**什么是 Redis？简述它的优缺点？**

Redis 的全称是：Remote Dictionary.Server，本质上是一个 Key-Value 类型的内存数据库，很像memcached，整个数据库统统加载在内存当中进行操作，定期通过异步操作把数据库数据 flush 到硬盘上进行保存。因为是纯内存操作，Redis 的性能非常出色，每秒可以处理超过 10 万次读写操作，是已知性能最快的Key-Value DB。Redis 的出色之处不仅仅是性能，Redis 最大的魅力是支持保存多种数据结构，此外单个 value 的最大限制是 1GB，不像 memcached 只能保存 1MB 的数据，因此 Redis 可以用来实现很多有用的功能。Redis 的主要缺点是数据库容量受到物理内存的限制，不能用作海量数据的高性能读写，因此 Redis 适合的场景主要局限在较小数据量的高性能操作和运算上。

**Redis 用什么语言开发的？**

Redis是一个开源的使用ANSI [C语言](https://baike.baidu.com/item/C%E8%AF%AD%E8%A8%80)编写、支持网络、可基于内存亦可持久化的日志型、Key-Value[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728)，并提供多种语言的API

**Redis 用在哪里？**

缓存，nosql，session共享，注册中心、

**消息队列**（lpush插入一条，brpop阻塞弹出，publish订阅发布模式），

**分布式锁**（利用redis setNx(如果不存在成功，存在不操作)和expire（设置过期时间））

**//上锁**

jedis.set(String lock, String requestid, String nx, String ex, int time)

lock：锁名

requestid：确保上锁和获取锁是同一个客户端、

nx：代表setNx就是不存在操作，存在不操作、

ex：代表要设置过期时间，值为第五个参数、

time：为过期时间值

**//获取锁 jedis ：redis客户端 lockKey: 锁，也就是key值 requestId：客户端身份，用于解锁时验证身份**

**requestId使用原因:（比如客户端A加锁，一段时间之后客户端A解锁，在执行jedis.del()之前，锁突然过期了，此时客户端B尝试加锁成功，然后客户端A再执行del()方法，则将客户端B的锁给解除了**。**）**

  public static boolean releaseDistributedLock(Jedis jedis, String lockKey, String requestId) {

**//脚本，获取当前锁的value值，判断是否和获取锁的requesid相等**

String script = "if redis.call('get', KEYS[1]) == ARGV[1] then return redis.call('del', KEYS[1]) else return 0 end";

**//将脚本通过eval交由redis执行**

Object result = jedis.eval(script, Collections.singletonList(lockKey), Collections.singletonList(requestId));

**//如果返回成功则可以获取锁，否则不能**

if (RELEASE\_SUCCESS.equals(result)) {

return true;

}

return false;

}

**Redis 与 memcached 相比有哪些优势？**

* 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
* redis 的速度比 memcached 快很多 redis 的速度比 memcached 快很多
* redis 可以持久化其数据 redis 可以持久化其数据

**Redis支持的数据类型？**

**String：**可以包含任何数据,比如jpg图片或者序列化的对象,一个键最大能存储512M。

命令：set、get。

**List：**链表(双向链表)，增删快，提供了操作某一段元素的API。适用于：最新消息排行等功能；消息队列

命令：lpush（插入到头）、rpush(插入到尾)、lpop（弹出头）、rpop（弹出尾）、blpop和brpop（弹出阻塞，有值弹出，无值阻塞）。

**Set:** 集合。哈希表实现，元素不重复，为集合提供了求交集、并集、差集等操作。适用于：共同好友；利用唯一性，统计访问网站的所有独立ip；好友推荐时，根据tag求交集，大于某个阈值就可以推荐。

命令：sadd（新增，无序唯一）、srem（删除）、scard（返回数量）、sismember（查看全部元素）。

**Hash:** 字典。键值对集合，即编程语言中的Map类型。适合存储对象，并且可以像数据库中update一个属性一样只修改某一项属性值。适用于：存储、读取、修改用户属性。

命令：hset、hget、hmset、hmget、hlen。

**Sorted Set：**有序集合。将Set中的元素增加一个权重参数score，元素按score有序排列。数据插入集合时，已经进行天然排序。适用于：排行榜；带权重的消息队列。

命令：ZADD key score1 member1 [score2 member2]。

**BitMap，Geo（存储经纬度地理位置），HyperLogLog**

**Redis持久化方式**

**RDB:**保存某个时间点的数据，因此保存数据可能不完整，保存成dump.rdb文件，父进程生出一个子进程来生成rdb文件，当子进程完成对新rdb文件写入时替换掉旧的rdb

**AOF：**方式默认每秒钟备份1次，频率很高，它的操作方式是以追加的方式记录日志而不是数据，AOF 文件的体积通常要大于 RDB 文件的体积。AOF 的文件提及比较大，而且由于保存频率很高，所以整体的速度会比 RDB 慢一些，但是性能依旧很高。

**Redis 事务？**

[MULTI](https://www.runoob.com/redis/transactions-multi.html)（标记一个事务块的开始。）  =》redis命令 =》[EXEC](https://www.runoob.com/redis/transactions-exec.html) （执行所有事务块内的命令。）

[DISCARD](https://www.runoob.com/redis/transactions-discard.html) （取消事务，放弃执行事务块内的所有命令）

**Redis的淘汰策略：**

**volatile-lru：**使用LRU算法删除一个键（只对设置了生存时间的键）

**allkeys-lru：**使用LRU算法删除一个键

**volatile-random：**随机删除一个键（只对设置了生存时间的键）

**allkeys-random：**随机删除一个键

**volatile-ttl：**删除生存时间最近的一个键

**Noeviction：**不删除键，只返回错误

**Redis的淘汰策略中的LRU算法：**

(Least recently used，最近最少使用）算法根据数据的历史访问记录来进行淘汰数据，其核心思想是“如果数据最近被访问过，那么将来被访问的几率也更高”。维护一个链表，新加入的元素在头部，被使用元素移往头部，未使用的会在链表结尾淘汰

**什么是缓存穿透？如何避免？：**

一般的缓存系统，都是按照key去缓存查询，如果不存在对应的value，就应该去后端系统查找（比如DB）。一些恶意的请求会故意查询不存在的key,请求量很大，就会对后端系统造成很大的压力。这就叫做缓存穿透。

1：对查询结果为空的情况也进行缓存，缓存时间设置短一点，或者该key对应的数据insert了之后清理缓存。

2：对一定不存在的key进行过滤。**布隆过滤器**

**什么是缓存雪崩？何如避免？**

当缓存服务器重启或者大量缓存集中在某一个时间段失效，这样在失效的时候，会给后端系统带来很大压力。导致系统崩溃。

1：在缓存失效后，通过加锁或者队列来控制读数据库写缓存的线程数量。比如对某个key只允许一个线程查询数据和写缓存，其他线程等待。

2：做二级缓存，A1为原始缓存，A2为拷贝缓存，A1失效时，可以访问A2，A1缓存失效时间设置为短期，A2设置为长期

3：不同的key，设置不同的过期时间，让缓存失效的时间点尽量均匀。