# 1、什么是Redis?

Redis 是完全开源免费的， 遵守 BSD 协议， 是一个高性能的 key-value 数据库。

Redis 与其他 key - value 缓存产品有以下三个特点：

Redis 支持数据的持久化，可以将内存中的数据保存在磁盘中，重启的时候可以再次加载进行使用。

Redis 不仅仅支持简单的 key-value 类型的数据， 同时还提供 list， set， zset， hash 等数据结构的存储。

Redis 支持数据的备份， 即 master-slave 模式的数据备份。

Redis 优 势

性能极高 – Redis 能读的速度是 110000 次/s,写的速度是 81000 次/s 。

丰富的数据类型 – Redis 支持二进制案例的 Strings, Lists, Hashes, Sets 及Ordered Sets 数据类型操作。

原子 – Redis 的所有操作都是原子性的，意思就是要么成功执行要么失败完全不执行。单个操作是原子性的。多个操作也支持事务，即原子性，通过 MULTI 和 EXEC 指令包起来。

丰富的特性 – Redis 还支持 publish/subscribe, 通知, key 过期等等特性。

Redis 与其他 key-value 存储有什么不同?

Redis 有着更为复杂的数据结构并且提供对他们的原子性操作，这是一个不同于其他数据库的进化路径。Redis 的数据类型都是基于基本数据结构的同时对程序员透明， 无需进行额外的抽象。

Redis 运行在内存中但是可以持久化到磁盘，所以在对不同数据集进行高速读写时需要权衡内存， 因为数据量不能大于硬件内存。在内存数据库方面的另一个优点是，相比在磁盘上相同的复杂的数据结构，在内存中操作起来非常简单，这样 Redis 可以做很多内部复杂性很强的事情。同时， 在磁盘格式方面他们是紧凑的以追加的方式产生的， 因为他们并不需要进行随机访问。

## 2、Redis 的数据类型?

答： Redis 支持五种数据类型： string（字符串），hash（哈希）， list（列表）， set（集合） 及 zsetsorted set： 有序集合)。

我们实际项目中比较常用的是 string，hash 如果你是 Redis 中高级用户，还需要加上下面几种数据结构 HyperLogLog、Geo、Pub/Sub。

如果你说还玩过 Redis Module，像 BloomFilter，RedisSearch，Redis-ML，面试官得眼睛就开始发亮了。

## 3、使用Redis 有哪些好处?

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 O1)

2、支持丰富数据类型， 支持 string， list， set， Zset， hash 等

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

## 4、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

2、Redis 的速度比 Memcached 快很3、Redis 可以持久化其数据

3、更多面试题关注微信公众号：Java2B

## 5、Memcache 与Redis 的区别都有哪些?

1、存储方式 Memecache 把数据全部存在内存之中， 断电后会挂掉， 数据不能超过内存大小。 Redis 有部份存在硬盘上， 这样能保证数据的持久性

2、数据支持类型 Memcache 对数据类型支持相对简单。 Redis 有复杂的数据类型。

3、使用底层模型不同 它们之间底层实现方式 以及与客户端之间通信的应用协议不一样。 Redis 直接自己构建了 VM 机制 ， 因为一般的系统调用系统函数的话， 会浪费一定的时间去移动和请求。

## 6、Redis 是单进程单线程的?

答： 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。

## 7、一个字符串类型的值能存储最大容量是多少?

答： 512M

## 8、Redis 的持久化机制是什么？各自的优缺点?

Redis 提供两种持久化机制 RDB 和 AOF 机制:

1、RDBRedis DataBase)持久化方式： 是指用数据集快照的方式半持久化模式) 记录 redis 数据库的所有键值对,在某个时间点将数据写入一个临时文件， 持久化结束后， 用这个临时文件替换上次持久化的文件， 达到数据恢复。

优点：

1、只有一个文件 dump.rdb， 方便持久化。

2、容灾性好， 一个文件可以保存到安全的磁盘。

3、性能最大化， fork 子进程来完成写操作， 让主进程继续处理命令， 所以是 IO 最大化。使用单独子进程来进行持久化，主进程不会进行任何 IO 操作，保证了 redis 的高性能) 4.相对于数据集大时， 比 AOF 的启动效率更高。

