链滴

Impala 简介

作者：ZhangVincent

原文链接：https://ld246.com/article/1568697398451

来源网站：链滴

Impala是什么

Cloudera Impala是一个分布式的海量关系型数据查询引擎，能基于海量数据提供秒级查询能力。其有以下特点：

● 低延时。在执行SQL查询时，Impala不会将中间结果落磁盘，能省略巨大的IO开销，因此Impala往能在几秒钟内返回查询结果。相比于Hive，Impala更适用于交互式查询场景。

● 可与Hive共享元数据和底层存储数据。Hive的元数据同步到Impala之后，Impala可以基于该元数直接查询一张Hive表。由于当前大公司往往使用Hive作为数仓，因此，将Impala集成到现有的数据术体系中就会变得十分便捷。

● 支持通过JDBC/ODBC的方式访问。由于Impala支持通过JDBC/ODBC的方式访问，因此可将其海数据秒级查询能力开放给某些后台系统。

Impala的架构

在一个Impala集群中，所有的节点被分为3中角色：

● ImpalaD:基本上每个节点都会扮演ImpalaD的角色。一个Impalad由以下3个模块构成：

- Planner

Planner负责接收SQL并解析SQL，同时生成查询计划

- Cordinator

（1）Cordinator负责将查询计划转换为子任务，同时将子任务分发给Executor去执行

