**什么是Redis**

Redis是一个基于内存的高性能key-value数据库，可以存储键和五种不同类型的值之间的映射。键的类型只能为字符串，值支持五种数据类型：字符串、列表、集合、散列表、有序集合。端口6379

**Redis有哪些优缺点**

优点：1. 读写性能优异

2. 支持数据持久化，支持RDB和AOF两种持久化方式。

3.支持事务，Redis的所有操作都是原子性的，同时Redis还支持对几个操作合并后的原子性执行。

4.数据结构丰富，除了支持string类型的value外还支持hash、set、zset、list等数据结构。

缺点：数据库容量受到物理内存的限制

**为什么要用 Redis /为什么要用缓存**

高性能：操作内存速度快

高并发：能够承受的请求量大

**Redis为什么这么快**

内存存储：Redis使用内存存储

单线程实现：Redis使用单线程处理请求，避免了多个线程切换的开销

优化的数据结构：Redis使用hash结构，读取速度快

**Redis为什么设计成单线程**

避免锁和上下文切换的开销；单线程保证原子性，所有命令在单线程中串行执行；

局限性：无法充分利用多核CPU

**Redis有哪些数据类型**

Redis主要有5种数据类型，包括string，hash，list，set，zset，满足大部分的使用要求

string，（分布式锁，自增实现全局唯一id，计数器）

hash 是一个键值对集合，相当于Java中的HashMap（适用于存储用户信息、商品信息等复杂对象）

list 存储一列连续的数据，有序，有索引，相当于Java中的LinkedList，（可以使用列表实现简单的消息队列）

set 存储一列不重复的数据，无序，无索引，相当于Java中的HashSet

zset 存储一列不重复的数据，可排序，无索引，（排行榜，可以使用有序集合存储用户的得分，然后根据得分排序）

**Redis的应用场景**

实现分布式锁

通过list实现消息队列

String 自增运算，实现计数器

时间戳+redis里的序列号生成唯一id

缓存数据库的热点数据（经常会被查询，但是不经常被修改或者删除的数据）

缓存用户的登录信息

缓存jwt

会话缓存：统一存储多台应用服务器的会话信息

**什么是Redis持久化？**

持久化就是把内存的数据写到磁盘中去，防止服务宕机了内存数据丢失。

**Redis 的持久化机制是什么？各自的优缺点？**

RDB机制和 AOF 机制：

RDB是Redis默认的持久化方式。按照一定的时间将内存的数据以快照的形式保存到硬盘中。

优点：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。数据集大时，比aof启动效率高。

缺点：RDB 是每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。

AOF将写命令记录到日志文件中，当Redis重启时再次执行AOF文件中的命令来恢复数据。当两种方式同时开启时，数据恢复Redis会优先选择AOF恢复。

优点：数据安全，aof 可以配置 appendfsync 为always，aof 可以配置每进行一次命令操作就记录到 aof 文件

缺点：AOF 文件比 RDB 文件大，且恢复速度慢。数据集大时，比 rdb 启动效率低

**如何选择合适的持久化方式**

如不能承受数分钟以内的数据丢失，对业务数据非常敏感，选用AOF

如能承受数分钟以内的数据丢失，且追求大数据集的恢复速度，选用RDB

**Redis的过期键的删除策略**

Redis的过期策略就是指当Redis中缓存的key过期了，Redis如何处理。

过期策略通常有以下三种：

定时过期：通过定时器清除，创建一个定时器，到过期时间就会立即清除。该策略对内存很友好；但是会占用大量的CPU资源去处理过期的数据。对CPU不友好，对内存很友好

惰性过期：只有当访问一个key时，才会判断该key是否过期，过期则清除。该策略可以最大化地节省CPU资源。对CPU很友好，对内存不友好

定期过期：每隔一段时间，对expires字典进行检查，删除里面的过期键。

(expires字典保存所有设置了过期时间的key的过期时间数据，其中，key是指向键空间中的某个键的指针，value是该键的毫秒精度的UNIX时间戳表示的过期时间。键空间是指该Redis集群中保存的所有键。)