缺点:

1、数据安全性低。RDB 是间隔一段时间进行持久化，如果持久化之间 redis 发生故障， 会发生数据丢失。所以这种方式更适合数据要求不严谨的时候)

2、AOFAppend-only file)持久化方式： 是指所有的命令行记录以 redis 命令请求协议的格式完全持久化存储)保存为 aof 文件。

优点：

1、数据安全， aof 持久化可以配置 appendfsync 属性， 有 always， 每进行一次命令操作就记录到 aof 文件中一次。

2、通过 append 模式写文件， 即使中途服务器宕机， 可以通过 redis-check-aof 工具解决数据一致性问题。

3、AOF 机制的 rewrite 模式。AOF 文件没被 rewrite 之前（文件过大时会对命令进行合并重写）， 可以删除其中的某些命令（比如误操作的 flushall））

缺点：

1、AOF 文件比 RDB 文件大， 且恢复速度慢。

2、数据集大的时候， 比 rdb 启动效率低。

## 9、Redis 常见性能问题和解决方案：

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

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、尽量避免在压力很大的主库上增加从

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。

## 10、redis 过期键的删除策略?

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、定期删除:每隔一段时间程序就对数据库进行一次检查，删除里面的过期键。至 于要删除多少过期键， 以及要检查多少个数据库， 则由算法决定。

## 11、Redis 的回收策略（淘汰策略）

volatile-lru：从已设置过期时间的数据集（ server.db[i].expires）中挑选最近最少使用的数据淘汰

volatile-ttl： 从已设置过期时间的数据集（ server.db[i].expires） 中挑选将要过期的数据淘汰

volatile-random： 从已设置过期时间的数据集（ server.db[i].expires） 中任意选择数据淘汰

allkeys-lru： 从数据集（ server.db[i].dict） 中挑选最近最少使用的数据淘汰

allkeys-random： 从数据集（ server.db[i].dict） 中任意选择数据淘汰

no-enviction（驱逐）： 禁止驱逐数据

注意这里的 6 种机制，volatile 和 allkeys 规定了是对已设置过期时间的数据集淘汰数据还是从全部数据集淘汰数据， 后面的 lru、ttl 以及 random 是三种不同的淘汰策略， 再加上一种 no-enviction 永不回收的策略。

使用策略规则：

1、如果数据呈现幂律分布，也就是一部分数据访问频率高，一部分数据访问频率 低， 则使用 allkeys-lru

2、如果数据呈现平等分布， 也就是所有的数据访问频率都相同， 则使用allkeys-random

## 12、为什么 edis 需要把所有数据放到内存中?

答：Redis 为了达到最快的读写速度将数据都读到内存中，并通过异步的方式将数据写入磁盘。所以 redis 具有快速和数据持久化的特征。如果不将数据放在内存中， 磁盘 I/O 速度为严重影响 redis 的性能。在内存越来越便宜的今天， redis 将会越来越受欢迎。如果设置了最大使用的内存， 则数据已有记录数达到内存限值后不能继续插入新值。

## 13、Redis 的同步机制了解么?

答：Redis 可以使用主从同步，从从同步。第一次同步时，主节点做一次 bgsave， 并同时将后续修改操作记录到内存 buffer， 待完成后将 rdb 文件全量同步到复制节点， 复制节点接受完成后将 rdb 镜像加载到内存。加载完成后， 再通知主节点将期间修改的操作记录同步到复制节点进行重放就完成了同步过程。

## 14、Pipeline 有什么好处，为什么要用pipeline?

答：可以将多次 IO 往返的时间缩减为一次，前提是 pipeline 执行的指令之间没有因果相关性。使用 redis-benchmark 进行压测的时候可以发现影响 redis 的 QPS 峰值的一个重要因素是 pipeline 批次指令的数目。

## 15、是否使用过 Redis 集群，集群的原理是什么?

1. 、Redis Sentinal 着眼于高可用， 在 master 宕机时会自动将 slave 提升为master， 继续提供服务。
2. 、Redis Cluster 着眼于扩展性， 在单个 redis 内存不足时， 使用 Cluster 进行分片存储。

