问题导读

1.如何开启Fetch？

2.哪些情况下，hive执行不使用MapReduce？

3.本地模式适用于什么情况？

4.小表与大表新版本是否需要优化？

5.大表与大表空值比较多该如何优化？

6.如何在Map端join？

7.动态分区的作用是什么？

8.数据倾斜有哪些措施？

9.严格模式你认为什么情况下会用到？

1.Fetch抓取

1. set hive.fetch.task.conversion=more（默认）

Fetch 抓取是指，Hive 中对某些情况的查询可以不必使用 MapReduce 计算。

该属性设置为 more 以后，在全局查找、字段查找、limit 查找等都不走 MapReduce。 设置为none后所有类型的查找语句都要走MapReduce；

例如：

1 查询所有员工的所有信息

hive> select * from emp;

OK

1001.0 Tome null 5000.0 10.0

1002.0 Tome null 5000.0 10.0

7369.0 SMITH CLERK 4800.0 20.0

Time taken: 3.664 seconds, Fetched: 16 row(s)

2、查询员工信息：员工号 姓名 月薪

hive> select empno,ename,sal from emp;

OK

1001.0 Tome 5000.0

1002.0 Tome 5000.0

Time taken: 0.166 seconds, Fetched: 16 row(s)

3、查询员工信息：员工号 姓名 月薪 年薪

hive> select empno,ename,sal,sal*12 from emp;

OK

1001.0 Tome 5000.0 60000.0

1002.0 Tome 5000.0 60000.0

Time taken: 0.184 seconds, Fetched: 16 row(s)

2.本地模式

1. set hive.exec.mode.local.auto=true（开启本地模式）

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，本地模式只能运行一个reducer

①开启本地模式后需要设置local mr的最大输入数据量，当数据量小于这个值时采用local mr的方式

1. set hive.exec.mode.local.auto.inputbytes.max=134217728（默认）

②开启本地模式后需要设置local mr的最大输入文件个数，当数据量小于这个值时采用local mr的方式

1. set hive.exec.mode.local.auto.input.files.max=4（默认）

上面是开启的自动本地模式，Hive通过条件判断是否通过本地模式运行mapreduce任务。当然也有完全本地模式，需要开启下面选项

1. SET mapreduce.framework.name = local;
2. SET mapred.local.dir=/tmp/username/mapred/local；

如果是小集群建议完全开启，如果不是，不建议开启完全本地模式。

3.表的优化

3.1小表join大表（小表需要在左边.）

注:(新版的 hive 已经对小表 JOIN 大表和大表 JOIN 小表进行了优化。小表 放在左边和右边已经没有明显区别。)

3.2大表join大表

当一个表内有许多空值时会导致MapReduce过程中,空成为一个key值,对应的会有大量的value值, 而一个key的value会**一起**到达reduce造成内存不足;所以要想办法过滤这些空值.

【这里你是否明白一起达到，因为空值太多导致都到了同一个reduce，然后造成内存暴增，所以需要过滤】

(1).通过查询所有不为空的结果

```
1. insert overwrite table jointable select n.* from
2.    (select * from nullidtable where id is not null ) n left join ori o on n.id = o.id;
```

(2).查询出空值并给其赋上随机数,避免了key值为空

```
1. insert overwrite table jointable
2. select n.* from nullidtable n full join ori o on
3. case when n.id is null then concat('hive', rand()) else n.id end = o.id;
```

由于是随机数，所以可能会分配找到不同的reduce中，所以可以解决数据倾斜。

3.3MapJoin

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。

设置MapJoin

```
1. set hive.auto.convert.join = true(默认)
```

大表小表的阀门值设置(默认25M以下认为是小表):

```
1. set hive.mapjoin.smalltable.filesize=25000000;
```

3.4Group 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聚合都在reduce端完成，很多聚合操作都可以现在Map端进行部分聚合，最后在Reduce段得到结果

