Redis主要有哪些功能？

1.哨兵（Sentinel）和复制（Replication） Redis服务器毫无征兆的罢工是个麻烦事，如何保证备份的机器是原始服务器的完整备份呢？这时候就需要哨兵和复制。

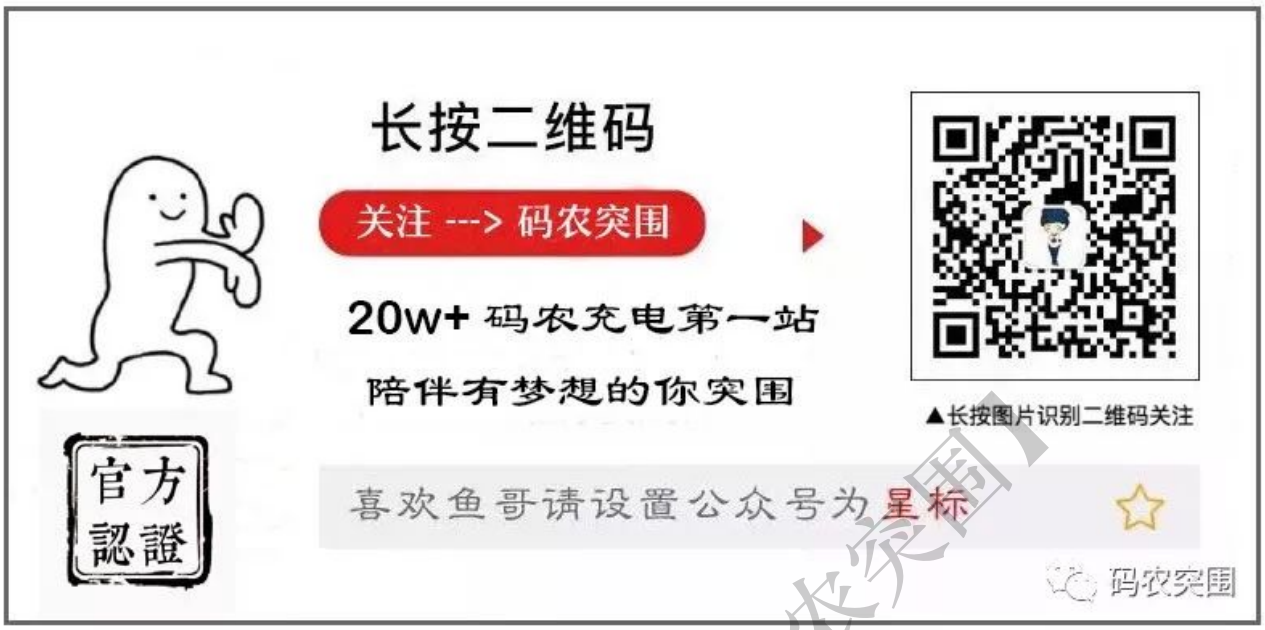
Sentinel可以管理多个Redis服务器，它提供了监控，提醒以及自动的故障转移的功能，Replication则是负责让一个Redis服务器可以配备多个备份的服务器。 Redis也是利用这两个功能来保证Redis的高可用的

2.事务 很多情况下我们需要一次执行不止一个命令，而且需要其同时成功或者失败。redis对事务的支持也是源自于这部分需求，即支持一次性按顺序执行多个命令的能力，并保证其原子性。(多个指令不支持)

3.LUA脚本 在事务的基础上，如果我们需要在服务端一次性的执行更复杂的操作（包含一些逻辑判断），则lua就可以排上用场了

4.持久化 redis的持久化指的是redis会把内存的中的数据写入到硬盘中，在redis重新启动的时候加载这些数据，从而最大限度的降低缓存丢失带来的影响。

5.集群（Cluster） 单台服务器资源的总是有上限的，CPU资源和IO资源我们可以通过主从复制，进行读写分离，把一部分CPU和IO的压力转移到从服务器上，这也有点类似mysql数据库的主从同步。 在Redis官方的分布式方案出来之前，有twemproxy和codis两种方案，这两个方案总体上来说都是依赖proxy来进行分布式的。



