOG.warn("########################" + configuration.get(MRJobConfig.MAP_IN
PUT_FILE));
            Text filenameKey = new Text(configuration.get(MRJobConfig.MAP_INPUT_FILE));
            context.write(filenameKey, value);
        }
    }

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

    @Override
    public int run(String[] args) throws Exception {
        Configuration conf = new Configuration(getConf());
        conf.set("mapreduce.input.fileinputformat.split.maxsize", ONE_MB * 32);
```

```
        Job job = Job.getInstance(conf);
        FileInputFormat.setInputPaths(job, args[0]);
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.setJarByClass(HadoopTest.class);
        job.setInputFormatClass(CombineTextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        job.setMapperClass(TextFileMapper.class);
        return job.waitForCompletion(true) ? 0 : 1;
    }
}
```

上面的程序很简单，其实就是将HDFS上多个小文件合并到大文件中，并再每行存储了这行数据的文件路径。程序运行的结果如下：

```
hdfs://iteblogcluster/user/iteblog/user/dt=2017-04-25/1.txt    iteblog
hdfs://iteblogcluster/user/iteblog/user/dt=2017-04-25/2.txt    zhangsan
hdfs://iteblogcluster/user/iteblog/user/dt=2017-04-25/3.txt    lisi
```

可以看到最终结果将三个文件里面的内容合并到一个文件中。注意体会mapreduce.input.fileinputformat.split.maxsize参数的设置，大家可以不设置这个参数并且和设置这个参数运行情况对比，观察Map Task的个数变化。

本博客文章除特别声明，全部都是原创！
转载本文请加上：转载自过往记忆（https://www.iteblog.com/）
本文链接:【】（）