explain sql查看sql的执行计划

Parquet Block 要选择合适的大小，太大不能充分利用集群的并行能力，太小文件的目录太复杂

新导入数据后，可以执行一下统计信息：

compute incremental stats dw.dw\_profile\_tag\_user\_cookie\_service partition ( data\_date = '20180829');  
compute incremental stats dw.dw\_profile\_tag\_user\_userid\_service partition ( data\_date = '20180829');

最基本的应对性能调优的方案，主要包括：使用Parquet格式存储数据、分区粒度要确定好，保证整个表的分区数不要太多（目录不要太多），每一个分区下不要存在过多的小文件（选择合适的Parquet文件大小），收集统计信息使得查询优化器能够选择更好的查询方案，最后要学会使用EXPLAIN和Profile功能分析性能问题所在。

show table stats dw.dw\_profile\_tag\_user\_userid\_service

查看 impala表信息，包括存储格式，hdfs，数据大小等等

show tables

显示所有表

 1、SQL优化，使用之前调用执行计划  
　　• 2、选择合适的文件格式进行存储  
　　• 3、避免产生很多小文件（如果有其他程序产生的小文件，可以使用中间表）  
　　• 4、使用合适的分区技术，根据分区粒度测算  
　　• 5、使用compute stats进行表信息搜集  
　　• 6、网络io的优化：  
　　　　– a.避免把整个数据发送到客户端  
　　　　– b.尽可能的做条件过滤  
　　　　– c.使用limit字句  
　　　　– d.输出文件时，避免使用美化输出  
　　• 7、使用profile输出底层信息计划，在做相应环境优化

　　• 8、如果是刷新表的新增元数据要使用refresh 表名 来刷新，不要使用impala-shell -r 或  invalidate metadata

　　• 9、如果执行SQL的结果内容较多的话可以使用 impala-shell -B 将一些不必要的样式输出去掉

1）根据数据大小选择合适的文件格式。对于每个表或分区都达很多G的数据，使用Parquet是性能表现最好的。   
2）基于数据大小选择分区粒度。通常来说，最好保证每个分区的数据不小于256MB。过度分区会导致查询很多不需的分区而造成查询太长。理想情况下，表的分区数要低于3万个。在每个分区目录下，保存的文件应该是几个大文件，而不要保存很多小文件。如果接收的数据本身是很多小文件，要考虑使用INSERT …SELECT语句将一个表或分区的数据复杂到其他表或分区，这会压缩文件达到较少的数量。   
3）选择一个合适的Parquet block尺寸。   
4）将传输结果回客户端的最小负载最小化。   
5）验证你的查询的逻辑是不是高效的。执行之前，通过EXPLAIN检测一下。   
6）验证查询的性能特征。IO、内存使用、网络带宽、CPU利用率等等。   
7）使用合适的操作系统设置

impala不支持预编译 报错

java.sql.SQLException: Method not supported

at org.apache.hive.jdbc.HiveStatement.getMaxFieldSize(HiveStatement.java:498)

|  |
| --- |
| <select id="analyzeCategoryStructure" statementType="STATEMENT" resultType="com.jollychic.data.yuce.common.model.usercategory.dos.impala.AnalyzeCategoryStructureDO">  ${analyzeCategoryStructureSql} </select> |

statmentType="STATEMENT"

mybatis，在不用预编译的时候，即设置参数statmentType="STATEMENT"，需要用${}获取参数值，不能用#{}

impala函数

指数函数

pow(基数，指数)

平均值函数

avg(数据)

取小于数据的最大整数

floor(数据)

取大于数据的最小整数

ceil(数据)

求和

sum(数据)

最大值

max(数据)

最小值

min(数据)

四舍五入

round(数据，保留的小数位数)

新导入数据，查询报错：

INVALIDATE METADATA 表名

查看分区表的所有分区

SHOW PARTITIONS dw.dw\_category\_userid\_analyze;

删除分区指定分区

alter table 表名 drop partition (分区字段 <=330295)

刷新表

refresh 表名;

新建分群表

create table dw.dw\_category\_userid\_analyze(tagid string comment '标签id', userid string comment 'userid', tagweight double comment '标签值')

PARTITIONED BY (categoryId int COMMENT '分群id') comment '分群（基于userid）分析表'

create table dw.dw\_category\_cookieid\_analyze(tagid string comment '标签id', cookieid string comment 'cookieid', tagweight double comment '标签值')

PARTITIONED BY (categoryId int COMMENT '分群id') comment '分群（基于cookieid）分析表'

查看表的状态

show table stats 表名;

新插入数据，可以做一下优化

invalidate metadata db.table\  
  
  
compute incremental stats db.table partition (data\_date='20180813')

INVALIDATE METADATA 表名

INVALIDATE METADATA是用于刷新全库或者某个表的元数据，包括表的元数据和表内的文件数据，它会首先清楚表的缓存，然后从metastore中重新加载全部数据并缓存，该操作代价比较重，主要用于在hive中修改了表的元数据，需要同步到impalad

 invalidate metadata操作比refresh要重量级

 如果涉及到表的schema改变，使用invalidate metadata [table]

 如果只是涉及到表的数据改变，使用refresh [table]

 如果只是涉及到表的某一个分区数据改变，使用refresh [table] partition [partition]

 禁止使用invalidate metadata什么都不加，宁愿重启catalogd。 --------------------- 本文来自 教练\_我要踢球 的CSDN 博客 ，全文地址请点击：https://blog.csdn.net/yu616568/article/details/72780346?utm\_source=copy

compute incremental stats db.table partition (data\_date='20180813')

COMPUTE INCREMENTAL STATS（DDL）命令

该命令增量的对各分区进行统计，并且当分区变动时只对变化部分进行扫描，并更新相应的统计信息

看了一下会缩减很多数据量，有点想压缩的感觉

set request\_pool='root.userprofile' 设置请求使用资源池

java访问impala使用指定资源池

jdbc:impala://localhost:18000[/default2];AuthMech=3;

UID=cloudera;PWD=cloudera;MEM\_LIMIT=1000000000;REQUEST\_POOL=myPool