1.开启Map端聚合参数设置

是否在Map段进行聚合，默认为true

1. hive.map.aggr = true

2.在Map端进行聚合操作的条目数

1. hive.groupby.mapaggr.checkinterval = 100000

3.有数据倾斜的时候进行负载均衡（默认是false）

1. hive.groupby.skewindata = true

注：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。

3.5Count(Distinct)去重统计

Count Distinct是使用了一个mapreduce ，当数据较少时无影响当数据较大时 只使用一个MapReduce将很难完成job。可以通过Group BY分组实现多个MapReduce共同完成。更多参考

https://blog.csdn.net/xyh1re/article/details/81813995

3.6.行列过滤

- 列处理: 在select中，只拿需要的列，尽量使用分区过滤，少用select*
- 行处理: 在分区剪裁中，当使用外关联时，如果将副表的过滤条件写在where后面那么就会先全表关联，之后再过滤。

实例：

（1）测试先关联两张表，再用 where 条件过滤

```
1. hive (default)> select o.id from bigtable bjoin ori o on o.id = b.idwhere o.id <= 10;
```

（2）通过子查询后，再关联表

```
1. hive (default)> select b.id from bigtable b        join (select id from ori where id <= 10 ) o on b.id = o.id;
```

3.7.动态分区

关系型数据库中，对分区表 Insert 数据时候，数据库自动会根据分区字段的值，将数据插入到相应的分区中，Hive 中也提供了类似的机制，即动态分区(Dynamic Partition)，只不过，使用 Hive 的动态分区，需要进行相应的配置。

首先要设置的属性

```
1. set hive.exec.dynamic.partition = true;
2. set hive.exec.dynamic.partition.mode = nonstrict;
3. set hive.exec.max.dynamic.partitions = 1000;
4. set hive.exec.max.dynamic.partitions.pernode = 100;
5. set hive.exec.max.created.files = 100000;
6. set hive.error.on.empty.partition = false;
```

模拟动态分区

```
1. insert overwrite table ori_partitioned_target partition (p_time)
2. select id, time, uid, keyword, url_rank, click_num, click_url, p_time from ori_partitioned;
```

4.数据倾斜

4.1合理设置Map数

设置切片值： set mapreduce.input.fileinputformat.split.maxsize=

4.2小文件进行合并

在 map 执行前合并小文件，减少 map 数：CombineHiveInputFormat 具有对小文件进行

合并的功能（系统默认的格式）。HiveInputFormat 没有对小文件合并功能。

```
1. set hive.input.format=
org.apache.hadoop.hive.ql.io.CombineHiveInputFormat;
```

4.3复杂文件增加Map数

```
1. set mapreduce.job.maps
```

4.4合理设置Reduce数

①调整reduce的个数方法一

（1）每个Reduce处理的数据默认是256MB

```
1. hive.exec.reducers.bytes.per.reducer=256000000
```

（2）每个任务最大的reduce数，默认为1009

```
1. hive.exec.reducers.max=1009
```

（3）计算reduce数的公式

N=min(参数2，总输入数据量/参数1)

②调整reduce个数的方法二

```
1. set mapreduce.job.reduces=
```

③reduce个数不是越多越好

1）过多的启动和初始化 reduce 也会消耗时间和资源；

2）另外，有多少个 reduce，就会有多少个输出文件，如果生成了很多个小文件，那么如果这些小文件作为下一个任务的输入，则也会出现小文件过多的问题；在设置 reduce 个数的时候也需要考虑这两个原则：处理大数据量利用合适的 reduce 数；使单个 reduce 任务处理数据量大小要合适；

4.5并行执行

通过设置参数 hive.exec.parallel 值为 true，就可以开启并发执行。不过，在共享集群中，需要注意下，如果 job 中并行阶段增多，那么集群利用率就会增加。

