# 1.Redis支持的数据类型

## 1.1.String字符串：

格式：set key value

string类型是二进制安全的，意思是redis的string可以包含任何数据。比如jpg图片或者序列化的对象。

string类型是redis最基本的数据类型，一个键最大能存储512MB。

## 1.2.Hash(哈希)

格式：hmset name key1 value1 key2 value2

redis hash是一个键值对的集合。

Redis hash是一个string类型的field和value的映射表，hash特别适合用于存储对象。

## 1.3.List(列表)

redis列表是简单的字符串列表，按照插入顺序排序。你可以添加一个元素到列表的头部(左边)或者右边。

格式：lpush name value 在key对应的list的头部添加字符串元素。

格式：rpush name value在key对应的list的尾部添加字符串元素。

格式：lrem name index key对应list中删除count个和value相同的元素。

格式：llen name 返回key对应list的长度。

## 1.4.Set(集合)

格式：sadd name value

redis的set是string类型的无序集合。

集合是通过哈希表实现的，所以添加、删除、查找的复杂度都是O(1)。

## 1.5.zset(sorted set有序集合)

格式：zadd name score value

redis zset和set一样也是string类型元素的集合，且不允许重复的成员。

不同的是每个元素都会关联一个double类型的分数。redis正是通过分数来为集合中的成员进行从小到大的排序。

zset的成员是唯一的，但是却可以重复。

# 2.Redis持久化

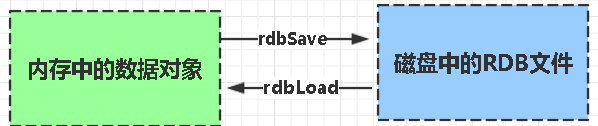
持久化就是把内存中的数据写到磁盘中去，防止服务宕机了内存数据丢失。

Redis提供了两种持久化方式：RDB(默认)和AOF

## 2.1.RDB

rdb就是Redis DataBase的缩写。

功能核心函数rdbSave(生成rdb文件)和rdbLoad(从文件加载内存)两个函数。



## 2.2.AOF

AOF是Append-only file缩写。

每当执行服务器(定时)任务或者函数时flushAppendOnlyFile函数都会被调用，这个函数执行以下两个工作：

WRITE：根据条件，将aof\_buf中的缓存写入到AOF文件。

SAVE：根据条件，调用fsync或fdatasync函数，将AOF文件保存到磁盘中。

## 2.3.存储结构

内容是redis通讯协议(RESP)格式的命令文本存储。

## 2.4.比较

1. AOF文件比RDB文件更新频率高，优先使用AOF还原数据。
2. 比RDB更安全也更大。
3. RDB性能比AOF好。
4. 如果两个都配了优先加载AOF。

## 2.5.Redis通讯协议

RESP 是redis客户端和服务端之前使用的一种通讯协议；

# 3.Redis架构模式

## 3.1.单机版

特点：简单。

问题：1.内存容量有限。2.处理能力有限。3.无法高可用。

## 3.2.主从复制

Redis的复制(replication)功能允许用户根据一个Redis服务器来创建任意多个该服务器的复制品，其中被复制的服务器为主服务器(master)，而同过复制创建出来的服务器复制品则为从服务器(slave)。只要出从服务器直接的网络连接正常，主从服务器两者会具有相同的数据，主服务器就会一直将发生在自己身上的数据更新同步给从服务器，从而一直保证主从服务器的数据相同。

特点：master/slave角色、master/slave数据相同、降低master读压力

问题：无法保证高可用、没有解决master写的压力

## 3.3哨兵模式

Redis sentinel是一个分布式系统中监控redis主从服务器，并在主服务器下线时自动进行故障转移。其中三个特性：

监控(monitoring)：sentinel会不断检查你的主服务器和从服务器是否运作正常。

提醒(notification)：当被监控的某个Redis服务器出现问题时，sentinel可以通过API向管理员或者其他应用程序发送通知。

自动故障转移(automatic failover)：当一个主服务器不能正常工作时，sentinel会开始一次自动故障迁移工作。

特点：保证高可用、监控各个节点、自动故障迁移

缺点：主从模式，切换需要时间丢数据、没有解决master写的压力。

## 3.4.集群(proxy型)

Twemproxy是一个Twitter开源的一个Redis和memcache快速/轻量级代理服务器；Twemproxy是一个快速的单线程代理程序，支持Memcache ASCII协议和redis协议。

特点：多种hash算法、支持失败节点自动删除、后端Sharding分片逻辑对业务透明，业务方的读写方式和操作单个Redis一致。

缺点：增加了新的proxy，需要维护其高可用、failover逻辑需要自己实现，其本身不能支持故障的自动转移可扩展性差，进行扩容缩容都需要手动干预。

## 3.5集群(直连型)

从Redis3.0以后版本都支持redis-cluster集群，redis-cluster采用无中心结构，每个节点保存数据和整个集群状态，每个节点都和其他节点连接。

