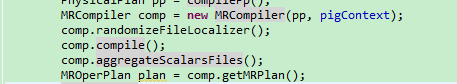
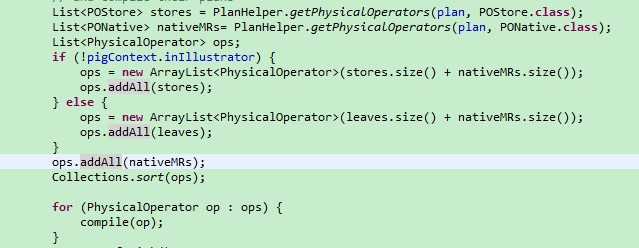
Physical plan to MROperatorPlan

转换的流程

1. 对于physical plan到mroperator plan的转换的入口类是MRCompiler，具体demo如下

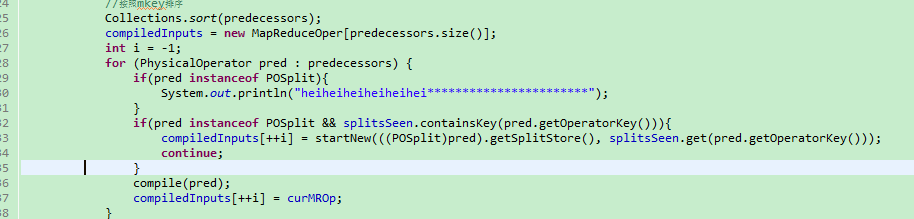


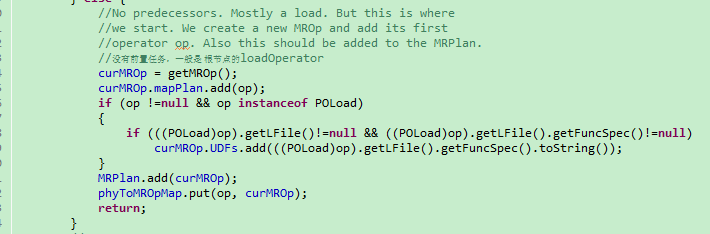
1. 具体的转换从最末端的physicalplan(一般是OPStore，且就一个)开始编译成不同的MRPlan操作树(也是一个DAG)



1. Compile方法

这个方法首先递归将自己的pre operator plan(一般都是一直到最初的load operator)转换成MRPlan





然后再调用自己的op.visit(this)(这里面的this就是MRCompiler，其自生也是一个访问者)方法，将自生的operator添加/新增到MRPlan中 。

这里面的会涉及到curMROp的一个变量，是指其前置翻译成的MROperator，在翻译自己时候会看自己的操作是否可以直接加到这个MR的map或者reduce中（一般的filter、for-each等都是可以的），而不用自己再新起一个MRJob 。

phyToMROpMap.put(op, curMROp);，这个变量保存的是每个physical op 到 mr opde 映射关系 。

1. 待补充

具体visit算法规则如下

1. 分裂操作:visitSplit(POSplit op)

在上一个MROperator中添加一个store到tmp file的操作，然后结束这个MROperator，新起一个后继的MROperator，将OPLoad（文件是刚才的那个文件）放入到mapplan中 。

1. 原始加载操作:visitLoad(POLoad op)

可以直接放到pre的MR job中的

1. nativa操作:visitNative(PONative op)
2. 具体的存储操作:visitStore(POStore op)

Store是相同的，直接插入到pre的MR中即可 。

1. 过滤操作: visitFilter(POFilter op)

Filter同上

1. nested cross: visitCross(POCross op)

同上

1. stream的操作：visitStream(POStream op)

同上

1. limit操作: visitLimit(POLimit op)

在map端和reduce端都插入limit操作，若是上一个MROP只有map，则将reduce 并发设置为1，并在reduce插入个limit

1. visitLocalRearrange(POLocalRearrange op)

插入到map的叶子节点

或者插入到reduce的叶子节点，然后再插入一个store tmp file操作，然后再新起一个空的MROP接着这个 。

1. visitCollectedGroup(POCollectedGroup op)
2. for-each : visitPOForEach(POForEach op)

可以随意插入到上一个的Map/Reduce的操作中

1. 全局排序:visitGlobalRearrange(POGlobalRearrange op)
2. Join : visitPackage(POPackage op)
3. visitUnion(POUnion op)
4. visitFRJoin(POFRJoin op)
5. 合并join:visitMergeJoin(POMergeJoin joinOp)
6. Disitinct:visitDistinct(PODistinct op)
7. 倾斜join: visitSkewedJoin(POSkewedJoin op)
8. Sort：visitSort(POSort op)

结束上一个MROperator，将其文件输出到文件系统中，启动一轮新的mapreduce统计文件的statistic ，然后在启动一轮mr来进行sort

1. Counter: visitCounter(POCounter op)
2. visitRank(PORank op)