Pig的load以及Store调研文档

Pig通过Load读数据的接口以及流程

loadFun

relativeToAbsolutePath(String location, Path curDir)

setLocation(String location, Job job)

这个会在调用完setPartitionFilter后再调用一次来更新这个数据值。这个方法某种意义上起到了个设置conf的作用，其会再setpartfilter、getinputformat、setudfcontextsignature等地方伴随使用

getInputFormat()

获得inputformat

getLoadCaster()

定位比较模糊，是获得一个从byte[]转换成具体的value的转换器，但是需要自己的代码中调用，默认的话是utft转换的那种，不是front/back必调函数 。

prepareToRead(RecordReader reader, PigSplit split)

getNext()

读取的时候返回一个个的tuple

setUDFContextSignature(String signature)

LoadMetadata

getSchema(String location, Job job)

在front和back端都会调用，这个方法每次都是返回本次load的所有列，不应该改收到投影操作的pushProjection的方法影响

getStatistics(String location, Job job)

获得统计信息

getPartitionKeys(String location, Job job)

setPartitionFilter(Expression partitionFilter)

LoadPushDown

List<OperatorSet> getFeatures()

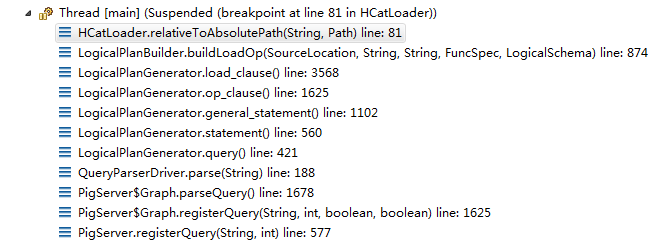
目前只支持投影操作的列裁剪，暂不支持谓词过滤下推的操作

pushProjection(RequiredFieldList requiredFieldList)

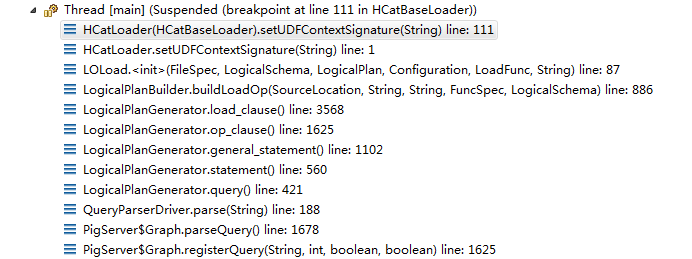
列裁剪的下推操作，在逻辑执行计划中，将需要的列设置过来的

Debug load整个一套运行流程

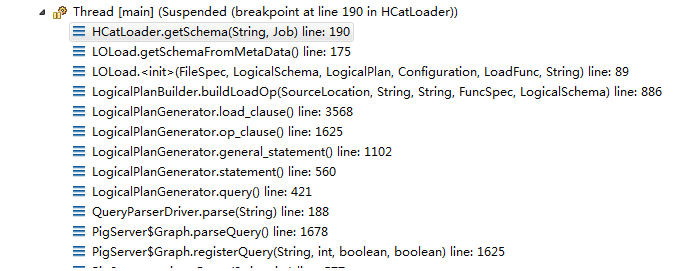
1. relativeToAbsolutePath



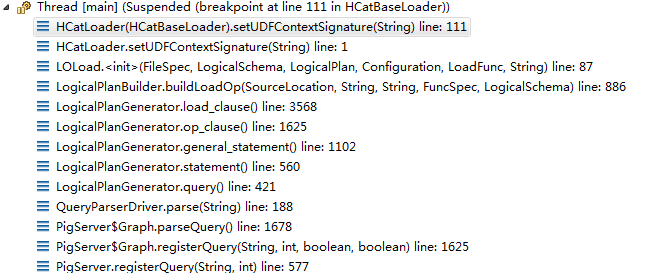
1. setUDFContextSignature（W\_1-0）



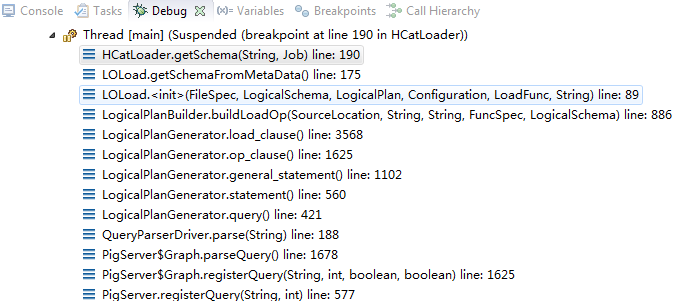
1. getSchema(id=41)



