Redis 是一个非关系形的数据库 他是基于内存的，我们为什么要用redis的呢 redis的优势就是他的读写非常快 读每秒110000次 写每秒81000次

1 redis不仅支持k/v类型的数据，同时还提供了string list，set，zset，hash数据结构的存储。  
2、redis支持集群化，即master-slave模式的主从复制  
3、redis支持数据的持久化，可将内存数据保存在磁盘中，重启的时候可以再次加载进行使用

String 数据类型的使用场景

用set方法存一个key 存放一些信息 用get方法进行获取

Hsah hash特别适合用于存储对象(应为对象可能会包含很多属性)

Redis 有一个redisTemplate：是spring 封装了 RedisTemplate 对象来进行对redis的各种操作，它支持所有的 redis 原生的 api。

我用redisTemplate里面string中的一个get set 方法就是opsForValue（ops噢弗valus）

这方法的作用就是 用来判断接口的幂等性 前台生成一个随机数

Redis hash 是一个 string 类型的 field 和 value 的映射表，hash 特别适合用于存储对象。

Redis 列表是简单的字符串列表，按照插入顺序排序。你可以添加一个元素到列表的头部（左边）或者尾部（右边）。

Set 哈希表实现,元素不重复 是个无序的

Sorted Set(有序集合) 数据插入集合时,已经进行天然排序

Redis 两种持久话方案 默认的是rdb

Rdb ：rdb是对redis中的数据进行一个快照的模式，就是以时间间隔的形式将内存中的数据写入的硬盘中。实际操作过程是fork一个子进程，先将数据集写入到一个临时文件，临时文件写完以后在替换之前的文件，用二进制 压缩存储。

配置的话就是在redis.conf中，找到save的关键字，来指定每隔多长时间有多少个key发生变化就将其持久化。

Aof：aof是将每次写操作对应的指令 保存到后缀名为aof的文本文件中。

Rdb的性能要比aof高

Rdb的文件中直接存储的是数据 也就是说在redis服务启动的时候直接加载rdb文件中数据就可以啦。

而aof文件中存储的是指令 那么redis服务启动的时候先加载aof中指令，在依次去执行这些指令。

Aof的实用性 和可靠性要比rdb高

Rdb呢就是说如果redis发生故障 那么还没来的急持久化的数据就是被丢失，因为他是按时间间隔来进行持久化的 而aof他能够做的数据不容易丢失，你也可以理解为他是实时更新，因为他是redis每发生一次变化就以指令的方式保存在文件中啦。

Redis 的三种状况

雪蹦： 大量的key 同一时间失效

穿透：恶意访问一个你缓存 ，数据库一定不存在的数据

击穿：一个热门的key 突然失效 大量的并发请求直接击穿你的数据库啦

解决方案就是：将热门的key设置永不过期

占空间，内存消耗大，并且不能保持数据最新

我可以去做一个定时器 定时去更新热门的key

1. 设置一个互斥锁 不过不建议这样做

1. 代码复杂度增大

2. 存在死锁的风险

Redis 的集群化

主从复制 哨兵 集群部署

主从复制就是：单台的redis可能会发生单点故障，一旦单点故障 数据就会丢失。

主从复制就将一台redis的数据去复制给多台redis服务器 实现一主多从 他还数据的复制是单项的 只能由主节点到从节点。

主从复制的几个优点吧

数据沉于：实现了数据的备份

故障恢复：如果我的主节点出现了问题 那么我的从节点就提供服务 实现快速的故障恢复。

**负载均衡 ： 主从模式 是读写分离的 如果在写少读多的情况下 ，读redis数据的时候可以从多个redis服务分担读取 可以提高了redis服务的并发量。**

哨兵 ：就是主从复制存在一个问题，故障恢复无法自动化 。哨兵就是解决这个问题的 哨兵就是一个监听 ,哨兵也是一个独立的redis节点，不过他不存储数据，只支持部分命令。**它能够自动完成故障发现和故障转移，并通知应用方，从而实现高可用**。

