Hive11 调优

> 学习一下Hive背后的理论知识以及底层的一些实现细节，会让用户更加高效地使用Hive

使用EXPLAIN

```
hive> DESCRIBE onecol;
number int

hive> SLELECT * FROM onecol;
5
5
4

hive> SELECT SUM(number) FROM onecol;
14
```

现在，在前面例子中最后一个查询语句前加上EXPLAIN关键字。然后这个本身并不会执行

```
hive> EXPLAIN SELECT SUM(number) FROM onecol;
```

打印出抽象语法树

表明Hive是如何将查询解析成token(符号)和literal(字面值)的。

```
ABSTRACT SYNTAX TREE:
(TOK_QUERY
    (TOK_FROM (TOK_TABREF (TOK_TABNAME onecol)))
    (TOK_INSRT (TOK_DESTINATION (TOK_DIR TOK_TMP_FILE))
    (TOK_SELECT
        (TOK_SELEXPR
            (TOK_FUNCTION sum (TOK_TABLE_OR_COL number))))))
```

Hive会将输出写入到一个临时文件中：

```
'(TOK_INSERT (TOK_DESTINATION (TOK_DIR TOK_TMP_FILE)))'
```

接下来可以看到列名number， 以及表名 onecol， 以及sun函数。

STAGE PLAN

```
STAGE DEPENDENCIES
    Stage-1 is a root stage
    Stage-0 is a root stage
```

Stage-1 包含了这个job的大部分处理过程

```
STAGE PLANS:
  Stage: Stage-1
    Map Reduce
      Alias -> Map Operator Tree:
        onecol
          TableScan
            alias: onecol
            Select Operator
              expressions:
                    expr: number
                    type: int
              outputColumnNames: number
              Group By Operator
                aggregations:
                      expr: sum(number)
                bucketGroup: false
                mode: hash
                outputColumnNames: _col0
                Reduce Output Operator
                  sort order:
                  tag: -1
                  value expressions:
                        expr: _col0
                        type: bigint
```

Map阶段发生的事件:

> - 触发一个MapReduce job
> - TableScan 以这个表作为输入，产生一个 number 输出
> - Group By Operator 会应用到 sum(number)， 产生输出字段 _col0

```
Reduce Operator Tree:
  Group By Operator
    aggregations:
          expr: sum(VALUE._col0)
    bucketGroup: false
    mode: mergepartial
    outputColumnNames: _col0
    Select Operator
      expressions:
            expr: _col0
            type: bigint
```

```
outputColumnNames: _col0
File Output Operator
  compressed: false
  GlobalTableId: 0
  table:
      input format: org.apache.hadoop.mapred.TextInputFormat
      output format:
          org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
```

> reduce阶段发生的事件：
>
> - Group by Operator 对 _col0 进行 sum 操作
> - File Output Operator 输出格式

Stage-0 没有任何操作

因为这个job没有LIMIT语句，因此Stage-0阶段是一个没有任何操作的阶段

```
Stage: Stage-0
  Fetch Operator
    limit: -1
```

EXPLAIN EXTENDED

> 使用 EXPLAIN EXTENDED 语句可以产生更多的输出信息

限制调整

> LIMIT 语句是大家经常使用到的， 经常使用CLI的用户都会使用到。在很多情况下LIMIT 语句还是需要执行整个查询语句，然后再返回部分结果的。因为这种情况通常是浪费的，所以应该尽可能地避免出现这种情况。

Hive有一个配置属性可以开启，当使用LIMIT 语句时，其可以对源数据进行抽样：

```
<property>
    <name> hive.limit.optimize.enable</name>
    <value>true</true>
    <desription>Whether to enable to optimization to try a smaller subset of
data for simple LIMIT first.</description>
</property>
```

一旦属性 hive.limit.optimize.enable 的值设置为 true ， 那么还会有两个参数可以控制这个操作，也就是 hive.limit.row.max.size 和 hive.limit.optimize.limit.file:

```
<property>
    <name> hive.limit.row.max.size</name>
    <value>10000</value>
    <description> When trying a smaller subset of data for simple LIMIT,hoow
much size we need to gurantee each row to have at least.
    </description>
</property>

<property>
    <name> hive.limit.optimize.limit.file</name>
    <value>10</value>
    <description> When trying a smaller subset of data for simple LIMIT,maxium
number of files we can sample.
    </description>
</property>
```