Redis中同时使用了惰性过期和定期过期两种过期策略。

**key的过期时间和永久有效分别怎么设置？**

EXPIRE和PERSIST命令。

**我们知道通过expire来设置key 的过期时间，那么对过期的数据怎么处理呢?**

1.定时去清理过期的缓存；

2.当有用户请求过来时，再判断这个请求所用到的缓存是否过期，过期的话就去底层系统得到新数据并更新缓存。

**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**

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。

**Redis的内存淘汰策略有哪些**

noeviction：当内存不足以容纳新写入数据时，新写入操作会报错。

allkeys-lru：当内存不足以容纳新写入数据时，在键空间中，移除最近最少使用的key。（这个是最常用的）

allkeys-random：当内存不足以容纳新写入数据时，在键空间中，随机移除某个key。

volatile-lru：当内存不足以容纳新写入数据时，在设置了过期时间的键空间中， 移除最近最少使用的key。

volatile-random：当内存不足以容纳新写入数据时，在设置了过期时间的键空间中，随机移除某个key。

volatile-ttl：当内存不足以容纳新写入数据时，在设置了过期时间的键空间中， 有更早过期时间的key优先移除。

**Redis主要消耗什么物理资源？**

内存。

**Redis的内存用完了会发生什么？**

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。）

**什么是事务？**

事务是一系列操作组成的工作单元，该工作单元内的操作是不可分割的，要么所有操作都做，要么所有操作都不做。

**Redis事务的概念**

redis事务就是一次性、顺序性、排他性的执行一个队列中的一系列命令。

**Redis事务的三个阶段**

事务开始 MULTI、命令入队、事务执行 EXEC

事务执行过程中，如果服务端收到有EXEC、DISCARD、WATCH、MULTI之外的请求，将会把请求放入队列中排队

**Redis事务相关命令**

MULTI开启一个事务

EXEC执行事务

DISCARD取消事务

**事物的四大特性(ACID)介绍一下?**

原子性：要么所有操作都做，要么所有操作都不做

一致性：执行事务前后，数据的完整性必须保持一致。(拿转账来说，假设用户A和用户B两者的钱加起来一共是5000，那么不管A和B之间如何转账，事务结束后两个用户的钱相加起来应该还得是5000，这就是事务的一致性)

隔离性：多个事务并发执行时，一个事务的执行不应影响其他事务的执行

持久性：事务对数据的改变是持久的

Redis的事务总是具有ACID中的一致性和隔离性，其他特性是不支持的。当服务器运行在AOF持久化模式下，并且appendfsync选项的值为always时，事务也具有耐久性。