哨兵的基本原理的：

定时任务 ：每个哨兵节点维护了三个定时任务。分别是：

通过向主从节点发送info命令来获取最新的主从结构

通过发布订阅功能获取其他节点的信息

通过向其他节点发送ping命令进行心跳检查 判断是否下线

主观下线 ：在心跳检测的定时任务中，如果其他节点超过一定时间没有回复，哨兵节点就会将其进行主观下线

**客观下线：**哨兵节点在对主节点进行主观下线后，就是我们在进行配置哨兵的时候会配置一个多少节点同意就对主节点下线 注意是（主节点没有心跳 会询问其他的从节点 主节点的状态 当从节点到达一个数值的时候就下线啦）

**选举领导者哨兵节点：**当主节点被判断客观下线以后，各个哨兵节点会进行协商，选举出一个领导者哨兵节点，并由该领导者节点对其进行故障转移操作。

如果当原来的主节点在启动以后他会变成新主节点的从节点

集群模式：Sentinel模式基本可以满足一般生产的需求，具备高可用性。**但是当遇到单机内存，并发和流量等瓶颈的时候，主从模式或Sentinel模式就不能满足需求了**，这个时候需要**对存储的数据进行分片**，**将数据存储到多个Redis实例中**。

集群是redis官方提供的分布式存储方案 是在3.0版本推出的

Redis就是把多台的redis服务部署在一起 Redis集群采用**无中心结构**，集群由多个节点组成的

Redis集群是两个方面 高可用 数据分片

**数据分区**：数据分区(或称数据分片)是Redis集群最核心的功能。

Redis集群将数据分散到多个节点，一方面突破了Redis单机内存大小的限制，存储容量大大增加；另一方面每个主节点都可以对外提供读服务和写服务，大大提高了集群的响应能力。

**高可用**：集群支持主从复制和主节点的自动故障转移（与哨兵类似）；当任一节点发生故障时，集群仍然可以对外提供服务。

**Redis集群的主从复制模型**

**Redis集群的数据分片**

## 我做过一个io类型的后台管理系统 用的是一个ssm框架。负责的一个功能块是用户管理。其中呢 我印象挺深刻的是做过一个这样的需求：根据用户的生日去给用户发送一条生日邮件。刚开始我直接把数据从mysql库里面直接查出来啦，因为是一个管理系统用户量没那么大 所以就直接查出来。把数据放到redis里面，从redis里面把数据查询出来，在求出当前时间的毫秒值 减去 当年的毫秒值 ，同理 用户生日的毫秒值 减去 生日那年的毫秒值，去比较得出来的毫秒值是否相等，如果相等就把这条数据的邮箱放到一个list的集合中，在把集合放到redis里面。因为每天都要发送 我就写啦一个 Timer（泰麽）****TimerTask（泰麽塔式柯）他们都是java中提供定时调用的两个类 Timer是一种定时器工具，用来在一个后台线程计划执行指定任务，而TimerTask一个抽象类，它的子类代表一个可以被Timer计划的任务。在定时器中设置每天凌晨进行一次调用。一开始我在网上封装了一个邮箱的工具类，工具类是可以调用一次去发送多个邮箱，但存在一个问题 就是如果用户已经把邮箱注销啦，那么这样写程序就会报错，会报一个邮箱不存在。那么这样的写法就不成立啦 最终呢我创建一个线程池 设置线程的数量，依次去调用邮箱的封装类，我已经把邮箱的一次发送多个变成单个发送，这样也提高了程序的性能。最终在测试的时候我发现一个漏洞，就是说如果计算毫秒值的情况下，那么会有一定的差别，如果在去细致化的去计算毫秒值的话 那代码的复杂性就又提高，最后我就直接在sql数据库查询数据的时候，写啦一个**MONTH（慢）函数，**