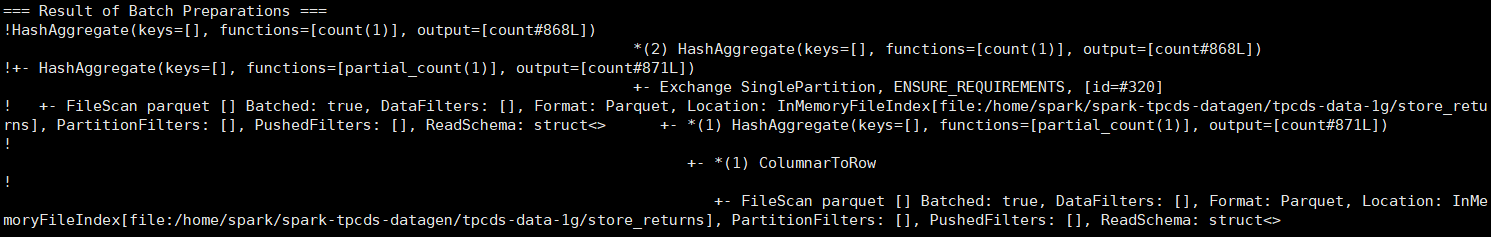
1. 使用Q1作为演示SQL
   1. SQL执行图（注：运行结果始终显示unable to infer schema for parquet. it must be specified manually）



* 1. RULE列表

|  |
| --- |
| org.apache.spark.sql.catalyst.analysis.Analyzer$ResolveRelations |
| org.apache.spark.sql.catalyst.analysis.CleanupAliases |
| org.apache.spark.sql.catalyst.analysis.EliminateSubqueryAliases |
| org.apache.spark.sql.catalyst.optimizer.ColumnPruning |
| org.apache.spark.sql.execution.exchange.EnsureRequirements |
| org.apache.spark.sql.execution.ApplyColumnarRulesAndInsertTransitions |
| org.apache.spark.sql.execution.CollapseCodegenStages |

* 1. 规则描述

1. org.apache.spark.sql.catalyst.optimizer.ColumnPruning

该规则在逻辑计划中尝试删除不需要读取的列，将删除后的列进行Project下推，以减少磁盘IO与网络IO的开销。

在SPARK3.2中，支持以下几种列裁剪：

1. Project/Aggregate/Expand 操作
2. 无用的AttachDistributedSequence
3. DeserializeToObject的叶子节点中的无用列
4. Aggregate/Expand/Generate/ScriptTransformation的叶子中的无用列
5. 不需要的references
6. 生成的不需要的嵌套字段
7. Left Existence Join右边的不需要的属性
8. 不必要的窗口表达式
9. 带有CTE（Common Table Expression）裁剪
10. 去掉没有对其进行操作的列

另外，在经过列裁剪规则处理后，由于Filter前的Project 会与PushPredicatesThroughProject规则冲突，该规则会对计划树自底向上遍历删除Filter前的Project。

1. org.apache.spark.sql.execution.CollapseCodegenStages

该规则为物理链式计划中支持CODEGEN的节点，将它们生成的代码进行合并，其中CODEGEN以SHUFFLE为界划分为不同STAGE，每个STAGE有不同的codegenStageId。

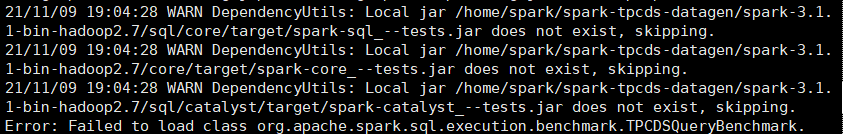
具体步骤如下：

对于PLAN的输出域长度为1并且输出域首元素类型是Spark SQL 表达式求值的 JVM 对象的情况，对其所有叶子节点递归调用insertWholeStageCodegen，如果新生成的PLAN与旧的不同，则用新的替换旧的PLAN。

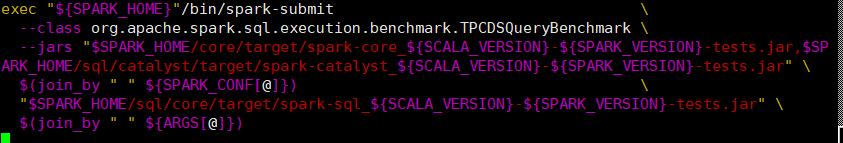
对于支持CODEGEN的PLAN，隐藏其所有支持CODEGEN叶子节点的SPARKPLAN（递归调用insertInputAdapter，为其所有不支持CODEGEN、SORTMERGEJOINEXEC、SHUFFLEDHASHJOINEXEC的叶子节点封装一层InputAdapter，其中InputAdapter用于从支持CODEGEN的子树隐藏SPARKPLAN），封装为WholeStageCodegenExec。

