# Hive

## Hive原理

#### **（★★★）Hive的工作原理**

1. 用户提交查询等任务给Driver。

2. 编译器获得该用户的任务Plan。

3. 编译器Compiler根据用户任务去MetaStore中获取需要的Hive的元数据信息。

4. 编译器Compiler得到元数据信息，对任务进行编译，先将HiveQL转换为抽象语法树，然后将抽象语法树转换成查询块，将查询块转化为逻辑的查询计划，重写逻辑查询计划，将逻辑计划转化为物理的计划（MapReduce）, 最后选择最佳的策略。

5. 将最终的计划提交给Driver。

6. Driver将计划Plan转交给ExecutionEngine去执行，获取元数据信息，提交给JobTracker或者SourceManager执行该任务，任务会直接读取HDFS中文件进行相应的操作。

7. 获取执行的结果。

8. 取得并返回执行结果。

#### （★）Hive有哪些保存元数据的方式，每个有什么特点？

1）. 内嵌模式：将元数据保存在本地内嵌的derby数据库中，内嵌的derby数据库每次只能访问一个数据文件，也就意味着它不支持多会话连接。

2）. 本地模式：将元数据保存在本地独立的数据库中（一般是mysql），这可以支持多会话连接。

3） .远程模式：把元数据保存在远程独立的mysql数据库中，避免每个客户端都去安装mysql数据库。

#### **（★★★）三种元数据存储方式**

参考上题的hive元数据存储

#### （★★）hive 如何权限控制？

1 Metastore Server中基于存储的授权

2 HiveServer2中基于SQL标准的授权

3使用Apache Ranger和Sentry进行授权

4旧的默认Hive授权（传统模式）

详情见官方文档[Hive—LanguageManual+Authorization](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Authorization)

#### （★★）hive 底层与数据库交互原理

Hive中有两类数据：表数据和元数据。和关系型数据库一样，元数据可以看做是描述数据的数据，包括

1.hive表的数据库名、表名、字段名称与类型、分区字段与类型

2.表及分区的属性、存放位置等 。

元数据存储路径和內表一样，分为本地和远程，可通过hive-site.xml文件设置

参考博客：https://blog.csdn.net/victorzzzz/article/details/81874674

Hive与mysql的关系：只是借用mysql来存储hive中的表的元数据信息，称为metastore

#### （★）Hive是按照什么粒度划分的

按照数据的粒度大小，hive数据可以被组织成：

1）databases： 避免不同表产生命名冲突的一种命名空间

2）tables：具有相同scema的同质数据的集合

3）partitions：一个表可以有一个或多个决定数据如何存储的partition key

4）buckets（或clusters）：在同一个partition中的数据可以根据某个列的hash值分为多个bucket。partition和bucket并非必要，但是它们能大大加快数据的查询速度。

详见 https://blog.csdn.net/w3045872817/article/details/78166612

#### （★）数据仓库与hive的联系

Hive:是一个数据仓库构建的工具，他可以通过derby或者mysql等关系型数据库来记录数据仓库元数据描述信息，可以将结构化的数据映射为一张数据库表，并提供类SQL查询功能，本质将SQL转换为MaoReduce程序。

## Hive数据导入

#### **（★★★）Hive数据导入的几种方式**

1.本地文件导入到Hive表；

2.Hive表导入到Hive表;

3.HDFS文件导入到Hive表;

4.创建表的过程中从其他表导入;