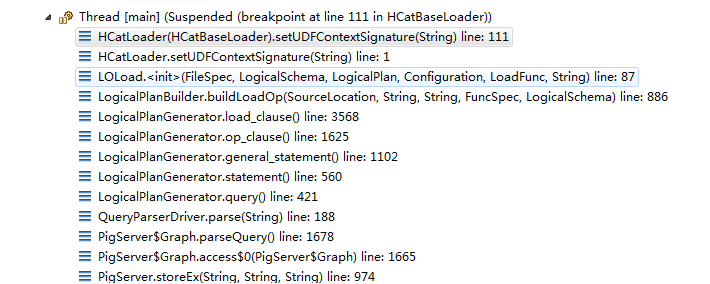
1. setUDFContextSignature（W\_1-1）



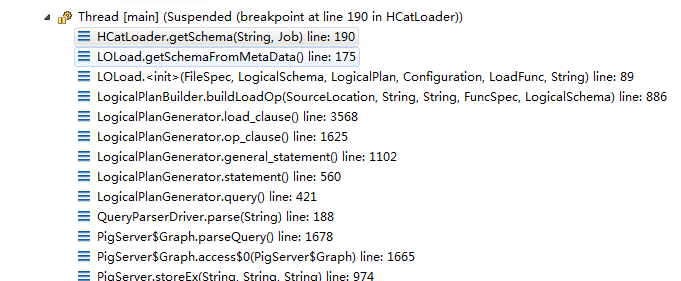
1. getSchema(id=76)



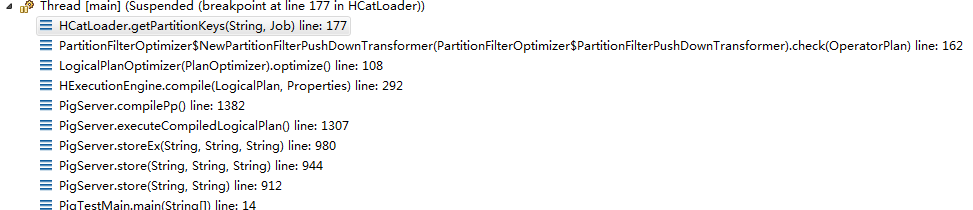
1. setUDFContextSignature(W\_1-2)



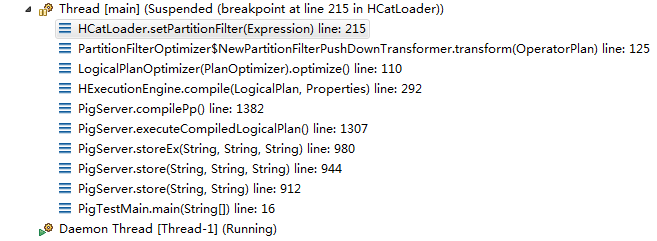
1. getSchema(id=87)



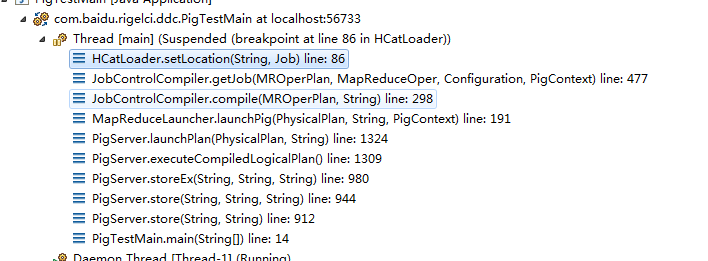
1. getPartitionKeys(id=99)