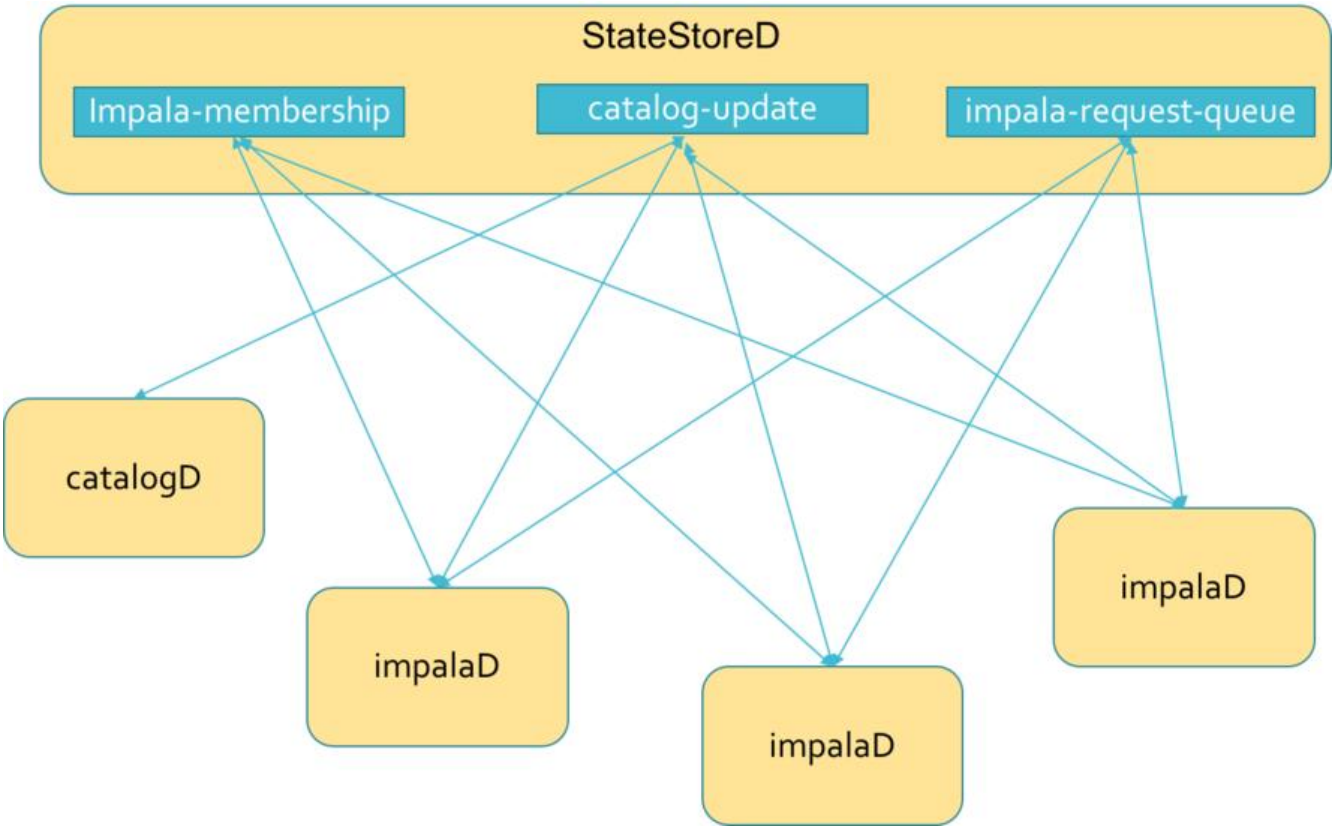
（2）Cordinator负责将Executor的执行结果返回给用户

（3）将用户提交的DDL操作转发给CatalogD处理

● CatalogD：一个集群中，只会有一个节点作为CatalogD，负责与Hive MetaStore通信，从而维护的元数据。由于Impala集群只有一个CatalogD，而impalad会将DDL操作都转交给catalogd执行，此Impala本身不适合于执行表数据或者表结构的变更操作，最好只将Impala做一个纯查询的工具使。

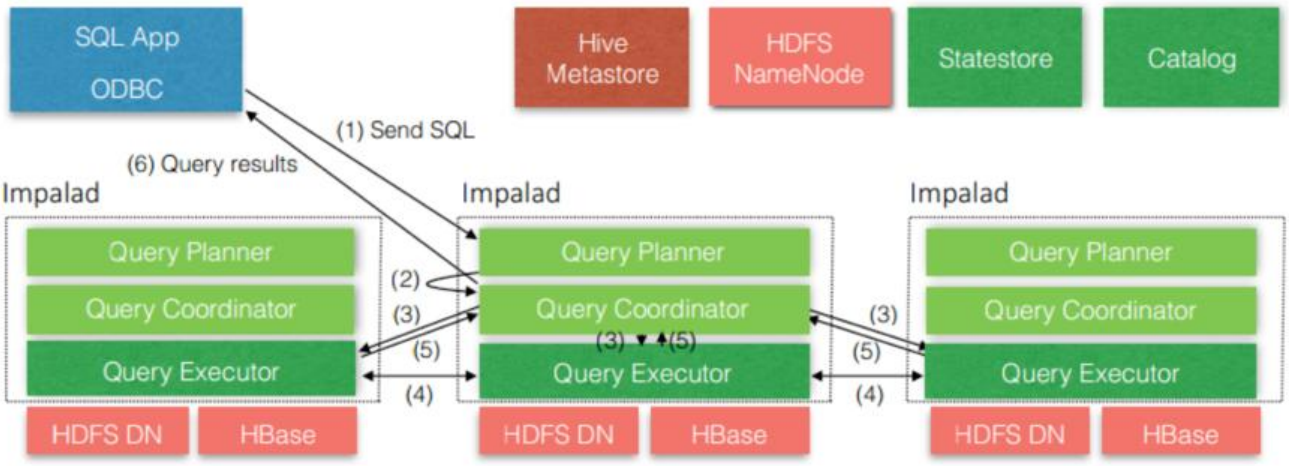
● StateStoreD：一个集群中，只会有一个节点作为StateStoreD，负责整个集群状态信息的同步。StaeStoreD 中会维护impala-membership/catalog update/impala-request-queue 等多个主题，某节点在订阅对应等主题后，可以向statestoreD上传主题变更后的内容，statestored则会将变更后的题内容向其他订阅了该主题的节点广播。例如，集群中所有的节点（包括catalogd）都会订阅catalog update 主题，当catalogd刷新元数据后，catalogd会将最新的元数据信息推送向statestored的catalg update 主题，而statestored会向集群中其他节点（都订阅了该主题）广播最新的主题内容。

