# Redis

## **redis是什么？**

Redis = Remote Dictionary Server，即远程字典服务。

## **redis五大基本类型**

Redis是一个开源，内存存储的数据结构服务器，可用作数据库，高速缓存和消息队列代理。它支持字符串、哈希表、列表、集合、有序集合，位图，hyperloglogs等数据类型。内置复制、Lua脚本、LRU收回、事务以及不同级别磁盘持久化功能，同时通过Redis Sentinel提供高可用，通过Redis Cluster提供自动分区。

由于redis类型大家很熟悉，且网上命令使用介绍很多，下面重点介绍五大基本类型的底层数据结构与应用场景，以便当开发时，可以熟练使用redis。

### 1 String（字符串）

1.String类型是redis的最基础的数据结构，也是最经常使用到的类型。

而且其他的四种类型多多少少都是在字符串类型的基础上构建的，所以String类型是redis的基础。

2.String 类型的值最大能存储 512MB，这里的String类型可以是简单字符串、

复杂的xml/json的字符串、二进制图像或者音频的字符串、以及可以是数字的字符串

**应用场景**

1、缓存功能：String字符串是最常用的数据类型，不仅仅是redis，各个语言都是最基本类型，因此，利用redis作为缓存，配合其它数据库作为存储层，利用redis支持高并发的特点，可以大大加快系统的读写速度、以及降低后端数据库的压力。

2、计数器：许多系统都会使用redis作为系统的实时计数器，可以快速实现计数和查询的功能。而且最终的数据结果可以按照特定的时间落地到数据库或者其它存储介质当中进行永久保存。

3、统计多单位的数量：eg，uid：gongming count：0 根据不同的uid更新count数量。

4、共享用户session：用户重新刷新一次界面，可能需要访问一下数据进行重新登录，或者访问页面缓存cookie，这两种方式做有一定弊端，1）每次都重新登录效率低下 2）cookie保存在客户端，有安全隐患。这时可以利用redis将用户的session集中管理，在这种模式只需要保证redis的高可用，每次用户session的更新和获取都可以快速完成。大大提高效率。

### 2 List（列表）

1.list类型是用来存储多个有序的字符串的，列表当中的每一个字符看做一个元素

2.一个列表当中可以存储有一个或者多个元素，redis的list支持存储2^32次方-1个元素。

3.redis可以从列表的两端进行插入（pubsh）和弹出（pop）元素，支持读取指定范围的元素集，

或者读取指定下标的元素等操作。redis列表是一种比较灵活的链表数据结构，它可以充当队列或者栈的角色。

4.redis列表是链表型的数据结构，所以它的元素是有序的，而且列表内的元素是可以重复的。

意味着它可以根据链表的下标获取指定的元素和某个范围内的元素集。

**应用场景**

1、消息队列：reids的链表结构，可以轻松实现阻塞队列，可以使用左进右出的命令组成来完成队列的设计。比如：数据的生产者可以通过Lpush命令从左边插入数据，多个数据消费者，可以使用BRpop命令阻塞的“抢”列表尾部的数据。

2、文章列表或者数据分页展示的应用。比如，我们常用的博客网站的文章列表，当用户量越来越多时，而且每一个用户都有自己的文章列表，而且当文章多时，都需要分页展示，这时可以考虑使用redis的列表，列表不但有序同时还支持按照范围内获取元素，可以完美解决分页查询功能。大大提高查询效率。