> 确定：可能输入中有用的数据永远不会被处理到。

JOIN优化

> 背景：如果所有表中只有一张表是小表，那么可以在最大的表通过mapper的时候将小表完全放到内存中。

> 原因：Hive可以和内存中的小表进行逐一匹配，从而省略掉常规连接操作所需要的reduce过程。

```
hive> set hive.auto.convert.join=true;

hive>SELECT /*+ MAPJOIN(d)*/ s.ymd, s.symbol, s.price_close, d.dividend
    >FROM stocks s JOIN dividends d ON s.ymd = d.ymd AND s.symbol = d.symbol
    >WHERE s.symbol = 'AAPL';
```

这个执行速度快了大概30%；

> 如果所有表中有一个表足够小，是完全可以载入到内存中的，那么这个时候Hive可以执行一个map-side JOIN ，这样可以减少reduce过程。有时甚至可以减少某些map task任务。有时候即使某些表不适合载入内存也可以使用map-JOIN,因为减少reduce阶段可能比将不太好的表分发到每个map task 中会带来更多的好处。

本地模式

> 大多数的Hadoop Job是需要Hadoop提供的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，这种情况下，为查询触发执行任务的时间消耗可能会比实际job的执行时间要更多。这种情况下，Hive可以通过本地模式再单台机器上处理所有任务。对于小数据集，执行时间可以明显被缩短。

- 临时启用本地模式

```
hive> set oldjobtracker=${hiveconf:mapred.job.tracker};

hive> set mapred.job.tracker=local;

hive> set mapred.tmp.dir = /home/edward/tmp;

hive> SELECT * FROM prople WHERE firstname=bob;
...

hive> set mapred.job.tracker=${oldjobtracker};
```

用户可以通过设置属性 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。用户通常可以将这个配置写在 $HOME/.hiverc文件中。

- 所有用户都使用这个配置，将这个配置项加到 $HIVE_HOME/conf/hive-site.xml中：

```
<property>
    <name> hive.exec.mode.local.auto</name>
    <value>true</value>
    <description> Let hive determine whether to run in local mode
automatically
    </description>
</property>
```

并行执行

> Hive会将一个查询转化为一个或者多个阶段，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执行过程中需要其他阶段。默认情况下， Hive一次只能执行一个阶段。不过有些阶段是可以并行执行的。

通过设置参数hive.exec.parallel值设为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中并行执行的阶段增多，那么集群利用率就会增加：

```
<property>
    <name> hive.exec.mode.parallel</name>
    <value>true</value>
    <description> Whteteher to execute jobs in parallel
    </description>
</property>
```

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产生意想不到的不好的影响的查询。

通过设置属性 hive.mapred.mode 值为 strict 可以禁止3种类型的查询。

对于分区表

禁止：除非WHERE语句中含有分区字段过滤条件来限制数据范围，否则不允许执行。（用户不允许扫描所有分区）。

原因： 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。

```
hive> SELECT DISTINCT(planner_id) FROM fracture_ins WHERE planner_id=5;

FAILED:Error in semantic analysis:No Partition Predicate Found for Alias
"fracture_ins" Table "fracture_ins"
```

如果这个语句WHERE语句增加了一个分区过滤条件：

```
hive> SELECT DISTINCT(planner_id) FROM fracture_ins
    > WHERE planner_id=5 AND hit_date=20120101;

... normal result ...
```

对于 ORDER BY 语句

禁止： 对于使用了 ORDER BY 语句的查询，要求必须使用LIMIT 语句。

原因：ORDER BY 为了执行排序过程会将所有的数据分发到同一个 redurce中处理，强制要求用户增加这个LINMIT语句可以防止reducer额外执行很长一段时间

```
hive> SELECT * FROM fracture_ins WHERE hit_date > 2012 ORDER BY planner_id;

FAILED:Error in semantic analysis: line 1:56 In strict mode, limit must be
specified if ORDER BY is present planner_id
```

只需要增加LIMIT就可以解决这个问题

```
hive> SELECT * FROM fracture_ins WHERE hit_date>2012 ORDER BY planner_id
    > LIMIT 10000;

... normal result ...
```