原文链接：Impala 简介



一条查询SQL是如何执行的

总体步骤



如图所示，一个查询请求会经历如下过程：

（1）请求提交到某个impalad

（2）impalad到query planner模块负责解析sql，生成执行计划

（3）impalad到Cordinator 模块将执行计划转化为分布式任务，并分发给多个executor执行

（4）executor执行子任务。在执行过程中，各executor可能会互相交换数据

（5）各executor将执行结果返回给最开始impalad的cordinator模块

（6）cordinator模块会处理最终的结果（如取全局最小的100条记录）并返回给客户端

原文链接：Impala 简介

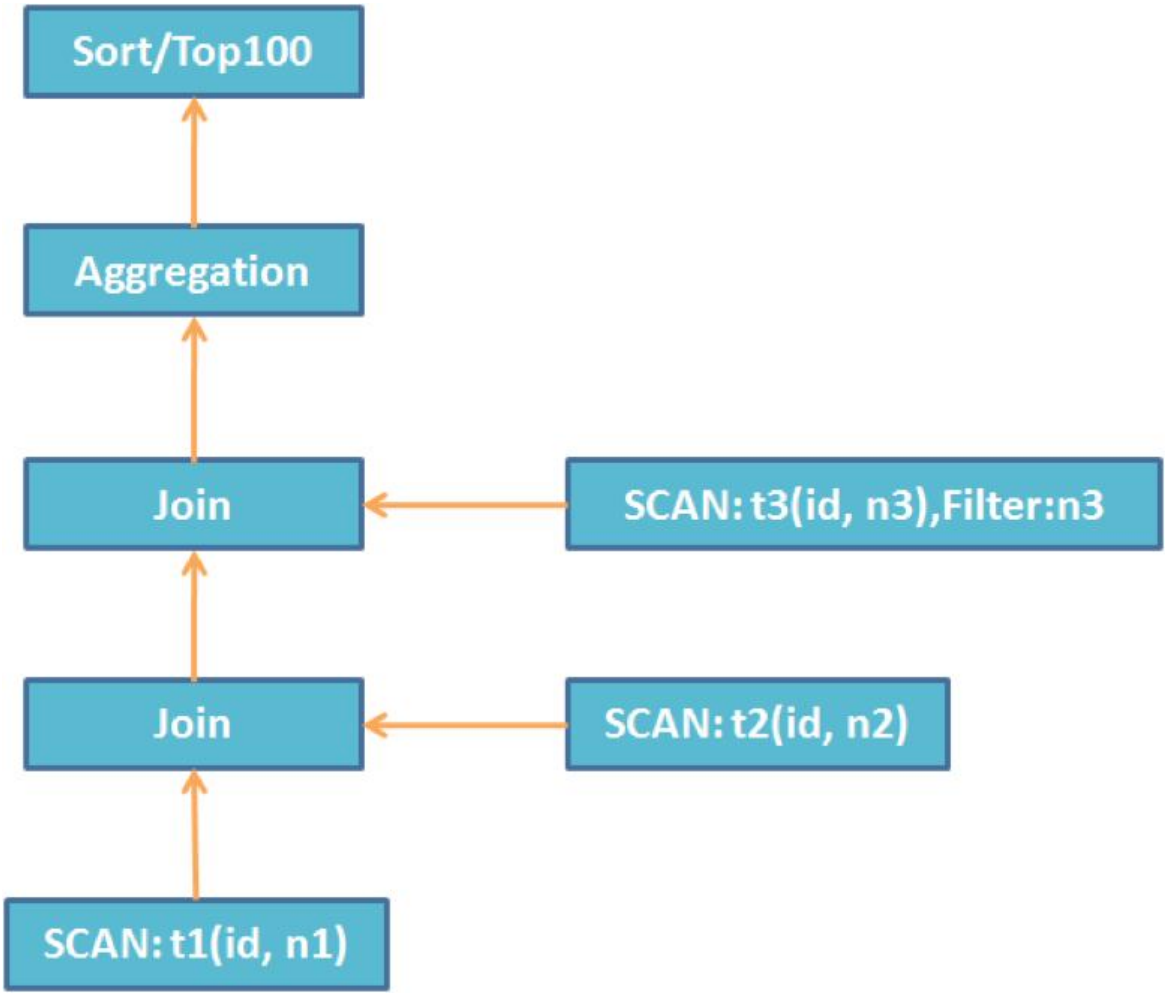
执行计划的生成

在一个查询SQL的执行过程中，涉及到执行计划的生成。生成执行计划，具体的又可以分为两步：（）生成单节点的执行计划（2）将单节点的执行计划转换为分区可并行的执行计划。我们以以下sql为解释执行计划的生成。

```
select   t1.n1 , t2.n2 , count(1) as c
from     t1
join     t2 on t1.id = t2.id
join     t3 on t1.id = t3.id
where    t3.n3 between 'a' and 'f'
group    by t1.n1,t2.n2
order    by c desc
limit    100
```