### 3 Set（集合）

1.redis集合（set）类型和list列表类型类似，都可以用来存储多个字符串元素的集合。

2.但是和list不同的是set集合当中不允许重复的元素。而且set集合当中元素是没有顺序的，不存在元素下标。

3.redis的set类型是使用哈希表构造的，因此复杂度是O(1)，它支持集合内的增删改查，

并且支持多个集合间的交集、并集、差集操作。可以利用这些集合操作，解决程序开发过程当中很多数据集合间的问题。

**应用场景**

1、标签：比如我们博客网站常常使用到的兴趣标签，把一个个有着相同爱好，关注类似内容的用户利用一个标签把他们进行归并。

2、共同好友功能，共同喜好，或者可以引申到二度好友之类的扩展应用。

3、统计网站的独立IP。利用set集合当中元素不唯一性，可以快速实时统计访问网站的独立IP。

数据结构

set的底层结构相对复杂写，使用了intset和hashtable两种数据结构存储，intset可以理解为数组。

### 4 sorted set（有序集合）

redis有序集合也是集合类型的一部分，所以它保留了集合中元素不能重复的特性，但是不同的是，有序集合给每个元素多设置了一个分数。

redis有序集合也是集合类型的一部分，所以它保留了集合中元素不能重复的特性，但是不同的是，

有序集合给每个元素多设置了一个分数，利用该分数作为排序的依据。

**应用场景**

1、 排行榜：有序集合经典使用场景。例如视频网站需要对用户上传的视频做排行榜，榜单维护可能是多方面：按照时间、按照播放量、按照获得的赞数等。

2、用Sorted Sets来做带权重的队列，比如普通消息的score为1，重要消息的score为2，然后工作线程可以选择按score的倒序来获取工作任务。让重要的任务优先执行。

### 5 hash（哈希）

Redis hash数据结构 是一个键值对（key-value）集合,它是一个 string 类型的 field 和 value 的映射表，

redis本身就是一个key-value型数据库，因此hash数据结构相当于在value中又套了一层key-value型数据。

所以redis中hash数据结构特别适合存储关系型对象

**应用场景**

1、由于hash数据类型的key-value的特性，用来存储关系型数据库中表记录，是redis中哈希类型最常用的场景。一条记录作为一个key-value，把每列属性值对应成field-value存储在哈希表当中，然后通过key值来区分表当中的主键。

2、经常被用来存储用户相关信息。优化用户信息的获取，不需要重复从数据库当中读取，提高系统性能。

## **Redis 的三种特殊数据类型**

Geo：Redis3.2推出的，地理位置定位，用于存储地理位置信息，并对存储的信息进行操作。

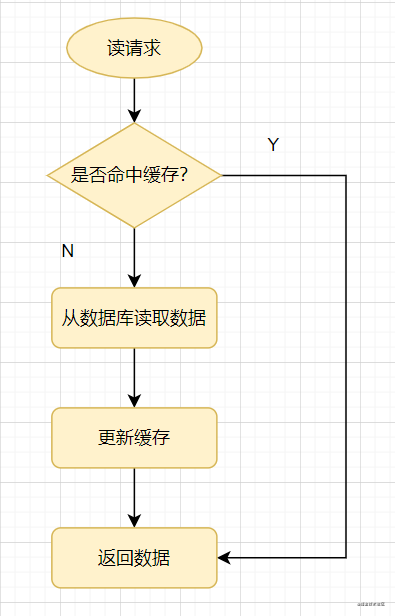
HyperLogLog：用来做基数统计算法的数据结构，如统计网站的UV。

Bitmaps ：用一个比特位来映射某个元素的状态，在Redis中，它的底层是基于字符串类型实现的，可以把bitmaps成作一个以比特位为单位的数组

## 什么是缓存击穿、缓存穿透、缓存雪崩？

### 缓存穿透问题

先来看一个常见的缓存使用方式：读请求来了，先查下缓存，缓存有值命中，就直接返回；缓存没命中，就去查数据库，然后把数据库的值更新到缓存，再返回。



**缓存穿透**：指查询一个一定不存在的数据，由于缓存是不命中时需要从数据库查询，查不到数据则不写入缓存，这将导致这个不存在的数据每次请求都要到数据库去查询，进而给数据库带来压力。

通俗点说，读请求访问时，缓存和数据库都没有某个值，这样就会导致每次对这个值的查询请求都会穿透到数据库，这就是缓存穿透。

**缓存穿透一般都是这几种情况产生的**：

**业务不合理的设计**：比如大多数用户都没开守护，但是你的每个请求都去缓存，查询某个userid查询有没有守护。

**业务/运维/开发失误的操作**：比如缓存和数据库的数据都被误删除了。

**黑客非法请求攻击**：比如黑客故意捏造大量非法请求，以读取不存在的业务数据。

**如何避免缓存穿透呢？ 一般有三种方法：**

1.如果是非法请求，我们在API入口，对参数进行校验，过滤非法值。

2.如果查询数据库为空，我们可以给缓存设置个空值，或者默认值。但是如有有写请求进来的话，需要更新缓存哈，以保证缓存一致性，同时，最后给缓存设置适当的过期时间。（业务上比较常用，简单有效）

3.使用布隆过滤器快速判断数据是否存在。即一个查询请求过来时，先通过布隆过滤器判断值是否存在，存在才继续往下查。

布隆过滤器原理：它由初始值为0的位图数组和N个哈希函数组成。一个对一个key进行N个hash算法获取N个值，在比特数组中将这N个值散列后设定为1，然后查的时候如果特定的这几个位置都为1，那么布隆过滤器判断该key存在。

