Apache Spark 2.0 在作业完成时却花费很长时间结束

现象

大家在使用 Apache Spark 2.x 的时候可能会遇到这种现象：虽然我们的 Spark Jobs 已经全部完成了，但是我们的程序却还在执行。比如我们使用 Spark SQL 去执行一些 SQL，这个 SQL 在最后生成了大量的文件。然后我们可以看到，这个 SQL 所有的 Spark Jobs 其实已经运行完成了，但是这个查询语句还在运行。通过日志，我们可以看到 driver 节点正在一个一个地将 tasks 生成的文件移动到最终表的目录下面，当我们作业生成的文件很多的情况下，就很容易产生这种现象。本文将给大家介绍一种方法来解决这个问题。

为什么会造成这个现象

Spark 2.x 用到了 Hadoop 2.x，其将生成的文件保存到 HDFS 的时候，最后会调用了 saveAsHadoopFile，而这个函数在里面用到了 FileOutputCommitter，如下：

```
def saveAsHadoopFile(
    path: String,
    keyClass: Class[_],
    valueClass: Class[_],
    outputFormatClass: Class[_ <: OutputFormat[_, _]],
    conf: JobConf = new JobConf(self.context.hadoopConfiguration),
```

```
    codec: Option[Class[_ <: CompressionCodec]] = None): Unit = self.withScope {

  ........

  // Use configured output committer if already set
  if (conf.getOutputCommitter == null) {
    hadoopConf.setOutputCommitter(classOf[FileOutputCommitter])
  }

  ........

}
```

问题就出在了 Hadoop 2.x 的 FileOutputCommitter 实现，FileOutputCommitter 里面有两个值得注意的方法：commitTask 和 commitJob。在 Hadoop 2.x 的FileOutputCommitter 实现里面，mapreduce.fileoutputcommitter.algorithm.version 参数控制着 commitTask 和 commitJob 的工作方式。具体代码如下（为了说明方便，我去掉了无关紧要的语句，完整代码可以参见 FileOutputCommitter.java）：

```
public void commitTask(TaskAttemptContext context, Path taskAttemptPath)
  throws IOException {

   ........

  if (taskAttemptDirStatus != null) {
    if (algorithmVersion == 1) {
      Path committedTaskPath = getCommittedTaskPath(context);
      if (fs.exists(committedTaskPath)) {
        if (!fs.delete(committedTaskPath, true)) {
          throw new IOException("Could not delete " + committedTaskPath);
        }
      }
      if (!fs.rename(taskAttemptPath, committedTaskPath)) {
        throw new IOException("Could not rename " + taskAttemptPath + " to "
            + committedTaskPath);
      }
      LOG.info("Saved output of task '" + attemptId + "' to " +
          committedTaskPath);
    } else {
      // directly merge everything from taskAttemptPath to output directory
      mergePaths(fs, taskAttemptDirStatus, outputPath);
      LOG.info("Saved output of task '" + attemptId + "' to " +
```

```
          outputPath);
      }
    } else {
      LOG.warn("No Output found for " + attemptId);
    }
  } else {
    LOG.warn("Output Path is null in commitTask()");
  }
}

public void commitJob(JobContext context) throws IOException {
      ........
      jobCommitNotFinished = false;
     ........
}

protected void commitJobInternal(JobContext context) throws IOException {
    ........
    if (algorithmVersion == 1) {
      for (FileStatus stat: getAllCommittedTaskPaths(context)) {
        mergePaths(fs, stat, finalOutput);
      }
    }
   ........
}
```

大家可以看到 commitTask 方法里面，有个条件判断 algorithmVersion == 1，这个就是 mapreduce.fileoutputcommitter.algorithm.version 参数的值，默认为1；如果这个参数为1，那么在 Task 完成的时候，是将 Task 临时生成的数据移到 task 的对应目录下，然后再在 commitJob 的时候移到最终作业输出目录，而这个参数,在 Hadoop 2.x 的默认值就是 1！这也就是为什么我们看到 job 完成了，但是程序还在移动数据，从而导致整个作业尚未完成，而且最后是由 Spark 的 Driver 执行 commitJob 函数的，所以执行的慢也是有到底的。

而我们可以看到，如果我们将 mapreduce.fileoutputcommitter.algorithm.version 参数的值设置为 2，那么在 commitTask 执行的时候，就会调用 mergePaths 方法直接将 Task 生成的数据从 Task 临时目录移动到程序最后生成目录。而在执行 commitJob 的时候，直接就不用移动数据了，自然会比默认的值要快很多。

注意，其实在 Hadoop 2.7.0 之前版本，我们可以将 mapreduce.fileoutputcommitter.algorithm.version 参数设置为非1的值就可以实现这个目的，因为程序里面并没有限制这个值一定为2,。不过到了

Hadoop 2.7.0，mapreduce.fileoutputcommitter.algorithm.version 参数的值必须为1或2，具体参见 MAPREDUCE-4815。

怎么在 Spark 里面设置这个参数

问题已经找到了，我们可以在程序里面解决这个问题。有以下几种方法：

- 直接在 conf/spark-defaults.conf 里面设置 spark.hadoop.mapreduce.fileoutputcommitter.algorithm.version 2，这个是全局影响的。
- 直接在 Spark 程序里面设置，spark.conf.set("mapreduce.fileoutputcommitter.algorithm.version", "2")，这个是作业级别的。
- 如果你是使用 Dataset API 写数据到 HDFS，那么你可以这么设置 dataset.write.option("mapreduce.fileoutputcommitter.algorithm.version", "2")。

不过如果你的 Hadoop 版本为 3.x，mapreduce.fileoutputcommitter.algorithm.version 参数的默认值已经设置为2了，具体参见 MAPREDUCE-6336 和 MAPREDUCE-6406。

因为这个参数对性能有一些影响，所以到了 Spark 2.2.0，这个参数已经记录在 Spark 配置文档里面了 configuration.html，具体参见 SPARK-20107。