HybridDB for MySQL

最佳实践

最佳实践

分区设计

分区表的设计

用户存有海量数据的表应该按照数据规模进行拆解，表的数据将拆解成多个数据分区独立存储，通常的设计原则是：

主键(Primary Key)

单实例数据库不要求表一定要有主键，但是对于分布式数据库，主键则是必须的，以保证一行数据是全局唯一的，避免迁移过程出现问题。如果用户没有特殊的性能需求或业务耦合问题，则应将主键设计为无意义的整数值或者自增的；

分区键(Partition Key)

用户的分区表必须按照一种维度进行数据划分，用户在按照分区键维度进行查询时，就能做到线性性能增长，分区键通常有如下选择方法：

1) 按业务ID切分，如用户ID、商品ID等，适合每个业务ID的数据较均匀且查询简单的场景；

2) 按多个业务维度切分，用户建立多张表存入相同数据，但是每张表按照不同业务维度切分，查询时根据过滤条件选择不同的表，以提升访问性能，适合查询复杂且单一切分方式不能满足需求的场景；

3) 按自增主键切分，若表分区方式为分区表，主键为自增，且该字段同时为分区键，则此时写入为随机分区，按照非主键查询时需要读取所有分区，适合有数据偏斜且写多读少的场景；

4) 至少保证在分区键上有一个索引；

业务列

其它的列统归为此类，仅用于存储数据，通常要考虑：1) 列不宜过长，够用即可，临时加载的长列会消耗额外内存，影响查询性能；2) 准确定义列的类型，避免查询时使用的类型与表定义的类型不匹配，从而进行类型转换，以致不能正确使用索引；3) 优先使用timestamp类型代替其它时间日期类型，且使用时严格遵守mysql的

时间日期格式；

主键索引

若主键是自增类型，则主键索引不会对整体性能优化有改善，若主键与业务相关，则可以对最频繁使用的SQL语句的查询条件建立主键索引，这样可以提升性能；

辅助索引

其他情况下，如果不能通过将SQL语句拆解成单分区的，且不能利用主键进行索引优化时，需要对全部分区进行扫描，此时可以对这部分全分区扫描的语句的查询条件建立索引，使得在每个分区上进行访问时，仍然能取得较高的性能

*注：HybridDB for MySQL 目前暂不支持广播表（广播表的数据在每个数据分区均有相同副本，用于与分区表进行join）

优化建议

常用优化包括

1) 合理的选择拆分字段。选择拆分字段时需要综合考虑查询性能、分布式事务、热点、数据迁移等多个因素；

2) 掌握SQL的执行计划，尤其是核心SQL。对于不确定的SQL应在分布式数据库执行 ‘explain sql’ 命令，确定SQL有没有跨分区、有没有改写以及底层有没有合适的索引，合并时是否进行了排序和分组动作；

3) 对底层MySQL建立合适的索引，这一点看似与分布式数据库无关，但却是最重要的。分布式数据库的高性能依赖于底层数据库的高性能，而对底层数据库性能来说，建立需要的索引是重要的环节；

4) 确保语句能正确使用到索引，例如查询条件能被索引完全覆盖到，保证分区键上有索引等；

5) 查询尽量在单机完成，最为简单的方式就是在分区字段上指定等值条件，使操作只发送到一个后台数据库节点。若不指定，则操作需要发送到每个后台节点，可能导致性能大幅下降；

6) 尽量避免分布式事务和分布式查询；

7) 同时使用其他适用于MySQL的优化手段；

设计表结构的过程

1) 预估数据量、访问规模，准备测试数据，测试原始数据库的性能基线；

2) 分析和设计表结构、约束、索引；

3) 分析和设计分区方式和分区字段；

4) 分析和设计常用sql语句访问的频度和分区数量

5) 分析和设计需要聚合、排序、分组、条件过滤的字段；

6) 重新在分布式数据测试性能基线；

7) 调整分布式数据库的配置参数，重新观察系统；

8) 调整读写比例、并发活跃连接数，重新观察系统；

9) 比较结果，找出正确的优化方式；

HybridDB for MySQL设计和实践优化建议

当用户在使用HybridDB for MySQL进行数据库设计和实践的过程中，我们有如下建议：

分区键的选择

分区键是数据库控制数据分布的维度，以该条件进行等值查询，查询范围只会限制在一个存储分区上，通常选择查询最频繁的列，或数据分布最均匀的列。

批量插入

写入数据时，建议以insert into tb (f1, f2, ...) values (v11, v12, ...),(v21, v22, ...) ...语法批量写入，这样会提升性能，减少更新事务中网络的开销。

创建适当的索引

HybridDB for MySQL的索引设计与MySQL一样，需要在最常用的查询维度上创建索引，索引包含的列从左到右依次为等值条件列、范围条件列或join列、排序列、投影列，尽量提前设计索引，表数据量加大时索引会变慢。

大小表分开

为提升系统整体的性能和稳定性，用户应合理地将大表和小表拆分到不同的数据库中，HybridDB for MySQL适合存储大表，而将小表交给RDS for MySQL存储，大表可以使用sharding分区技术合理利用资源，而不影响小表。

分区键高并发执行

在某些场景下，用户期望拥有更高的吞吐和并发，进行快速的数据批量存取。HybridDB for MySQL支持查询force partition语法，用户可直接查询存储分区的数据，独立并行地查询多个存储分区，这样可以大幅提升整体性能。