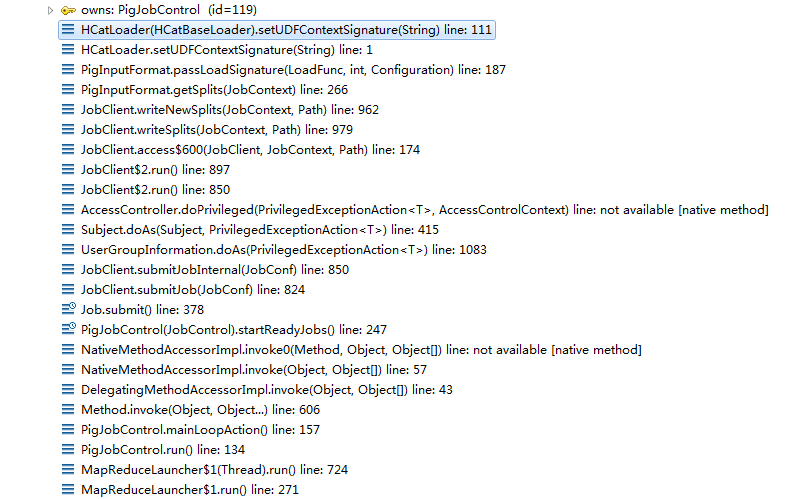
1. setPartitionFilter(pdate == '2014-03-09'),optimizer时候调用



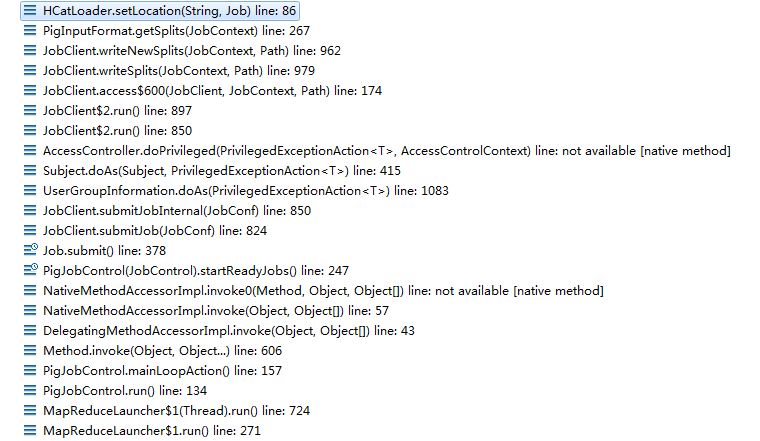
1. setLocation(id=107)



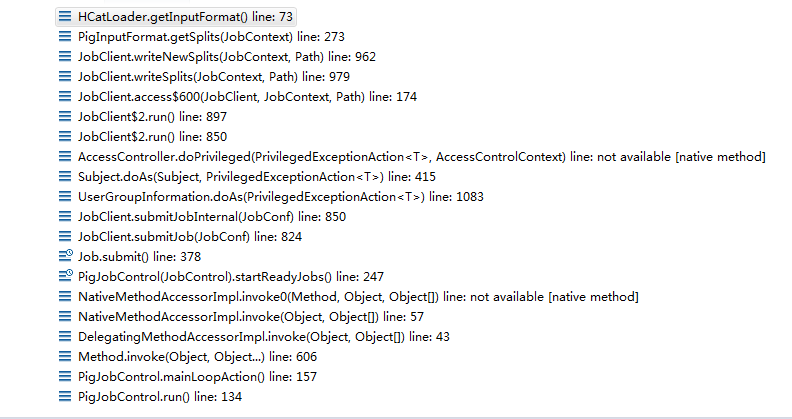
1. setUDFContextSignature(W\_1-2)



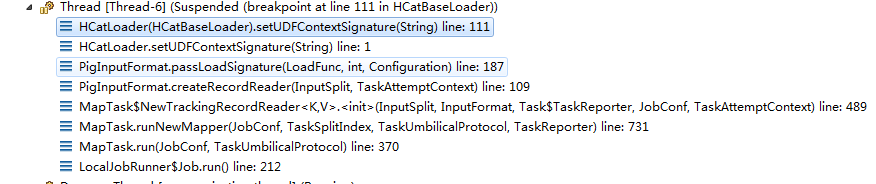
1. setLocation



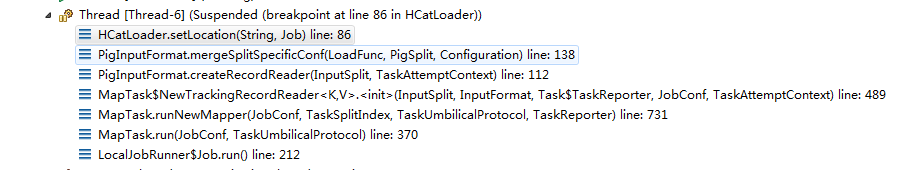
1. getInputFormat



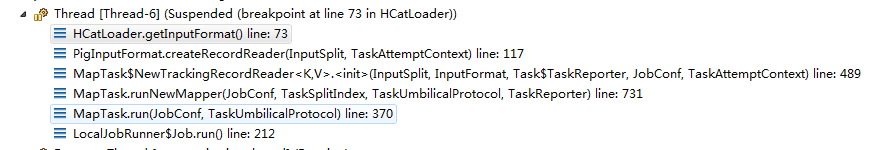
1. setUDFContextSignature(W\_1-2)



