1. 前言

Impala是Cloudera公司主导开发的新型查询系统，它提供SQL语义，能查询存储在Hadoop的HDFS和HBase中的PB级大数据。已有的Hive系统虽然也提供了SQL语义，但由于Hive底层执行使用的是MapReduce引擎，仍然是一个批处理过程，难以满足查询的交互性。相比之下，Impala的最大特点也是最大卖点就是它的快速。但也依赖Hive（impala元数据都存储在Hive的metastore中,或者说两者元数据共存）

来自〈https://baike.baidu.com/item/Impala/7458017?fr=aladdin〉

2.Impala核心组件

2.1 Impala Daemon

Impala的核心组件是运行在各个节点上面的impalad这个守护进程（Impala daemon），它负责读写数据文件，接收从impala-shell、Hue、JDBC、ODBC等接口发送的查询语句，并行化查询语句和分发工作任务到Impala集群的各个节点上，同时负责将本地计算好的查询结果发送给协调器节点（coordinator node）。

你可以向运行在任意节点的Impala daemon提交查询，这个节点将会作为这个查询的协调器（coordinator node），其他节点将会传输部分结果集给这个协调器节点。由这个协调器节点构建最终的结果集。在做实验或者测试的时候为了方便，我们往往连接到同一个Impala daemon来执行查询，但是在生产环境运行产品级的应用时，我们应该循环（按顺序）的在不同节点上面提交查询，这样才能使得集群的负载达到均衡。

Impala daemon不间断的跟statestore进行通信交流，从而确认哪个节点是健康的能接收新的工作任务。它同时接收catalogd daemon（从Impala 1.2之后支持）传来的广播消

息来更新元数据信息，当集群中的任意节点create、alter、drop任意对象、或者执行INSERT、LOAD DATA的时候触发广播消息。

2.2 Impala Statestore

Impala Statestore检查集群各个节点上Impala daemon的健康状态，同时不间断地将结果反馈给各个Impala daemon。这个服务的物理进程名称是statestored，在整个集群中我们仅需要一个这样的进程即可。如果某个Impala节点由于硬件错误、软件错误或者其他原因导致离线，statestore就会通知其他的节点，避免其他节点再向这个离线的节点发送请求。

由于statestore是当集群节点有问题的时候起通知作用，所以它对Impala集群并不是有关键影响的。如果statestore没有运行或者运行失败，其他节点和分布式任务会照常运行，只是说当节点掉线的时候集群会变得没那么健壮。当statestore恢复正常运行时，它就又开始与其他节点通信并进行监控。

2.3 Impala Catalog

Imppalla catalog服务将SQL语句做出的元数据变化通知给集群的各个节点，catalog服务的物理进程名称是catalogd，在整个集群中仅需要一个这样的进程。由于它的请求会跟statestore daemon交互，所以最好让statestored和catalogd这两个进程在同一节点上。