Redis支持哪几种数据类型？

支持多种类型的数据结构

1.string：最基本的数据类型，二进制安全的字符串，最大512M。

2.list：按照添加顺序保持顺序的字符串列表。

3.set：无序的字符串集合，不存在重复的元素。

4.sorted set：已排序的字符串集合。

5.hash：key-value对的一种集合。

类型	最大存储数据量
string	512M
list	2^32 - 1
set	2^32 - 1
sorted set	2^32 -1
hash	2^32 - 1

Redis是单进程单线程的？

Redis是单进程单线程的，Redis利用队列技术将并发访问变为串行访问，消除了传统数据库串行控制的开销。

Redis为什么是单线程的？

多线程处理会涉及到锁，而且多线程处理会涉及到线程切换而消耗CPU。因为CPU不是Redis的瓶颈，Redis的瓶颈最有可能是机器内存或者网络带宽。单线程无法发挥多核CPU性能，不过可以通过在单机开多个Redis实例来解决。

其它开源软件采用的模型

1.Nginx：多进程单线程模型 2.Memcached：单进程多线程模型

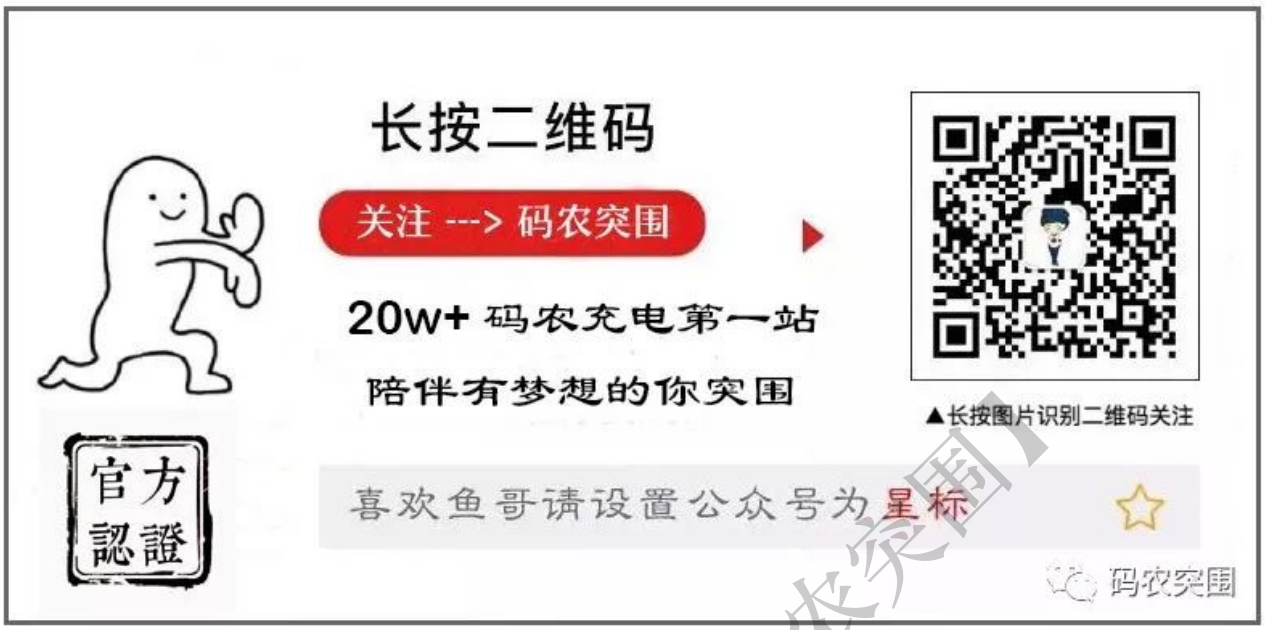
使用Redis的优势？

1.速度快，因为数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度都是O(1)

