Hive调优

高校大数据课程系列

ENTER

课程目标

Course objectives

1 掌握Hive优化方式

2 掌握Join优化案例

3 掌握参数设置优化案例

本章任务

Task of this chapter

1 Hive调优方式

2 Join优化案例

3 参数设置案例

Hive调优方式

1.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。

```
<property>
  <name>hive.fetch.task.conversion</name>
  <value>more</value>
  <description>
   Expects one of [none, minimal, more].
   Some select queries can be converted to single FETCH task minimizing latency.
   Currently the query should be single sourced not having any subquery and should not have
   any aggregations or distincts (which incurs RS), lateral views and joins.
   0. none : disable hive.fetch.task.conversion
   1. minimal : SELECT STAR, FILTER on partition columns, LIMIT only 所有的都要走MR
   2. more    : SELECT, FILTER, LIMIT only (support TABLESAMPLE and virtual columns)
  </description>
</property>
```

Hive调优方式

1.把hive.fetch.task.conversion设置成none，然后执行查询语句，都会执行mapreduce程序。

```
hive (default)> set hive.fetch.task.conversion=none;
hive (default)> select * from score;
hive (default)> select s_score from score;
hive (default)> select s_score from score limit 3;
```

1.把hive.fetch.task.conversion设置成more，然后执行查询语句，如下查询方式都不会执行mapreduce程序。

```
hive (default)> set hive.fetch.task.conversion=more;
hive (default)> select * from score;
hive (default)> select s_score from score;
hive (default)> select s_score from score limit 3;
```

Hive调优方式

2. 大多数的Hadoop Job是需要Hadoop提供完整的可扩展性来处理大数据集的。不过，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。

```
set hive.exec.mode.local.auto=true; //开启本地mr
//设置local  mr的最大输入数据量，当输入数据量小于这个值时采用local  mr的方式，默认为134217728，即128M
set hive.exec.mode.local.auto.inputbytes.max=51234560;
//设置local mr的最大输入文件个数，当输入文件个数小于这个值时采用local mr的方式，默认为4
set hive.exec.mode.local.auto.input.files.max=10;
```

Hive调优方式

2.开启本地模式，并执行查询语句。

```
hive (default)> set hive.exec.mode.local.auto=true;
hive (default)> select * from score cluster by s_id;
18 rows selected (1.568 seconds)
```

2.关闭本地模式，并执行查询语句。

```
hive (default)> set hive.exec.mode.local.auto=false;
hive (default)> select * from score cluster by s_id;
18 rows selected (11.865 seconds)
```

Hive调优方式

3.Join原则：

1）小表Join大表，

将key相对分散，并且数据量小的表放在join的左边，这样可以有效减少内存溢出错误发生的几率；再进一步，可以使用Group让小的维度表（1000条以下的记录条数）先进内存。在map端完成reduce。

●select count(distinct s_id) from score;

●select count(s_id) from score group by s_id; 在map端进行聚合，效率更高

2）多个表关联时，最好分拆成小段，避免大sql（无法控制中间Job）

3）大表Join大表

●有时join超时是因为某些key对应的数据太多，而相同key对应的数据都会发送到相同的reducer上，从而导致内存不够。此时我们应该仔细分析这些异常的key，很多情况下，这些key对应的数据是异常数据，我们需要在SQL语句中进行过滤。

●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布到不同的reducer上

Hive调优方式

4.MapJoin：

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。

开启MapJoin参数设置：

（1）设置自动选择Mapjoin

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

（2）大表小表的阈值设置（默认25M以下认为是小表）：

```
set hive.mapjoin.smalltable.filesize=25123456;
```

Hive调优方式

4.Group 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;
```

Hive调优方式

5.并行执行：

Hive会将一个查询转化成一个或者多个阶段。这样的阶段可以是MapReduce阶段、抽样阶段、合并阶段、limit阶段。或者Hive执行过程中可能需要的其他阶段。默认情况下，Hive一次只会执行一个阶段。不过，某个特定的job可能包含众多的阶段，而这些阶段可能并非完全互相依赖的，也就是说有些阶段是可以并行执行的，这样可能使得整个job的执行时间缩短。不过，如果有更多的阶段可以并行执行，那么job可能就越快完成。

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。

```
set hive.exec.parallel=true;                    //打开任务并行执行
set hive.exec.parallel.thread.number=16;  //同一个sql允许最大并行度，默认为8。
```

Hive调优方式

6.严格模式：

Hive提供了一个严格模式，可以防止用户执行那些可能意想不到的不好影响的查询。通过设置属性hive.mapred.mode值为默认是非严格模式nonstrict 。开启严格模式需要修改hive.mapred.mode值为strict，开启严格模式可以禁止3种类型的查询。

```
<property>
    <name>hive.mapred.mode</name>
    <value>strict</value>
</property>
```

1）对于分区表，除非where语句中含有分区字段过滤条件来限制范围，否则不允许执行。换句话说，就是用户不允许扫描所有分区。进行这个限制的原因是，通常分区表都拥有非常大的数据集，而且数据增加迅速。没有进行分区限制的查询可能会消耗令人不可接受的巨大资源来处理这个表。

2）对于使用了order by语句的查询，要求必须使用limit语句。因为order by为了执行排序过程会将所有的结果数据分发到同一个Reducer中进行处理，强制要求用户增加这个LIMIT语句可以防止Reducer额外执行很长一段时间。

3）限制笛卡尔积的查询。对关系型数据库非常了解的用户可能期望在执行JOIN查询的时候不使用ON语句而是使用where语句，这样关系数据库的执行优化器就可以高效地将WHERE语句转化成那个ON语句。不幸的是，Hive并不会执行这种优化，因此，如果表足够大，那么这个查询就会出现不可控的情况。

Hive调优方式

7.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。