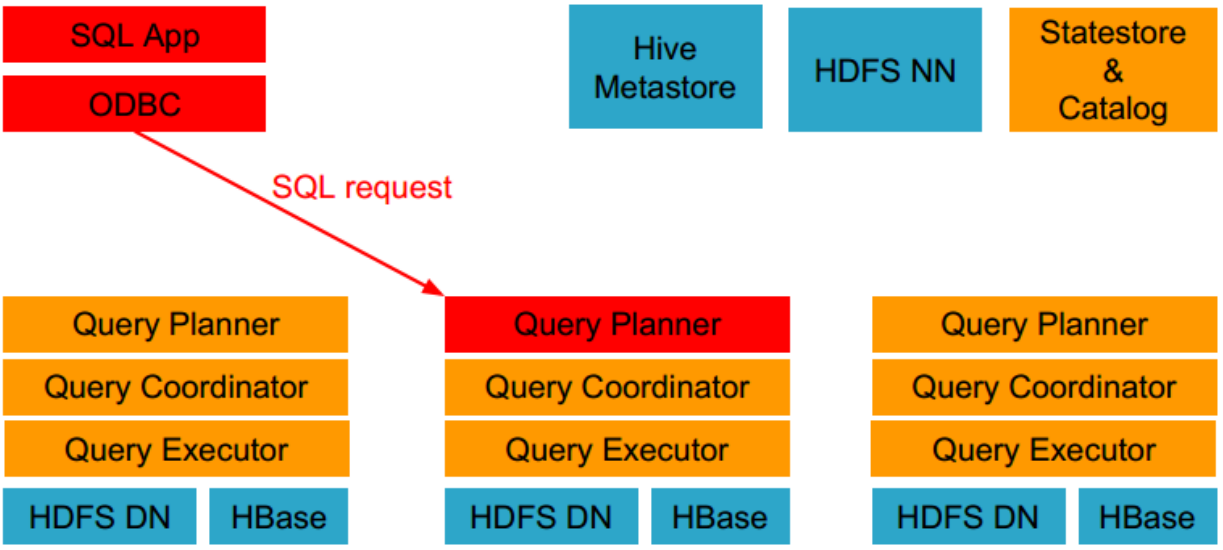
Impala 1.2中加入的catalog服务减少了REFRESH和INVALIDATE METADATA语句的使用。在之前的版本中，当在某个节点上执行了CREATE DATABASE、DROP DATABASE、CREATE TABLE、ALTER TABLE、或者DROP TABLE语句之后，需要在其它的各个节点上执行命令INVALIDATE METADATA来确保元数据信息的更新。同样的，当你在某个节点上执行了INSERT语句，在其它节点上执行查询时就得先执行REFRESH table_name这个操作，这样才能识别到新增的数据文件。需要注意的是，通过Impala执行的操作带来的元数据变化，有了catalog就不需要再执行REFRESH和INVALIDATE METADATA，但如果是通过Hive进行的建表、加载数据，则仍然需要执行REFRESH和INVALIDATE METADATA来通知Impala更新元数据信息。