2.支持丰富数据类型，支持string，list，set，sorted set，hash

3.支持事务，操作都是原子性，所谓的原子性就是对数据的更改要么全部执行，要么全部不执行

4.丰富的特性：可用于缓存，消息，按key设置过期时间，过期后将会自动删除



Redis单点吞吐量

单点TPS达到8万/秒，QPS达到10万/秒，补充下TPS和QPS的概念

1.QPS: 应用系统每秒钟最大能接受的用户访问量

每秒钟处理完请求的次数，注意这里是处理完，具体是指发出请求到服务器处理完成功返回结果。可以理解在server中有个counter，每处理一个请求加1，1秒后counter=QPS。

2.TPS： 每秒钟最大能处理的请求数

每秒钟处理完的事务次数，一个应用系统1s能完成多少事务处理，一个事务在分布式处理中，可能会对应多个请求，对于衡量单个接口服务的处理能力，用QPS比较合理。

Redis相比memcached有哪些优势？

1.memcached所有的值均是简单的字符串，Redis作为其替代者，支持更为丰富的数据类型

2.Redis的速度比memcached快很多

3.Redis可以持久化其数据

4.Redis支持数据的备份，即master-slave模式的数据备份。

Redis有哪几种数据淘汰策略？

在Redis中，允许用户设置最大使用内存大小server.maxmemory，当Redis 内存数据集大小上升到一定大小的时候，就会施行数据淘汰策略。

1.volatile-lru:从已设置过期的数据集中挑选最近最少使用的淘汰