```
<property>
  <name>mapreduce.job.jvm.numtasks</name>
  <value>10</value>
</property>
```

本章任务

Task of this chapter

1 Hive调优方式

2 Join优化案例

3 参数设置案例

任务1 Join优化案例

任务背景

用户只提交按HiveQL语法规则，建立业务需要声明的内容，Hive就会将其转换成本地或MapReduce job。在不了解Hive内部运行原理的情况下，更利于工程师们专注于手头上的事情。而内部复杂的查询解析、规划、优化和执行过程是由Hive开发团队多年的艰难工作来实现的，不过大部分时间用户可以直接无视这些内部逻辑。

但当进行一些系统设计或者应付大量数据时，学习下Hive背后的理论知识以及底层的一些实现细节，会更有利于高效、正确地使用Hive。如设置Map与Reduce数量，如何更有效地编写JOIN等等。

任务1 Join优化案例

任务需求

1）创建大表join_optimize_stocks（股票交易所编码、股票代码、股票交易日期、股票开盘价、股票最低价、股票收盘价、股票交易量和股票成交价）并导入数据。

2）创建小表 join_optimize_dividends（股票交易所编码、股票代码、股票交易日期、股息）并导入数据。

3）通过大表与大表连接分析，验证Hive表不能加入内存时的JOIN连接的MapReduce执行过程。

4）通过大表与小表连接分析，验证Hive小表加入内存的JOIN连接省略Reduce过程的优化策略。

任务1 Join优化案例

任务分析

启动Hadoop服务，查看安全模式的状态。启动Hive服务，进入Hive命令行客户端。创建大表join_optimize_stocks，创建小表 join_optimize_dividends并导入数据。通过大表与大表连接分析，验证Hive表不能加入内存时的JOIN连接的MapReduce执行过程。验证Hive小表加入内存的JOIN连接省略Reduce过程的优化策略。退出Hive环境，停止Hadoop服务。

任务1 Join优化案例

任务步骤

1、启动Hadoop服务,启动Hive服务

2、创建大表join_optimize_stocks（股票交易所编码、股票代码、股票交易日期、股票开盘价、股票最低价、股票收盘价、股票交易量和股票成交价）并导入数据。

3、创建小表 join_optimize_dividends（股票交易所编码、股票代码、股票交易日期、股息）并导入数据。

4、通过大表与大表连接分析，验证Hive表不能加入内存时的JOIN连接的MapReduce执行过程。

5、通过大表与小表连接分析，验证Hive小表加入内存的JOIN连接省略Reduce过程的优化策略。

任务1 Join优化案例

任务结果

Hive命令行客户端正常启动，成功执行相应Hive语句。

本章任务

Task of this chapter

1 Hive调优方式

2 Join优化案例

3 参数设置案例

任务2 参数设置案例

任务背景

用户只提交按HiveQL语法规则，建立业务需要声明的内容，Hive就会将其转换成本地或MapReduce job。在不了解Hive内部运行原理的情况下，更利于工程师们专注于手头上的事情。而内部复杂的查询解析、规划、优化和执行过程是由Hive开发团队多年的艰难工作来实现的，不过大部分时间用户可以直接无视这些内部逻辑。

但当进行一些系统设计或者应付大量数据时，学习下Hive背后的理论知识以及底层的一些实现细节，会更有利于高效、正确地使用Hive。如设置Map与Reduce数量，如何更有效地编写JOIN等等。

任务2 参数设置案例

任务需求

1）创建表set_param_stocks（股票交易所编码、股票代码、股票交易日期、股票开盘价、股票最低价、股票收盘价、股票交易量和股票成交价）并导入数据。

2）hive模式命令窗口参数设置底层运行的框架，对比查询数据所花费时间。只对当前会话有效。

3）hive-site.xml文件参数设置底层运行的框架，对比查询数据所花费时间，对所有hive用户永久生效。

任务2 参数设置案例

任务分析

启动Hadoop服务，查看安全模式的状态。启动Hive服务，进入Hive命令行客户端。创建表set_param_stocks，hive模式命令窗口参数设置底层运行的框架，hive-site.xml文件参数设置底层运行的框架。对比查询数据所花费时间。退出Hive环境，停止Hadoop服务。

任务2 参数设置案例

任务步骤

1、启动Hadoop服务，启动Hive服务。

2、创建表set_param_stocks（股票交易所编码、股票代码、股票交易日期、股票开盘价、股票最低价、股票收盘价、股票交易量和股票成交价）并导入数据。

3、hive模式命令窗口参数设置底层运行的框架，对比查询数据所花费时间。只对当前会话有效。

4、hive-site.xml文件参数设置底层运行的框架，对比查询数据所花费时间，对所有hive用户永久生效。

5、停止Hadoop服务，退出Hive环境。

任务2 参数设置案例

任务结果

Hive命令行客户端正常启动，成功执行相应Hive语句。

谢谢观看

THANKS FOR WATCHING