### 缓存雪奔问题

**缓存雪奔**： 指缓存中数据大批量到过期时间，而查询数据量巨大，请求都直接访问数据库，引起数据库压力过大甚至down机。

缓存雪奔一般是由于大量数据同时过期造成的，对于这个原因，可通过均匀设置过期时间解决，即让过期时间相对离散一点。如采用一个较大固定值+一个较小的随机值，5小时+0到1800秒酱紫。

Redis 故障宕机也可能引起缓存雪奔。这就需要构造Redis高可用集群啦。

### 缓存击穿问题

**缓存击穿**： 指热点key在某个时间点过期的时候，而恰好在这个时间点对这个Key有大量的并发请求过来，从而大量的请求打到db。

缓存击穿看着有点像，其实它两区别是，缓存雪奔是指数据库压力过大甚至down机，缓存击穿只是大量并发请求到了DB数据库层面。可以认为击穿是缓存雪奔的一个子集吧。有些文章认为它俩区别，是区别在于击穿针对某一热点key缓存，雪奔则是很多key。

解决方案就有两种：

1.**使用互斥锁方案**。缓存失效时，不是立即去加载db数据，而是先使用某些带成功返回的原子操作命令，如(Redis的setnx）去操作，成功的时候，再去加载db数据库数据和设置缓存。否则就去重试获取缓存。

2. **“永不过期**”，是指没有设置过期时间，但是热点数据快要过期时，异步线程去更新和设置过期时间。

## **Redis的常用应用场景**

### 缓存

我们一提到redis，自然而然就想到缓存，国内外中大型的网站都离不开缓存。合理的利用缓存，比如缓存热点数据，不仅可以提升网站的访问速度，还可以降低数据库DB的压力。并且，Redis相比于memcached，还提供了丰富的数据结构，并且提供RDB和AOF等持久化机制，强的一批。

### 排行榜

当今互联网应用，有各种各样的排行榜，如电商网站的月度销量排行榜、社交APP的礼物排行榜、小程序的投票排行榜等等。Redis提供的zset数据类型能够实现这些复杂的排行榜。

比如，用户每天上传视频，获得点赞的排行榜可以这样设计：

1.用户Jay上传一个视频，获得6个赞，可以酱紫：

zadd user:ranking:2021-03-03 Jay 3

2.过了一段时间，再获得一个赞，可以这样：

zincrby user:ranking:2021-03-03 Jay 1

3.如果某个用户John作弊，需要删除该用户：

zrem user:ranking:2021-03-03 John

4.展示获取赞数最多的3个用户

zrevrangebyrank user:ranking:2021-03-03 0 2

### 计数器应用

各大网站、APP应用经常需要计数器的功能，如短视频的播放数、电商网站的浏览数。这些播放数、浏览数一般要求实时的，每一次播放和浏览都要做加1的操作，如果并发量很大对于传统关系型数据的性能是一种挑战。Redis天然支持计数功能而且计数的性能也非常好，可以说是计数器系统的重要选择。

### 共享Session

如果一个分布式Web服务将用户的Session信息保存在各自服务器，用户刷新一次可能就需要重新登录了，这样显然有问题。实际上，可以使用Redis将用户的Session进行集中管理，每次用户更新或者查询登录信息都直接从Redis中集中获取。

