## 1、什么是Redis?

回答：Redis是一个基于内存的高性能key-value数据库；

应用场景：1）会话缓存（Session Cache） 2）全页缓存（FPC） 3）队列 4）排行榜/计数器 5）发布/订阅

## 2、使用Redis有哪些好处?

回答：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O（1）；

2）支持丰富的数据类型，支持string，list，set，sorted set，hash；

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；

## 3、Redis的特点?

回答：Redis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，另外 Redis 也可以对存入的 Key-Value 设置 expire 时间。

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。

## 4、为什么 Redis 需要把所有数据放到内存中？

回答：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 速度会严重影响 redis 的性能。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。

## 5、Redis 常见的性能问题怎么解决？

回答：1） Master 最好不要做任何持久化工作，如 RDB 内存快照和 AOF 日志文件；

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次；

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内；

4） 尽量避免在压力很大的主库上增加从库；

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。

## 6、Redis 与 memcached 有什么区别？

回答：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型；

2）redis 的速度比 memcached 快很多；

3）redis 可以持久化其数据；

## 7、Redis 有哪些数据结构？

回答：常用的五种数据结构（string、list、set、hash、zset）

1）string：可以是字符串，整数或者浮点数，对整个字符串或者字符串中的一部分执行操作，对整个整数或者浮点执行自增（increment）或者自减（decrement）操作。

2）list：一个链表，链表上的每个节点都包含了一个字符串，出链表的两端推入或者弹出元素，根据偏移量对链表进行修剪（trim），读取单个或者多个元素，根据值查找或者移除元素。

3）set：包含字符串的无序收集器（unordered collection）、并且被包含的每个字符串都是独一无二的。添加，获取，移除单个元素，检查一个元素是否存在于集合中，计算交集，并集，差集，从集合里面

随机获取元素。

4) hash：包含键值对无序散列表，添加，获取，移除单键值对，获取所有键值对。

5) zset：字符串成员(member)与浮点数分值(score)之间的有序映射，元素的排列顺序由分值的大小决定。添加，获取，删除单个元素，根据分值范围(range)或者成员来获取元素。

## 8、Redis 持久化方案区别以及优缺点？

回答：redis 提供了两种持久化的方式，分别是 RDB（Redis DataBase）和 AOF（Append Only File）。

RDB 方式：是一种快照式的持久化方法，将某一时刻的数据持久化到磁盘中。

1) redis 在进行数据持久化的过程中，会先将数据写入到一个临时文件中，待持久化过程都结束了，才会用这个临时文件替换上次持久化好的文件。正是这种特性，让我们可以随时来进行备份，因为快照文件总是完整可用的。

2) 对于 RDB 方式，redis 会单独创建（fork）一个子进程来进行持久化，而主进程是不会进行任何 IO 操作的，这样就确保了 redis 极高的性能。

3) 如果需要进行大规模数据的恢复，且对于数据恢复的完整性不是非常敏感，那 RDB 方式要比 AOF 方式更加的高效。

AOF 方式：是将执行过的写指令记录下来，在数据恢复时按照从前到

后的顺序再将指令执行一遍。

1）AOF命令以redis协议追加保存每次写的操作到文件末尾.Redis还能对AOF文件进行后台重写，使得AOF文件的体积不至于过大.默认的AOF持久化策略是每秒钟fsync一次（fsync是指把缓存中的写指令记录到磁盘中），因为在这种情况下，redis仍然可以保持很好的处理性能，即使redis故障，也只会丢失最近1秒钟的数据。

2）如果在追加日志时，恰好遇到磁盘空间满、inode满或断电等情况导致日志写入不完整，也没有关系，redis提供了redis-check-aof工具，可以用来进行日志修复。

3）因为采用了追加方式，如果不做任何处理的话，AOF文件会变得越来越大，为此，redis提供了AOF文件重写（rewrite）机制，即当AOF文件的大小超过所设定的阈值时，redis就会启动AOF文件的内容压缩，只保留可以恢复数据的最小指令集。举个例子或许更形象，假如我们调用了100次INCR指令，在AOF文件中就要存储100条指令，但这明显是很低效的，完全可以把这100条指令合并成一条SET指令，这就是重写机制的原理。

4）在进行AOF重写时，仍然是采用先写临时文件，全部完成后再替换的流程，所以断电、磁盘满等问题都不会影响AOF文件的可用性。

## 9、如何来维护集群之间的关系，或者说集群之间如何建立连接？

回答：1）所有的redis节点彼此互联（PING-PONG机制），内部使用二进制协议优化传输速度和带宽.

2）节点的 fail 是通过集群中超过半数的节点检测失效时才生效.

3）客户端与 redis 节点直连，不需要中间 proxy 层. 客户端不需要连接集群所有节点，连接集群中任何一个可用节点即可

4）redis-cluster 把所有的物理节点映射到 [0-16383] slot 上，cluster 负责维护 node<->slot<->value

## 10、Redis 如何序取实体？

回答：存储的时候需要将实体序列化，然后就可以当字符串一样存储，取数据时也一样，取出来的数据需要反序列化。

## 11、Redis 保留时间多久？

回答：如果未设置则一直存在，除非服务停掉且没有保存到磁盘。如果已手动或自动保存过，则再次启动服务还会存在。

EXPIRE <KEY> <TTL> ：将键的生存时间设为 ttl 秒

PEXPIRE <KEY> <TTL> ：将键的生存时间设为 ttl 毫秒

EXPIREAT <KEY> <timestamp> ：将键的过期时间设为 timestamp 所指定的秒数时间戳

PEXPIREAT <KEY> <timestamp>：将键的过期时间设为 timestamp 所指定的毫秒数时间戳.

## 12、Redis 挂掉后怎么办？介绍 Redis 是怎么实现高可用的？

回答：主要取决于，你是把 redis 作为缓存还是 nosql，如果是缓存

那丢了也无所谓，从别的地方恢复重建就行了，如果是nosql的话，redis是有snapshot和aof的机制来保证数据持久化的。

## 13、Redis有事务吗，简单的说一下？

回答：1） 开启：以MULTI开始一个事务

2） 入队：将多个命令入队到事务中，找到这些命令并不会立即执行，而是放到等待执行事务队列里面

3） 执行：由EXEC命令触发事务

三个特性：单独的隔离操作：事务的所有命令都会序列化、按顺序地实行

没有隔离级别的概念：事务提交前任何指令都不会被实际执行

不保证原子性：redis同一个事务中如果有一条命令执行失败，其后的命令仍然会被执行，不回滚