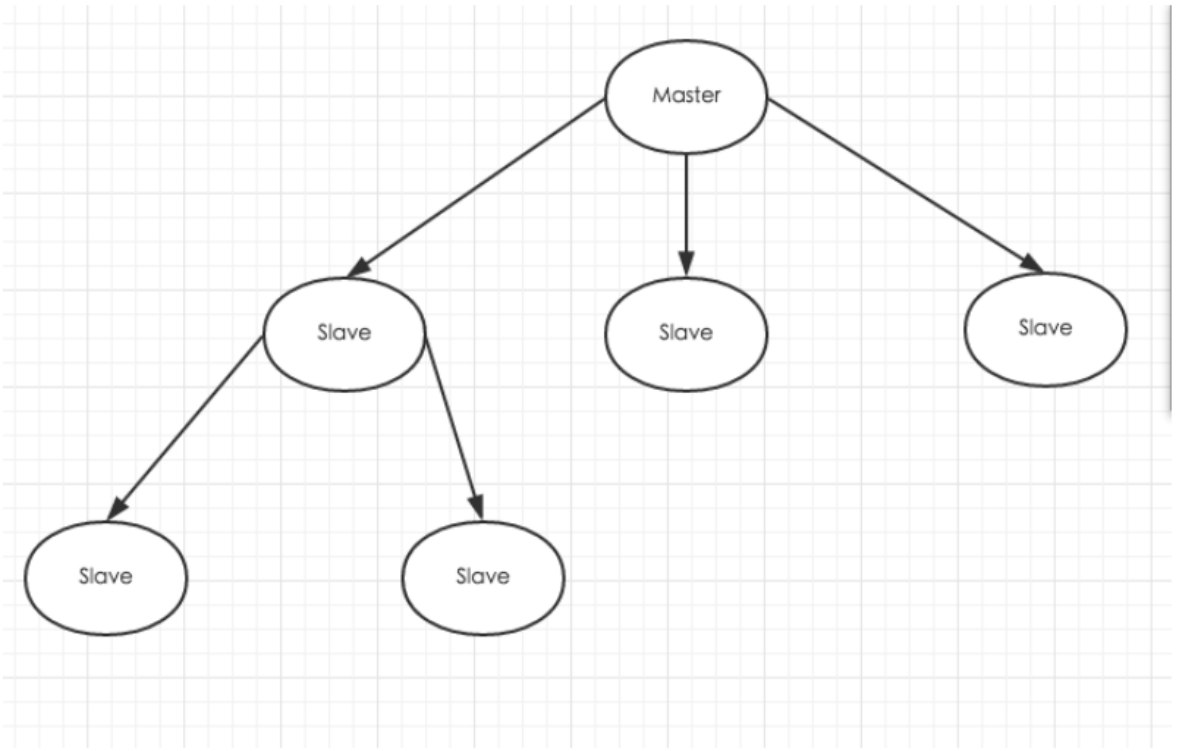
主从的意义：

1.redis要达到高可用、高并发，只有单个redis是不够的，单个redis也就只能支持几万的QPS，所以必须以集群的形式提供服务，而集群中又以多个主从组成。

2.主从是以多个redis集合在一起，以一个master多个slave为模式对外提供服务，master主要以写为主，slave提供读，即是读写分离的情况，以读多写少为准，如果写比较多的情况一般就以异步的形式提供服务。



Master与Slave通讯及数据复制原理分析

redis replication核心机制

1.redis采用异步的形式复制数据到slave，从redis2.8开始，slave会周期性地确认自己每次复制的数据量。

2.一个master node可以连接多个slave node

3.slave node也可以连接其他slave node

4.slave node复制数据时不会block master node的正常工作

5.slave node复制数时也不会block自己的查询操作，它会用旧的数据对外提供服务，但复制完成后，需要删除旧的数据，加载新的数据，这个时候会暂停对外服务。

6. slave node主要用来进行横向扩容，扩容可以提高读的吞吐量。

master持久化对主从架构的安全保障的意义

1.如果采用了主从架构，就必须开启master node持久化。

2.不要把slave node作为master的热备份，如果那样的话，如果master没有持久化，master宕机后重启了，master上的数据是空的，会覆盖salve node中的数据，这样数据就丢了。