5.通过sqoop将mysql库导入到Hive表

## Hive表

#### **（★★★）Hive几种表之间的区别**：

Hive表有受控表(内部表)、外部表、分区表、桶表四种。

**内部表**，就是一般的表，前面讲到的表都是内部标，当表定义被删除的时候，表中的数据随之一并被删除。

**外部表**，数据存在与否和表的定义互不约束，仅仅只是表对hdfs上相应文件的一个引用，当删除表定义的时候，表中的数据依然存在

**分区表**，在Hive Select查询中一般会扫描整个表内容，会消耗很多时间做没必要的工作。有时候只需要扫描表中关心的一部分数据，因此建表时引入了partition概念。分区表指的是在创建表时指定的partition的分区空间

**分桶表**，对于每一个表（table）或者分区， Hive可以进一步组织成桶，也就是说桶是更为细粒度的数据范围划分。Hive也是 针对某一列进行桶的组织。Hive采用对列值哈希，然后除以桶的个数求余的方式决定该条记录存放在哪个桶当中。

### 内部表外部表

#### （★）Hive内部表和外部表的区别

**内部表**：加载数据到Hive所在的HDFS目录，删除时候，元数据和数据文件都删除

**外部表**：不加载数据到Hive所在的HDFS目录，删除时候，只删除表结构

#### （★）生产环境中为什么建议使用外部表？

删除外部表的仅仅会删除元数据，HDFS上的文件不会被删除，这样的好处是操作人员不小心删除表不会将数据也删除。

### 分区分桶

#### （★）HIVE分区和分桶，有什么好处

**分区:**

1、一个表可以拥有一个或者多个分区，每个分区以文件夹的形式单独存在表文件夹的目录下。

2、表和列名不区分大小写。

3、分区是以字段的形式在表结构中存在，通过describetable命令可以查看到字段存在， 但是该字段不存放实际的数据内容，仅仅是分区的表示（伪列） 。

**分桶:**

对于每一个表（table）或者分区， Hive可以进一步组织成桶，也就是说桶是更为细粒度的数据范围划分。Hive也是 针对某一列进行桶的组织。Hive采用对列值哈希，然后除以桶的个数求余的方式决定该条记录存放在哪个桶当中。

**好处:**

（1）获得更高的查询处理效率。桶为表加上了额外的结构，Hive 在处理有些查询时能利用这个结构。具体而言，连接两个在（包含连接列的）相同列上划分了桶的表，可以使用 Map 端连接 （Map-side join）高效的实现。比如JOIN操作。对于JOIN操作两个表有一个相同的列，如果对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。

（2）使取样（sampling）更高效。在处理大规模数据集时，在开发和修改查询的阶段，如果能在数据集的一小部分数据上试运行查询，会带来很多方便。

#### （★★）Hive的动态分区和静态分区有什么区别

静态分区是分区的时候指定所分的区，不能自动分区，分区的条件必须自己指定

动态分区可以在程序运行的时候以某一列变量的值进行分区，动态分区必须通过设置打开：如

set hive.exec.dynamic.partition=true;

set hive.exec.dynamic.partition.mode=nonstrict;

#### （★）假如一个分区的数据主表错误怎么通过Hivesql删除HDFS

Alter table ptable drop partition(daytime=’20140911’，city=’bj’)

元数据，数据文件都删除，但是目录daytime=20140911还在

## Hive查询

### UDF

#### （★★）添加自定义UDF的时候都是临时的，怎么永久添加？

1. 将打包好的jar包上传到HDFS上

2. CREATE FUNCTION [db\_name.]function\_name AS class\_name

[USING JAR|FILE|ARCHIVE 'file\_uri' [, JAR|FILE|ARCHIVE 'file\_uri'] ];

file\_uri:是hdfs上的jar包目录

#### （★★★）UDF函数(User Defined Funtion ):用户自定义函数

当Hive提供的内置函数无法满足你的业务处理需要时，此时就可以考虑使用用户自定义函数，用户自定义函数UDF。编写一个UDF，需要继承UDF类，并实现evaluate()函数。在查询执行过程中，查询中对应的每个应用到这个函数的地方都会对这个类进行实例化。对于每行输入都会调用到evaluate()函数。而evaluate()函数处理的值会返回给Hive。同时用户是可以重载evaluate方法的。Hive会像Java的方法重载一样，自动选择匹配的方法。

详细讲解，参考博客：

https://www.cnblogs.com/chushiyaoyue/p/6632090.html?utm\_source=itdadao&utm\_medium=referral

