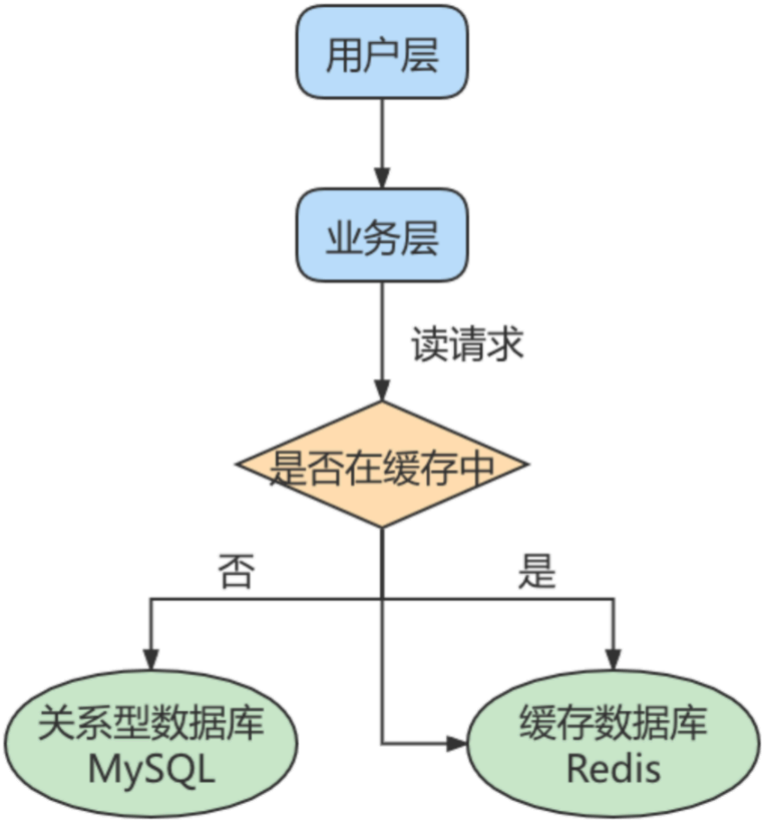
Part 18 主从复制

1. 主从复制概述

1.1 如何提升数据库并发能力

在实际工作中，我们常常将 Redis 作为缓存与 MySQL 配合来使用，当有请求的时候，首先会从缓存中进行查找，如果存在就直接取出。如果不存在再访问数据库，这样就 提升了读取的效率，也减少了对后端数据库的 访问压力。Redis的缓存架构是 高并发架构 中非常重要的一环。



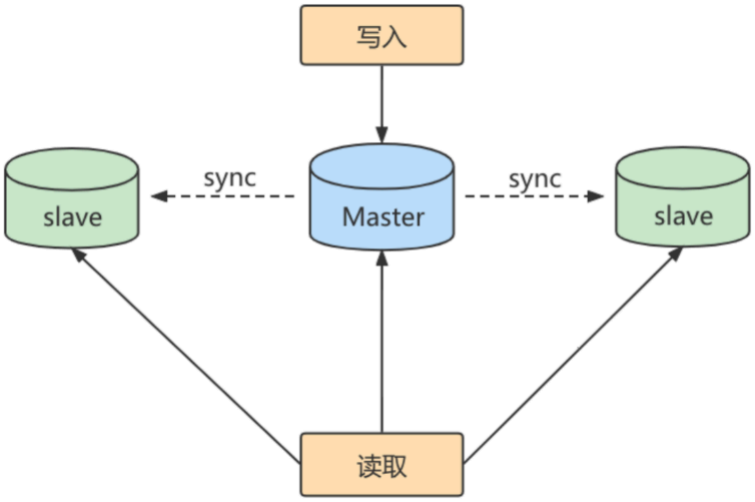
此外，一般应用对数据库而言都是“读多写少”，也就说对数据库读取数据的压力比较大，有一个思路就是采用数据库集群的方案，做 主从架构、进行 读写分离，这样同样可以提升数据库的并发处理能力。但并不是所有的应用都需要对数据库进行主从架构的设置，毕竟设置架构本身是有成本的。

