##### Redis 数据持久化

redis 是基于内存的，但是redis也会定时的将这些数据持久化到 dump.rdb文件中，然后也会读取rdb文件

在redis.conf中是可以配置dump.rdb（二进制文件）文件的，如果rdb文件换了 那么就等于redis 的库中数据换了 ，Redis启动时在 加载持久化文件 包括rdb 和 aof 文件

###### RDB

默认数据持久化方式 ： rdb

Rdb持久化方式的特点 ：每隔一段时间持久化一次 当宕机时会导致数据丢失 ， fork线程占用资源

主从复制集群是关不掉的

原理: redis会单独创建（fork）一个 与 当前进程一模一样的子线程来进行持久化

这个子进程所有的数据结构（变量、环境变量、程序计数器等）都和原进程一模一样，

先会将原有的rdb数据写到临时文件中，然后原文件进行持久化，待持久化结束了 再 用这个临时文件（在dir中）替换上次的持久化文件 替换好后就自动删除，在这个过 程中 主进程不进行任何io操作 ，从而提高了性能

###### Aof

1. Aof持久化

根据每次操作命令日志将数据追加到aof文件中 ，虽然是日志的形式，但是aof 保存的是重复命令的最后一个命令 ，并不会记录过程，如果修改key的值，那么在下 一次重写aof文件的时候 就会修改这key对应的值

1. Aof持久化特点

宕机时不会导致数据丢失，如果数据丢失可以通过 redis-check-aof工具检查找 回但是由于是日志日志的形式持久化数据会导致aof文件过大、效率降低

(3)Redis 提供了 aof文件重写机制来解决了文件过大的问题

重写机制：对现有的文件压缩，以rdb的形式保存将内存减小

1. 配置配置触发重写时机：auto-aof-rewrite-min-size 64mb(默认，生成环境都是几个G) 当文件大小达到64兆，会自动触发重写
2. 配置如何重写：auto-aof-rewrite-percentge 100(默认) 增长率 当文件第一次到64mb时 ，会触发重写机制，文件只有32mb ，如果文件再增长32时 就再次重写 就会压缩一半到43 ，然后增长43后 又会再次重写

C、手动调用重写bgrewriteaof 命令

原理 ： 当执行aof重写命令时 ，由于另外还有其他的命令在执行， 主线程会把这时的数据写到缓冲区2，当redis重写并压缩完原来的aof（二进制）文件后 ， 然后把缓冲区的执行命令的日志数据加载到新的aof中 从而保证的数据的一致性

缺点： 兼容性差 ，只支持4.0之后的版本使用 ，4.0之前不认识aof文件

##### Redis数据备份方式

Redis 支持数据的备份，即 master-slave 模式的数据备份，且异步 只要主节点备份完毕 即默认成功（这样可能导致数据不一致问题 ：master节点处理数据完毕 异步还未同步至 从节点，master节点挂了 此时数据就不存在 ）

##### 不同类型的不同使用场景

###### String

1. 分布式锁

service 执行时间 大于 锁的过期时间 可以使用redisssion 进行锁续命。

锁续命原理 ： 在执行业务时新开一个线程定时查看，如果未执行完重新设置过 期时间redission过期时间默认30s ， 锁续命查看时间 1/3

1. 计数器

###### Hash ： map 嵌套

###### List ：有序可重复的双向list

Lpush 、Lpop 从左边添加 或者 删除

Rpush、Rpop 从右边添加删除

Blopo ： 从左边 拿出一个元素 ， 没有就等待

Bropo ： 从右边 拿出一个元素 ， 没有就等待

带阻塞的 可以做监听的功能， 有就拿 没有就等待

Lrange 开始位置 结束位置 展示前几条数据

1. 实现 栈的功能 （后进先出）
2. 实现 队列功能 （先进先出） 生产消费
3. 实现延迟队列 消费队列，没有就阻塞等待

###### SET ： 无序不可重复

1. 抽奖功能

参与抽奖的人 存至集合 sadd key userId

查看抽奖人数 smemers key

随机送集合中抽取 若干名中奖用户 ：

* + 1. srandmemery key count(抽几个人) 人数不从集合中删除
    2. Spop key(抽几个人) 人数从集合中删除

1. 点赞功能

点赞 Sadd key value(userId)

取消点赞 srem key value(userId)

检查用户是否点赞 sismember key value(userId)

获取点赞列表 ： 查询 set 集合

获取点赞用户数 ： scard key

###### Zset : key 、value （value中 key、value）

1. 求交集、并集、差集 功能

##### 一个String 默认最大存储大小 512M

##### redis 常见性能解决方案

1、Master 最好不要写内存快照，如果 Master 写内存快照，save 命令调度 rdbSave

函数，会阻塞主线程的工作，当快照比较大时对性能影响是非常大的，会间断性

暂停服务

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秒

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

4、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过期建删除策略设置

1、定时删除:在设置键的过期时间的同时，创建一个定时器 timer(). 让定时器在键

的过期时间来临时，立即执行对键的删除操作。

2、惰性删除:放任键过期不管，但是每次从键空间中获取键时，都检查取得的键是

否过期，如果过期的话，就删除该键;如果没有过期，就返回该键。

3、当内存超过maxMemory时 ，触发主动清理策略

##### redis键回收策略 ： 从内存中淘汰无用的key 提高性能

**设置key 过期时间**

1. volatile-lru：从设置过期时间的数据集中挑选出最近最少使用的数据淘汰
2. volatile-ttl：从设置过期时间的数据集中挑选将要过期的数据淘汰，ttl值越大越优先被淘汰

(3)volatile-random：从设置过期时间的数据中任意挑选一个key删除，只有在内存达到限制无法写入未设置过期时间的数据时

**未设置key 过期时间**

(1)allkeys-lru：从数据集中挑选最近最少使用的数据淘汰，该策略要淘汰的key面向的是全体key集合，而非过期的key集合

(2) allkeys-random：从数据集(server.db[i].dict）中选择任意数据淘汰

(3)no-enviction：禁止驱逐数据，也就是当内存不足以容纳新入数据时，新写入操作就会报错，请求可以继续进行，线上任务也不能持续进行采用no-enviction策略可以保证数据不被丢失，这也是系统默认的一种淘汰策略。

##### Redis 同步机制

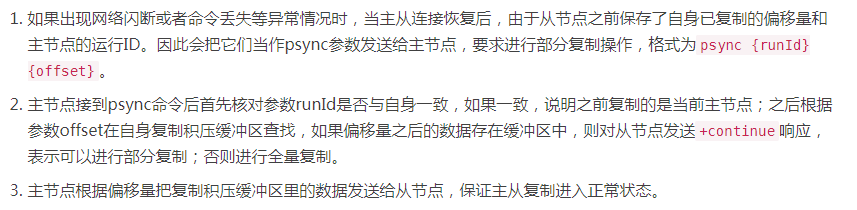
1. 全量复制

场景: 当 slave第一次连接 master时 ,或者master 第一次作为slave时



1. 增量复制

场景: 由于网络异常 或者 命令丢失时 ，当主从连接后从节点进行增量复制



##### pipeLine 批量执行命令使用

多个命令批量一次执行，提高效率 前提： 批量执行的命令没有因果关系性

##### scan 命令使用

由于redis单线程，在生产环境使用keys 命令查询键 会导致性能下降 可以使用scan 高效的分页显 示数据