### 窗口函数

#### （★）Hive的开窗函数

over(partition by ......)主要和聚合函数sum()、count()、avg()等结合使用，实现分组聚合的功能

带上group by的hive sql语句只能显示与分组聚合相关的字段，而带上over(partition by ......)的hive sql语句能显示所有字段.。

#### （★）RANK、DENSE\_RANK以及ROW\_NUMBER的区别

ROW\_NUMBER() 是没有重复值的排序(即使两条记录相同，序号也不重复的)，不会有同名次。

DENSE\_RANK() 是连续的排序，两个第二名仍然跟着第三名。

RANK()是跳跃排序，两个第二名下来就是第四名。

### 综合

#### （★）请用一个sql语句返回两张表的差集

select ★ from tb1 where id not in (select id from tb2)

#### （★）什么是左连接?

LEFT JOIN 关键字会从左表那里返回所有的行，即使在右表中没有匹配的行

RIGHT JOIN 关键字会右表那里返回所有的行，即使在左表中没有匹配的行

#### （★★）多表join如何处理?

#### （★）Hive如何解析JSON串

将json以字符串的方式整个入Hive表，然后通过使用UDF函数解析已经导入到hive中的数据，比如使用LATERAL VIEW json\_tuple的方法，获取所需要的列名。

在导入之前将json拆成各个字段，导入Hive表的数据是已经解析过得。这将需要使用第三方的SerDe。

#### （★★）Hive中UDF UDAF UDTF的区别。

UDF：操作作用于单个数据行，并且产生一个数据行作为输出。大多数函数都属于这一类。

UDAF 接受多个输入数据行，并且产生一个输出数据行。向COUNT和MAX这样的函数就是聚集函数。

UDTF 操作作用于单个数据行，并且产生多个数据行-------一个表作为输出。Lateral view explore()

#### （★★）Hive什么时候不能用combiner？ //P

比如使用avg函数的时候，就不能用combiner,一些在map端聚合会和正确结果不一样的时候，不能用combiner。

#### （★）使用Hive实现WordCount

（1） 创建表并关联数据：

create external table wordcount(line string) row format delimited fields terminated by '\n' stored as textfile location '/input/wordcount';

（2） 创建data.txt文件，并上传至HDFS中的/input/wordcount目录下

vi ruozedata.txt

hello,ruoze

hello,jepsondb

hello,www.ruozedata.com

hi,man

hi,gril

（3） 上传命令hdfs dfs -put data.txt /input/wordcount

（4） 拆分：把每行数据差分成单词，这里需要用到一个Hive的内置表生成函数（UDTF）explode(array),参数是array，其实就是行变多列

select word, count(★) from (select explode(split(line, ",")) as word from wordcounts) t group by word;

### Hive函数对比

#### （★）Hive 里面用什么代替 in 查询

hive不支持in 子查询的用法，可以考虑用left join的方式来替换in

#### （★）having和where的区别

Where 子句是用来指定 "行" 的条件的，而Having 子句是指定 “组” 的条件的，

即

Where 子句 = 指定行所对应的条件

Having 子句 = 指定组所对应的条件

#### （★）union all和 union的区别

Union 和 Union All 的区别之一在于对重复结果的处理。

UNION 在进行表链接后会筛选掉重复的记录， 所以在表链接后会对所产生的结果集进行排序运算， 删除重复的记录再返回结果。 实际大部分应用中是不会产生重复的记录， 最常见的是过程表与历史表 UNION。

而 UNION ALL 只是简单的将两个结果合并后就返回。 这样， 如果返回的两个结果集中有重复的数据， 那么返回的结果集就会包含重复的数据了。

从效率上说， UNION ALL 要比 UNION 快很多， 所以， 如果可以确认合并的两个结果集中不包含重复的数据的话， 那么就使用 UNION ALL

#### （★）Hive 的 sort by 和 order by 的区别

sort by：不是全局排序，其在数据进入reducer前完成排序

order by：会对输入做全局排序，因此只有一个reducer（多个reducer无法保证全局有序）。只有一个reducer，会导致当输入规模较大时，需要较长的计算时间。