对于笛卡尔积的限制

禁止：JOIN 和 WHERE连接

原因：对于关系型数据库非常了解的用户可能期望在执行JOIN查询的时候不适用ON 语句而是WHERE语句。这样关系型数据库的执行优化器就可以高效将WHERE语句转化成那个ON语句。但是Hive并没有这个优化。

```
hive> SELECT * FROM fracture_act JOIN fracture_ads
    > WHERE fracture_act.planner_id = fracture_ads.planner_id;

FALLED: Error in semantic analysis: In strict mode, cartesian product is not
allowed. If you readlly want to perform the operation, +set hive.mapred.mode =
nonstrict+
```

下面这个才是正确的JOIN 和ON 语句的查询：

```
hive> SELECT * FROM fracture_act JOIN fracture_ads
    > ON (fracture_act.planner_id = fracture_ads.planner_id);

... normal results ...
```

调整 mapper 和 reducer个数

> Hive 通过将查询划分成一个或者多个MapReduce任务达到并行的目的。每个任务都可能具有多个mapper 和 reducer 任务，其中至少有一些可以并行执行的。确定最佳的个数取决于多个变量。
>
> 背景：如果有太多的mapper 或 reducer 任务，就会导致启动阶段、调度和运行job过程中产生过多的开销；如果设置的数量太少，那么就可能没有充分利用好集群内在的并行性；

当执行查询具有reduce过程的时候，CLI控制台会打印出调优后的reducer个数。

```
hive> SELECT pixel_id , cont FROM fracture_ins WHERE hit_date = 20120119
    > GROUP BY pixel_id;

Total MapReduce jobs = 1
Launching Jobs = 1
Number of reduce tasks not specified. Estimated from input data size: 3
...
```

Hive 是按照输入的数据量的大小来确定reducer个数的，我们可以通过dfs -count 命令来计算输入量大小，这个命令和Linux中的du -s 命令类似；其可以计算指定目录夏所有数据的总大小：

```
$ hadoop dfs -count /user/media6/fracture/ins/* | tail -3
1  8 261408737  hdfs://.../usr/media6/fracture/ins/hit_date=20120118
1  7 2612318737  hdfs://.../usr/media6/fracture/ins/hit_date=20120118
1  17 262218737  hdfs://.../usr/media6/fracture/ins/hit_date=20120118
```

属性 hive.exec.reducers.bytes.per.reducer 默认值是 1GB 。如果将这个属性调整为 750MB的话，那么下面这个任务Hive就会使用 4 个 reducer：

```
hive> set hive.exec.reducers.bytes.per.reducer=750000000;

hive> SELECT pixel_id, count(1) FROM fracture_ins WHERE hit_date = 20120119
Total MapReduce jobs = 1
Launching Jobs = 1
Number of reduce tasks not specified. Estimated from input data size: 4
```

默认值通常是比较适合的，但是也有下面两种情况：

- 如果map阶段产生的数据量非常多，那么根据输入的数据量大小来确定的reducer个数就会显得很少
- map阶段可能会过滤掉输入数据集中很大一部分的数据而这时可能需要少量的reducer就满足计算了。

一种快速的验证方法就是将 reducer个数设置为固定的值，而无需Hive来计算这个值。可以通过设置mapred.reducer.tasks的值为不同的值来确定是使用较多还是较少的reducer来缩短执行时间。

> 当在共享集群上处理大任务时，为了控制资源利用情况，属性 hive.exec.reducers.max显得非常重要，一个Hadoop集群可以提供的map 和 reducer资源个数是固定的。某个大job可能会消耗完所有的插槽，导致其他job无法执行。
>
> 通过设置hive.exec.reducers.max可以阻止某个查询消耗太多的reduce资源。有必要将这个属性配置到 $HIVE_HOME/conf/hive-site.xml 文件中。这个属性大小的一个建议的计算公式如下：

（集群总 Reduce槽位个数 * 1.5 ） / （执行中的查询的平均个数）

JVM 重用

> 对Hive的性能具有非常大的影响，特别是对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。当执行的job包含成百上千个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文件中进行设置：

```
<property>
    <name>mapred.job.reuse.jvm.num.task</name>
    <value>10</value>
    <description> How many tasks to run per jvm. IF set to -1, there is no
limit.
    </description>
</properity>
```

这个功能的一个缺点是：开启JVM重用会一直占用使用到的task插槽，以便进行重用。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 消耗的时间多得多，那么保留的插槽就会一直空闲着无法被其他的job使用，直到所有的task都结束了才会释放。

