**1.Fetch抓取（Hive可以避免进行MapReduce）**

Hive中对某些情况的查询可以不必使用MapReduce计算。例如：SELECT \* FROM employees;在这种情况下，Hive可以简单地读取employee对应的存储目录下的文件，然后输出查询结果到控制台。

在hive-default.xml.template文件中hive.fetch.task.conversion默认是more，老版本hive默认是minimal。

该属性修改为more以后，在全局查找、字段查找、limit查找等都不走mapreduce。

**2.开启本地模式**

有时Hive的输入数据量是非常小的。在这种情况下，为查询触发执行任务时消耗可能会比实际job的执行时间要多的多。对于大多数这种情况，Hive可以通过本地模式在单台机器上处理所有的任务。对于小数据集，执行时间可以明显被缩短。

用户可以通过设置hive.exec.mode.local.auto的值为true，来让Hive在适当的时候自动启动这个优化。

**3.Join优化**

3.1mapjoin

大表join小表

如果不指定MapJoin或者不符合MapJoin的条件，那么Hive解析器会将Join操作转换成Common Join，即：在Reduce阶段完成join。容易发生数据倾斜。可以用MapJoin把小表全部加载到内存在map端进行join，避免reducer处理。

map端join的参数设置：

设置自动选择mapjoin

set hive.auto.convert.join = true; -- 默认为true

大表小表的阈值设置：

set hive.mapjoin.smalltable.filesize= 25000000;

小表的输入文件大小的阈值（以字节为单位）;如果文件大小小于此阈值，它将尝试将common join转换为map join。

不等连接

select f.a,f.b from A f join B t where f.a > t.a；将连接条件写到where中mr会进行笛卡尔积，效率特别低，如果使用mapjoin将在map端进行不等值的join操作，进而提高效率

3.2大表join大表

3.2.1空key过滤，过滤掉为空的key，例如：SELECT a.\* FROM (SELECT \* FROM nullidtable WHERE id IS NOT NULL ) a JOIN ori b ON a.id = b.id;

3.2.2空key随机转换

有时虽然某个key为空对应的数据很多，但是相应的数据不是异常数据，必须要包含在join的结果中，此时我们可以表a中key为空的字段赋一个随机的值，使得数据随机均匀地分不到不同的reducer上。

为了解决这种情况，我们可以通过hive的rand函数，随记的给每一个为空的id赋上一个随机值，这样就不会造成数据倾斜。例如：

SELECT a.\* FROM nullidtable a LEFT JOIN ori b ON CASE WHEN a.id IS NULL THEN concat('hive', rand()) ELSE a.id END = b.id;

3.3Bucket-Mapjoin分桶join

大表join大表时，不足以放到内存中，如果想在map端进行join就要使用bucket-mapjoin。

实现方式：

两个join表在join key上都做hash bucket，并且把你打算复制的那个（相对）小表的bucket数设置为大表的倍数。这样数据就会按照key join，做hash bucket。小表依然复制到所有节点，Map join的时候，小表的每一组bucket加载成hashtable，与对应的一个大表bucket做局部join，这样每次只需要加载部分hashtable就可以了。

作用：

1.提高join查询效率

2.快速检测sql是否可用，可以达到快速开发目的

抽样检测

select \* from t\_bucket tablesample(bucket x out of y on xx);

如果表有别名tablesample函数紧跟着表名之后。

x表示从哪个bucket进行抽样,桶计数从1开始.y用来计算抽取数据的量，计算方式为分桶数/y.假设我们一共分了128个桶,y设置为32,则表示要抽取4个bucket,如果x为12,则抽取的数据来自于12/16/20/24….y的值可以为桶个数的公约数,可以为任意值.

条件：

1） set hive.optimize.bucketmapjoin = true;  
2） 一个表的bucket数是另一个表bucket数的整数倍  
3） bucket列 == join列  
4） 必须是应用在map join的场景中

注意：要向分桶表中填充成员，需要将 hive.enforce.bucketing 属性设置为 true。

分区表和分桶表的区别

分区表将一个文件按照分区字段放到不同的分区目录，提升查询性能，分区字段是额外的字段跟表字段无关

分桶表将一个文件按照分桶字段拆分多个文件，在特殊场景提升多表查询的性能，分桶字段是表的某个字段来进行分桶

3.4SMB JOIN sort merge bucket join

基于bucket-mapjoin的**有序bucket**，可实现在map端完成join操作，可以有效地减少或避免shuffle的数据量。

原理：首先进行排序，继而合并，然后放到所对应的bucket中去，bucket是hive中和分区表类似的技术，就是按照key进行hash，相同的hash值都放到相同的buck中去。在进行两个表联合的时候。我们首先进行分桶，在join会大幅度的对性能进行优化。也就是说，在进行联合的时候，是table1中的一小部分和table1中的一小部分进行联合，table联合都是等值连接，相同的key都放到了同一个bucket中去了，那么在联合的时候就会大幅度的减小无关项的扫描。

条件：

|  |  |
| --- | --- |
| **bucket mapjoin** | **SMB join** |
| set hive.optimize.bucketmapjoin = true; | --写入数据强制分桶  set hive.enforce.bucketing=true;  --写入数据强制排序  set hive.enforce.sorting=true;  --开启bucketmapjoin  set hive.optimize.bucketmapjoin = true;  --开启SMB Join  set hive.auto.convert.sortmerge.join=true;  set hive.optimize. bucketmapjoin.sortedmerge=true;  set hive.auto.convert.sortmerge.join.noconditionaltask=true; |
| 一个表的bucket数是另一个表bucket数的整数倍 | 小表的bucket数**=**大表bucket数 |
| bucket列 == join列 | Bucket 列 == Join 列 == **sort 列** |
| 必须是应用在map join的场景中 | 必须是应用在bucket mapjoin 的场景中 |

**4.SQL优化**

3.3.1列裁剪：读数据的时候只读查询所需的列，例如：select a,b from z where c < 10;

3.3.2分区裁剪：查询的时候加上分区的条件

3.3.3group by

默认情况下，Map阶段同一Key数据分发给一个reduce，当一个key数据过大时就倾斜了。并不是所有的聚合操作都需要在Reduce端完成，很多聚合操作都可以先在Map端进行部分聚合，最后在Reduce端得出最终结果。

开启Map端聚合参数设置

--（1）是否在Map端进行聚合，默认为True

set hive.map.aggr = true;

--（2）在Map端进行聚合操作的条目数目

set hive.groupby.mapaggr.checkinterval = 100000;

--（3）有数据倾斜的时候进行负载均衡（默认是false）

set hive.groupby.skewindata = true;

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。

3.3.4count(distinct)

数据量大的情况下，由于COUNT DISTINCT操作需要用一个Reduce Task来完成，这一个Reduce需要处理的数据量太大，就会导致整个Job很难完成，一般COUNT DISTINCT使用先GROUP BY再COUNT的方式替换

3.3.5笛卡尔积

尽量避免笛卡尔积，即避免join的时候不加on条件，或者无效的on条件，Hive只能使用1个reducer来完成笛卡尔积。例如select \* from a join b where a.id = b.id;应使用select \* from a join b on a.id = b.id; mysql对join做了

优化，会自动识别将where转换为on