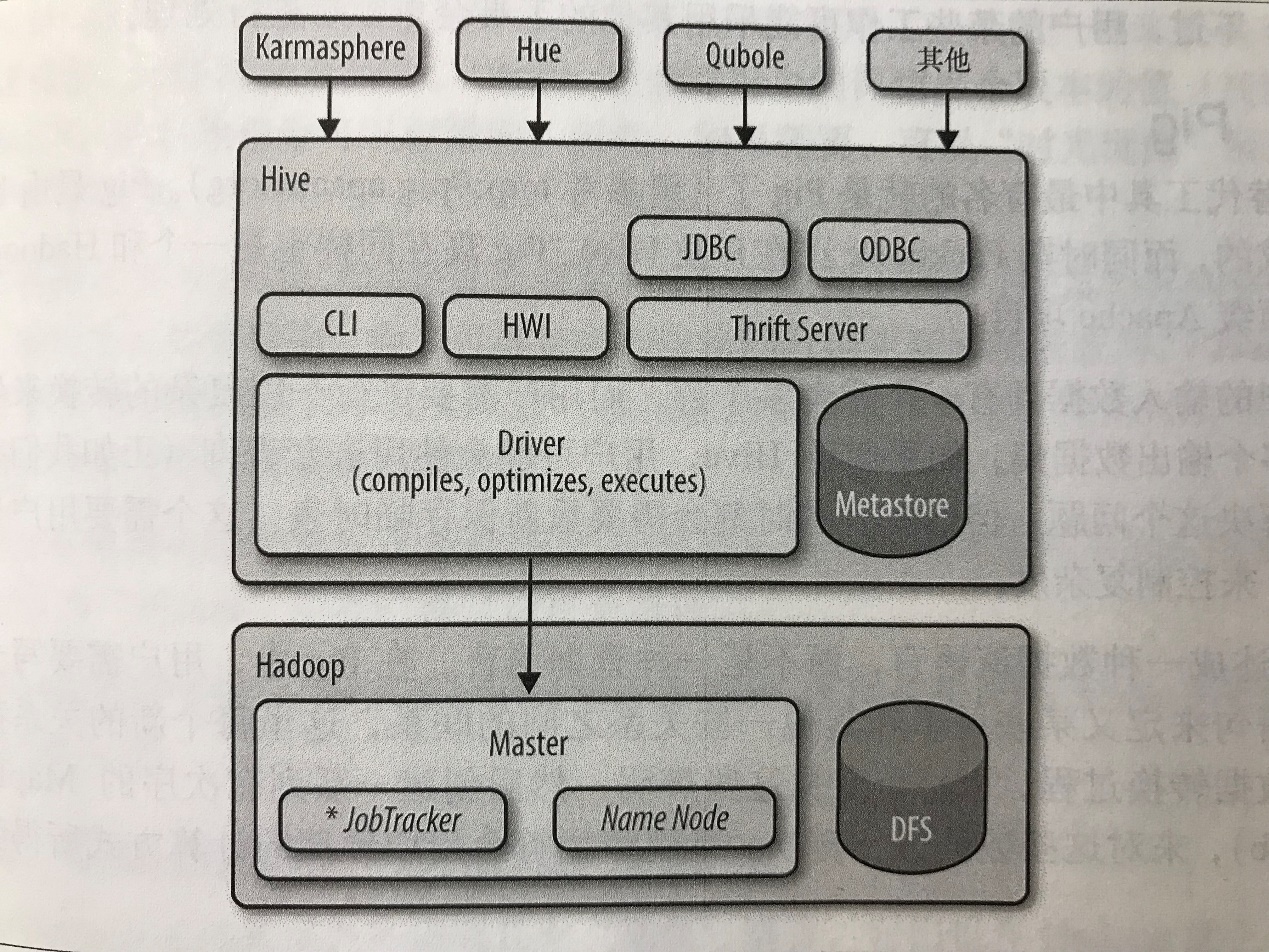
# 一、组成模块



【1】用户接口，CLI,WUI,CLIENT

【2】Hive 中的元数据包括表的名字，表的列和分区及其属性，表的属性（是否为外部表等），表的数据所在目录等

【3】解释器、编译器、优化器完成 HQL 查询语句从词法分析、语法分析、编译、优化以及查询计划的生成。生成的查询计划存储在 HDFS 中，并在随后有 MapReduce 调用执行

【4】MapReduce框架实现SQL基本操作的原理

<https://tech.meituan.com/2014/02/12/hive-sql-to-mapreduce.html>

# 二、使用技巧

【1】



**SELECT \* FROM shorter\_join where fistername=‘join’**

【2】

（1）Hive支持索引，但是Hive的索引与关系型数据库中的索引并不相同，

比如，Hive不支持主键或者外键。

（2）Hive索引可以建立在表中的某些列上，以提升一些操作的效率，

例如减少MapReduce任务中需要读取的数据块的数量

（3）维护索引需要存储空间，创建索引需要消耗计算资源。因此需要权衡利弊

【3】



模式设计，使用分桶算法：coolumn.hashCode % (桶数)，提供join这两个表都进行了桶操作。那么将保存相同列值的桶进行JOIN操作就可以，可以大大较少JOIN的数据量