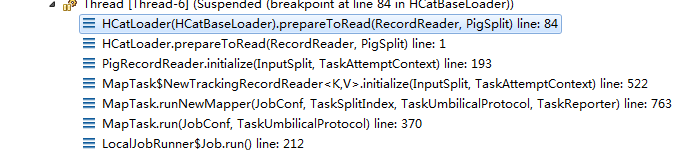
1. setLocation



1. getInputFormat



1. setUDFContextSignature(W\_1-2)
2. prepareToRead



1. 待补充

Ceshi

Stroe流程的方法与作用

StoreFunc

getOutputFormat()

获得outputFormat，多次被调用

setStoreLocation (String location, Job job)

设置输出路径，也是多次被调用，相当配置job.conf的作用

checkSchema(ResourceSchema s)

上层传入的带写入数据schema信息，在这里讲schema写入udfContext中，以便后来用

prepareToWrite(RecordWriter writer)

初始化writer，由于writer信息太少所以只能写之前，初始化其它的一些对象

setStoreFuncUDFContextSignature(String signature)

设置signature

cleanupOnFailure(String location, Job job)

cleanupOnSuccess(String location, Job job)

以上两个方法都是在整个结束的时候，成功/失败的时候调用的close方法，但是对于writer等的close清理，不必要在这里调用，这已经交由mr框架来处理。

interface StoreMetadata

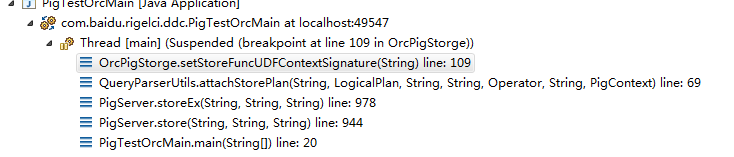
storeStatistics(ResourceStatistics stats, String location, Job job)

storeSchema(ResourceSchema schema, String location, Job job)

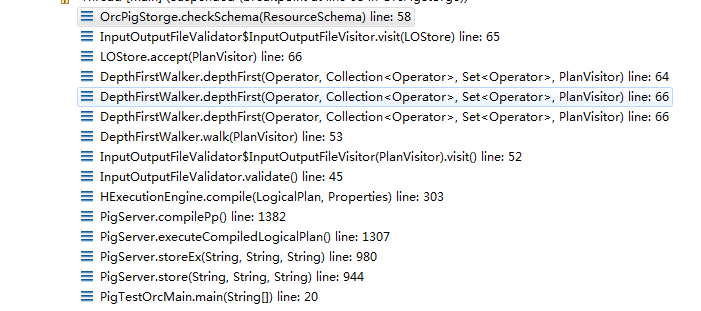
设置元数据的两个方法，在这里用途不是很大

Debug 一次Store的运行流程

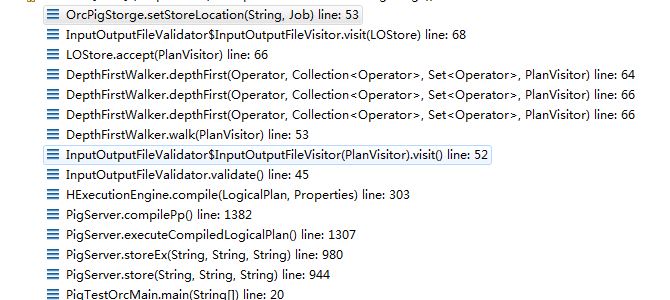
1. setStoreFuncUDFContextSignature（H\_1-3）、



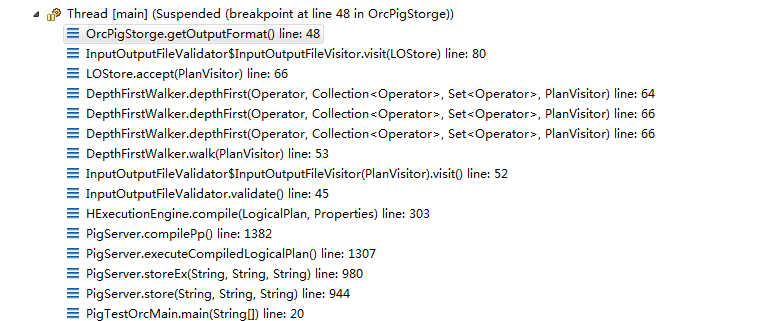
1. checkSchema(将待存储的schema存储下来)



