Map-Reduce源码分析

自定义对应Job类模板的代码作用为

- 声明Configuration对象， 并通过该对象设置运行模式为本地模式，若想运行在Windows环境下，需要设置为本地模式
- 通过Configuration对象以及Job类的getInstance()方法获得job对象，并设置job对象的作业主类、作业名称、Reduce数量、 Map写出的Key 与Value的类型和Map与Reduce的处理类,而且通过FileIInputFormat类设置作业从哪里读取数据以及作业计算完的数据写到哪里，最后调用job的waitForCompletion()方法作业提交到集群

Split部分

Split部分主要用到三个类及其类中的对应方法

- Job类
  - Job类继承自JobContextImpl类,并且实现了**JobContext**接口
  - Job类中涉及到的属性
    - JobState类型的state属性，JobState是枚举，共有两种实例，一种为DEFINE，另一种为RUNNING，默认为DEFINE
    - Cluster类型的cluster属性，代表集群对象
    - JobStatus类型的status属性
    - UserGroupInformation类型的ugi属性，代表用户与组的信息对象
  - Job类中使用到的方法
    - waitForCompletion()方法
      - 方法参数为boolean，返回值为boolean
      - 方法流程
        - 判断Job对象的状态，若状态为DEFINE，便执行**submit()方法(该方法为重点方法，后续会展开来讲)**
        - 判断参数
          - 若参数为true，执行monitorAndPrintJob()方法，即监听并打印工作
          - 若参数为false，则通过cluster对象的参数从客户端获取完成轮询间隔，并根据该轮询间隔询问作业状态
        - 全部执行完成，便返回true
    - submit()方法
      - 该方法无参无返回值
      - 方法流程
        - 通过ensureState()方法,，确认作业的状态是否为DEFINE
        - 通过setUserNewAPI()方法,，设置使用新API
        - 通过connec()方法，实例化cluster属性，即连接集群
        - 根据集群的文件系统(Cluster的属性)与客户端(Cluster的属性)获得JobSubmitter对象
        - 调用ugi的方法，并最终调用JobSubmitter对象的**submitJobInternal方法(该方法为重点方法，后续会展开来讲)**，用其返回值修改status属性
        - 修改state属性为RUNNING
        - 打印日志
- JobSubmitter类

- 没有需要注意的继承或者实现关系以及属性
- submitJobInternal()方法
  - 方法参数为Job类型对象与Cluster类型对象；方法返回值JobStatus
  - 方法流程
    - 通过checkSpecs()方法，检查作业输出规格
    - 通过job对象实例化Configuration对象，即获取配置
    - 通过addMRFrameworkToDistributedCache()方法，将Configuration对象作为参数，若有需要则将作业添加至分布式缓存
    - 根据Cluster对象以及Configuration对象获取代表与工作相关的工作路径的Path对象jobStagingArea
    - 获取服务器地址
    - 生成Jobld对象
    - 将Jobld添加至Job对象
    - 根据jobStagingArea对象与Jobld对象获取代表加载作业相关所需要的路径的Path对象submitJobDir
    - 声明值为空的JobStatus对象
    - 进行权限的设置或者校验
    - 调用copyAndConfigureFiles()方法，完成计算向数据靠拢
    - 调用LOG属性的debug方法，来记录日志
    - 声明maps变量，代表需要使用的Mapper数量，将job对象与submitJobDir对象作为参数调用**writeSplits()方法(该方法为重点方法，后续会展开来讲)**，获得切片数量，并将其赋予maps变量，即Mapper数量与切片数量一致
    - 跳过部分代码
    - 调用submitClient属性的submitJob()方法来完成真正的提交，且提交到YARN集群，此处需注意**提交方式有两种，一种为提交到本地，一种为提交到Yarn集群**，因为submitClient属性为ClientProtocol类型，而ClientProtocol为接口，其实现类有两个，分别为代表本地的LoaclJobRunner与代表Yarn集群的YARNRunner，所以submitJob()方法有两种实现，即提交方式有两种，一种为提交到本地，一种为提交到Yarn集群
- wirteSplit()方法
  - 方法参数为JobContext类型对象与Path类型对象；返回值为int类型，代表Mapper的个数
    - JobContext为接口，使用其实现类Job类，代表作业
    - Path类型对象代表加载作业相关所需要的路径
  - 方法流程
    - 声明变量maps
    - 判断用户使用新的API还是旧的API
      - 使用新的便调用**writeNewSplits()方法(该方法为重点方法，后续会展开来讲)**，并将其返回值赋maps变量
      - 使用旧的便调用writeOldSplits()方法，并将其返回值赋予maps变量
- writeNewSplits()方法
  - 参数与返回值和wirteSplit()方法相同,不做赘述
  - 方法流程
    - 根据job对象获取Configuration对象，即获取配置
    - 根据job对象，并使用反射来初始化InputFormat抽象类(此处会用到FileInputFormat类)，其引用为input
    - 调用input引用的**getSplits()方法(该方法为重点方法，且实际调用的是FileInputFormat类的getSplits()方法,所以在FileInputFormat()类中展开来**