**Redis事务支持隔离性吗**

Redis是单进程程序，并且它保证在执行事务时，不会对事务进行中断，事务可以运行直到执行完所有事务队列中的命令为止。因此，Redis 的事务是总是带有隔离性的。

redis是单线程的，并且不会对事务进行中断

**Redis事务保证原子性吗，支持回滚吗**

Redis中，单条命令是原子性执行的，但事务不保证原子性，且没有回滚。事务中任意命令执行失败，其余的命令仍会被执行。

**Redis事务其他实现**

基于Lua脚本，Redis可以保证脚本内的命令一次性、按顺序地执行，

其同时也不提供事务运行错误的回滚，执行过程中如果部分命令运行错误，剩下的命令还是会继续运行完

**redis哨兵（2）**

哨兵模式

集群监控：负责监控 redis master 和 slave 进程是否正常工作。

消息通知：如果某个redis实例有故障，那么哨兵负责发送消息作为报警通知给管理员。

故障转移：如果 master node 挂掉了，会自动转移到 slave node 上。

配置中心：如果故障转移发生了，通知 client 客户端新的 master 地址。

哨兵的核心知识

哨兵至少需要 3 个实例，来保证自己的健壮性。

哨兵 + redis 主从的部署架构，是不保证数据零丢失的，只能保证 redis 集群的高可用性。

**redis主从复制的核心原理(1)**

当启动一个 slave节点的时候，它会发送一个PSYNC命令给master节点。如果是初次连接就会触发一次全量复制。

master会启动后台线程，生成一份RDB快照文件，同时将新收到的写命令缓存在内存中。

master将生成好的RDB文件发给 slave，slave 会先写入本地磁盘，再加载到内存中，

master 会将内存中缓存的写命令发送到 slave，

slave 也会同步这些数据。

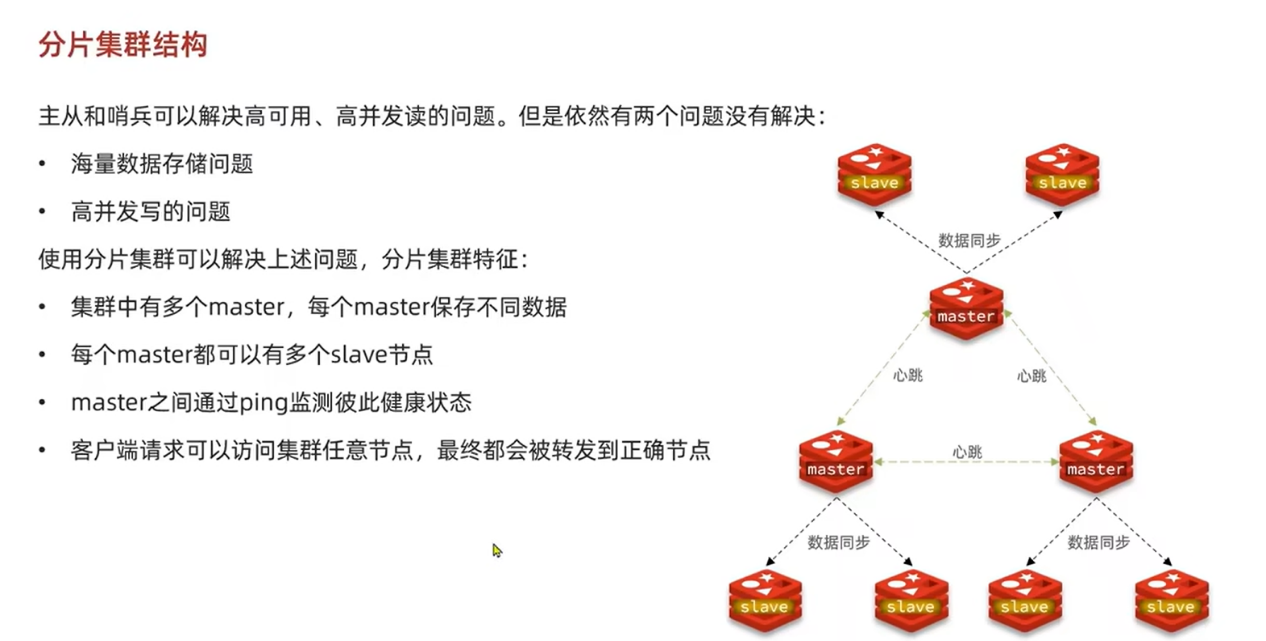
如果断开了连接，重连之后 master node 仅会复制给 slave 部分缺少的数据。

**Redis的集群方式有哪些？**

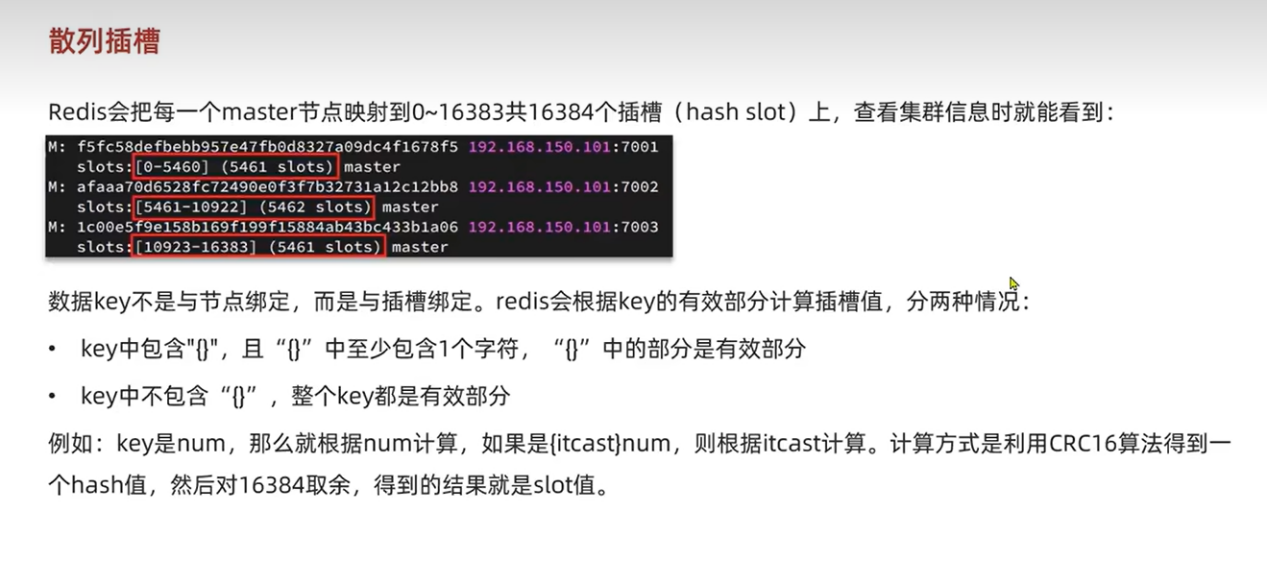
主从集群，主库用来写数据，从库用来读数据。结合哨兵，可以在主库宕机时从新选主，目的是保证Redis的高可用。