3.即使采用了高可用机制，slave自动接管了master，但是也可能sentinal还没有检测到master failure，master node就重启了，还是有可能导致所有的slave node被清空。

主从架构核心原理

1.当启动一个slave node的时候，它会发送一个PSYNC命令给master

2.如果这是slave node重现连接master，master会将缺少的数据发送给slave，如果是第一次连接master，则会触发一次full resynchronization，开始full resynchronization的时候，master启动一个后台线程，先将现有的数据生成一个临时的rdb文件，生成文件后，master会将这个rdb文件发送给slave，slave会先把这个rbd文件存放到本地磁盘，然后再加载到内存中，然后master会将生成rdb这段时间接收到的在内存中的数据发送给slave，slave也会接收这份数据。

3.slave如果跟master网络故障，断开了，当重新连接上后，master发现有多个slave都来重新连接，master会生成一个rdb文件，将这个文件同时发送给多个slave node。

主从复制的断点续传

1.redis从2.8开始就支持断点续传功能，即当slave和master断开后，重新连接时，会继续从上一次断开的点继续传送数据，而不是full resynchronization。

2.master会在内存中创建一个backlog，master和slave都会保存一个offset，slave还有一个master id，offset就是保存在backlog中的，如果slave和master网络断开，重新连接后slave会让master offset开始续传。但是如果没有找到offset，则会触发full resynchronization。

无磁盘化复制

1.master在内存中直接创建rdb，然后直接发送给slave，不会存入本地磁盘。

2.参数配置：

repl-diskless-sync

repl-diskless-sync-delay，等待一定市场在复制，因为要等更多的slave重新连接。

过期key处理

slave不会有过期key，只有master有过期key，如果master过期了一个可以或者通过LRU算法淘汰了一个key，那么master会模拟发送了一个del命令给slave。

复制完整流程：

1.slave node的启动，仅仅保存了master node的信息，包括master的host和ip，但是复制还没有开始，host和ip都配置在redis.conf中。

2.slave内部有一个自动任务，每秒中会自动检查一次是否有master需要连接，如果发现有master连接，则master发送socket连接。

3.slave发送ping命令给master

4.口令认证，如果master设置了requirepass，那么slave必须发送requireauth口令过去进行认证

5.master node第一次触发全量复制命令，将所有的数据发送给slave。

6.master后续持续将写命令异步发送给slave。

数据异步相关核心机制：

1.指的是第一次slave全量复制，这个过程的一些细节机制。

2.master和slave都会维护一个offset，master会自身不断得累加offset，slave也会不断的累加offset

3.slave每秒都会发送自己的offset给master，同时master也会保存每个slave的offset。这个主要是master和slave都要知道各自的offset，这样才能知道哪些数据不一致。

4.master会有一个backlog，默认是1M，master给slave复制数据时，也会将数据同步到backlog中一份。

5.master run id

在客户端可以通过命令info server命令看到master run id，如果根据master的host+ip定位master node是不靠谱的，如果master发生重启或者数据发生了改变，那么slave应该根据不同的run id区分，run id不同就做全量复制，相同就做续传，如果需要不改变run id重启，则使用redis-cli debug reload命令。

6.slave 使用psync从master进行复制，psync runid offset，master会根据自身的情况适时响应信息，可能时fullresync runid offset触发全量复制，可能时continue触发增量复制。

全量复制

1.master执行bgsave，在本地生成一份rdb快照文件。

2.master把rdb文件发送给salve，如果rdb复制超过60s（repl-timeout参数可以配置），那么slave就会认为复制失败，可以适当调节这个参数。
3.master会将生成rdb这段时间内接收到的在内存中的数据发送给slave，slave也会接收这份数据。
4.client-output-buffer-limit slave 256MB 64MB 60，如果在复制期间，内存缓冲区持续超过60MB或者一次性超过256MB，那么停止复制，复制失败。
5.slave接收到数据后，清空旧的数据，然后重新加载rdb到内存中，加载期间，基于旧的数据对外提供服务。
6.如果开启了aof，那么会立即执行berewriteaof，重新aof
7.rdb生成，rdb通过网络拷贝，slave旧数据的清理，slave aof rewrite都很耗时，如果复制数据在4G-6G之间，那么复制时间很可能超过1分半到2分钟