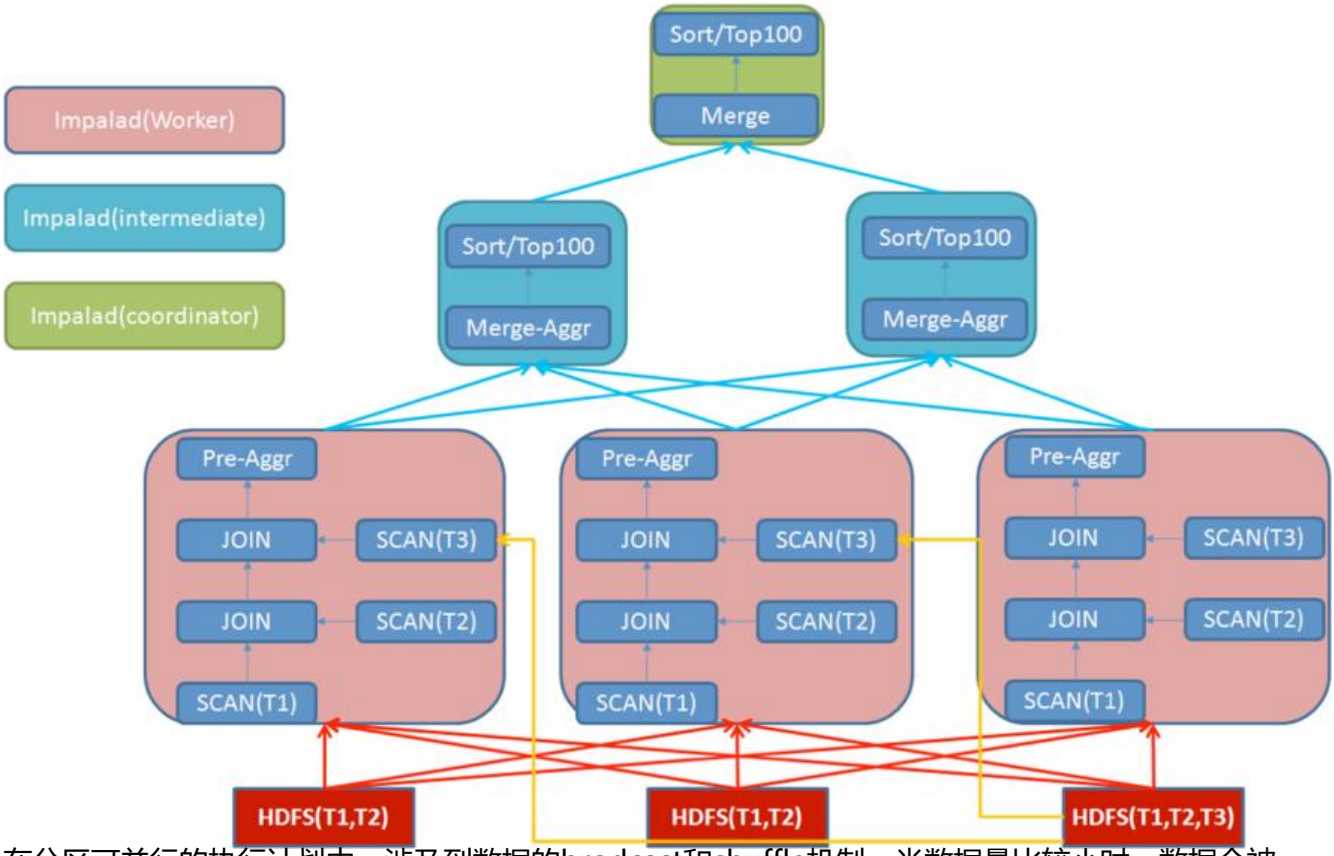
生成单节点的执行计划

一个单节点的执行计划如下图所示，涉及到scan/join/aggregation/sort等操作



将单节点的执行计划转换为分区可并行的执行计划

转化后的分区可并行执行计划如下图所示。



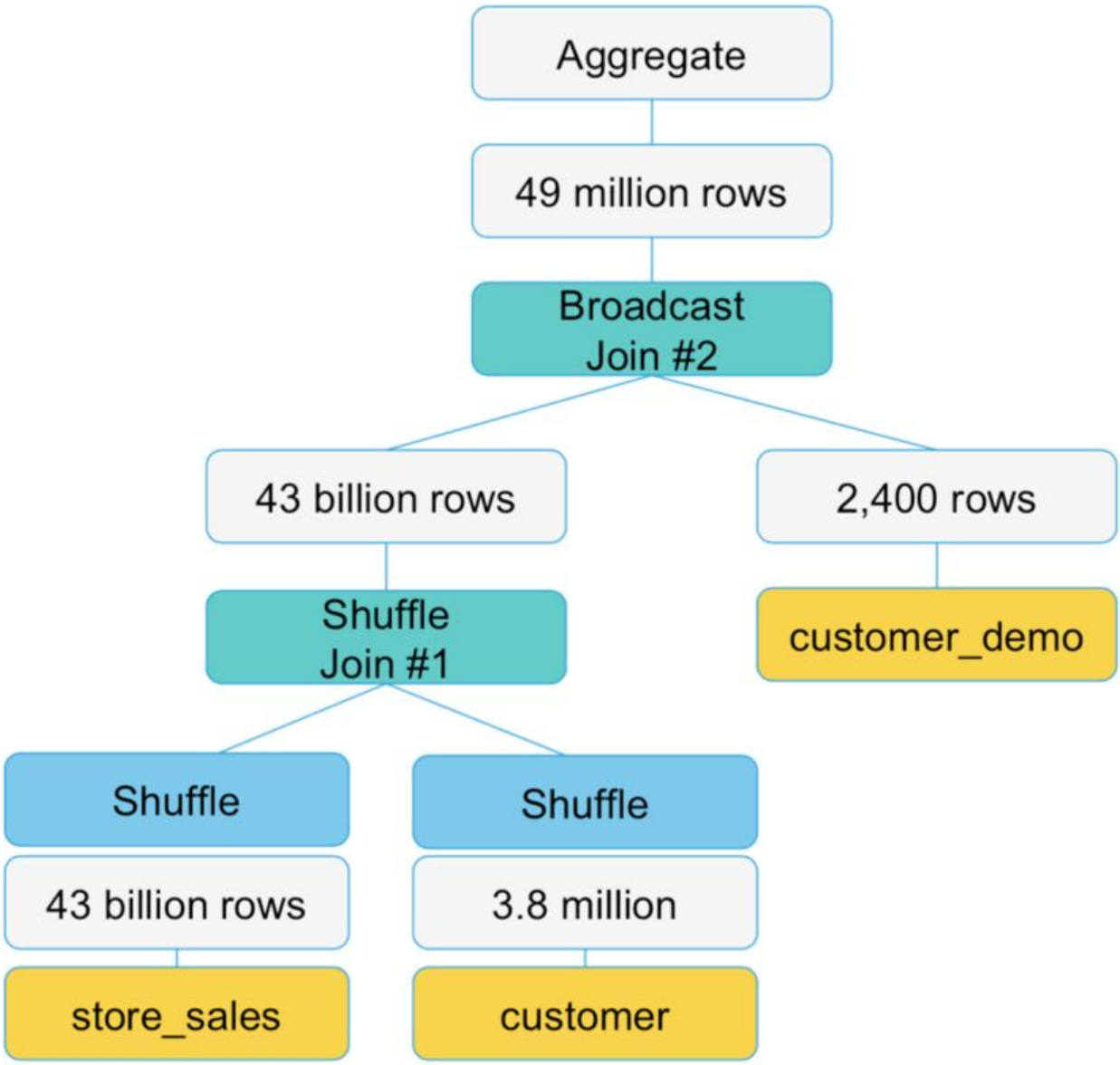
在分区可并行的执行计划中，涉及到数据的brodcast和shuffle机制。当数据量比较小时，数据会被播到所有的节点上（图中t3表的数据比较少，t3中满足条件的所有数据，都会被广播到所有的impalad worker节点）；当数据量较大时，数据会被hash到不同到节点（图中t1和t2表中数据比较多，因此据会根据join的id字段做hash，相同hash值的数据会被分配到相同到impalad worker节点；worker行完毕后，数据会根据n1和n2字段做hash，将worker到结果分配到不同到intermediate节点做进一合并）。

