## 1 什么是Redis？

Redis是一个key-value存储系统。和Memcached类似，它支持存储的value类型相对更多，包括string(字符串)、list(链表)、set(集合)、zset(sorted set --有序集合)和hash（哈希类型）。这些数据类型都支持push/pop、add/remove及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，redis支持各种不同方式的排序。与memcached一样，为了保证效率，数据都是缓存在内存中。区别的是redis会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了master-slave(主从)同步。

Redis 是一个高性能的key-value数据库。 redis的出现，很大程度补偿了memcached这类key/value存储的不足，在部分场合可以对关系数据库起到很好的补充作用。它提供了Java，C/C++，C#，PHP，JavaScript，Perl，Object-C，Python，Ruby，Erlang等客户端，使用很方便。

Redis支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得Redis可执行单层树复制。存盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有帮助。

可以充当缓存、队列等作用。

## 2 Redis两种持久化的方式的比较。

答：Redis的持久化方式有两种，AOF和RDB。

RDB持久化方式能够在指定的时间间隔能对你的数据进行快照存储.

AOF持久化方式记录每次对服务器写的操作,当服务器重启的时候会重新执行这些命令来恢复原始的数据,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。

两种区别就是，一个是持续的用日志记录写操作，crash（崩溃）后利用日志恢复；一个是平时写操作的时候不触发写，只有手动提交save命令，或者是shutdown关闭命令时，才触发备份操作。

选择的标准，就是看系统是愿意牺牲一些性能，换取更高的缓存一致性（aof），还是愿意写操作频繁的时候，不启用备份来换取更高的性能，待手动运行save的时候，再做备份（rdb）。rdb这个就更有些 最终一致性（eventually consistent）的意思了。

**3、说说Redis哈希槽的概念？**

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。

**4、Redis集群最大节点个数是多少？**

16384个

**5、Redis中的管道有什么用？**

一次请求/响应服务器能实现处理新的请求即使旧的请求还未被响应。这样就可以将多个命令发送到服务器，而不用等待回复，最后在一个步骤中读取该答复。

这就是管道（pipelining），是一种几十年来广泛使用的技术。例如许多POP3协议已经实现支持这个功能，大大加快了从服务器下载新邮件的过程。

**6、怎么理解Redis事务？**

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。

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。

**7、Redis事务相关的命令有哪几个？**

MULTI、EXEC、DISCARD、WATCH

**8、Redis如何做内存优化？**

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.

**9 什么是RPC？**

答案：RPC（Remote Procedure Call）—[远程过程调用](https://baike.baidu.com/item/%E8%BF%9C%E7%A8%8B%E8%BF%87%E7%A8%8B%E8%B0%83%E7%94%A8)，它是一种通过[网络](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C)从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。[RPC协议](https://baike.baidu.com/item/RPC%E5%8D%8F%E8%AE%AE)假定某些[传输协议](https://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E5%8D%8F%E8%AE%AE)的存在，如TCP或UDP，为通信程序之间携带信息数据。在OSI[网络通信](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E9%80%9A%E4%BF%A1)模型中，RPC跨越了[传输层](https://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E5%B1%82)和[应用层](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E5%B1%82)。RPC使得开发包括网络[分布式](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F)多程序在内的应用程序更加容易。

**10 HttpClient与JSONP区别？**

1发送请求的位置不同.

JSONP:浏览器解析js发出的请求

HttpClient:是业务层java模拟发出Http请求

2返回值的格式不同

JSONP:返回数据时必须要有回调函数.

HttpClient:直接返回JSON数据即可.

3代码的层级不同

JSONP:代码层级一般3层

HttpClient:一般5层.

**11 什么是nginx？**

答：Nginx (engine x) 是一个高性能的[HTTP](https://baike.baidu.com/item/HTTP)和[反向代理](https://baike.baidu.com/item/%E5%8F%8D%E5%90%91%E4%BB%A3%E7%90%86/7793488)服务。

反向代理方式是指以代理服务器来接收Interrnet上的连接请求，然后将请求转发给内部网络上的服务器；并将从服务器上得到的结果返回给Internet上请求连接的客户端，此时代理服务器对外就表现为一个服务器。

当一个代理服务器能够代理外部网络上的主机，访问内部网络时，这种代理服务的方式称为反向代理服务，当做服务器的替身，这样，代理服务器就在安全数据库和可能的恶意攻击之间提供了又一道屏障。

防火墙通路只允许代理服务器有权进行访问。

可以使用多个代理服务器来处理对一个高用量内容服务器的请求，这样做的好处是内容服务器可以处理更高的负载，并且比其独自工作时更有效率。

配置文件主要由四部分组成：main(全区设置)，server(主机配置)，upstream(负载均衡服务器设置)，和location(URL匹配特定位置设置)。

有1全局变量2事件配置3 http参数4虚拟主机基本设置5 Nignx状态监控6反向代理7负载均衡8 URL重写9 IP限制

主要了解虚拟主机配置，反向代理，负载均衡，url重新