特点：

1. 无中心架构(不存在哪个节点影响性能瓶颈)，少了proxy层。
2. 数据按照slot存储分布在多个节点，节点间数据共享，可动态调整数据分布。
3. 可扩展性，可线性扩展到1000个节点，节点可动态添加或删除。
4. 高可用性，部分节点不可用时，集群仍可用。可以通过增加slave做备份数据副本。
5. 实现故障自动failover，节点之间通过gossip协议交换状态信息，用投票机制完成slave到master的角色提升。

缺点：

1. 资源隔离性较差，容易出现相互影响的情况。
2. 数据通过异步复制，不保证数据的强一致性。

# 4.Redis有哪几种淘汰策略

1.noevaition：返回错误当内存达到限制，并且客户端尝试执行会让更多内存被使用的命令。

2.allkeys-lru：尝试回收最少使用的键(LRU)，使得新添加的数据有空间存放。

3.volatile-lru：尝试回收最少使用的键(LRU)，但仅限于在过期集合的键，使得新添加的数据有空间存放。

4.allkeys-random：回收随机的键使得新添加的数据有空间存放。

5.volatile-random：回收随机的键使得新添加的数据有空间存放，但仅限于在过期集合的键。

6.volatile-ttl：回收在过期集合的键，并且优先回收存活时间(TTL)较短的键，使得新添加的数据有空间存放。

# 5.字符串类型的值能存储的最大容量

512M

# 为什么Redis需要把所有数据放到内存中

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，如果不将数据放到内存中，磁盘IO速度会严重影响redis的性能。在内存越来越便宜的今天，redis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极限后不能继续插入新值。

# 7.Redis集群方案什么情况下会导致整个集群不可用

在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这个范围内的槽不一样。

# 8.如何保证redis中的数据都是热点数据

Redis内存数据集大小上升到一定大小的时候，就会实行数据淘汰策略。其实面试官除了考察redis，不少公司都很重视高并发高可用的技术，特别是一线互联网公司，分布式、JVM、Spring源码分析、微服务等知识点已是面试的必考题了。

# 9.Redis有哪些适合的场景？

## 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Memcaches)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使用Redis来缓存会话的文档。甚至广为人知的Magento也提供了Redis的插件。

## 2.全页缓存(FPC)

除了基本的会话token之外，Redis还提供了很简便的RPC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的本地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。

## 3.队列

Redis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。

## 4.排行榜/计数器

Redis在内存中对数字进行递增和递减的操作实现的非常好。集合(Set)和有序集合(SortedSet)也使得我们在执行这些操作的时候变得非常简单，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’。

## 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不多。

# 10.Redis支持的Java客户端有哪些

Redisson、Jedis、lettuce等，官方推荐使用Redisson。

# 11.Redis哈希槽

Redis集群中没有使用一致性hash，而是引入了哈希槽的概念，Redis集群中有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，集群中的每个节点负责一部分hash槽。

# 12.Redis集群的主从复制模型是怎样的

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。

# 13.Redis集群会有写操作丢失吗

Redis并不能保证数据的强一致性，这意味着这在实际中集群在特定的条件下可能会丢失写操作。

# 14.Redis集群之间是如何复制的

异步复制。

# 15.Redis集群最大节点个数是多少

16384个

# 16.Redis集群如何选择数据库

Redis集群目前无法做数据库选择，默认在0数据库。

# 17.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POP3协议已经实现支持了这个功能，大大加快了从服务器下载新邮件的过程。

# 18.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顺序化、按顺序的执行，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。

# 19.Redis事务相关的命令有哪几个

MULTI、EXEC、DISCARD、WATCH

# 20.Redis如何做内存优化

尽可能的使用散列表(hash)，散列表(是说散列表里面存储的数据较少)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。

# 21.Redis回收进程是如何工作的

一个客户端运行了新的命令，添加了新的数据。Redis检查内存的使用情况，如果大于maxmemory的限制，则根据设定好的策略进行回收

# 22.Redis加锁机制

# 23.Redis锁互斥机制

# 24.watch dog自动延期机制

# 25.可重入锁机制

# 26.释放锁机制

# 27.Redis实现分布式锁

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。如果在setnx之后执行expire之前进程以外crash或者重启维护了，那会怎样？setnx指令有非常复杂的参数，可以把把setnx和expire合成一条指令来执行。

# 28.Redis做异步队列

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。

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。

不能生产一次消费多次。

使用pub/sub主题订阅者模式，可以实现1:N的消息队列。

# 29.缓存穿透和缓存雪崩

缓存穿透：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。

如何避免缓存穿透：1.对于查询结果为空的情况也进行缓存，缓存时间设置的短一点，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。尽可能的把所有的可能存在的key放到一个大的bitmap中，查询时通过该bitmap过滤。

缓存雪崩：当缓存服务器重启或者大量缓存集中在某一时刻失效，这样在失效的时候，会给后端系统带来很大的压力。导致系统崩溃。

如何避免缓存雪崩：1.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，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。