2.volatile-ttr:从已设置过期的数据集中挑选将要过期的数据淘汰

3.volatile-random:从已设置过期的数据集中任意挑选数据淘汰

4.allkeys-lru:从数据集中挑选最近最少使用的数据淘汰

5.allkeys-random:从数据集中任意挑选数据淘汰

6.noenviction:禁止淘汰数据

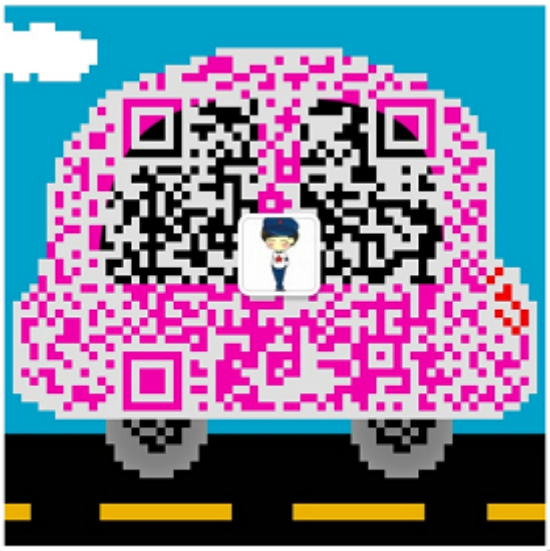
redis淘汰数据时还会同步到aof

Redis集群方案应该怎么做？都有哪些方案？

1.twemproxy

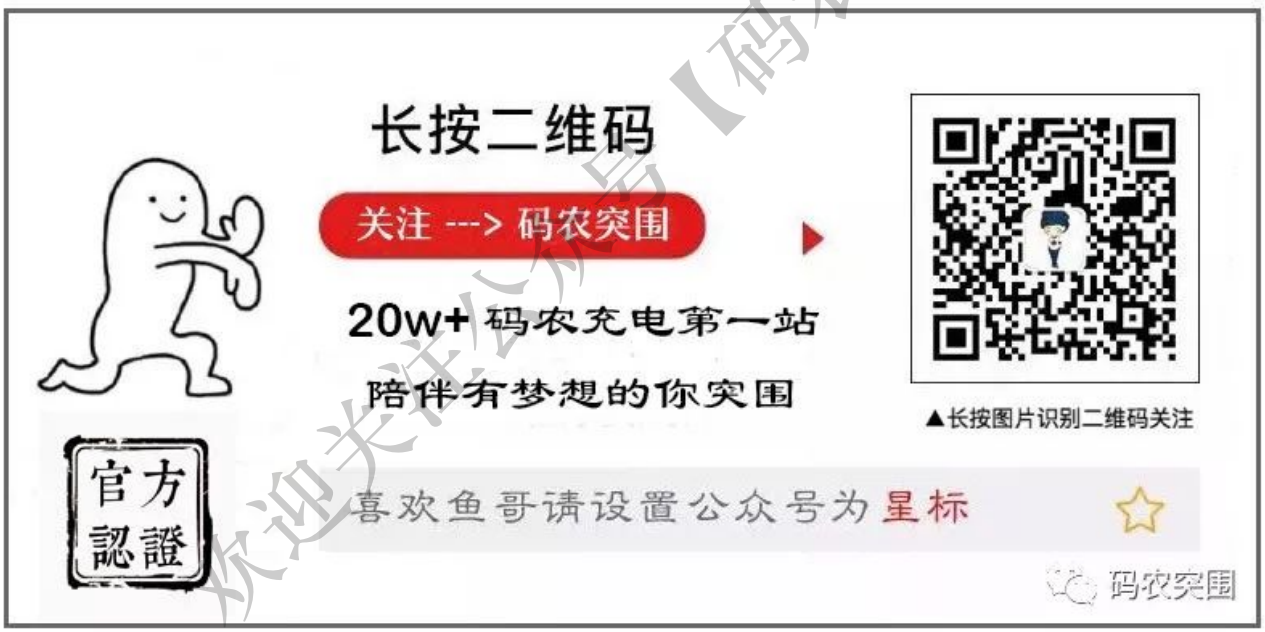
2.codis，目前用的最多的集群方案，基本和twemproxy一致的效果，但它支持在 节点数量改变情况下，旧节点数据可恢复到新hash节点。

3.Redis cluster3.0自带的集，特点在于他的分布式算法不是一致性hash，而是hash槽的概念，以及自身支持节点设置从节点。



扫一扫上面的二维码图案，加我微信

欢迎关注公号：码农突围（id：smartyuge）



Redis读写分离模型

通过增加Slave DB的数量，读的性能可以线性增长。为了避免Master DB的单点故障，集群一般都会采用两台Master DB做双机热备，所以整个集群的读和写的可用性都非常高。

读写分离架构的缺陷在于，不管是Master还是Slave，每个节点都必须保存完整的数据，如果在数据量很大的情况下，集群的扩展能力还是受限于单个节点的存储能力，而且对于Write-intensive类型的应用，读写分离架构并不适合。