运行时过滤

一般情况下，查询引擎会根据sql中提供的限制条件从存储引擎中读取数据（如果存储引擎提供了传谓词的接口到话，则直接传入限制条件；如果没有提供谓词接口到话，则查询引擎对读取上来到数据过滤），称之为谓词下推。而impala在谓词下推的基础上，做了进一步的优化。

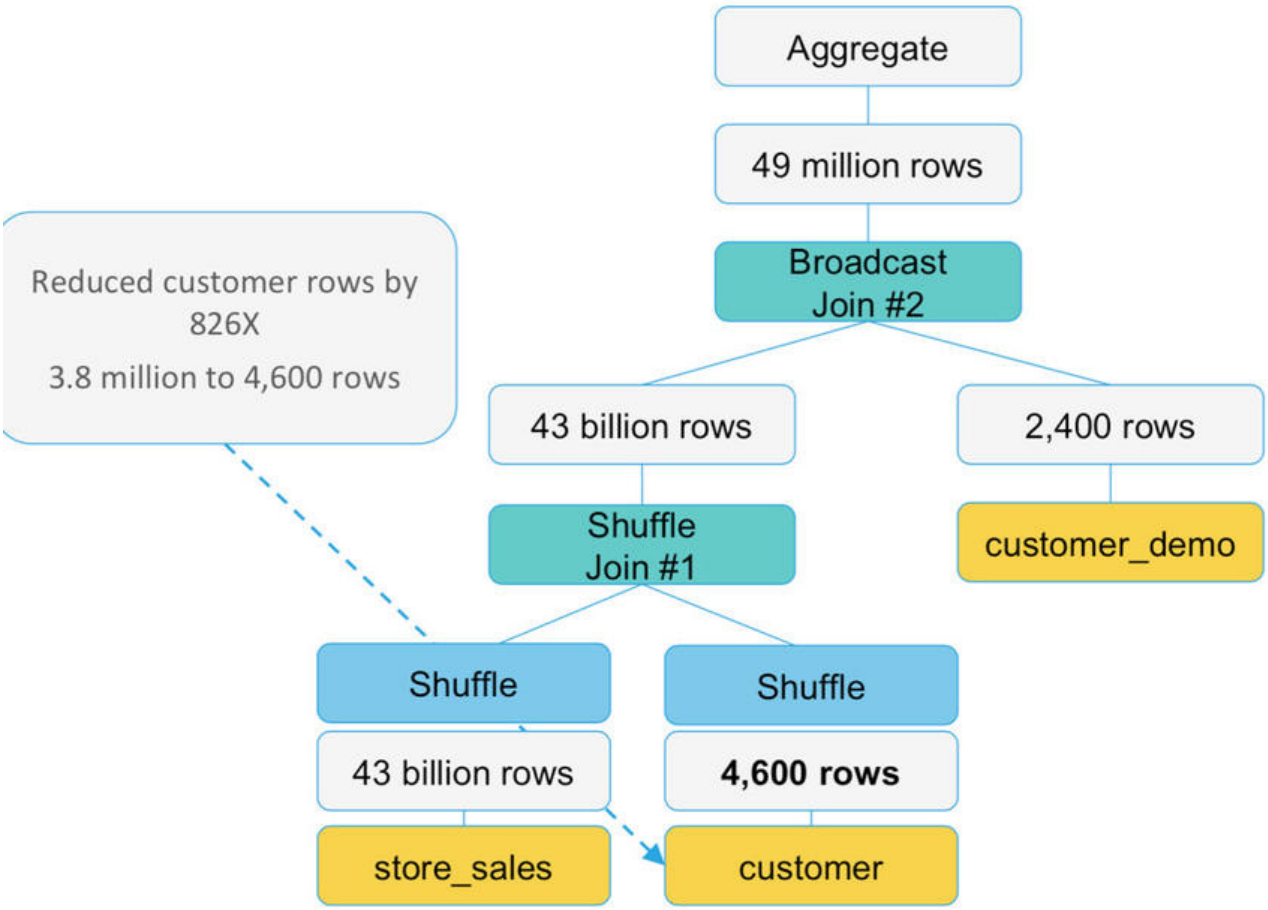
在impala中，考虑查询请求中join使用到两张表往往一个是大表一个是小表，而对小表的扫描要快于大表的扫描，这样可以先对小表执行扫描操作，将输出的记录交给JOIN节点，而大表则会主动等待一时间（默认是1s），JOIN节点会根据小表输出的记录计算出一个过滤条件，这个条件就是运行时过滤以下三张图展示了不使用运行过滤（第一张图）和使用运行时过滤（第二张图和第三张图）时，从存引擎中读取数据的情况。