# 三、性能优化

【1】limit限制调整

hive.limit.optimize.enable=true：优化LIMIT n语句

hive.limit.row.max.size=1000000：

hive.limit.optimize.limit.file=10：最大文件数

【2】MapJoin是Hive的一种优化操作，其适用于小表JOIN大表的场景。由于表的JOIN操作是在Map端且在内存进行的，所以其并不需要启动Reduce任务，也就不需要经过shuffle阶段，从而能在一定程度上节省资源提高JOIN效率。

set hive.auto. convert . join =  true ;  将小表刷入内存中

set hive.mapjoin.smalltable.filesize = 25000000 ; 刷入内存表的大小(字节)

注意：是false，会在必要的时候启动优化。需要将大表放在join右边，或者 使用 select /\*streamtable(table\_name)\*/ 指出

【3】当对3个或者更多个表进行join连接时，如果每个on子句都使用相同的连接键的话，那么只会产生一个MapReduce job。

【4】并行执行

hive会将一个查询转化为一个或多个阶段，包括：MapReduce阶段、抽样阶段、合并阶段、limit阶段等。默认情况下，一次只执行一个阶段。 不过，如果某些阶段不是互相依赖，是可以并行执行的。

set hive.exec.parallel=true,可以开启并发执行。

set hive.exec.parallel.thread.number=32; //同一个sql允许最大并行度，默认为8

【5】strict模式

--对分区表进行查询，在where子句中没有加分区过滤的话，将禁止提交任务(默认：nonstrict)set hive.mapred.mode=strict;

注：使用严格模式可以禁止3种类型的查询：

（1）对于分区表，不加分区字段过滤条件，不能执行

（2）对于order by语句，必须使用limit语句。

Order by为执行排序过程会将所有数据分发到一个reducer中进行处理，需要强制加limit

（3）限制笛卡尔积的查询

select \* from A JOIN B WHERR A.ID=B.ID 这里会走一个笛卡尔积，A\*B的数量，然后再查询相等的数量。（常用关系型数据库会优化成内连接）

select \* from A JOIN B ON A.ID=B.ID

【6】调整mapper 和reducer个数

Map/reduce执行时间：map/reduce 任务启动和初始化的时间+逻辑处理的时间。太多的mapper和reducer任务，会导致启动、调度、运行过程中开销过多。如果太少，没有利用好并行性，效率太低，所以保持平衡，效率最高。

调整map数量

通过以下方法来在map执行前合并小文件，减少map数

set mapred.max.split.size=100000000;

set mapred.min.split.size.per.node=100000000;

set mapred.min.split.size.per.rack=100000000;

set hive.input.format=org.apache.hadoop.hive.ql.io.CombineHiveInputFormat;（减少map的数量，来合并小文件，节省资源）

调整reduce

set hive.exec.reducers.bytes.per.reducer=1000000000; （1G）

或者

set mapred.reduce.tasks = 15;（设置reduce的数量）

【7】**JVM重用**

避免小文件的场景或者task特别多的场景，这类场景大多数执行时间都很短，因为hive调起mapreduce任务，JVM的启动过程会造成很大的开销，尤其是job有成千上万个task任务时，JVM重用可以使得JVM实例在同一个job中重新使用N次

set mapred.job.reuse.jvm.num.tasks=10; --10为重用个数

缺点：开启后会一直占用使用到的task卡槽，以便重用，可能某几个reduce task执行很慢，造成已经完成的task空闲占用卡槽。

【8】数据倾斜

任务进度长时间维持在99%（或100%），查看任务监控页面，发现只有少量（1个或几个）reduce子任务未完成。因为其处理的数据量和其他reduce差异过大。单一reduce的记录数与平均记录数差异过大，通常可能达到3倍甚至更多。 最长时长远大于平均时长

**（1）hive.map.aggr=true**

Map 端部分聚合，相当于Combiner

**hive.groupby.skewindata=true**

有数据倾斜的时候进行负载均衡，当选项设定为 true，生成的查询计划会有两个 MR Job。第一个 MR Job 中，Map 的输出结果集合会随机分布到 Reduce 中，每个 Reduce 做部分聚合操作，并输出结果，这样处理的结果是相同的 Group By Key 有可能被分发到不同的 Reduce 中，从而达到负载均衡的目的；第二个 MR Job 再根据预处理的数据结果按照 Group By Key 分布到 Reduce 中（这个过程可以保证相同的 Group By Key 被分布到同一个 Reduce 中），最后完成最终的聚合操作。

（2）生成随机数列 rand(100)

【9】执行计划

<https://blog.csdn.net/conggova/article/details/77985070>

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Explain>

【1】**hive.groupby.skewindata=true 应用场景**

**【2】shuf sort**