Hive调优

- What?Why?How?

Hive优化

- Hive 优化
  - 核心思想：把Hive SQL 当做Mapreduce程序去优化
  - 以下SQL不会转为Mapreduce来执行
    - select仅查询本表字段
    - where仅对本表字段做条件过滤
- Explain 显示执行计划
  - EXPLAIN [EXTENDED] query

Hive优化

- Hive抓取策略:
  - Hive中对某些情况的查询不需要使用MapReduce计算
- 抓取策略
  - Set hive.fetch.task.conversion=none/more;

Hive优化

- Hive运行方式：
  - 本地模式
  - 集群模式
- 本地模式
  - 开启本地模式：
    - set hive.exec.mode.local.auto=true;
    - 注意：
      - hive.exec.mode.local.auto.inputbytes.max默认值为128M
      - 表示加载文件的最大值，若大于该配置仍会以集群方式来运行！

Hive优化

- 并行计算
  - 通过设置以下参数开启并行模式：
    - set hive.exec.parallel=true;
  - 注意：hive.exec.parallel.thread.number
    - （一次SQL计算中允许并行执行的job个数的最大值）

Hive优化

- 严格模式
  - 通过设置以下参数开启严格模式：
    - set hive.mapred.mode=strict;
      - （默认为：nonstrict非严格模式）
  - 查询限制：
    - 1、对于分区表，必须添加where对于分区字段的条件过滤；
    - 2、order by语句必须包含limit输出限制；
    - 3、限制执行笛卡尔积的查询。

Hive优化

- Hive排序
  - Order By - 对于查询结果做全排序，只允许有一个reduce处理
    - （当数据量较大时，应慎用。严格模式下，必须结合limit来使用）
  - Sort By - 对于单个reduce的数据进行排序
  - Distribute By - 分区排序，经常和Sort By结合使用
  - Cluster By - 相当于 Sort By + Distribute By
    - （Cluster By不能通过asc、desc的方式指定排序规则；

      可通过 distribute by column sort by column asc|desc 的方式）

Hive优化

- Hive Join
  - Map Join：在Map端完成Join
    - 两种实现方式：
      - 1、SQL方式，在SQL语句中添加MapJoin标记（mapjoin hint）
        - 语法：

```
SELECT /*+ MAPJOIN(smallTable) */ smallTable.key, bigTable.value
FROM smallTable JOIN bigTable ON smallTable.key = bigTable.key;
```

      - 2、开启自动的MapJoin

Hive优化

- Hive Join
  - 自动的mapjoin
  - 通过修改以下配置启用自动的mapjoin：
    - set hive.auto.convert.join = true;
      - （该参数为true时，Hive自动对左边的表统计量，如果是小表就加入内存，即对小表使用Map join）
  - 相关配置参数：
    - hive.mapjoin.smalltable.filesize;
      - （大表小表判断的阈值，如果表的大小小于该值则会被加载到内存中运行）
    - hive.ignore.mapjoin.hint；
      - （默认值：true；是否忽略mapjoin hint 即mapjoin标记）

Hive优化

- Hive Join
  - 尽可能使用相同的连接键（会转化为一个MapReduce作业）
- 大表join大表
  - 空key过滤：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。
  - 空key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

Hive优化

- Map-Side聚合
  - 通过设置以下参数开启在Map端的聚合：
    - set hive.map.aggr=true;
  - 相关配置参数：
    - hive.groupby.mapaggr.checkinterval：
      - map端group by执行聚合时处理的多少行数据（默认：100000）
    - hive.map.aggr.hash.min.reduction：
      - 进行聚合的最小比例（预先对100000条数据做聚合，若聚合之后的数据量/100000的值大于该配置0.5，则不会聚合）
    - hive.map.aggr.hash.percentmemory：
      - map端聚合使用的内存的最大值
    - hive.groupby.skewindata
      - 是否对GroupBy产生的数据倾斜做优化，默认为false

Hive优化

- 合并小文件
  - 文件数目小，容易在文件存储端造成压力，给hdfs造成压力，影响效率
  - 设置合并属性
    - 是否合并map输出文件：hive.merge.mapfiles=true
    - 是否合并reduce输出文件：hive.merge.mapredfiles=true;
    - 合并文件的大小：hive.merge.size.per.task=256*1000*1000
- 去重统计
  - 数据量小的时候无所谓，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

Hive优化

- 控制Hive中Map以及Reduce的数量
  - Map数量相关的参数
    - mapred.max.split.size
      - 一个split的最大值，即每个map处理文件的最大值
    - mapred.min.split.size.per.node
      - 一个节点上split的最小值
    - mapred.min.split.size.per.rack
      - 一个机架上split的最小值
  - Reduce数量相关的参数
    - mapred.reduce.tasks
      - 强制指定reduce任务的数量
    - hive.exec.reducers.bytes.per.reducer
      - 每个reduce任务处理的数据量
    - hive.exec.reducers.max
      - 每个任务最大的reduce数

Hive优化

- Hive - JVM重用
  - 适用场景：
    - 1、小文件个数过多
    - 2、 task个数过多
  - 通过 set mapred.job.reuse.jvm.num.tasks=n; 来设置
    - （n为task插槽个数）
  - 缺点：设置开启之后，task插槽会一直占用资源，不论是否有task运行，直到所有的task即整个job全部执行完成时，才会释放所有的task插槽资源！