Hive SQL优化思路

Hive的优化主要分为：配置优化、SQL语句优化、任务优化等方案。

其中在开发过程中主要涉及到的可能是SQL优化这块。

优化的核心思想是：

- 减少数据量（例如分区、列剪裁）
- 避免数据倾斜（例如加参数、Key打散）
- 避免全表扫描（例如on添加加上分区等）
- 减少job数（例如相同的on条件的join放在一起作为一个任务）

HQL语句优化

1、使用分区剪裁、列剪裁

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。

```
select a.*
from test1 a
left join test2 b on  a.uid = b.uid
where a.ds='2020-08-10'
and b.ds='2020-08-10'
```

上面这个SQL主要是犯了两个错误：

1. 副表的过滤条件写在where后面，会导致先全表关联在过滤分区
2. on的条件没有过滤null值的情况，如果两个数据表存在大批量null值的情况，会造成数据倾斜。

```
select a.*
from test1 a
left join test2 b on (d.uid is not null and a.uid = b.uid and b.ds='2020-08-10')
where a.ds='2020-08-10'
```

如果null值也是需要的，那么需要在条件上转换，或者单独拿出来

```
select a.*
from test1 a
left join test2 b on (a.uid is not null and a.uid = b.uid and b.ds='2020-08-10')
where a.ds='2020-08-10'
union all
select a.* from test1 a where a.uid is null

或者
```

2、尽量不要用COUNT DISTINCT，因为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，虽然会多用一个Job来完成，但在数据量大的情况下，这个绝对是值得的。

```
select count(distinct uid)
from test
where ds='2020-08-10' and uid is not null

转换为

select count(a.uid)
from
(select uid from test where uid is not null and ds = '2020-08-10' group by uid) a
```

3、使用with as，因为拖慢hive查询效率出了join产生的shuffle以外，还有一个就是子查询，在SQL语句里面尽量减少子查询。with as是将语句中用到的子查询事先提取出来（类似临时表），使整个查询当中的所有模块都可以调用该查询结果。使用with as可以避免Hive对不同部分的相同子查询进行重复计算。

```
select a.*
from test1 a
left join test2 b on  a.uid = b.uid
where a.ds='2020-08-10'
and b.ds='2020-08-10'

可以转化为

with b
as
select uid
from test2
where ds = '2020-08-10' and uid is not null

select a.*
from test1 a
left join b on a.uid = b.uid
where a.ds='2020-08-10' and a.uid is not null
```

4、大小表的join，写有Join操作的查询语句时有一条原则：应该将条目少的表/子查询放在Join操作符的左边。原因是在Join操作的Reduce阶段，位于Join操作符左边的表的内容会被加载进内存，将条目少的表放在左边，可以有效减少发生OOM错误的几率。

但新版的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。

不过在做join的过程中通过小表在前可以适当的减少数据量，提高效率。

5、数据倾斜，数据倾斜的原理都知道，就是某一个或几个key占据了整个数据的90%，这样整个任务的效率都会被这个key的处理拖慢，同时也可能会因为相同的key会聚合到一起造成内存溢出。

数据倾斜只会发生在shuffle过程中。这里给大家罗列一些常用的并且可能会触发shuffle操作的算子：distinct、groupByKey、reduceByKey、aggregateByKey、join、cogroup、repartition等。出现数据倾斜时， 可能就是你的代码中使用了这些算子中的某一个所导致的。

hive的数据倾斜一般的处理方案：

```
常见的做法，通过参数调优：
set hive.map.aggr=true;
set hive.groupby.skewindata = ture;
```

当然这些优化都是针对SQL本身的优化，还有一些是通过参数设置去调整的，这里面就不再详细描述了。

但是优化的核心思想都差不多：

1. 减少数据量
2. 避免数据倾斜
3. 减少JOB数
4. 虚核心点：根据业务逻辑对业务实现的整体进行优化；
5. 虚解决方案：采用presto、 impala等专门的查询引擎，采用spark计算引擎替换MR/TEZ