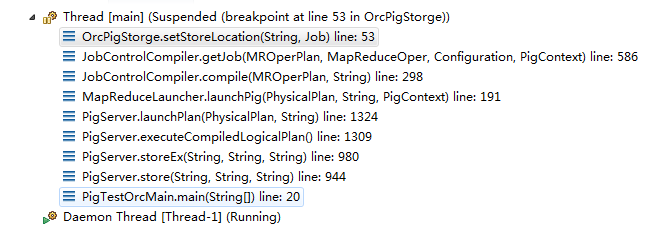
1. setStoreLocation



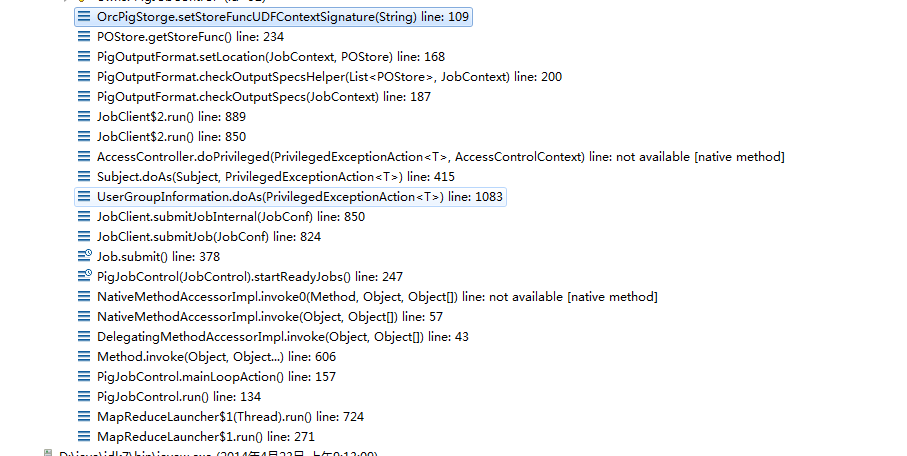
1. getOutputFormat



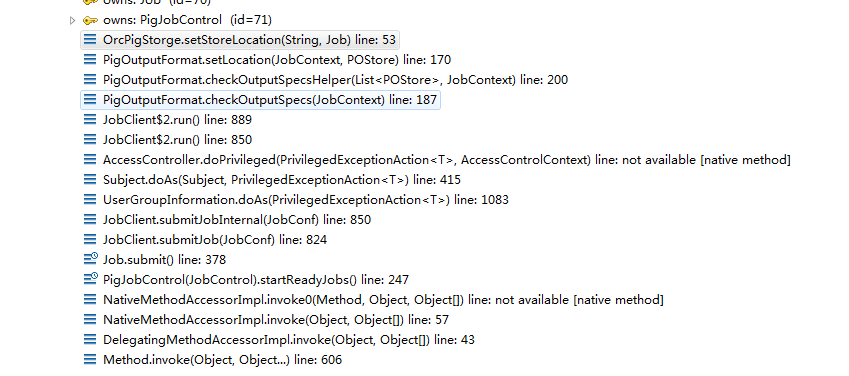
1. setStoreLocation



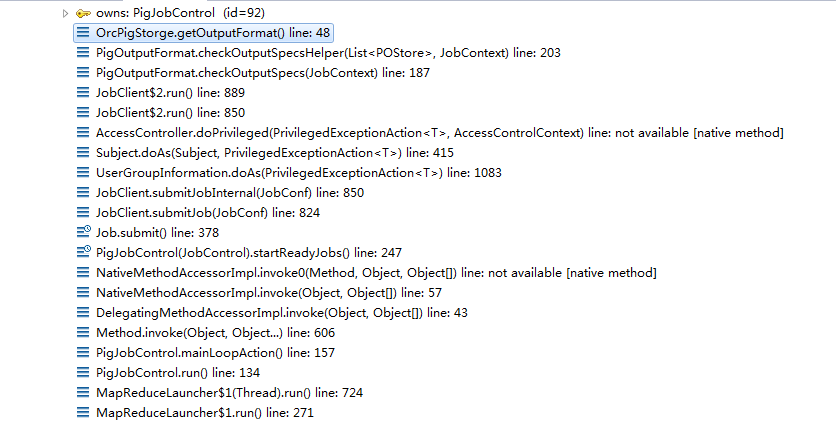
1. setStoreFuncUDFContextSignature(H\_1-3)



