1.Impala和Hive的关系

2. Impala相对于Hive所使用的优化技术

3. Impala与Hive的异同

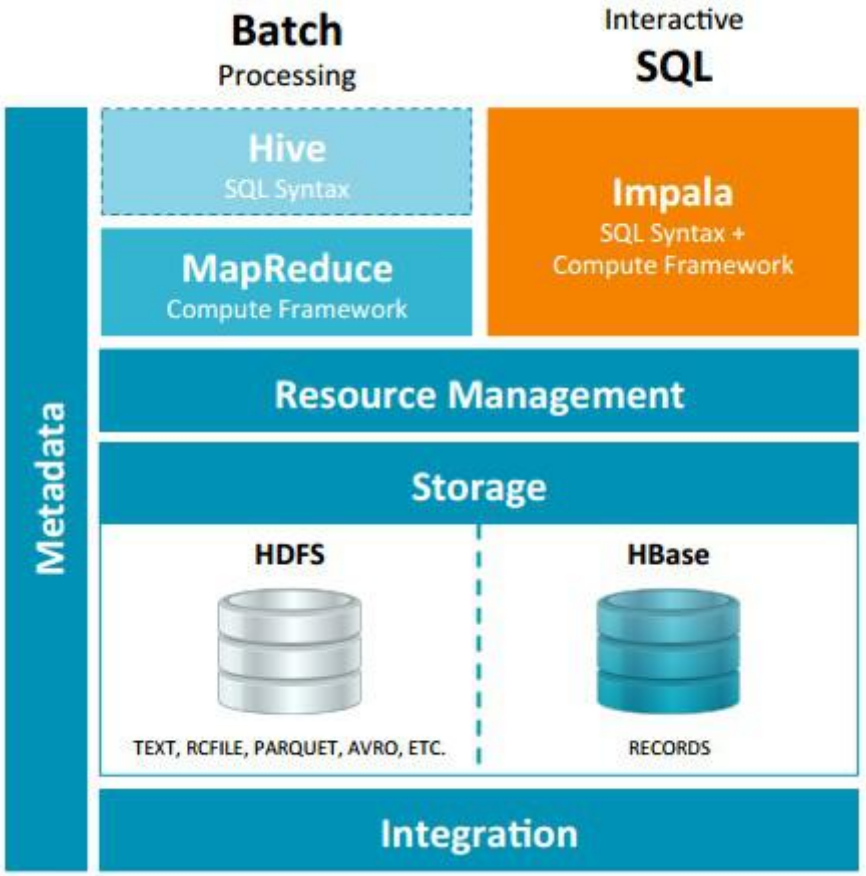
1.Impala和Hive的关系

Impala是基于Hive的大数据实时分析查询引擎，直接使用Hive的元数据库Metadata,意味着impala元数据都存储在Hive的metastore中。并且impala兼容Hive的sql解析，实现了Hive的SQL语义的子集，功能还在不断的完善中。

与Hive的关系

Impala 与Hive都是构建在Hadoop之上的数据查询工具各有不同的侧重适应面，但从客户端使用来看Impala与Hive有很多的共同之处，如数据表元数 据、ODBC/JDBC驱动、SQL语法、灵活的文件格式、存储资源池等。**Impala与Hive在Hadoop中的关系如下图**所示。

Hive适合于长时间的批处理查询分析，而Impala适合于实时交互式SQL查询，Impala给数据分析人员提供了快速实验、验证想法的大数 据分析工具。可以先使用hive进行数据转换处理，之后使用Impala在Hive处理后的结果数据集上进行快速的数据分析。



2. Impala相对于Hive所使用的优化技术

- 1、没有使用 MapReduce进行并行计算，虽然MapReduce是非常好的并行计算框架，但它更多的面向批处理模式，而不是面向交互式的SQL执行。与MapReduce相比：Impala把整个查询分成一执行计划树，而不是一连串的MapReduce任务，在分发执行计划后，Impala使用拉式获取 数据的方式获取结果，把结果数据组成按执行树流式传递汇集，减少的了把中间结果写入磁盘的步骤，再从磁盘读取数据的开销。Impala使用服务的方式避免 每次执行查询都需要启动的开销，即相比Hive没了MapReduce启动时间。
- 2、使用LLVM产生运行代码，针对特定查询生成特定代码，同时使用Inline的方式减少函数调用的开销，加快执行效率。
- 3、充分利用可用的硬件指令（SSE4.2）。
- 4、更好的IO调度，Impala知道数据块所在的磁盘位置能够更好的利用多磁盘的优势，同时Impala支持直接数据块读取和本地代码计算checksum。
- 5、通过选择合适的数据存储格式可以得到最好的性能（Impala支持多种存储格式）。

