

本课时我们来探讨读写分离如何在业务中落地。

读写分离是业务开发中常用的技术手段，在面试中也是热点问题，今天我们要讲的是在什么业务场景下需要读写分离，读写分离实现的机制，以及实际生产中应用读写分离要注意的问题。

什么时候需要读写分离

互联网大部分业务场景都是读多写少的，对于电商等典型业务，读和写的请求对比可能差了不少一个数量级。为了不让数据库的读成为业务瓶颈，同时也为了保证写库的成功率，一般会采用读写分离的技术来保证。

读写分离顾名思义，就是分离读库和写库操作，从 CRUD 的角度，主数据库处理新增、修改、删除等事务性操作，而从数据库处理 SELECT 查询操作。具体的实现上，可以有一主一从，一个主库配置一个从库；也可以一主多从，也就是一个主库，但是配置多个从库，读操作通过多个从库进行，支撑更高的读并发压力。

读写分离的实现是把访问的压力从主库转移到从库，特别在单机数据库无法支撑并发读写，并且业务请求大部分为读操作的情况下。如果业务特点是写多读少，比如一些需要动态更新的业务场景，应用读写分离就不合适了，由于 MySQL InnoDB 等关系型数据库对事务的支持，使得写性能不会太高，一般会选择更高性能的 NoSQL 等存储来实现。

MySQL 主从复制技术

读写分离是基于主从复制架构实现的，下面介绍一下 MySQL 中的主从复制技术。

binlog 日志

MySQL InnoDB 引擎的主从复制，是通过二进制日志 binlog 来实现。除了数据查询语句 select 以外，binlog 日志记录了其他各类数据写入操作，包括 DDL 和 DML 语句。

binlog 有三种格式：Statement、Row 及 Mixed。

- Statement 格式，基于 SQL 语句的复制

在 Statement 格式中，binlog 会记录每一条修改数据的 SQL 操作，从库拿到后在本地进行回放就可以了。

- Row 格式，基于行信息复制

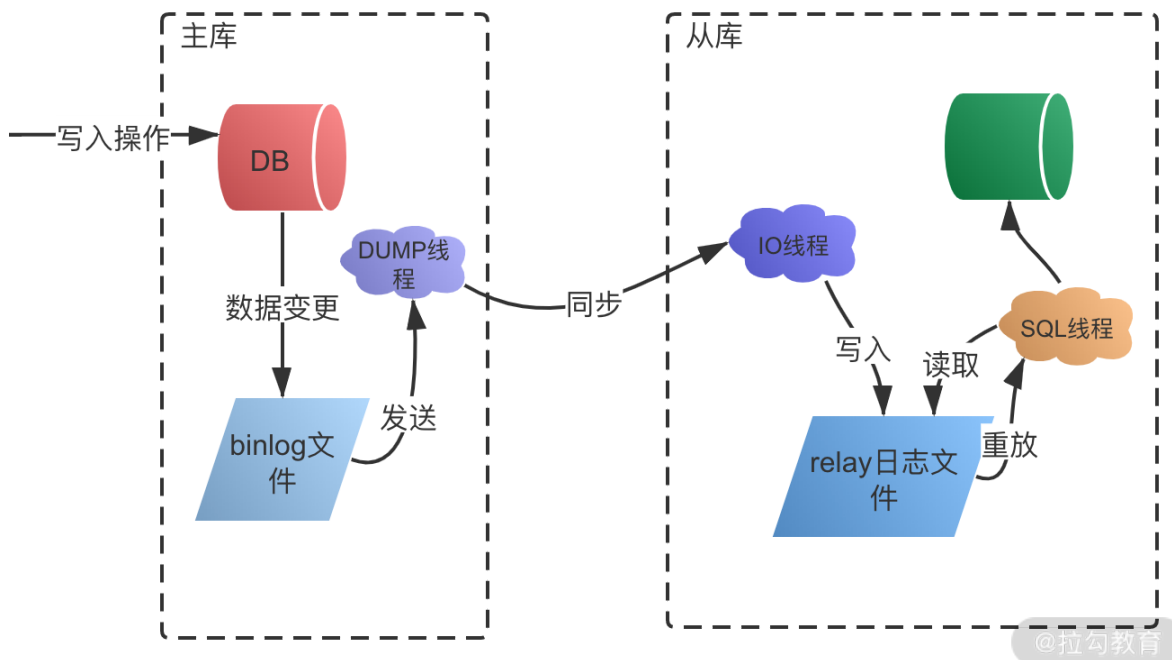
Row 格式以行为维度，记录每一行数据修改的细节，不记录执行 SQL 语句的上下文相关的信息，仅记录行数据的修改。假设有一个批量更新操作，会以行记录的形式来保存二进制文件，这样可能会产生大量的日志内容。