如果我们的目的在于提升数据库高并发访问的效率，那么

- 首先考虑的是如何优化 SQL和索引，这种方式简单有效
- 其次才是采用 缓存的策略，比如使用 Redis将热点数据保存在内存数据库中，提升读取的效率
- 最后才是对数据库采用 主从架构，进行读写分离。

1.2 主从复制的作用

- 提高数据库吞吐量
- 读写分离



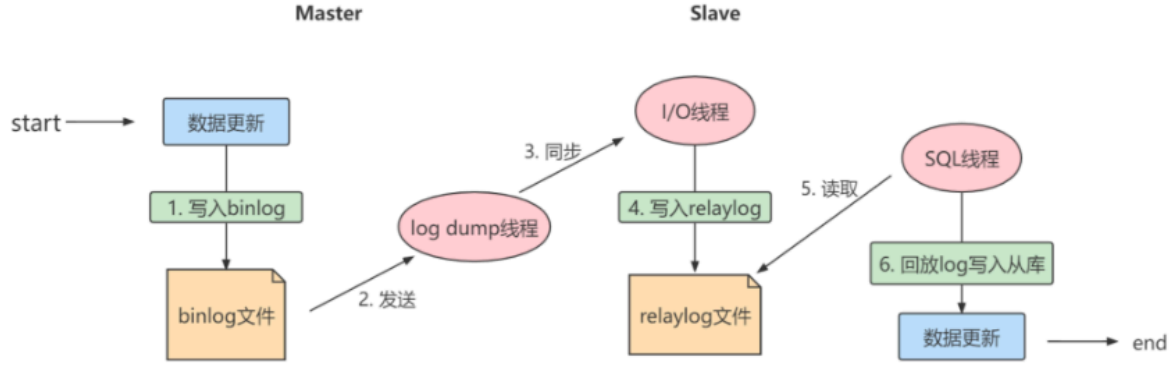
- 数据备份
  - 我们通过主从复制将主库上的数据复制到了从库上，相当于是一种热备份机制，也就是在主库正常运行的情况下进行的备份，不会影响到服务。
- 高可用性
  - 数据备份实际上是一种冗余的机制，通过这种冗余的方式可以换取数据库的高可用性，也就是当服务器出现 `故障` 或 `宕机` 的情况下，可以 `切换` 到从服务器上，保证服务的正常运行。
  - 关于高可用性的程度，我们可以用一个指标衡量，即正常可用时间/全年时间。比如要达到全年99.999%的时间都可用，就意味着系统在一年中的不可用时间不得超过 `365*24*60*(1-99.999%)=5.256` 分钟(含系统崩溃的时间、日常维护操作导致的停机时间等)，其他时间都需要保持可用的状态。
  - 实际上，更高的高可用性，意味着需要付出更高的成本代价。在现实中我们需要结合业务需求和成本来进行选择。