## Hive文件压缩

#### （★★★）**Hive的几种文件格式**

1.textfile

2.sequencefile

3.rcfile

4.orc

5.自定义格式

#### （★）hive 中的压缩格式 RCFile、 TextFile、 SequenceFile 各有什么区别？

**TextFile**

文本文件，可以压缩，lzo,...Bzip2压缩文本压缩比较大，时间长，可切割,如果压缩文件不可切割，则由一个map处理大文件

**SequenceFile**

序列文件，kv对，格式紧凑。适合mr的output，可压缩可切割。

该文件和text文件都行方式存储。反之，rcFile/orc/parquet文件格式可以列存储。

SET hive.exec.compress.output=true;

SET io.seqfile.compression.type=BLOCK

**RCFile**

Record Columnar File.

kv存储，类似于sequencefile，将数据文件水平切割多个组。

若干group存放在一个hdfs中，先保存所有行的第一列，第二列，以此类推。

该文件可切割.可以跳过不相关部分，更快得到数据，成本更低。

参考博客：https://blog.csdn.net/tian\_qing\_lei/article/details/77484712

## Hive数据倾斜

#### （★★）数据倾斜问题怎么解决？

数据倾斜产生的原因为分区之后某一个reduce运算的数量比较小，而某一个reduce运行的数据量比较大，造成两个reduce处理数据不平等

合理设置map数量

总结：可以影响map的数量的因素都有哪些？

在input文件夹中，每一个文件就是一个map；input文件的数量, input的文件的大小 。

在MR任务中一个切片就是一个map任务 。

设置切片大小： 在hadoop中

FileInputFormat.setMaxInputSplitSize(job, size);

FileInputFormat.setMinInputSplitSize(job, size);

设置reduce个数

set mapreduce.job.reduces

根据业务自定分区规则

## Hive小文件处理

#### （★★）Hive文件合并

（1） Map操作之前合并小文件

① 每个Map最大输入大小设置为2GB（单位：字节）

setmapred.max.split.size=2048000000

② 执行Map前进行小文件合并

sethive.input.format=org.apache.hadoop.hive.ql.io.CombineHiveInputFormat

（2） 输出时进行合并

① #在Map-only的任务结束时合并小文件

sethive.merge.mapfiles = true

② #在Map-Reduce的任务结束时合并小文件

sethive.merge.mapredfiles= true

③ #合并文件后的大小为1GB左右

sethive.merge.size.per.task = 1024000000

④ #当输出文件的平均大小小于1GB时，启动一个独立的map-reduce任务进行文件merge

sethive.merge.smallfiles.avgsize=1024000000

（3） 如果需要压缩输出文件，就需要增加一个压缩编解码器，同时还有两个压缩方式和多种压缩编码器，压缩方式一个是压缩输出结果，一个是压缩中间结果，按照自己的需求选择，我需要的是gzip就选择的GzipCodec，同时也可以选择使用BZip2Codec、SnappyCodec、LzopCodec进行压缩。压缩文件：

① sethive.exec.compress.output=true;

#默认为false，是否对输出结果进行压缩

② setmapred.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec;

#压缩格式设置

③ setmapred.output.compression.type=BLOCK;