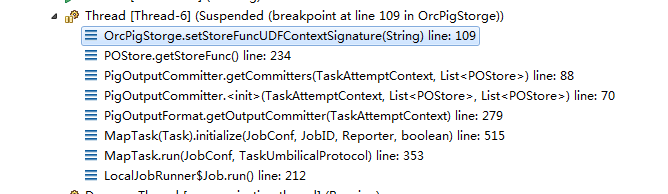
1. setStoreLocation



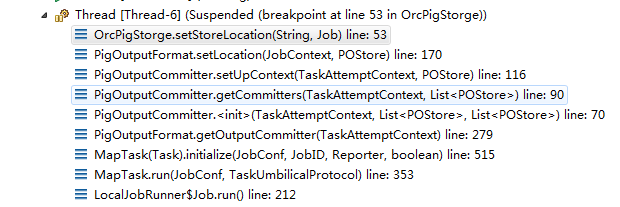
1. getOutputFormat



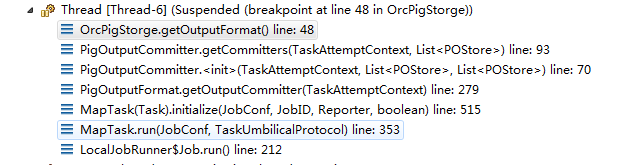
1. setStoreFuncUDFContextSignature(H\_1-3)



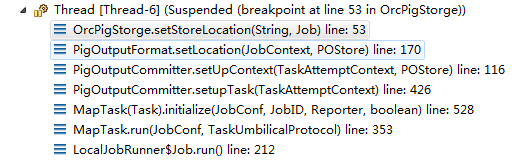
1. setStoreLocation



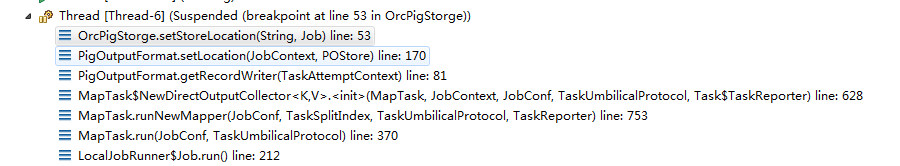
1. getOutputFormat



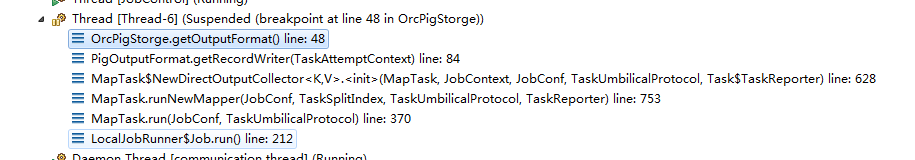
1. setStoreLocation



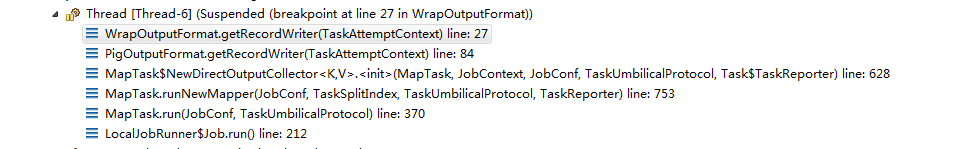
1. setStoreLocation



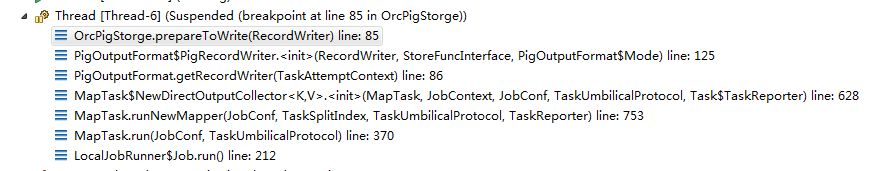
1. getOutputFormat



1. getRecordWriter



1. prepareToWrite



1. 待补充