讲)，获得**InputSplit类型**的切片列表
        - 将列表转为数组
        - 将数组排序，并使用SplitComparator()比较器进行排序，该比较器的比较规则为根据文件长度(文件长度代表了文件的大小)进行比较，文件长度越长，对应数组中的位置越靠前，即根据文件长度倒序排序
        - 返回数组的长度，来确保**一个分片split对应一个Mapper**
- FileInputFormat类
    - getSplits()方法
        - 参数为JobContext类型对象，返回值为InputSplit列表
        - 方法流程
            - 声明并实例化StopWatch对象，sw，即计时对象
            - 声明long类型的变量minSize，且其取值逻辑为若程序员设置了，便用自定义的,若没有设置，则其值默认为1
            - 声明long类型的变量maxSize，且其取值逻辑为若程序员设置了，便用自定义的，若没有设置，则其取值默认为long类型最大值
            - 初始化InputSplit列表，其引用为splits，即存放切片的容器
            - 初始化FileStatus列表，其引用为files
            - 根据FileStatus列表进行foreach循环
                - 进行是否是文件夹判断，若为文件夹便跳过
                - 获取文件路径，赋予Path类的path对象
                - 获取文件总长度，赋予long类型的length变量
                - 切分操作开始
                - 若文件总长度不为0，进行后续切分操作
                - 声明BlockLocation数组blkLocations，代表数据块的位置信息
                - 判断FileStatus是否为本地文件系统，若是本地文件系统，直接通过文件信息对数据块进行处理
                - 若不是本地文件系统，即使用的是hdfs分布式系统，则从分布式文件系统获取块信息
                - 判断是否能进行分割
                    - 若能分割
                        - 声明long类型blockSize变量，代表数据块的大小，默认128M
                        - 声明long类型splitSize变量，代表切片大小，进行逻辑处理。**调节切片大小的核心为：若想调大切片，将minSize调大，若想调小切片，将maxSize调小。**切片的默认大小为数据块的大小即128M
                        - 声明long类型bytesRemaining变量，代表文件剩余长度，由于目前为止还没有对文件进行任何切分操作，所以先将length赋予bytesRemaining变量
                        - 通过while循环，对文件进行切分操作，while循环的条件为**文件剩余长度是否为切分大小的1.1倍**
                            - 切分一个片，并将该切片添加至InputSplit列表，即放入splits引用中
                            - InputSplit列表中存放的本应是InputSplit对象，但是InputSplit是一个抽象类， 无法直接创建对象，所以InputSplit列表中存放的实际为FileSplit对象(FileSplit为

InputSplit抽象类的子类)。且InputSplit列表中并不是直接存放FileSplit对象，而是通过调用makeSplit()方法来完成，此方法需要五个参数，分别对应文件路径、起始位置、切片长度、块来源的主机信息与块来源的缓存信息，由此又可推断：切片操作并不是物理切片，而是进行了逻辑切片

                - 对剩余文件长度进行重新赋值
            - while循环结束后，对剩余文件长度进行判断，若不为0，则将剩余字节切为一个片，并放入InputSplit列表
        - 若不能分割(**有些文件是无法切分的，因为有一些特殊格式的文件连Block都无法切分**)
            - 通过日志来记录不能切分的数据块
            - 将整个块作为一个片去处理，即将整个块存入到InputSplit列表中
    - 若为空文件，便将一个空切片放入到InputSplit列表，放入InputSplit列表的并不是一个空对象，而是一个代表空切片的FileInput对象(个人理解)
- 结束计时，并将InputSplit列表返回