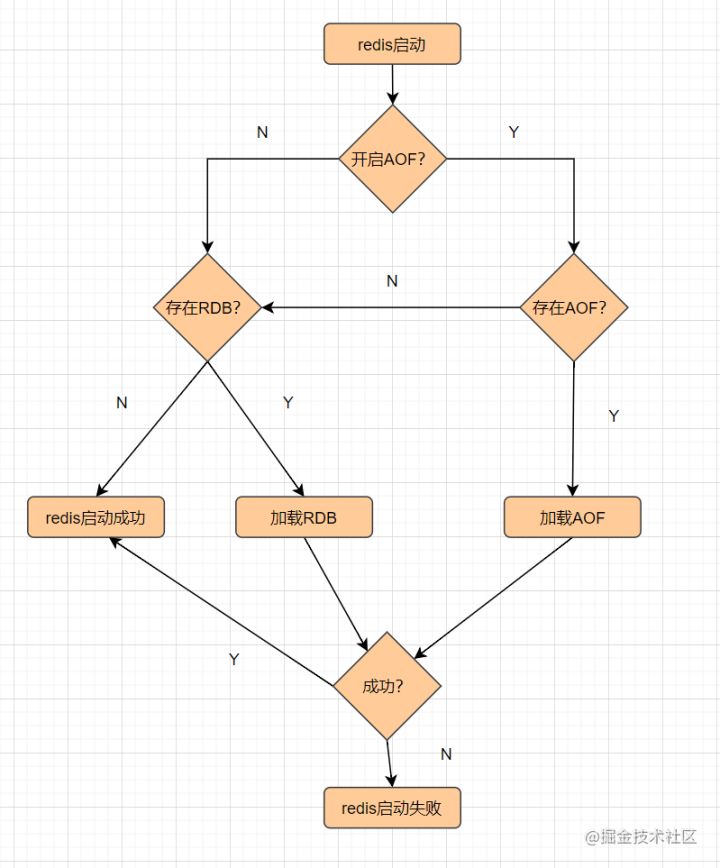
### 社交网络

赞/踩、粉丝、共同好友/喜好、推送、下拉刷新等是社交网站的必备功能，由于社交网站访问量通常比较大，而且传统的关系型数据不太适保存 这种类型的数据，Redis提供的数据结构可以相对比较容易地实现这些功能。

### 消息队列

消息队列是大型网站必用中间件，如ActiveMQ、RabbitMQ、Kafka等流行的消息队列中间件，主要用于业务解耦、流量削峰及异步处理实时性低的业务。Redis提供了发布/订阅及阻塞队列功能，能实现一个简单的消息队列系统。另外，这个不能和专业的消息中间件相比。

## **Redis 的持久化机制**

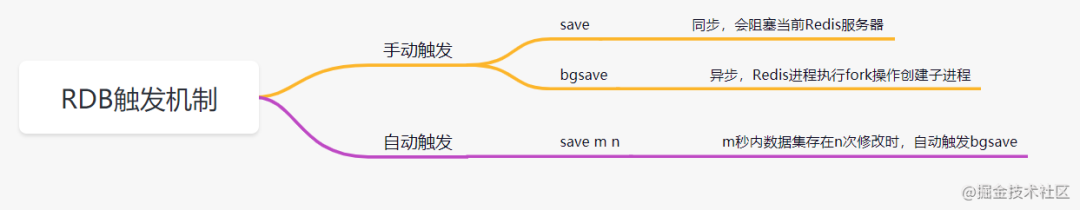


### **RDB**

RDB，就是把内存数据以快照的形式保存到磁盘上。

什么是快照?可以这样理解，给当前时刻的数据，拍一张照片，然后保存下来。

RDB持久化，是将内存中的数据集快照写入磁盘中，它是Redis默认的持久化方式。执行完操作后，在指定目录下会生成一个dump.rdb文件，Redis 重启的时候，通过加载dump.rdb文件来恢复数据。RDB触发机制主要有以下几种：



**RDB 的优点**

适合大规模的数据恢复场景，如备份，全量复制等

**RDB缺点**

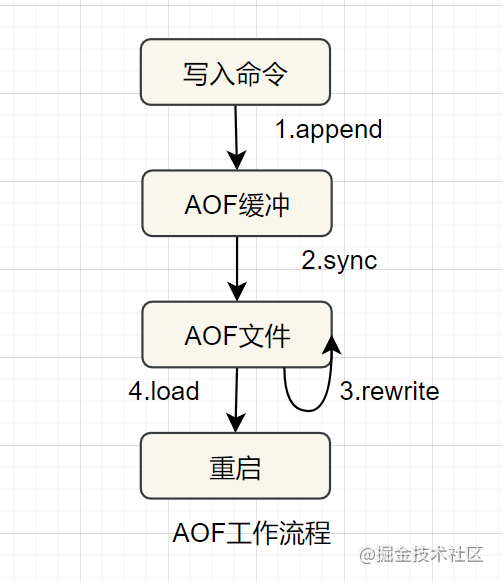
没办法做到实时持久化/秒级持久化。

新老版本存在RDB格式兼容问题

### AOF

AOF（append only file） 持久化，采用日志的形式来记录每个写操作，追加到文件中，重启时再重新执行AOF文件中的命令来恢复数据。它主要解决数据持久化的实时性问题。默认是不开启的。

AOF的工作流程如下：



**AOF的优点**

数据的一致性和完整性更高

**AOF的缺点**

AOF记录的内容越多，文件越大，数据恢复变慢。