分片集群，集群中有多个master。每个master保存不同数据。目的是让Redis能存储更多数据

**Redis分片集群（3）**

****

**Redis分片集群-插槽**

****

1. 将所有的存储空间切割成16384份，每个master节点保存一部分
2. 每个key通过CRC16校验后对16384取模来决定放置哪个槽

3. 数据先写入主节点，再同步到从节点

4. 读取数据时，当key没有分配在该节点上时，redis会返回转向指令，指向正确的节点

5. 扩容时时需要需要把旧节点的数据迁移一部分到新节点

**Redis是单线程的，如何提高多核CPU的利用率？**

可以在同一个服务器部署多个Redis的实例，并把他们当作不同的服务器来使用，在某些时候，无论如何一个服务器是不够的， 所以，如果你想使用多个

CPU，你可以考虑一下分片（shard）。

**为什么要做Redis分区？**

分区可以让Redis管理更大的内存，Redis将可以使用所有机器的内存

**为什么要用 Redis 而不用 map/guava 做缓存?**

缓存分为本地缓存和分布式缓存。Java自带的map或者guava实现的是本地缓存，在多实例的情况下，每个实例都需要各自保存一份缓存，缓存不具有一致性。使用 redis的是分布式缓存，在多实例的情况下，各实例共用一份缓存数据，缓存具有一致性。

**缓存雪崩**

大量请求无法在redis得到处理，从而打到数据库。大量的缓存实例在同一时间失效或同时过期，造成雪崩状态。

解决：1、过期时间增加随机数。

2、发生雪崩时进行服务降级。非核心数据直接返回默认值或错误。

3、限流熔断，当数据库负载突升时，暂停业务应用对缓存的访问。

**缓存击穿**

一条热点数据过期失效，大量请求突然打到数据库。

解决方法：1.热点数据不设置过期时间。2.加锁，大量并发只让一个去查，其他人等待，查完后缓存中就会有数据

**缓存穿透**

数据不在缓存中，也不在数据库中。

解决：1.null结果缓存，并加入短暂过期时间

2.使用布隆过滤器快速判断数据是否存在。

3.前端进行请求检查。

**Redis支持的Java客户端都有哪些？官方推荐用哪个？**

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

**分布式锁**

setnxex加锁，需要保证占位+过期时间的原子性，判断+删除的原子性。

SETNX

如果 key 不存在，则显示插入成功，可以用来表示加锁成功；

如果 key 存在，则会显示插入失败，可以用来表示加锁失败。

 EX，设置其过期时间

解锁要保证，判断+删除的原子性。

保证接口幂等性

幂等性：通俗的说就是一个接口, 多次发起同一个请求, 必须保证操作只能执行一次

**分布式缓存一致性**

失效模式：写完数据库，然后删缓存，等有请求进来读数据的时候，缓存中没有，就会查数据库，然后主动放到缓存里面。也叫触发主动更新。

双写模式：就是写完数据库之后再去写缓存

**为什么要使用redis而不是把数据放在本地内存中**

本地缓存有内存一致性问题，不同节点的本地内存数据，版本会不一致。

本地缓存的机器内存有限，redis集群模式，是分片的

**Redis的分布式锁实现方法**

Redis分布式锁主要是通过setnx命令实现的。key和value分别代表锁的名称和锁的值(线程id)。setnx 将 key 的值设为 value ，当且仅当 key 不存在。如果 SETNX 返回1，说明该进程获得锁。

如果 SETNX 返回0，说明其他进程已经获得了锁，进程不能进入临界区

**非redis实现分布式锁，基于数据库**

使用数据库实现分布式锁，主要是通过表的唯一索引不能重复的特性，保证某个key在表中只能存在一个，某个key插入表成功就获取到锁，删除这条数据就释放锁。

幂等就是一个操作，不论执行多少次，产生的效果和返回的结果都是一样的。

退还保证金的接口、案件id，校验是否已退还 -> 执行退还 ->

同一个案件id可能导致退还多次

同一时间该流程只能有一个能执行成功