来自〈http://www.cnblogs.com/chenz/articles/3947147.html〉

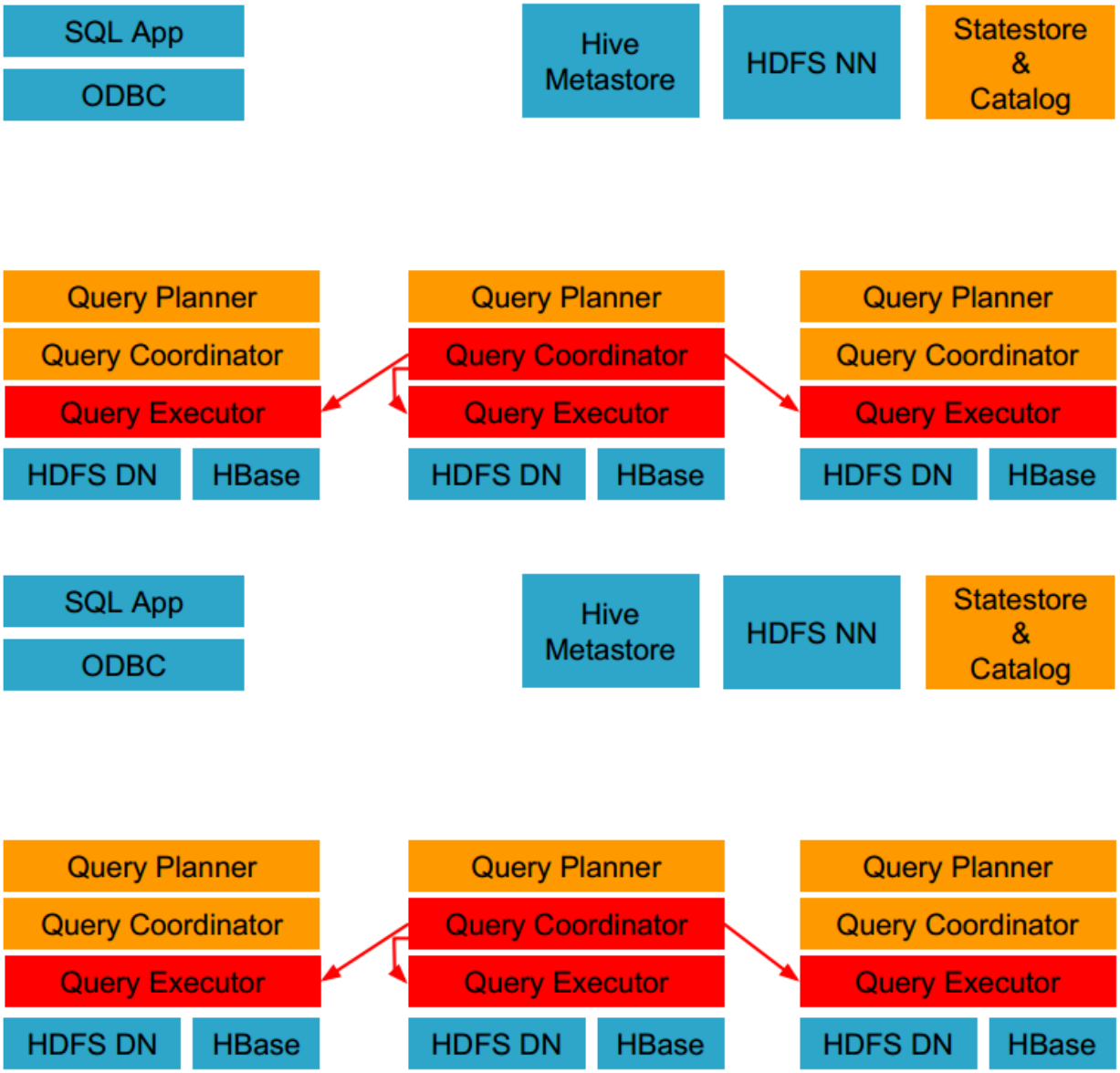
3. Impala执行步骤

Impala执行的查询有以下几个步骤：

1. 客户端通过ODBC、JDBC、或者Impala shell向Impala集群中的任意节点发送SQL语句，这个节点的impalad实例作为这个查询的协调器（coordinator）。