Redis数据分片模型

为了解决读写分离模型的缺陷，可以将数据分片模型应用进来。

可以将每个节点看成都是独立的master，然后通过业务实现数据分片。

结合上面两种模型，可以将每个master设计成由一个master和多个slave组成的模型。

Redis提供了哪几种持久化方式？

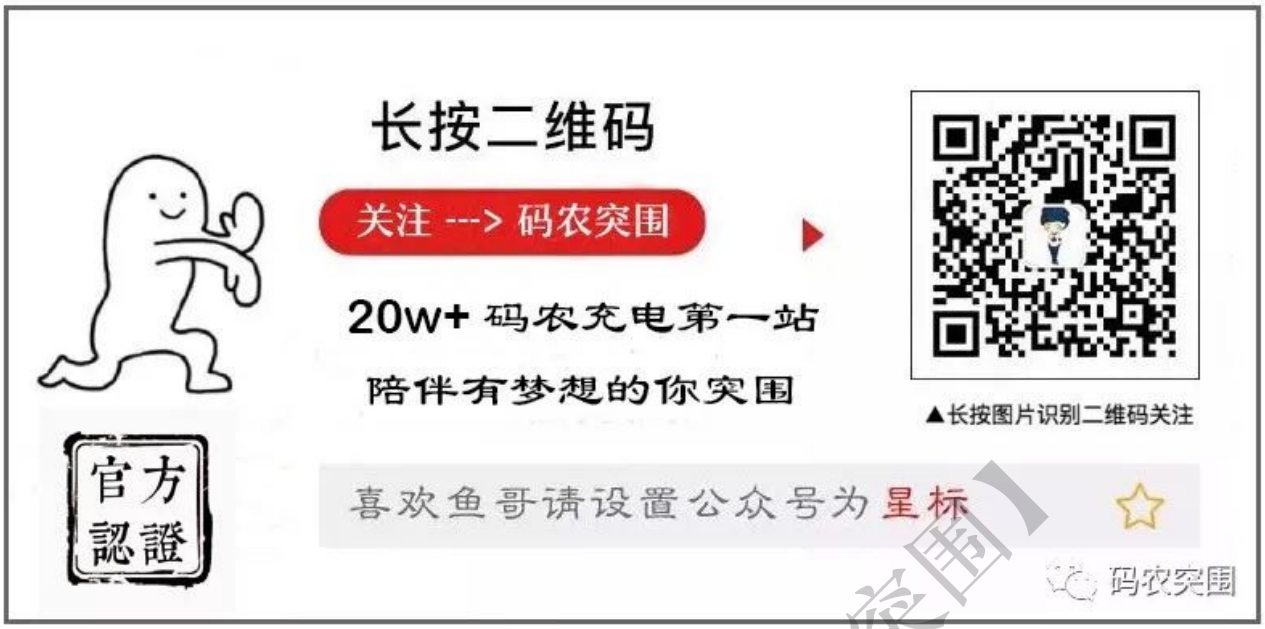
RDB持久化方式能够在指定的时间间隔能对你的数据进行快照存储

AOF持久化方式记录每次对服务器写的操作,当服务器重启的时候会重新执行这些命令来恢复原始的数据,AOF命令以Redis协议追加保存每次写的操作到文件末尾.Redis还能对AOF文件进行后台重写,使得AOF文件的体积不至于过大.

如果你只希望你的数据在服务器运行的时候存在,你也可以不使用任何持久化方式.

你也可以同时开启两种持久化方式, 在这种情况下, 当Redis重启的时候会优先载入AOF文件来恢复原始的数据,因为在通常情况下AOF文件保存的数据集要比RDB文件保存的数据集要完整.

最重要的事情是了解RDB和AOF持久化方式的不同,让我们以RDB持久化方式开始。



如何选择合适的持久化方式？

1.Redis主要提供了两种持久化机制：RDB和AOF；

2.RDB

默认开启，会按照配置的指定时间将内存中的数据快照到磁盘中，创建一个dump.rdb文件，Redis启动时再恢复到内存中。

Redis会单独创建fork()一个子进程，将当前父进程的数据库数据复制到子进程的内存中，然后由子进程写入到临时文件中，持久化的过程结束了，再用这个临时文件替换上次的快照文件，然后子进程退出，内存释放。

需要注意的是，每次快照持久化都会将主进程的数据库数据复制一遍，导致内存开销加倍，若此时内存不足，则会阻塞服务器运行，直到复制结束释放内存；都会将内存数据完整写入磁盘一次，所以如果数据量大的话，而且写操作频繁，必然会引起大量的磁盘I/O操作，严重影响性能，并且最后一次持久化后的数据可能会丢失；

3.AOF

以日志的形式记录每个写操作（读操作不记录），只需追加文件但不可以改写文件，Redis启动时会根据日志从头到尾全部执行一遍以完成数据的恢复工作。包括flushDB也会执行。主要有两种方式触发：有写操作就写、每秒定时写（也会丢数据）。

因为AOF采用追加的方式，所以文件会越来越大，针对这个问题，新增了重写机制，就是当日志文件大到一定程度的时候，会fork出一条新进程来遍历进程内存中的数据，每条记录对应一条set语句，写到临时文件中，然后再替换到旧的日志文件（类似rdb的操作方式）。默认触发是当aof文件大小是上次重写后大小的一倍且文件大于64M时触发。