索引

Bitmap索引

> bitmap索引普遍应用于排重后较少的列。

```
CREATE INDEX employees_index
ON TABLE employees(country)
AS 'BITMAP'
WITH DEFERRED REBUILD
IDXPROPERTIED('creator' = 'me', 'ceated_at' = 'some_time')
IN TABLE employees_index_table
PARTITIONED BY (country, name)
COMMENT 'Employees indexes by country and name.';
```

动态分区调整

开启严格模式

开启严格模式的时候，必须保证至少有一个分区是静态的。

```
<property>
    <name>hive.exec.dynamic.partition.mode</name>
    <vlaue>strict</value>
    <description> In strict mode, the user must specify at least one static
partition in case the user accidentally overwrites all partitions.</description>
</property>
```

然后可以增加一些相关的属性信息，通过下面属性来限制可以创建的最大动态分区个数：

```
<property>
    <name>hive.exec.max.dynamic.partitions</name>
    <vlaue>300000</value>
    <description> Maximum number of dynamic partitions allowed to be created in
total.</description>
</property>

<property>
    <name>hive.exec.max.dynamic.partitions.pernode</name>
    <vlaue>10000</value>
    <description> Maximum number of dynamic partitions allowed to be created in
each mapper/reducer node.</description>
</property>
```

还有一个配置是来控制DataNode上一次可以打开的文件的个数。这个参数在$HADOOP_HOME/conf/hdfs-site.xml配置文件中，默认值是256太小了。

```
<property>
    <name>dfs,datanode.max.xcievers</name>
    <value>8192</value>
</property>
```

推测执行

> 推测执行是Hadoop中的一个功能，其可以触发执行一些重复的任务(task).目的是通过加快获取单个tsak的结果以及进行侦测将执行慢的TakTracker加入到黑名单的方式来提高整体的任务执行效率。

在$HADOOP_HOME/conf/mapred-site.xml文件配置如下：

```
<property>
    <name>mapred.map.tasks.speculative.execution</name>
    <value>true</value>
    <description>If true, then multiple instances of some map tasks may be
excuted in parallel.</description>
</property>

<property>
    <name>mapred.reduce.tasks.speculative.execution</name>
    <value>true</value>
    <description>If true, then multiple instances of some reduce tasks may be
excuted in parallel.</description>
</property>
```

不过，Hive本身也是提供了配置项来控制 reduce-side的推测执行：

```
<property>
    <name>mapred.reduce.tasks.speculative.execution</name>
    <value>true</value>
    <description>If true, then multiple instances of some reduce tasks may be
excuted in parallel.</description>
</property>
```

单个MapReduce中多个 GROUP BY

> 另一个特别的优化试图查询中的多个GROUP BY 操作组装到单个MapReduce任务。

如果想要启动这个优化，需要一组常用的 GROUP BY 键

```
<property>
    <name>hive.multigroupby.singlemr</name>
    <value>false</value>
    <description>Whether to optimize multi group by query to generate single M/R
job plan.If the multi group by query has common group by keys,it will be
optimized to generate single M/R job.</description>
</property>
```

虚拟列

两种虚拟列：

- 用于将要进行划分的输入文件名
- 用于文件中的块内偏移量

```
hive> set hive.exec.rowoffset=true;

hive> SELECT INPUT_FILE_NAME, BLOCK_INSIDE_FILE, line
    > FROM hive_text WHERE line LIKE '%hive%' LIMIT 2;
har://file/user/hive/warehouse/hive_text/folder=docs/data.har/user/hive/warehous
e/hive_text/folder=docs/README.txt 2243
    http://hive.apache.org/

har://file/user/hive/warehouse/hive_text/folder=docs/data.har/user/hive/warehous
e/hive_text/folder=docs/README.txt 3646
    http://hive.apache.org/
```

第三种虚拟列提供了文件的行偏移量，这个需要通过如下参数显式地启用：

```
<property>
    <name>hive.exec.rowoffset</name>
    <value>true</value>
    <description>Whether to provide the row offset virtual column</description>
</property>
```

就可以在如下查询中使用了：

```
hive> SELECT INPUT_FILE_NAME, BLOCK_OFFSET_INSIDE_FILE,
    > ROW_OFFSET_INSIDE_BLOCK
    > FROM hive_text WHERE line LIKE '%hive%' limit 2;
file:/user/hive/warehouse/hive_text/folder=docs/README.txt 2243 0
file:/user/hive/warehouse/hive_text/folder=docs/README.txt 3646 0
```