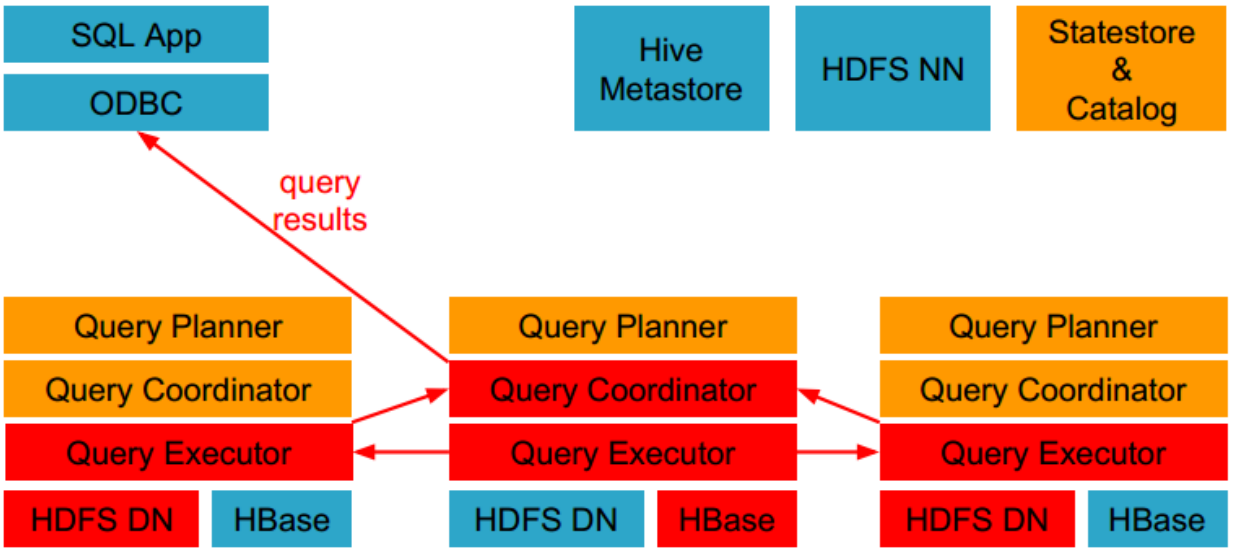


2. Impala解析和分析这个查询语句来决定集群中的哪个impalad实例来执行某个任务。



3. HDFS和HBase给本地的impalad实例提供数据访问。

4. 各个impalad向协调器impalad返回数据，然后由协调器impalad向client发送结果集。

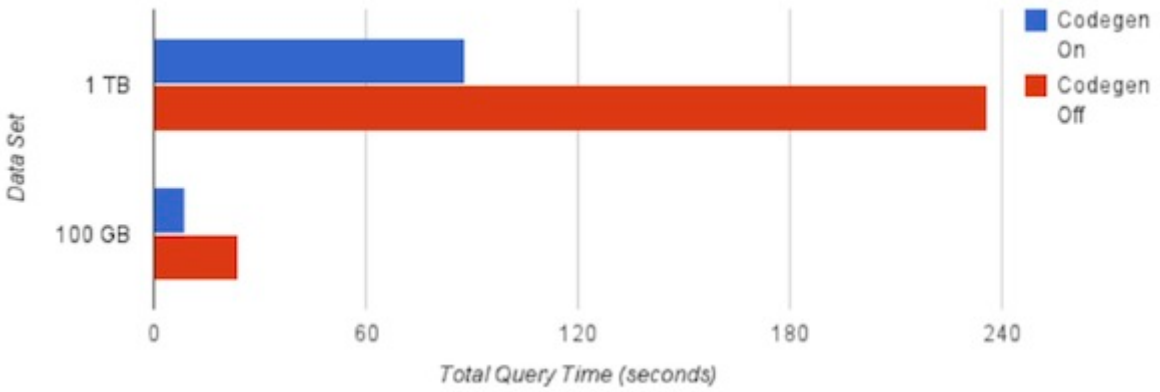


来自〈http://www.cnblogs.com/chenz/articles/3947147.html〉

Impala为什么比Hive速度快

Impala自称数据查询效率比Hive快几倍甚至数十倍，它之所以这么快的原因大致有以下几点:

- 真正的MPP查询引擎。
- 使用C++开发而不是Java，降低运行负荷。
- 运行时代码生成（LLVM IR），提高效率。



- 在执行SQL语句的时候，Impala不会把中间数据写入到磁盘，而是在内存中完成了所有的处理。
- 使用Impala的时候，查询任务会马上执行而不是生产Mapreduce任务，这会节约大量的初始化时间。

- Impala查询计划解析器使用更智能的算法在多节点上分布式执行各个查询步骤，同时避免了sorting和shuffle这两个非常耗时的阶段，这两个阶段往往是不需要的。
- Impala拥有HDFS上面各个data block的信息，当它处理查询的时候能够在各个datanode上面更均衡的分发查询。
- 另外一个关键原因是，Impala为每个查询产生汇编级的代码，当Impala在本地内存中运行的时候，这些汇编代码执行效率比其它任何代码框架都更快，因为代码框架会增加额外的延迟。

> *来自 <http://www.cnblogs.com/chenz/articles/3947147.html>*

注:

MPP:

MPP提供了另外一种进行系统扩展的方式，它由多个SMP服务器通过一定的节点互联网络进行连接，协同工作，完成相同的任务，从用户的角度来看是一个服务器系统。其基本特征是由多个SMP服务器(每个SMP服务器称节点)通过节点互联网络连接而成，每个节点只访问自己的本地资源(内存、存储等)，是一种完全无共享(Share Nothing)结构，因而扩展能力最好，理论上其扩展无限制。

在MPP系统中，每个SMP节点也可以运行自己的操作系统、数据库等。但和NUMA不同的是，它不存在异地内存访问的问题。换言之，每个节点内的CPU不能访问另一个节点的内存。节点之间的信息交互是通过节点互联网络实现的，这个过程一般称为数据重分配(Data Redistribution)。

SMP：

所谓对称多处理器结构，是指服务器中多个CPU对称工作，无主次或从属关系。各CPU共享相同的物理内存，每个 CPU访问内存中的任何地址所需时间是相同的，

来自〈https://www.2cto.com/net/201608/535126.html〉

LLVM IR:

提供与编程语言无关的优化和针对多种CPU的代码生成功能。

来自〈http://www.nagain.com/activity/article/4/〉