昆仑分布式数据库分布式查询优化原理

泽拓科技创始人 赵伟

Agenda

- 分布式数据库的查询处理面对的新挑战和新机遇
- 基于代价的查询优化（CBO）简介及其分布式扩展
- 昆仑分布式数据库的查询处理机制和策略
- 昆仑分布式数据库的查询处理具体技术

分布式数据库的查询优化面对的新挑战和新机遇

- 单机数据库： 数据存储在同一个计算机服务器中
  - 磁盘(ssd) -> 内存->CPU cache ->CPU
- 分布式数据库：数据存储在一组计算机服务器中，在另一组的服务器中用于计算结果
  - 磁盘 -> 内存（db buffer pool） -> CPU-> 网络设备buffer -> TCP/IP网络 -> 网络设备buffer -> CPU-> 内存（CN local buffer） -> CPU
  - 磁盘 -> 内存（db buffer pool） -> 网络设备buffer -> TCP/IP网络 -> 网络设备buffer -> 内存（CN local buffer） -> CPU (RDMA)
- 机遇：执行一个SQL SELECT 查询可以使用多个计算机服务器的硬件资源
  - 更多的CPU core
  - 更多的内存和总线带宽
  - 并行的IO设备和PCI总线带宽
  - 结论：多节点并行执行同一个SELECT 查询

分布式数据库的查询优化面对的新挑战

- 跨计算机搬动数据的时耗和资源开销
  - 信息传输速度的上限
    - 光在光纤介质中的速度约 2*10^8 米/秒
    - 跨计算机传输的时耗来源（RTT）
      - 介质传输（数百米（同数据中心40us），数十公里（同城4ms），到数千公里（跨省跨国0.4s））、
      - 数据包的转发&路由；数字信号的中继、放大、校验、转换
    - 数据（电子信号）在单机电路内的传输速度：0.6~0.9c，访存100ns
      - 电子信号在IC内传输延时 <<(远小于) 光信号跨服务器（同机房）的传输延时
  - 网络带宽有限&共享
    - 超过资源上限数据包需排队，进一步增大了延时
    - 带宽大量消耗会拖慢整个数据中心所有节点
  - **结论：传输的距离越近越快，传输的数据量越少越好**

基于代价的查询优化（CBO）简介

- CBO的本质是对某个特定的操作从多个选项方法中选择最优
- 单机数据库CBO的代价计算的因子（factors）
  - 计算开销（包含了CPU指令执行以及读写内存）：MonoIncF1(行数)
  - IO开销：MonoIncF2(页面数量)
- 单机数据库的主要操作的执行选项
  - 访问方法：全表扫描（SeqScan） VS 索引扫描（IndexScan，TIDBitmapScan）
  - 排序：IndexScan， TupleSort，MergeSort
  - 连接：NestLoop，HashJoin，MergeJoin
  - Agg分组：Sort，Hash
  - Union、Insersection、Difference：TIDBitmap 位图Union,Intersect, Except

基于代价的查询优化（CBO）的统计信息

- 基本的统计信息
  - 页的数量，行的数量，以及计算出的平均行长度avg_row_len = num_pages*pagesize/num_rows
  - 由cluster_mgr从存储节点同步到计算节点
  - 假定行等长，假定每列的字段值均匀分布
- 高级统计信息：直方图: 解决列值非平均分布的问题
  - 一个表的某一个或者一组列值的分布详情
    - 列值范围N等分，每份各自占总行数的比例
    - 行数N等分，每份的列值范围
  - 用于估算某个过滤条件、连接条件获取的行的数量
  - 由cluster_mgr从存储节点同步到计算节点

昆仑数据库分布式查询优化

- 分布式数据库的额外代价计算因子
  - 网络传输开销= MonoIncF3(跨节点搬动的数据量)
- 分布式数据库的主要操作的执行选项
  - 访问方法：RemoteScan，存储节点单表查询
    - 是否使用本地索引由每个存储节点自行决定
    - 过滤条件下推：避免或者减少无效数据行
    - Projection下推：避免传输不用的字段
    - 假设相同操作的实现性能相近，无条件优先下推
    - 无法下推的过滤条件：保留在计算节点过滤收到的行
    - 分解无法下推的Projection：func(x,y) -> x,y
    - 示例

昆仑数据库分布式查询优化

- 单表RemoteScan的查询优化：算子下推
  - 排序
    - 下推Sort，多个存储节点并行Sort->CN拉取多分片局部有序子结果->MergeSort（无需缓存数据， O(1) 的时间和空间复杂度）（√）
    - 拉取所有候选行到计算节点->Sort （X）
  - DISTINCT
    - 下推DISTINCT，局部去重，对于分区表，在CN中再次去重（√）
      - 更少的数据传输，并行去重
    - 拉取到CN中去重（X）
  - LIMIT
    - 单表：无条件下推：更小的数据传输量
    - 分区表 有ORDER BY：需要全局MergeSort，无法下推LIMIT
    - 分区表无ORDER BY: 不确定操作，无意义，忽略
  - 示例

昆仑数据库分布式查询优化

- 分布式数据库的执行选项：表连接
  - 左右单表在同一个shard RemoteJoin：下推Join，拉取结果集
  - 左右分区表：TupleShuffle->MergeAppend或MergeSort
    - 搬动一个表的每个分片过滤后数据到另一个表的每个分片做join
  - 上传 左表 和 右表，在CN内join
  - 局部有序时利于MergeJoin
  - 数据搬迁的代价对比
    - 搬动左表数据到右表存储节点：NRows_Left * avg_row_len_L * NShards_Right
    - 搬动右表数据：NRows_Right * avg_row_len_R * NShards_Left
    - 搬动左右表到计算节点：NRows_Left*avg_row_len_L + NRows_Right*avg_row_len_R

昆仑数据库分布式查询优化

- 分布式数据库的执行选项：Aggregation 聚集查询
  - 优先下推，避免拉取大量数据行，假定相同操作的实现性能相近
  - 无法下推时拉取数据行到计算节点执行
  - 多分片表：合并部分agg结果 MergeRemoteAgg
  - 合并方法
    - count，sum：加总
    - avg：下推sum和count，在CN中各自加总再除
    - max，min：取全局最大、最小
  - 示例

Q&A

Thank You

Zhao Wei