#一共三种压缩方式（NONE, RECORD,BLOCK），BLOCK压缩率最高，一般用BLOCK。

#### （★★）对于小文件的处理办法

我们可以通过一些配置项来使Hive在执行结束后对结果文件进行合并：

hive.merge.mapfiles 在map-only job后合并文件，默认true

hive.merge.mapredfiles 在map-reduce job后合并文件，默认false

hive.merge.size.per.task 合并后每个文件的大小，默认256000000

hive.merge.smallfiles.avgsize 平均文件大小，是决定是否执行合并操作的阈值，默认16000000

Hive在对结果文件进行合并时会执行一个额外的map-only脚本，mapper的数量是文件总大小除以size.per.task参数所得的值，触发合并的条件是：

根据查询类型不同，相应的mapfiles/mapredfiles参数需要打开；

结果文件的平均大小需要大于avgsize参数的值。

详见 https://blog.csdn.net/dxl342/article/details/50932062

## hive优化

#### **（★★★）性能调优（重点是 join 的时候如何放置大小表）**

这里只讲解hive大小表join优化性能，其他优化参考hive优化总结

1. 小、大表 join

在小表和大表进行join时，将小表放在前边，效率会高。hive会将小表进行缓存。

2．Mapjoin

使用mapjoin将小表放入内存，在map端和大表逐一匹配。从而省去reduce

当一个大表和一个或多个小表做JOIN时，最好使用MAPJOIN，性能比普通的JOIN要快很多。另外，MAPJOIN 还能解决数据倾斜的问题。 MAPJOIN的基本原理是：在小数据量情况下，SQL会将用户指定的小表全部加载到执行JOIN操作的程序的内存中，从而加快JOIN的执行速度。

参考博客：https://blog.csdn.net/u012036736/article/details/84978689

#### （★★）HIVE怎么优化？ //P

把hive sql当做mapreduce程序来读，理解hadoop的核心能力，是hive优化的根本。

参考博客：https://www.cnblogs.com/smartloli/p/4356660.html

##### 表的优化

在表的优化中第一点：当数据量比较大的时候常用的手段就是为拆分表，大表拆小表，分区表 ，临时表 。外部表

小表和大表join ,要把数据量的小的表放在join的左边，先进行缓存 ,这样减少表Join的时候内存的消耗量

##### 并行执行

并行执行与java多线程的异步和同步概念上差不多，在MR运行任务中，存在很多的MR任务可以进行执行。有些MR任务和下一个MR任务存在依赖关系，但是有些MR任务没有依赖关系。

例如：

存在依赖关系的MR 它输出就是下一个MR任务的输入。 对于没有依赖关系的MR任务就可以使用并行执行，在同一时间运行多个MR任务。这样在运行的过程中效 率就会得到提升

可以通过以下参数进行设置：

set hive.exec.parallel=true ; // 开启任务并行执行

设置多少个任务可以同时运行

set hive.exec.parallel.thread.number=8; //默认值为8个任务可以同时运行

##### 严格模式

Hive中提供有严格模式，为了防止一些查询。出现不好的影响。例如笛卡儿积。在严格模式下是不能运行的

<property>

<name>hive.mapred.mode</name>

<value>strict</value>

<description>

The mode in which the Hive operations are being performed.

In strict mode, some risky queries are not allowed to run. They include:

Cartesian Product.

No partition being picked up for a query.

Comparing bigints and strings.

Comparing bigints and doubles.

Orderby without limit.

</description>

</property>

默认值为：非严格模式 nonstrict

开启严格模式： strict

开启了严格模式，会对查询语句进行一些限制：

对于分区表： 必须存在where语句对分区表中分区字段进行条件过滤，否则，不允许执行该查询

对于使用order by的语句必须使用limit 进行限定,由于order by 之后所有的数据都会被分到一个reduce中那这样reduce操作的数据量太多了，可能时间过长卡死。所以为了防止reduce时间过程。在order by的时候必须给定 limit 减少redue处理的数据量

限制了笛卡儿积的查询 .主要在多表join中会出现。笛卡儿积的出现会造成性能极大的消耗

##### JVM重用

在hive执行计算任务的时候，会把的执行计划上传到yarn集群中进行提交，运行MR任务。每次进行任务的运行的时候都会开启一个JVM进程运行的MR任务。如果提交任务频繁过多就会造成JVM频繁的开启和关闭。在JVM开启和关闭的过程中会造成大量的资源浪费

在处理小文件的时候，由于map任务较多。所以JVM会频繁的开启和关闭。所以对于小文件的处理优化，主要减少JVM开启的次数

在 mapred-default.xml配置文件中有如下参数

<property>

<name>mapreduce.job.jvm.numtasks</name>

<value>10</value>

<description>How many tasks to run per jvm. If set to -1, there is

no limit.