## 16、Redis 集群方案什么情况下会导致整个集群不可用?

答： 有 A， B， C 三个节点的集群,在没有复制模型的情况下,如果节点 B 失败了， 那么整个集群就会以为缺少 5501-11000 这个范围的槽而不可用。

## 17、Redis 支持的Java 客户端都有哪些？官方推荐用哪个?

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

## 18、Jedis 与 Redisson 对比有什么优缺点?

答： Jedis 是 Redis 的 Java 实现的客户端， 其 API 提供了比较全面的 Redis 命令的支持；Redisson 实现了分布式和可扩展的 Java 数据结构，和 Jedis 相比，功能较为简单， 不支持字符串操作， 不支持排序、事务、管道、分区等 Redis 特性。Redisson 的宗旨是促进使用者对 Redis 的关注分离，从而让使用者能够将精力更集中地放在处理业务逻辑上。

## 19、Redis 如何设置密码及验证密码?

设置密码： config set requirepass 123456 授权密码： auth 123456

## 20、说说 Redis 哈希槽的概念?

答： Redis 集群没有使用一致性 hash,而是引入了哈希槽的概念， Redis 集群有16384 个哈希槽，每个 key 通过 CRC16 校验后对 16384 取模来决定放置哪个槽， 集群的每个节点负责一部分 hash 槽。

## 21、Redis 集群的主从复制模型是怎样的?

答： 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.

## 22、Redis 集群会有写操作丢失吗？为什么?

答：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。

## 23、Redis 集群之间是如何复制的?

答： 异步复制

## 24、Redis 集群最大节点个数是多少?

答： 16384 个。

## 25、Redis 集群如何选择数据库?

答： Redis 集群目前无法做数据库选择， 默认在 0 数据库。

## 26、怎么测试 Redis 的连通性?

答： 使用 ping 命令。

## 27、怎么理解 Redis 事务?

答：

1） 事务是一个单独的隔离操作： 事务中的所有命令都会序列化、按顺序地执行。事务在执行的过程中， 不会被其他客户端发送来的命令请求所打断。

2） 事务是一个原子操作： 事务中的命令要么全部被执行， 要么全部都不执行。

## 28、Redis 事务相关的命令有哪几个?

答： MULTI、EXEC、DISCARD、WATCH

## 29、Redis key 的过期时间和永久有效分别怎么设置?

答： EXPIRE 和 PERSIST 命令。

## 30、Redis 如何做内存优化?

答： 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. 更多面试题关注微信公众号：Java2B

## 31、Redis 回收进程如何工作的?

答： 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。一个新的命令被执行， 等等。所以我们不断地穿越内存限制的边界， 通过不断达到边界然后不断地回收回到边界以下。如果一个命令的结果导致大量内存被使用（ 例如很大的集合的交集保存到一个新的键）， 不用多久内存限制就会被这个内存使用量超越。

## 32、都有哪些办法可以降低 Redis 的内存使用情况呢?

答：如果你使用的是 32 位的 Redis 实例，可以好好利用 Hash,list,sorted set,set 等集合类型数据， 因为通常情况下很多小的 Key-Value 可以用更紧凑的方式存放到一起。

## 33、Redis 的内存用完了会发生什么?

答：如果达到设置的上限，Redis 的写命令会返回错误信息（ 但是读命令还可以正常返回。） 或者你可以将 Redis 当缓存来使用配置淘汰机制， 当 Redis 达到内存上限时会冲刷掉旧的内容。

## 34、一个 Redis 实例最多能存放多少的 keys？List、Set、Sorted Set 他们最多能存放多少元素?

答：理论上 Redis 可以处理多达 232 的 keys，并且在实际中进行了测试，每个实例至少存放了 2 亿 5 千万的 keys。我们正在测试一些较大的值。任何 list、set、和 sorted set 都可以放 232 个元素。换句话说， Redis 的存储极限是系统中的可用内存值。

