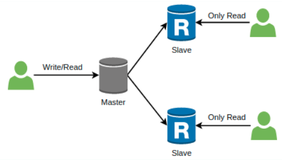
复制: 单领导者

主节点与从节点

- 主节点：主节点又称之为领导者节点、主库，接受客户端的读/写请求，并将数据的更改传播给所有的从节点
- 从节点：从节点又称之为只读节点、追随者或者从库，通常作为只读数据库，仅接受客户端的读请求，并接收或者拉取主节点的数据变更

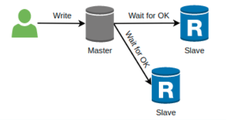


- 主从复制的优势
  - 由于从节点只处理只读请求，那么据此可实现数据的读写分离，将数据读取的负载均摊在多个从节点上，从而降低主库压力并提升系统效率
  - 我们默认为从节点和主节点的数据存在较少差异，那么此时从节点又可以作为容灾库使用。当主节点宕机机即可快速的使用从节点数据进行恢复
- 主从复制存在的问题
  - 同步复制 or 异步复制？
    - 同步复制能保证数据的严格一致性，但当从节点下线时，可能会出现服务不可用的情况
    - 异步复制能保证服务的可用性，但若从节点和主节点的复制延迟过大，将无法保证数据的一致性
  - 复制的数据采用何种格式？

同步复制与异步复制

同步复制

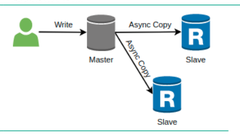
- 基本概念
  - 同步复制是指当用户向主节点写入数据后，主节点需要将此变更成功应用到集群中所有的从节点中，而后才返回成功给客户端



- 优势
  - 数据的同步复制可以保证数据的严格一致性，并且当主节点失效时可快速切换主节点
- 缺陷
  - 数据写入的效率随着从节点数量的增加而降低，并且如果出现某一个从节点下线，那么数据写入将永远不会成功，此时服务基本不可用
  - 数据的同步复制同时也会带来更高的复杂度，当对两个从节点进行同步复制时，前一个节点复制成功，而后一个节点复制失败，那么就需要对复制成功的节点进行额外的数据回滚操作

异步复制

- 基本概念
  - 异步复制是指当用户向主节点写入数据后，主节点立即返回成功给客户端，并将修改以异步的方式同步给其余从节点



- 优势
  - 异步复制能够最大程度地保证服务可用性，并且数据写入的成本并未增加
- 缺陷
  - 异步复制的一个最大问题就在于其复制延迟，由于网络的复杂性，无法保证从节点和主节点的复制延迟，所以会带来数据一致性的问题

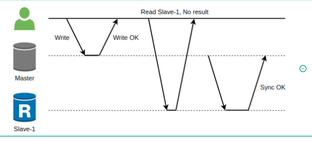
异步复制的一致性问题

最终一致性 (Eventually Consistency)

- 当数据副本采用异步复制的方式进行构建时，由于网络数据传输以及网络的复杂性存在，从库必然存在复制滞后的问题
- 但这个复制滞后并不是永久的，当主节点和从节点均正常运行且主节点无数据写入时，一段时间后从节点的进度仍然能追赶上主节点，使得数据再次一致，这就是最终一致性
- 某些场景下最终一致性的一致性级别能够满足需求，但是对于一些特定的操作，更希望得到强一致性的结果

读写一致性 (Read-after-write consistency)

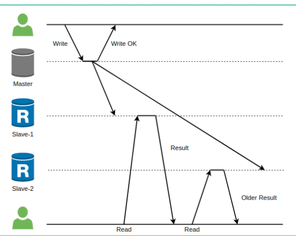
- 场景
  - 用户在主节点进行数据写入或者数据修改，Web Server返回成功后用户紧接着刷新页面，尝试获取刚才自己所做的修改
  - 由于是异步复制，可能存在或多或少的复制延迟，那么在请求读取数据时，可能读到从节点的旧数据。过了一会儿，从库同步了该修修改，用户再次读取时，成功获取了修改



  - 通常来说，用户读取自己的修改应该要保持较强的一致性，避免用户产生疑惑从而进行频繁的操作
- 可选解决方案
  - 1 追踪数据更新时间
    - 数据在修改成功后额外返回数据更新的时间戳，用户读取数据时服务端据此判断上次的数据更新到现在过去了多长时间。如果时间较短(从库可能未接收到更新数据)，则从主库读取数据，否则从从库读取数据
  - 2 建立元数据中心存储
    - 本质上仍然是追踪数据更新时间，只不过该更新时间建立在服务端，且对客户端不可见
    - 当应用具有多种终端(Web、APP)时，数据更新时间无法在多个终端进行共享，那么此时就需要建立中心存储
  - 3 尽可能地降低复制延迟
    - 通过优化数据复制的速度来降低主从节点间的差距，并建立完善的复制监控机制。客户端进行数据读取时可根据从节点延迟的大小进行加权选择

单调读一致性 (Monotonic-read Consistency)

- 当用户进行数据读取时，假设结果为A，那么后续的读取操作返回结果要么是A，要么是读取到更新的数据，不会读取到旧值
- 当集群中存在多个从节点时，每个从节点的复制延迟可能不尽相同，当用户在延迟小的从节点读取数据后，可能下一次在延迟较大的节点中进行读取，那么此时可能会读取到数据的旧值
- 场景



- 可选解决方案
  - 1 从特定的副本读取数据
    - 确保每个用户总是从同一个副本进行读取，例如采用对用户ID进行哈希取模的方式，使特定用户永远都只在一个数据副本上进行读取
    - 只不过此种方式无法处理热点Key的问题，同时也限定了较为特殊的数据读取