</description>

</property>

也可以在hive中临时设置JVM重用任务的运行数量

set mapreduce.job.jvm.numtasks

##### 推测执行

由于集群中的资源分配不均等，或者每个集群中节点的硬件性能，会导致某个任务运行的时间快或者某个任务运行的时间慢，或者某个任务在运行的时候直接卡死了

为了防止某些任务，在运行过程中，拖慢了整个MR任务的进度。在运行慢的任务节点上开启相同的任务，如果时间比原来的任务运行的快则直接输出推测运行的任务

设置开启推测执行的参数：

<property>

<name>mapreduce.map.speculative</name>

<value>true</value>

<description>If true, then multiple instances of some map tasks

may be executed in parallel.</description>

</property>

在hadoop中默认开启推测执行。推测执行不是说一卡死就开启任务必须运行%5以上才会开启推测执行

在hive中通过set参数也可以进行设置

set mapreduce.map.speculative=true

默认值为true

推测执行分为map端的推测执行以及reduce端的推测执行

<property>

<name>mapreduce.reduce.speculative</name>

<value>true</value>

<description>If true, then multiple instances of some reduce tasks

may be executed in parallel.</description>

</property>

##### 执行计划

Hive中提供的可以查看Hql语句的执行计划，在执行计划中会生成抽象语法树，在语法树中会显示HQL语句之间的依赖关系以及执行过程。通过这些执行的过程和依赖可以对HQL语句进行优化

##### 虚拟列

虚拟列本身是一个不存在列，在数据查询的时候，可以通过虚拟列去查询数据的的路径，以及数据的偏移量，这两个内容都是hive中为用户提供的虚拟列进行的查询

## Hive对比其他

#### （★★）Pig 和Hive语法有什么不同

①Language

在Hive中可以执行插入/删除 等操作，但是Pig中我没有发现有可以插入数据的方法

②Schemas

Hive中至少还有一个“表”的概念，但是Pig中我认为是基本没有表的概念，所谓的表建立在Pig Latin脚本中，对与Pig更不要提metadata了。

③Partitions

Pig中没有表的概念，所以说到分区对于Pig来说基本免谈，如果跟Hive说“分区”(Partition)他还是能明白的

参考博客：<https://blog.csdn.net/bluejoe2000/article/details/41476465>

#### （★）hive 跟 hbase 的区别是？

|  |  |  |
| --- | --- | --- |
| 区别点 | Hive | Hbase |
| 数据库类型 | 不是数据库 | NoSQL数据库 |
| 处理类型 | 不支持批处理（OLAP） | 支持实时数据流处理（OLTP） |
| 数据模式 | 支持所有 | 无数据模式 |
| 延迟 | 高延迟 | 低延迟 |
| 数据库模型 | 关系型DBMS | 宽列存储 |

#### （★）hive 相对于 Oracle 来说有那些优点？

Hive是一个数据仓库工具，而Oracle是数据库，Hive能够存储海量数据；

Hive还可以进行数据分析；

Hive免费；

#### （★）Spark Sql为什么比Hive快呢？

使用的计算框架不同：

Spark SQL使用的是Spark框架，而如果Hive没有配置Hive On Spark的话，默认是基于MapReduce的，这种方式从提交查询到返回结果需要相当长的时间，而Spark的执行能力原本就比MapReduce快，因此Spark SQL比Hive要快。

#### （★）Hive和HBase对接

Hive与HBase整合的实现是利用两者本身对外的API接口互相通信来完成的，其具体工作交由Hive的lib目录中的hive-hbase-handler-★.jar工具类来实现

详见 https://blog.csdn.net/carl810224/article/details/52382885