①set hive.exec.parallel=true; //打开任务并行执行

①set hive.exec.parallel.thread.number=16; //同一个 sql 允许最大并行度，默认为 8。

4.6严格模式

Hive严格模式，在严格模式下会对可能产生较大查询结果的语句做限制，禁止其提交执行。

1.切换严格模式

查看当前的模式：

1. hive> set hive.mapred.mode;
2. hive.mapred.mode is undefined

未定义即为false，即no-strict模式。

开启严格模式：

1. set hive.mapred.mode=strict;

关闭严格模式：

1. set hive.mapred.mode=undefined;

2、严格模式严格在哪里

对于严格模式数据量特别大的时候，可能会用到，但是有些限制，某些情况开启可以提高性能、安全等。

1. 对分区表的查询必须使用到分区相关的字段

分区表的数据量通常都比较大，对分区表的查询必须使用到分区相关的字段，不允许扫描所有分区，想想也是如果扫描所有分区的话那么对表进行分区还有什么意义呢。

当然某些特殊情况可能还是需要扫描所有分区，这个时候就需要记得确保严格模式被关闭。

```
hive> select * from t_score ;
FAILED: SemanticException Queries against partitioned tables without a partition filter are disabled for safety reasons. If you know what you are do
ing, please sethive.strict.checks.large.query to false and that hive.mapred.mode is not set to 'strict' to proceed. Note that if you may get errors
or incorrect results if you make a mistake while using some of the unsafe features. No partition predicate for Alias "t_score" Table "t_score"
```

1.png

2. order by必须带limit

因为要保证全局有序需要将所有的数据拉到一个Reducer上，当数据集比较大时速度会很慢。个人猜测可能是设置了limit N之后就会有一个很简单的优化算法：每个Reducer排序取N然后再合并排序取N即可，可大大减少数据传输量。

```
hive> select * from t_score order by score;
FAILED: SemanticException 1:31 Order by-s without limit are disabled for safety reasons. If you know what you are doing, please sethive.strict.check
s.large.query to false and that hive.mapred.mode is not set to 'strict' to proceed. Note that if you may get errors or incorrect results if you make
 a mistake while using some of the unsafe features.. Error encountered near token 'score'
```

1.png

3. 禁止笛卡尔积查询（join必须有on连接条件）

hive不会对where中的连接条件优化为on，所以join必须带有on连接条件，不允许两个表直接相乘。

```
hive> select * from t_score, t_foobar ;
FAILED: SemanticException Cartesian products are disabled for safety reasons. If you know what you are doing, please sethive.strict.checks.cartesian
.product to false and that hive.mapred.mode is not set to 'strict' to proceed. Note that if you may get errors or incorrect results if you make a mi
stake while using some of the unsafe features.
```

2.png

4.JVM 重用

JVM 重用是 Hadoop 调优参数的内容，其对 Hive 的性能具有非常大的影响，特别是对于很难避免小文件的场景或 task 特别多的场景，这类场景大多数执行时间都很短。

Hadoop 的默认配置通常是使用派生 JVM 来执行 map 和 Reduce 任务的。这时 JVM 的启动过程可能会造成相当大的开销，尤其是执行的 job包含有成百上千 task任务的情况。JVM重用可以使得 JVM 实例在同一个 job 中重新使用 N 次。 N 的值可以在 Hadoop 的 mapred-site.xml 文件中进行配置。通常在 10-20 之间，具体多少需要根据具体业务场景测试得出。

```
1. <property>
2.     <name>mapreduce.job.jvm.numtasks</name>
3.     <value>10</value>
4.     <description>
5.         How many tasks to run per jvm. If set to -1, there is no limit.
6.     </description>
7. </property>
```

这个功能的缺点是，开启 JVM 重用将一直占用使用到的 task 插槽，以便进行重用，直到任务完成后才能释放。如果某个“不平衡的”job 中有某几个 reduce task 执行的时间要比其他 Reduce task 消耗的时间多的多的话，那么保留的插槽就会一直空闲着却无法被其他的 job使用，直到所有的 task 都结束了才会释放。