1.4运行时遇到的问题

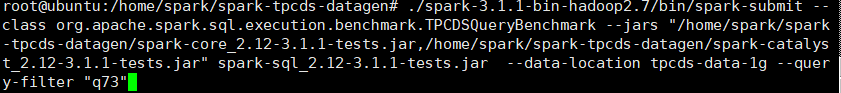
（1）加载不到test jars



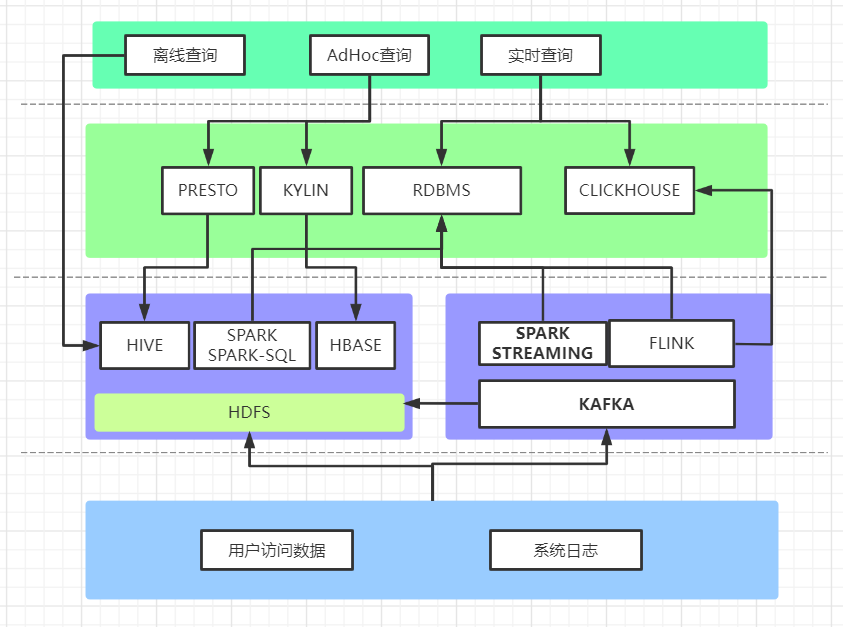
原因：run-tpcds-benchmark中spark-submit所指定的testsjar包路径非实际jar包存放路径



解决方法:在SHELL中直接运行spark-submit



2.



LAMBDA架构

优点：

1. 容错能力强。能够保证实时处理出现问题之后整个计算数据不被破坏，并可以保证最终一致性。
2. 实时处理计算量比较小，其运行代价可控，可在晚上进行批量处理，将实时处理与批处理在时间上进行错峰处理，缓解集群压力
3. 扩展能力强。可通过添加机器的方式进行水平扩展

缺点：

（1） 批处理与流处理算法需要实现两次，代码维护代价高，部署复杂，另外由于数据口径不一致可能会最终输出数据不一致

（2） 随着批处理数据规模的增加，有可能会出现批处理时间窗口无法处理完待处理数据的问题（如一天之内无法完成计算前一天的增量数据）

（3） 一般的批处理的设计会生成大量中间结果表，原始数据的膨胀会同时造成中间表的存储开销变大，造成存储成本上升

3.HDFS读写过程：

客户端写：

（1） client远程调用NameNode 的create方法向 NameNode 请求上传文件，NameNode 检查目标文件是否已存在、父目录是否存在、client是否有权限写入、在本地记录EditLog，如果这4项有一项失败，则向client 直接返回错误。  
（2）向client 返回可以上传。  
（3） client远程调用NameNode 的addBlock方法以获取向哪些DataNode写入数据。  
（4） NameNode 返回 3 个 DataNode 节点，假设分别为 dn1、dn2、dn3。  
（5） client通过 FSDataOutputStream 模块向dn1 请求上传数据， dn1 收到请求会继 续调用dn2，然后 dn2 调用 dn3，将这个通信管道建立完成。  
（6）dn1、dn2、dn3 逐级应答client。  
（7） client开始往 dn1 上传第一个 Block，以 Packet 为单位，dn1 收到一个 Packet 就会传给 dn2，dn2 传给 dn3。

（8）当第1个 Block 传输完成之后， 向client返回应答信息，客户端按步骤3-7继续传输剩下的Block。

（9） client 发送完成信号给NameNode。

客户端读：

（1） client远程调用NameNode 的XX 方法请求查询文件元数据，NameNode返回文件块所在的 DataNode 地址。  
（2）client按就近原则挑选一台 DataNode服务器，请求建立输入流。  
（3）DataNode 开始传输数据给client（以 Packet 为单位来做校验）。  
（4）client以 Packet 为单位接收，先在本地缓存，然后写入目标文件。