- 6、最大使用内存，中间结果不写磁盘，及时通过网络以stream的方式传递。

3. Impala与Hive的异同

- **数据存储：使用相同的存储数据池都支持把数据存储于HDFS, HBase。**
- **元数据：两者使用相同的元数据。**
- **SQL解释处理：比较相似都是通过词法分析生成执行计划。**

执行计划：

- Hive: 依赖于MapReduce执行框架，执行计划分成 map->shuffle->reduce->map->shuffle->reduce…的模型。如果一个Query会 被编译成多轮MapReduce，则会有更多的写中间结果。由于MapReduce执行框架本身的特点，过多的中间过程会增加整个Query的执行时间。
- Impala: 把执行计划表现为一棵完整的执行计划树，可以更自然地分发执行计划到各个Impalad执行查询，而不用像Hive那样把它组合成管道型的map->reduce模式，以此保证Impala有更好的并发性和避免不必要的中间sort与shuffle。

数据流：

- Hive: 采用推的方式，每一个计算节点计算完成后将数据主动推给后续节点。
- Impala: 采用拉的方式，后续节点通过getNext主动向前面节点要数据，以此方式数据可以流式的返回给客户端，且只要有1条数据被处理完，就可以立即展现出来，而不用等到全部处理完成，更符合SQL交互式查询使用。

内存使用：

- Hive: 在执行过程中如果内存放不下所有数据，则会使用外存，以保证Query能顺序执行完。每一轮MapReduce结束，中间结果也会写入HDFS中，同样由于MapReduce执行架构的特性，shuffle过程也会有写本地磁盘的操作。

- Impala：在遇到内存放不下数据时，当前版本1.0.1是直接返回错误，而不会利用外存，以后版本应该会进行改进。这使用得Impala目前处理Query会受到一 定的限制，最好还是与Hive配合使用。Impala在多个阶段之间利用网络传输数据，在执行过程不会有写磁盘的操作（insert除外）。

调度：

- Hive：任务调度依赖于Hadoop的调度策略。
- Impala：调度由自己完成，目前只有一种调度器simple-schedule，它会尽量满足数据的局部性，扫描数据的进程尽量靠近数据本身所在的物理机器。调度器 目前还比较简单，在SimpleScheduler::GetBackend中可以看到，现在还没有考虑负载，网络IO状况等因素进行调度。但目前 Impala已经有对执行过程的性能统计分析，应该以后版本会利用这些统计信息进行调度吧。

容错：

- Hive：依赖于Hadoop的容错能力。
- Impala：在查询过程中，没有容错逻辑，如果在执行过程中发生故障，则直接返回错误（这与Impala的设计有关，因为Impala定位于实时查询，一次查询失败， 再查一次就好了，再查一次的成本很低）。但从整体来看，Impala是能很好的容错，所有的Impalad是对等的结构，用户可以向任何一个Impalad提交查询，如果一个Impalad失效，其上正在运行的所有Query都将失败，但用户可以重新提交查询由其它Impalad代替执行，不 会影响服务。对于State Store目前只有一个，但当State Store失效，也不会影响服务，每个Impalad都缓存了State Store的信息，只是不能再更新集群状态，有可能会把执行任务分配给已经失效的Impalad执行，导致本次Query失败。

适用面：

- Hive：复杂的批处理查询任务，数据转换任务。
- Impala：实时数据分析，因为不支持UDF，能处理的问题域有一定的限制，与Hive配合使用,对Hive的结果数据集进行实时分析。

来自〈https://www.cnblogs.com/zlslch/p/6785207.html?utm_source=itdadao&utm_medium=referral〉