## 35、MySQL 里有 2000w 数据，redis 中只存 20w 的数据，如何保证redis 中的数据都是热点数据?

答： 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。相关知识： Redis 提供 6 种数据淘汰策略：

volatile-lru：从已设置过期时间的数据集（server.db[i].expires）中挑选最近最

少使用的数据淘汰

volatile-ttl： 从已设置过期时间的数据集（server.db[i].expires） 中挑选将要过期的数据淘汰

volatile-random： 从已设置过期时间的数据集（server.db[i].expires） 中任意选择数据淘汰

allkeys-lru： 从数据集（server.db[i].dict） 中挑选最近最少使用的数据淘汰

allkeys-random： 从数据集（server.db[i].dict） 中任意选择数据淘汰

no-enviction（驱逐）： 禁止驱逐数据

## 36、Redis 最适合的场景?

1、会话缓存（Session Cache）

最常用的一种使用 Redis 的情景是会话缓存（session cache）。用 Redis 缓存会话比其他存储（如 Memcached）的优势在于：Redis 提供持久化。当维护一个不是严格要求一致性的缓存时， 如果用户的购物车信息全部丢失， 大部分人都会不高兴的， 现在， 他们还会这样吗？ 幸运的是， 随着 Redis 这些年的改进， 很容易找到怎么恰当的使用 Redis 来缓存会话的文档。甚至广为人知的商业平台 Magento 也提供 Redis 的插件。

2、全页缓存（FPC）

除基本的会话 token 之外， Redis 还提供很简便的 FPC 平台。回到一致性问题， 即使重启了 Redis 实例， 因为有磁盘的持久化， 用户也不会看到页面加载速度的下降，这是一个极大改进，类似 PHP 本地 FPC。 再次以 Magento 为例，Magento 提供一个插件来使用 Redis 作为全页缓存后端。 此外， 对 WordPress 的用户来说， Pantheon 有一个非常好的插件 wp-redis， 这个插件能帮助你以最快速度加载你曾浏览过的页面。

3、队列

Reids 在内存存储引擎领域的一大优点是提供 list 和 set 操作， 这使得 Redis 能作为一个很好的消息队列平台来使用。Redis 作为队列使用的操作，就类似于本地程序语言（如 Python）对 list 的 push/pop 操作。 如果你快速的在 Google 中搜索“ Redis queues”， 你马上就能找到大量的开源项目， 这些项目的目的就是利用 Redis 创建非常好的后端工具， 以满足各种队列需求。例如， Celery 有一个后台就是使用 Redis 作为 broker， 你可以从这里去查看。

4， 排行榜/计数器

Redis 在内存中对数字进行递增或递减的操作实现的非常好。集合（Set） 和有序集合（Sorted Set） 也使得我们在执行这些操作的时候变的非常简单，Redis 只是正好提供了这两种数据结构。所以， 我们要从排序集合中获取到排名最靠前的 10 个用户– 我们称之为“ user_scores”， 我们只需要像下面一样执行即可： 当然，这是假定你是根据你用户的分数做递增的排序。如果你想返回用户及用户的分数，你需要这样执行： ZRANGE user_scores 0 10 WITHSCORES Agora Games 就是一个很好的例子， 用 Ruby 实现的， 它的排行榜就是使用 Redis 来存储数据的， 你可以在这里看到。

5、发布/订阅

最后（但肯定不是最不重要的）是 Redis 的发布/订阅功能。发布/订阅的使用场景确实非常多。我已看见人们在社交网络连接中使用， 还可作为基于发布/订阅的脚本触发器， 甚至用 Redis 的发布/订阅功能来建立聊天系统！

## 37、假如 Redis 里面有 1 亿个key，其中有 10w 个key 是以某个固定的已知的前缀开头的，如果将它们全部找出来?

答： 使用 keys 指令可以扫出指定模式的 key 列表。

对方接着追问： 如果这个 redis 正在给线上的业务提供服务， 那使用 keys 指令会有什么问题?