2. 主从复制的原理

`Slave` 会从 `Master` 读取 `binlog` 来进行数据同步。

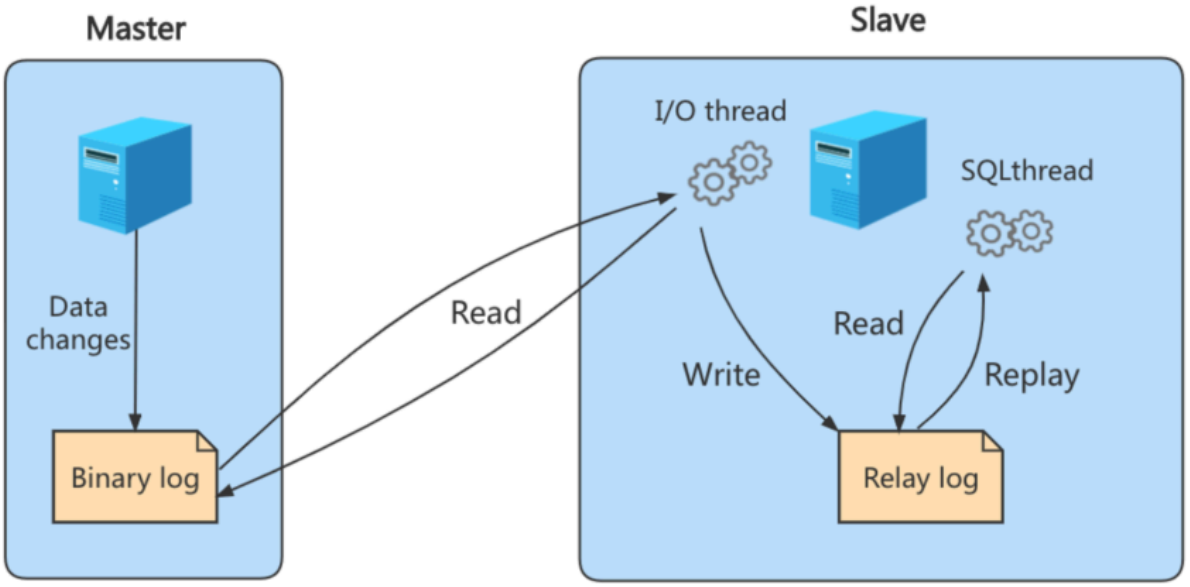
2.1 原理剖析

实际上主从同步的原理就是基于 binlog 进行数据同步的。在主从复制过程中，会基于 `3 个线程` 来操作，一个主库线程，两个从库线程。



- 主库线程
  - `二进制日志转储线程`（Binlog dump thread）是一个主库线程。当从库线程连接的时候， 主库可以将二进制日志发送给从库，当主库读取事件（Event）的时候，会在 Binlog 上 `加锁`，读取完成之后，再将锁释放掉。

- 从库线程
  - `从库 I/O 线程会连接到主库`，向主库发送请求更新 Binlog。这时从库的 I/O 线程就可以读取到主库的二进制日志转储线程发送的 Binlog 更新部分，并且拷贝到本地的中继日志 （Relay log）。
  - `从库 SQL 线程会`读取从库中的中继日志，并且执行日志中的事件，将从库中的数据与主库保持同步。



> 注意：不是所有版本的MySQL都默认开启服务器的二进制日志。在进行主从同步的时候，我们需要先检查服务器是否已经开启了二进制日志。

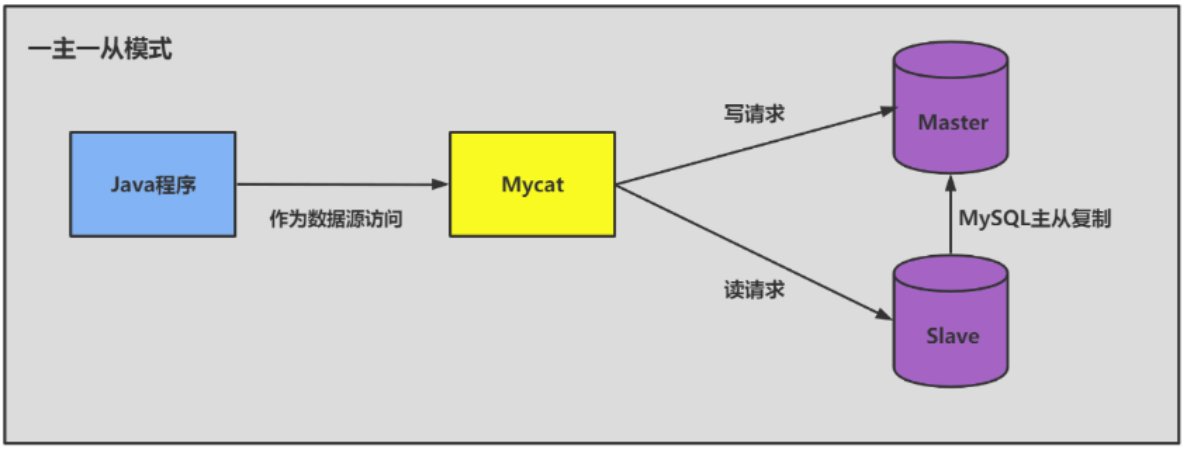
复制最大的问题：延时

2.2 复制的基本原则

- 每个`Slave`只有一个`Master`
- 每个`Slave`只能有一个唯一的服务器ID
- 每个`Master`可以有多个`Slave`

3. 一主一从架构搭建

一台`主机`用于处理所有`写请求`，一台从机负责所有读请求，架构图如下：



剩余部分见原笔记。

4. 同步数据一致性问题

主从同步的要求：

- 读库和写库的数据一致(最终一致)；
- 写数据必须写到写库；
- 读数据必须到读库(不一定)；

4.1 理解主从延迟问题

进行主从同步的内容是二进制日志，它是一个文件，在进行`网络传输`的过程中就一定会`存在主从延迟`（比如 500 ms），这样就可能造成用户在从库上读取的数据不是最新的数据，也就是主从同步中的`数据不一致性`问题。