当两种方式同时开启时，数据恢复Redis会优先选择AOF恢复。一般情况下，只要使用默认开启的RDB即可，因为相对于AOF，RDB便于进行数据库备份，并且恢复数据集的速度也要快很多。

开启持久化缓存机制，对性能会有一定的影响，特别是当设置的内存满了的时候，更是下降到几百reqs/s。所以如果只是用来做缓存的话，可以关掉持久化。

Redis常见性能问题和解决方案？

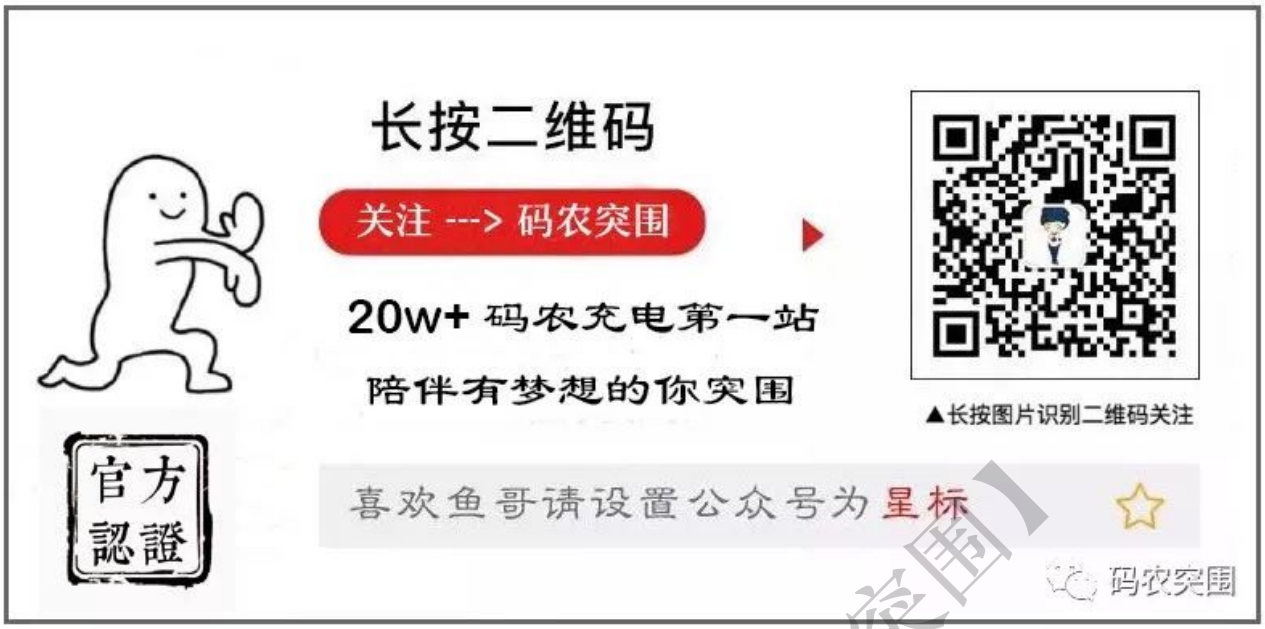
1.Master最好不要做任何持久化工作，如RDB内存快照和AOF日志文件

2.如果数据比较重要，某个Slave开启AOF备份数据，策略设置为每秒同步一次

3.为了主从复制的速度和连接的稳定性，Master和Slave最好在同一个局域网内

4.尽量避免在压力很大的主库上增加从库

5.主从复制不要用图状结构，用单向链表结构更为稳定，即：Master <- Slave1 <- Slave2 <- Slave3... 这样的结构方便解决单点故障问题，实现Slave对Master的替换。如果Master挂了，可以立刻启用Slave1做Master，其他不变。



Redis支持的Java客户端都有哪些？官方推荐用哪个？

Redisson、Jedis、lettuce等等，官方推荐使用Redisson。

Redis哈希槽的概念？

Redis集群没有使用一致性hash,而是引入了哈希槽的概念，当需要在 Redis 集群中放置一个 key-value 时，根据 CRC16(key) mod 16384的值，决定将一个key放到哪个桶中。

Redis集群最大节点个数是多少？

Redis集群预分好16384个桶(哈希槽)

Redis集群的主从复制模型是怎样的？

为了使在部分节点失败或者大部分节点无法通信的情况下集群仍然可用，所以集群使用了主从复制模型,每个节点都会有N-1个复制品.