这个时候你要回答 redis 关键的一个特性：redis 的单线程的。keys 指令会导致线程阻塞一段时间， 线上服务会停顿， 直到指令执行完毕， 服务才能恢复。这个时候可以使用 scan 指令， scan 指令可以无阻塞的提取出指定模式的 key 列表， 但是会有一定的重复概率， 在客户端做一次去重就可以了， 但是整体所花费的时间会比直接用 keys 指令长。

## 38、如果有大量的 key 需要设置同一时间过期，一般需要注意什么?

答：如果大量的 key 过期时间设置的过于集中，到过期的那个时间点，redis 可能会出现短暂的卡顿现象。一般需要在时间上加一个随机值， 使得过期时间分散一些。

## 39、使用过 Redis 做异步队列么，你是怎么用的?

答：一般使用 list 结构作为队列，rpush 生产消息，lpop 消费消息。当 lpop 没有消息的时候， 要适当 sleep 一会再重试。

如果对方追问可不可以不用 sleep 呢?

list 还有个指令叫 blpop，在没有消息的时候，它会阻塞住直到消息到来。如果对方追问能不能生产一次消费多次呢? 使用 pub/sub 主题订阅者模式， 可以实现1:N 的消息队列。

如果对方追问 pub/sub 有什么缺点?

在消费者下线的情况下，生产的消息会丢失，得使用专业的消息队列如 RabbitMQ 等。

如果对方追问 redis 如何实现延时队列?

我估计现在你很想把面试官一棒打死如果你手上有一根棒球棍的话， 怎么问的这么详细。但是你很克制，然后神态自若的回答道：使用 sortedset，拿时间戳作为score，消息内容作为 key 调用 zadd 来生产消息，消费者用 zrangebyscore 指令获取 N 秒之前的数据轮询进行处理。到这里， 面试官暗地里已经对你竖起了大拇指。但是他不知道的是此刻你却竖起了中指， 在椅子背后。

## 40、使用过 Redis 分布式锁么，它是什么回事?

先拿 setnx 来争抢锁， 抢到之后， 再用 expire 给锁加一个过期时间防止锁忘记了释放。

这时候对方会告诉你说你回答得不错， 然后接着问如果在 setnx 之后执行 expire 之前进程意外 crash 或者要重启维护了， 那会怎么样?

这时候你要给予惊讶的反馈： 唉， 是喔， 这个锁就永远得不到释放了。紧接着你需要抓一抓自己得脑袋， 故作思考片刻， 好像接下来的结果是你主动思考出来的， 然后回答： 我记得 set 指令有非常复杂的参数， 这个应该是可以同时把 setnx 和expire 合成一条指令来用的！ 对方这时会显露笑容， 心里开始默念： 摁， 这小子还不错。

## 41、如何实现集群中的 session 共享存储?

Session 是运行在一台服务器上的，所有的访问都会到达我们的唯一服务器上，这样我们可以根据客户端传来的 sessionID，来获取 session，或在对应 Session 不存在的情况下（ session 生命周期到了/用户第一次登录），创建一个新的 Session； 但是， 如果我们在集群环境下， 假设我们有两台服务器 A， B， 用户的请求会由Nginx 服务器进行转发（ 别的方案也是同理）， 用户登录时， Nginx 将请求转发至服务器 A 上，A 创建了新的 session，并将 SessionID 返回给客户端，用户在浏览其他页面时，客户端验证登录状态， Nginx 将请求转发至服务器 B， 由于 B 上并没有对应客户端发来 sessionId 的 session，所以会重新创建一个新的 session，并且再将这个新的 sessionID 返回给客户端， 这样， 我们可以想象一下， 用户每一次操作都有 1/2 的概率进行再次的登录， 这样不仅对用户体验特别差， 还会让服务器上的 session 激增， 加大服务器的运行压力。

为了解决集群环境下的 seesion 共享问题， 共有 4 种解决方案：

1. 粘性 session

粘性 session 是指 Ngnix 每次都将同一用户的所有请求转发至同一台服务器上， 即将用户与服务器绑定。

1. 服务器 session 复制

即每次 session 发生变化时， 创建或者修改， 就广播给所有集群中的服务器， 使所有的服务器上的 session 相同。

1. session 共享

缓存 session， 使用 redis， memcached。4.session 持久化

将 session 存储至数据库中， 像操作数据一样才做 session。

## 42、memcached 与redis 的区别?

1、Redis 不仅仅支持简单的 k/v 类型的数据，同时还提供 list，set，zset， hash 等数据结构的存储。而 memcache 只支持简单数据类型，需要客户端自己处理复杂对象

2、Redis 支持数据的持久化， 可以将内存中的数据保持在磁盘中， 重启的时候可以再次加载进行使用（ PS： 持久化在 rdb、aof）。