- Mixed 格式，混合模式复制

Mixed 格式，就是 Statement 与 Row 的结合，在这种方式下，不同的 SQL 操作会区别对待。比如一般的数据操作使用 row 格式保存，有些表结构的变更语句，使用 statement 来记录。

主从复制过程

MySQL 主从复制过程如下图所示：



- 主库将变更写入 binlog 日志，从库连接到主库之后，主库会创建一个 log dump 线程，用于发送 bin log 的内容。
- 从库开启同步以后，会创建一个 IO 线程用来连接主库，请求主库中更新的 bin log，I/O 线程接收到主库 binlog dump 进程发来的更新之后，保存在本地 relay 日志中。
- 接着从库中有一个 SQL 线程负责读取 relay log 中的内容，同步到数据库存储中，也就是在自己本地进行回放，最终保证主从数据的一致性。

读写分离要注意的问题

分布式系统通过主从复制实现读写分离，解决了读和写操作的性能瓶颈问题，但同时也增加了整体的复杂性。我们看一下引入主从复制后，额外需要注意哪些问题。

主从复制下的延时问题

由于主库和从库是两个不同的数据源，主从复制过程会存在一个延时，当主库有数据写入之后，同时写入 binlog 日志文件中，然后从库通过 binlog 文件同步数据，由于需要额外执行日志同步和写入操作，这期间会有一定时间的延迟。特别是在高并发场景下，刚写入主库的数据是不能马上在从库读取的，要等待几十毫秒或者上百毫秒以后才可以。

在某些对一致性要求较高的业务场景中，这种主从导致的延迟会引起一些业务问题，比如订单支付，付款已经完成，主库数据更新了，从库还没有，这时候去从库读数据，会出现订单未支付的情况，在业务中是不能接受的。

为了解决主从同步延迟的问题，通常有以下几个方法。

- 敏感业务强制读主库

在开发中有部分业务需要写库后实时读数据，这一类操作通常可以通过强制读主库来解决。

- 关键业务不进行读写分离

对一致性不敏感的业务，比如电商中的订单评论、个人信息等可以进行读写分离，对一致性要求比较高的业务，比如金融支付，不进行读写分离，避免延迟导致的问题。

主从复制如何避免丢数据

假设在数据库主从同步时，主库宕机，并且数据还没有同步到从库，就会出现数据丢失和不一致的情况，虽然这是一个极端场景，一般不会发生，但是 MySQL 在设计时还是考虑到了。

MySQL 数据库主从复制有异步复制、半同步复制和全同步复制的方式。

- 异步复制

异步复制模式下，主库在接受并处理客户端的写入请求时，直接返回执行结果，不关心从库同步是否成功，这样就会存在上面说的的问题，主库崩溃以后，可能有部分操作没有同步到从库，出现数据丢失问题。

- 半同步复制

在半同步复制模式下，主库需要等待至少一个从库完成同步之后，才完成写操作。主库在执行完客户端提交的事务后，从库将日志写入自己本地的 relay log 之后，会返回一个响应结果给主库，主库确认从库已经同步完成，才会结束本次写操作。相对于异步复制，半同步复制提高了数据的安全性，避免了主库崩溃出现的数据丢失，但是同时也增加了主库写操作的耗时。

- 全同步复制

全同步复制指的是在多从库的情况下，当主库执行完一个事务，需要等待所有的从库都同步完成以后，才完成本次写操作。全同步复制需要等待所有从库执行完对应的事务，所以整体性能是最差的。

总结

今天分享了读写分离的业务场景，MySQL 的主从复制技术，包括 binlog 的应用、主从复制的延时问题、数据库同步的不同机制等。

读写分离只是分布式性能优化的一个手段，不是任何读性能瓶颈都需要使用读写分离，除了读写分离，还可以进行分库分表，以及利用缓存，文件索引等 NoSQL 数据库来提高性能，这些内容在后面的课时中会讲到。

精选评论

****安:**

做好数据库监控也很重要

***峥:**

讲的太到位了!

***锯:**

既然三种主从复制都有弊端的话，我们在生产上应该选择哪种方式进行呢?

讲师回复:

根据业务对一致性的要求，一般采用异步或者半同步就可以

****淼:**

老师，binlog不是innodb引擎的东西吧

讲师回复:

binlog 是 MySQL Server 维护的，根据存储引擎的不同记录格式会有区别。

****弗:**

感谢分享!