4.2 主从延迟问题原因

网络正常情况下，日志从主库传给从库所需的时间是很短的，主备延迟的主要来源是备库接收完binlog和执行完这个事务之间的时间差。

主备延迟最直接的表现是，从库消费中继日志（relay log）的速度，比主库生产binlog的速度要慢。

造成原因：一是从库的机器性能比主库要差，二是从库的压力大，三是大事务的执行。

4.3 如何减少主从延迟

若想要减少主从延迟的时间，可以采取下面的办法：

1. 降低多线程大事务并发的概率，优化业务逻辑
2. 优化SQL，避免慢SQL，`减少批量操作`，建议写脚本以update-sleep这样的形式完成。
3. `提高从库机器的配置`，减少主库写binlog和从库读binlog的效率差。
4. 尽量采用`短的链路`，也就是主库和从库服务器的距离尽量要短，提升端口带宽，减少binlog传输的网络延时。
5. 实时性要求的业务读强制走主库，从库只做灾备，备份。

4.4 如何解决一致性问题

如果操作的数据存储在同一个数据库中，那么对数据进行更新的时候，可以对记录加写锁，这样在读取的时候就不会发生数据不一致的情况。但这时从库的作用就是备份，并没有起到读写分离，分担主库读压力的作用。

读写分离情况下，解决主从同步中数据不一致的问题， 就是解决主从之间`数据复制方式`的问题，如果按照数据一致性`从弱到强来`进行划分，有以下 3 种复制方式。

4.4.1 异步复制

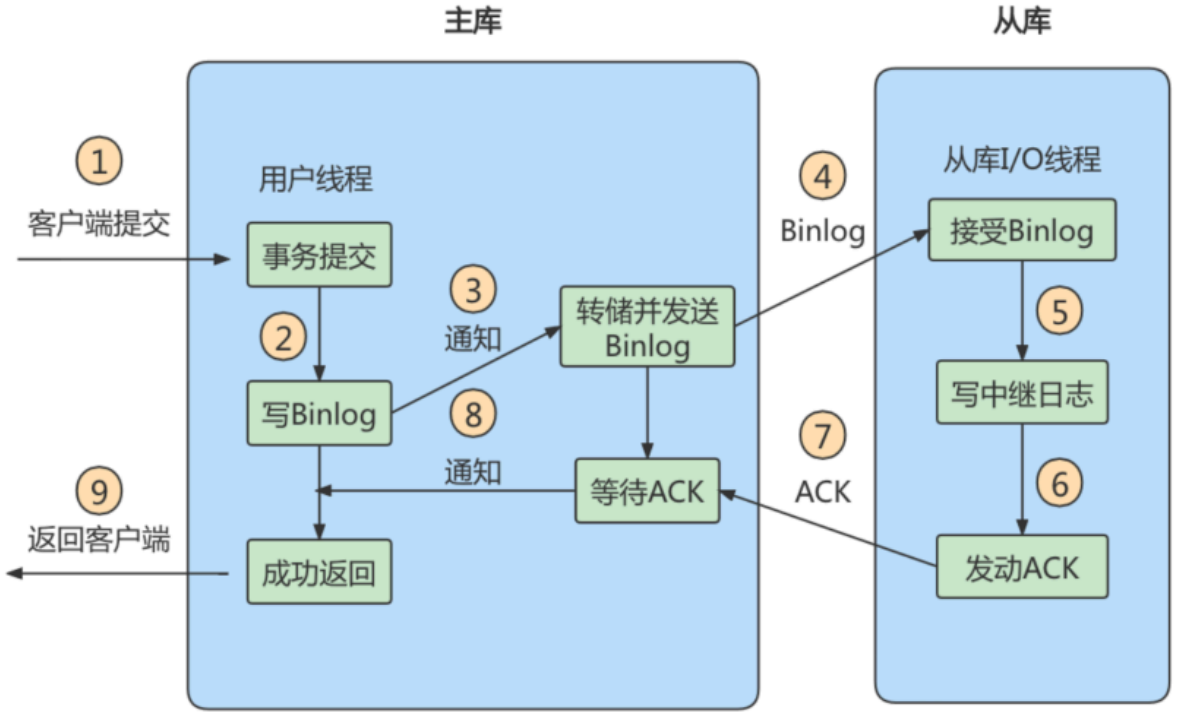
异步模式就是客户端提交COMMIT之后不需要等从库返回任何结果，而是直接将结果返回给客户端，这样做的好处是不会影响主库写的效率，但可能会存在主库宕机，而Binlog还没有同步到从库的情况，也就是此时的主库和从库数据不一致。这时候从从库中选择一个作为新主，那么新主则可能缺少原来主服务器中已提交的事务。所以，这种复制模式下的数据一致性是最弱的。