Redis集群会有写操作丢失吗？为什么？

Redis并不能保证数据的强一致性，这意味这在实际中集群在特定的条件下可能会丢失写操作。

Redis集群之间是如何复制的？

异步复制

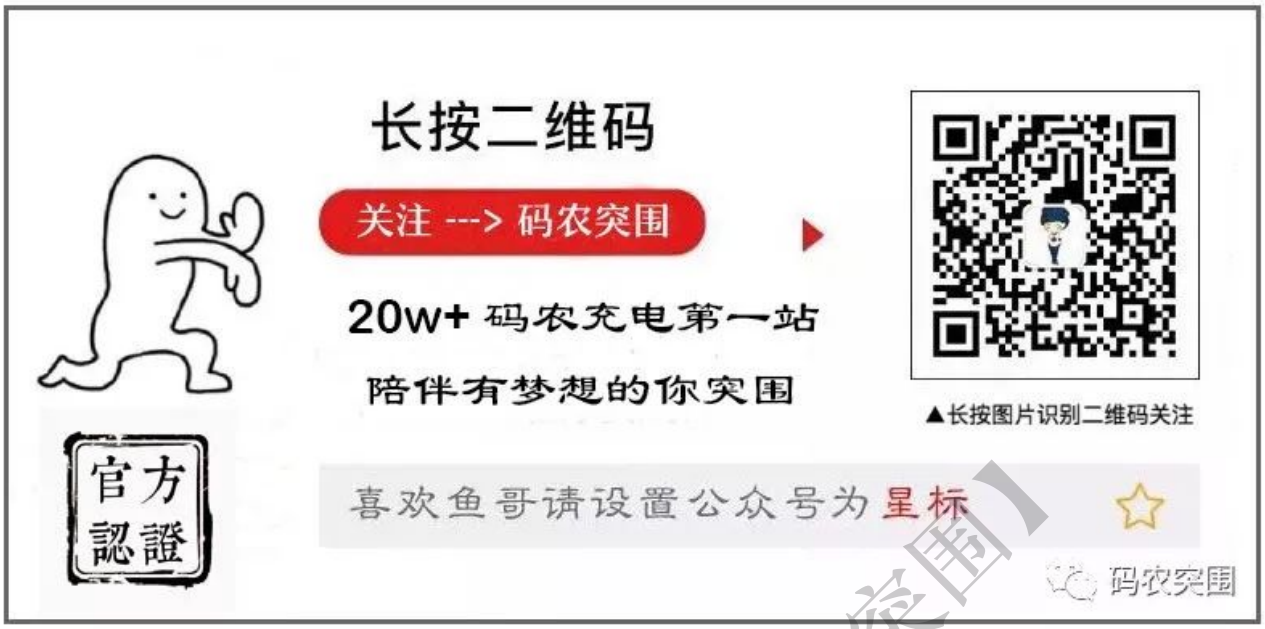
Redis如何做内存优化？

尽可能使用散列表（hashes），散列表（是说散列表里面存储的数少）使用的内存非常小，所以你应该尽可能的将你的数据模型抽象到一个散列表里面。比如你的web系统中有一个用户对象，不要为这个用户的名称，姓氏，邮箱，密码设置单独的key,而是应该把这个用户的所有信息存储到一张散列表里面.

Redis回收进程如何工作的？

一个客户端运行了新的命令，添加了新的数据。

Redi检查内存使用情况，如果大于maxmemory的限制, 则根据设定好的策略进行回收。



Redis回收使用的是什么算法？

LRU算法

Redis有哪些适合的场景？

1.Session共享(单点登录)

2.页面缓存

3.队列

4.排行榜/计数器

5.发布/订阅