在不使用运行时过滤的情况下，分别会从3张表中读取430亿/380万/2400条记录。

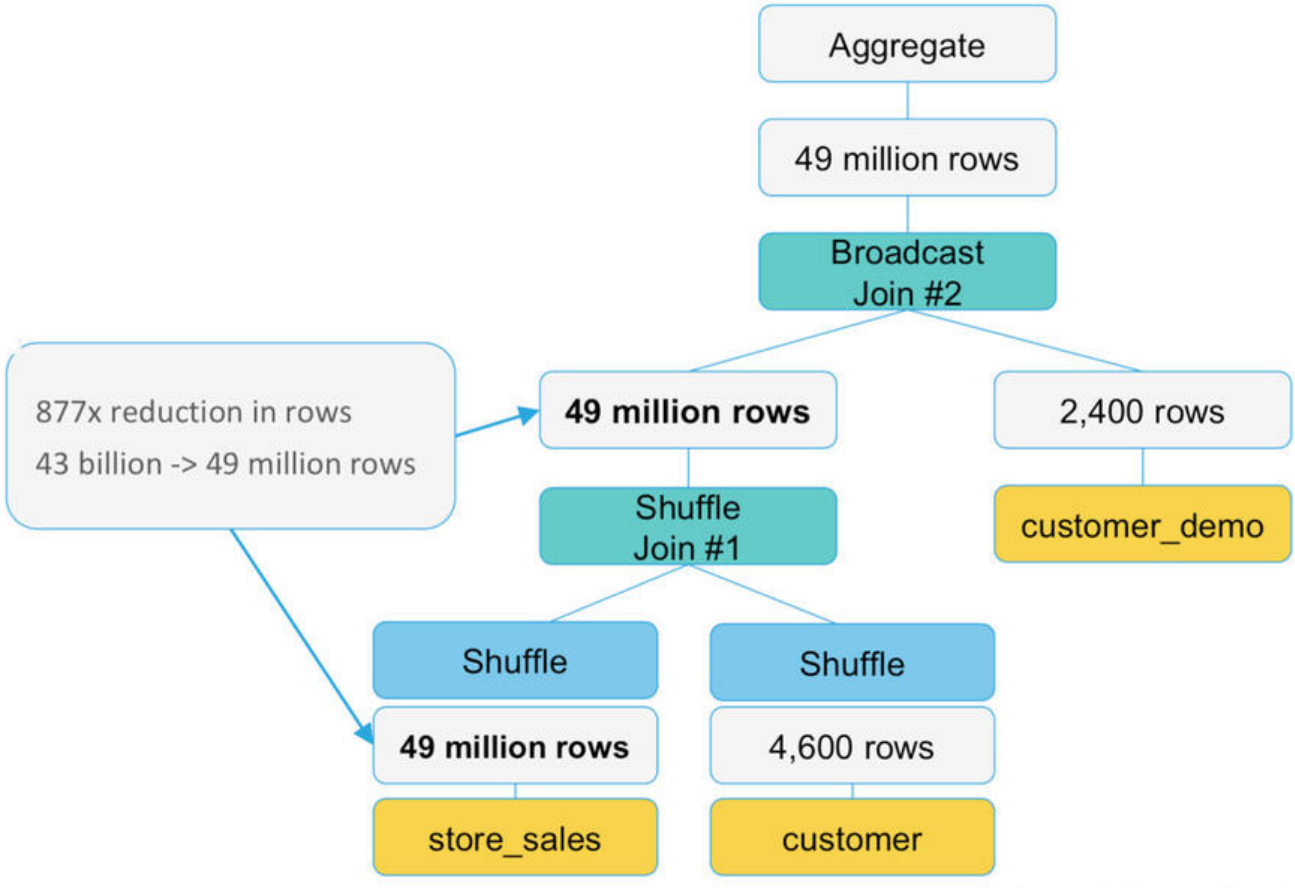


在使用运行时过滤的情况下，首先扫描customer_demo表，读取出2400条记录。根据这2400条记生成的运行时过滤条件作用到读取customer表，最终只需要从customer表中读取出4600条记录。

原文链接：Impala 简介



根据从customer表中读取出的4600条记录，生成运行时过滤条件，作用到store_sales表，最终从store_sales表中只需要读取出4900万条记录。



原文链接：Impala 简介

一些优化查询速度的技巧

统计表信息

通过执行 compute stats table_name 命令，可以对表的情况做一些统计（如表大小/每列最大最值/每列不同值个数等信息）。这些统计信息会被impala使用到，作用与执行计划生成过程（如决定否使用运行时过滤，使用brodcast还是shuffle进行join）。更精确的统计信息，有利于生成更合理的行计划。

使用explain获取执行计划

与mysql类似，通过 explain + sql ，可以查看某个sql等执行计划。

使用hint关键字指定join类型

当sql执行计划中的join类型选取不合理时，用户可以通过hint关键字手动指定join类型。例如：

```
select STRAIGHT_JOIN c_custkey,count(o_orderkey)
from customer
join [shuffle] orders
on c_custkey = o_custkey and o_comment not like '%[WORD1]%[WORD2]%'
group by c_custkey
order by c_custkey limit 10;
```

避免产生很多小文件

impala查询的表，往往是一张hive表。对hive表的多个插入操作，最好合并为一个大的插入操作，这可以防止生成多个小文件（太多的小文件，会影响读取性能）。

为数据存储选择合适的文件格式

通常对于大数据量来说，Parquet文件格式是最佳的