4.4.2 半同步复制

MySQL5.5版本之后开始支持半同步复制的方式。原理是在客户端提交COMMIT之后不直接将结果返回给客户端，而是等待至少有一个从库接收到了Binlog，并且写入到中继日志中，再返回给客户端。

这样做的好处就是提高了数据的一致性，当然相比于异步复制来说，至少多增加了一个网络连接的延迟，降低了主库写的效率。

在MySQL5.7版本中还增加了一个rpl_semi_sync_master_wait_for_slave_count参数，可以对应答的从库数量进行设置，默认为1，也就是说只要有1个从库进行了响应，就可以返回给客户端。如果将这个参数调大，可以提升数据一致性的强度，但也会增加主库等待从库响应的时间。



4.4.3 组复制

异步复制和半同步复制都无法最终保证数据的一致性问题，无法满足对数据一致性要求高的场景。MGR很好地弥补了这两种复制模式的不足。组复制技术，简称 MGR（MySQL Group Replication）。是 MySQL 在 5.7.17 版本中推出的一种新的数据复制技术，这种复制技术是基于 Paxos 协议的状态机复制。

首先我们将多个节点共同组成一个复制组，在`执行读写（RW）事务`的时候，需要通过一致性协议层（Consensus 层）的同意，也就是读写事务想要进行提交，必须要经过组里“大多数人”（对应 Node 节点）的同意，大多数指的是同意的节点数量需要大于 （N/2+1），这样才可以进行提交，而不是原发起方一个说了算。而针对`只读（RO）事务`则不需要经过组内同意，直接 COMMIT 即可。

在一个复制组内有多个节点组成，它们各自维护了自己的数据副本，并且在一致性协议层实现了原子消息和全局有序消息，从而保证组内数据的一致性。

5. 知识延伸

在主从架构的配置中，如果想要采取读写分离的策略，我们可以`自己编写程序`，也可以通过`第三方的中间件`来实现。

- 自己编写程序的好处就在于比较自主，我们可以自己判断哪些查询在从库上来执行，针对实时性要求高的需求，我们还可以考虑哪些查询可以在主库上执行。同时，程序直接连接数据库，减少了中间件层，相当于减少了性能损耗。
- 采用中间件的方法有很明显的优势，功能强大，使用简单。但因为在客户端和数据库之间增加了中间件层会有一些性能损耗，同时商业中间件也是有使用成本的。我们也可以考虑采取一些优秀的开源工具。

① Cobar属于阿里B2B事业群，始于 2008 年，在阿里服役 3 年多，接管3000+个MySQL数据库的schema,集群日处理在线SQL请求 50 亿次以上。由于Cobar发起人的离职，Cobar停止维护。

② Mycat是开源社区在阿里cobar基础上进行二次开发，解决了cobar存在的问题，并且加入了许多新的功能在其中。青出于蓝而胜于蓝。

③ OneProxy基于MySQL官方的proxy思想利用c语言进行开发的，OneProxy是一款商业收费的中间件。舍弃了一些功能，专注在性能和稳定性上。

④ kingshard由小团队用go语言开发，还需要发展，需要不断完善。

⑤ Vitess是Youtube生产在使用，架构很复杂。不支持MySQL原生协议，使用需要大量改造成本。

⑥ Atlas是 360 团队基于mysql proxy改写，功能还需完善，高并发下不稳定。

⑦ MaxScale是mariadb（MySQL原作者维护的一个版本） 研发的中间件

⑧ MySQLRoute是MySQL官方Oracle公司发布的中间件