|  |  |
| --- | --- |
| **问题** | **答案** |
| hive 数据倾斜？ | 现象：通常发生在涉及到数据shuffle的操作（如group by，count(distinct)，join等），由于数据分布不均匀，出现其他reducer已经完成，某一节点reduce需要处理大量的数据，很长时间仍然没有结束。相关设置：  \* set hive.exec.reducers.max=200;  \* set mapred.reduce.tasks=200; ---增大Reduce个数  \* set hive.groupby.mapaggr.checkinterval=100000 ;--这个是group的键对应的记录条数超过这个值则会进行分拆,值根据具体数据量设置  \* set hive.groupby.skewindata=true; --如果是group by过程出现倾斜 应该设置为true  \* set hive.skewjoin.key=100000; --这个是join的键对应的记录条数超过这个值则会进行分拆,值根据具体数据量设置  \* set hive.optimize.skewjoin=true;--如果是join 过程出现倾斜 应该设置为true |
| 什么时候应该使用Sort by而不是Order by？ | 一般我们总应该使用Sort by，当输出数据集的全局顺序不是那么重要时。如果你要输出一个很大的数据集，Sort by会使用多个reducer进行排序，只保证输出的数据在同一个reducer内是有序的，而Order by只会使用一个reducer来排序，会消耗很长的时间。 |
| 调优：distribute by 与 cluster by |  |
| Hive分区？ | \* 分区目的：在创建表的时候将数据分块管理，提高查询效率  \* 分区实现：不同的分区在hdfs上体现为不同的数据目录  \* 分区方式：基本原则：按照经常使用的where查询条件字段分区，如日期，业务字段等  \* 动态分区：在运行时才能确定的分区值，一般是数据中的某个字段  \* 静态分区：在创建时即确定的分区值，显式指定  动态分区的分区键的值是在运行时计算的，事先并不清楚分区的数量。如果源表不是一张分区表，要找出所有分区键的值并插入分区表是一件很麻烦的事情，这个时候就应该使用动态分区。 |
| 为什么需要分桶？ | 首先可以提高Map Join的效率，左右表同一join键的数据都在一个节点上处理，其次可以提高查询效率，再次可以让取样（Sampling）过程更加有效率。 |
| 分桶里的数据是如何分布的？ | 通过计算哈希函数(分桶的列值) 取模 (桶的数量) 的值来决定每行数据的桶的编号。通常哈希函数的选择取决于分桶键的数据类型，比如对于整数类型的分桶列来说，哈希函数的返回值就是列值本身。 |
| Hive是否适合OLTP系统？为什么？ | 不，因为不提供insert和update语句 |
| Hive表的数据存放在什么地方？ | 在HDFS的/user/hive/warehouse目录下，可以修改hive-site.xml的配置hive.metastore.warehouse.dir来修改下。 |
| Hive的metastore是什么？ | 存放了Hive库和表的元数据信息，一般会把元数据储存到关系型数据库上，如MySQL和Postgres。不把元数据信息储存到HDFS的原因是因为读写很慢，延迟较高。 |
| Hive 内部（Managed）表和外部（External）表的区别？ | Drop内部表的时候，数据文件和元数据也一起删除了。  Drop外部表的时候，只删除元数据，数据文件保留。 |
| 如何往Hive系统里导入数据文件? | 如果文件已经保存在HDFS上了,而且文件格式是已知的,只需要定义一张外部表就可以了. 如果文件不在HDFS上, 可以先使用hdfs命令将文件拷贝到HDFS,在建立外部表关联, 或者可以使用load data local命令来导入, 但是需要先创建好内部表. 注意, 如果文件格式是未知的, 那么是无法正确定义Hive表的结构来处理这些文件的. |
| Hive系统架构是由哪几个组件组成的? | a) Hive Client: 包括CLI工具, 还支持JDBC连接  b) Metastore: 存储表的结构、文件位置等元数据  c) Query Processor: 解析器, 将SQL转换成DAG执行, 每个步骤是一个MapReduce Job (当执行引擎是MR时)  d) 执行引擎: 负责执行DAG, 并输出结果和返回 |
| Hive的序列化器和反序列化器的用途? | Hive通过SerDe来实现读取和写入表的记录:  HDFS文件 -> InputFileFormat -> <key, value> -> Deserializer -> Row object  Row object -> Serializer -> <key, value> -> OutputFileFormat -> HDFS files |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |