- Redis有什么优点？
- Redis有什么缺点
- Redis和Memcached的区别
- 请说说 Redis 的线程模型？
- 为什么 Redis 单线程模型也能效率这么高？
- Redis 是单线程的，如何提高多核 CPU 的利用率？
- 知道redis的持久化吗都有什么缺点优点啊? ?具体底层实现呢?
- Redis 有几种数据“过期”策略？
- Redis 有哪几种数据“淘汰”策略？
- Redis 回收进程如何工作的？
- 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？
- 聊聊 Redis 使用场景
- 如何使用 Redis 实现分布式锁？
- 如何使用 Redis 实现消息队列？
- 什么是 Redis Pipelining ？
- 如何实现 Redis CAS 操作？
- Redis 集群都有哪些方案？
- 什么是 Redis 主从同步？
- 如何使用 Redis Sentinel 实现高可用？
- 如果使用 Redis Cluster 实现高可用？
- Redis Cluster 方案什么情况下会导致整个集群不可用？
- Redis Cluster 会有写操作丢失吗？为什么？
- Redis 集群如何选择数据库？
- 请说说生产环境中的 Redis 是怎么部署的？
- 什么是 Redis 分区？
- 你知道有哪些 Redis 分区实现方案？
- Redis 有哪些重要的健康指标？
- 如何提高 Redis 命中率？
- 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？
- 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？
- Redis 常见的性能问题都有哪些？如何解决？

- Redis的key是如何寻址的?
- 如何保证数据库与redis缓存一致的

Redis有什么优点？

> 1.速度快

因为数据存储在内存中，类似HashMap，优势就是查找和操作的时间复杂度是0(1)。

Redis实际上就是一个Key-Value类型的内存数据库，它和Menchahed很像。整个数据库全部加载到内存中进行操作，定期通过异步的方式flush到硬盘上。

> 2.支持丰富的数据类型

支持String、List、Hash、Set、Sorted set

Redis的出色支出不仅仅是性能，最大的魅力是它所支持的多种数据结构，它的单个Value最大存储为1GB，不像Mencached只能存储1MB的数据，我们可以使用Redis来实现非常多的功能，例如：

- 使用它的list来做FIFO的双向链表，实现一个轻量级、高性能的队列服务
- 用它的set做高性能的tag系统

> 3.丰富的特性

- 订阅发布
- key过期策略
- 事务
- 支持多个DB
- 计数
- ...

> 4.数据持久化

支持两种持久化存储方案（RDB和AOF），解决内存数据库中最担心Redis挂掉数据丢失问题

Redis有什么缺点

- 1.由于Redis是内存数据库，所以但台机器存储的数据量和机器本身的内存大小有关
- 2.修改配置文件需要重启，将硬盘中的数据加载到内存中，时间比较久，在这个过程中，不能对外提供服务

Redis和Memcached的区别

> 1.Redis支持复杂的数据结构

- Memcached只支持简单的字符串
- Redis支持多种数据结构，

> 2.Redis原生支持集群模式

- Redis在3.0之后，支持集群模式
- Memcached没有原生的集群模式，需要依靠客户端来往集群分片中写数据

> 3.性能对比

- Redis使用单核而Memcached可以使用多核，所以平均每一个核上Redis在存储小数据时性能更高
- 在存储100K以上的数据时Memcached的性能要高。

> 4.内存使用率

- 1.简单的 Key-Value存储的话，Memcached的内存利用率更高，可以使用类似内存池。
- 如果 Redis 采用hash结构来做key-value存储，由于其组合式的压缩， 其内存利用率会高于 Memcached 。
- Redis 采用的是包装的malloc/free，相较于Memcached的内存管理方法 tcmalloc / jmalloc 来说，要简单很多 。

> 5.网络IO模型

- Memcached是多线程的，非阻塞IO复用的网络模型，原型上接近Nginx
- Redis 使用单线程的 IO 复用模型，自己封装了一个简单的 AeEvent 事件处理框架，主要实现了 epoll, kqueue 和 select ，更接近 Apache 早期的模式。

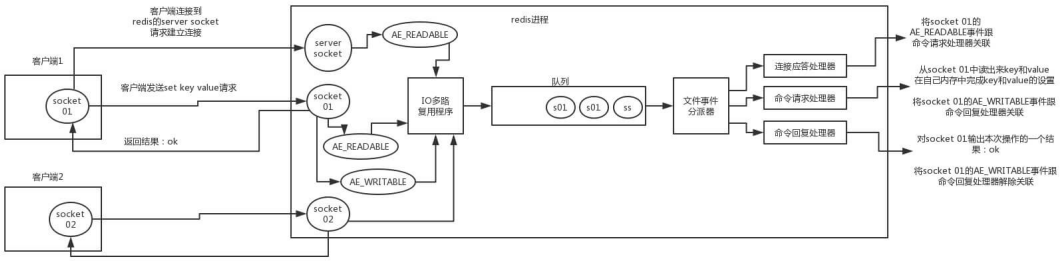
> 6.持久化存储

- Memcached 不支持持久化存储，重启时，数据被清空。
- Redis 支持持久化存储，重启时，可以恢复已持久化的数据。

也推荐阅读下 《脚踏两只船的困惑 - Memcached 与 Redis》 。

请说说 Redis 的线程模型？

Redis内部有一个单线程的文件处理器，所以Redis是单线程的模型，多路IO复用程序监听多个Socket，会将Socket产生的事件放入队列中，事件分配器每次从对列中取出一个事件，把该事件交给对应的处理器质性处理



- 客户端 socket01 向 redis 的 server socket 请求建立连接，此时 server socket 会产生一个AE_READABLE 事件，IO多路复用程序监听到 server socket 产生的事件后，将该事件压入队列中。文件事件分派器从队列中获取该事件，交给连接应答处理器。连接应答处理器会创建一个能与客户端通信的 socket01，并将该 socket01 的 AE_READABLE 事件与命令请求处理器关联。
- 假设此时客户端发送了一个 set key value 请求，此时 redis 中的 socket01 会产生 AE_READABLE事件，IO多路复用程序将事件压入队列，此时事件分派器从队列中获取到该事件，由于前面 socket01 的 AE_READABLE 事件已经与命令请求处理器关联，因此事件分派器将事件交给命令请求处理器来处理。命令请求处理器读取 socket01 的 key value 并在自己内存中完成 key value 的设置。操作完成后，它会将 socket01 的 AE_WRITABLE 事件与令回复处理器关联。
- 如果此时客户端准备好接收返回结果了，那么 redis 中的 socket01 会产生一个 AE_WRITABLE事件，同样压入队列中，事件分派器找到相关联的命令回复处理器，由命令回复处理器对socket01输入本次操作的一个结果，比如 ok，之后解除 socket01 的 AE_WRITABLE 事件与命令回复处理器的关联。

为什么 Redis 单线程模型也能效率这么高？

- 1.纯内存操作
- 2.非阻塞IO，多路复用机制
- 3.单线程操作，避免多线程频繁切换问题
    - 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
- 4.Redis全程使用Hash结构，读取速度快，还有一些特殊的数据结构，对数据结构进行了优化，对短数据进行压缩存储。

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 ，你可以考虑一下分区。

知道redis的持久化吗都有什么缺点优点啊？?具体底层实现呢?

> RDB：对Redis中数据进行周期性备份

- 会生成多个数据文件，每个文件代表某一时刻Redis数据，这种方式非常适合做冷备份
- 对外提供读写服务的影响非常小，可以让Redis保持高性能，Redis有一个子进程，让子进程执行磁盘IO操作，进行RDB持久化；
- 因为每次子进程执行RDB快照数据文件生成的时候，如果数据文件特别大，可能会导致对客户端提供的服务暂停数毫秒，甚至数秒；
- 如果想要Redis故障时，尽可能的少丢失数据，那么RDB没有AOF好，一般来说RDB每个5分钟或更长时间执行一次，这个时候如果宕机，会丢失最近5分钟内的数据；

> AOF：对每条写入命令作为日志，以Append-only的方式写入文件，在Redis重启的时候，通过回访日志文件实现数据恢复；

- AOF可以更好的防止数据丢失，因为AOF会每隔1秒通过一个线程，执行一次fsynch，所以如果宕机，那么最多丢失1秒内的数据；
- AOF日志是一Append-only的方式进行写入，所以没有任何磁盘寻址的开销，写入性能非常高，而且文件不容易破坏，即使文件损坏也很容易恢复
- AOF日志文件即使过大，出现后台重写操作，也不会影响客户端的读写性能，因为在rewrite log的时候，会对其中的文件进行压缩，创建出一份需要恢复数据的最小日志出来，在创建新日志文件的时候，老的日志文件还是照常写入，当新的merge后的日志文件ready的时候，再交换新老日志文件即可
- 对于同一份日志文件来说，AOF日志文件要比RDB文件大的多
- AOF 开启后，支持的写 QPS 会比 RDB 支持的写 QPS 低，因为 AOF 一般会配置成每秒 fsync 一次日志文件，当然，每秒一次 fsync，性能也还是很高的。（如果实时写入，那么 QPS 会大降，redis 性能会大大降低）

> RDB和AOF到底该如何选择

- 根据RDB和AOF的优缺点，我们知道仅仅使用RDB的方式会导致数据丢失很多，仅仅使用AOF的方式，因为恢复速度AOF没有RDB快的多，RDB每次简单粗暴的备份数据快照，更加健壮
- Redis支持同事开启两种持久化方式，用AOF保证数据不丢失，作为数据恢复的第一选择，用RDB方式来做不同程度的冷备份

另外，RDB 和 AOF 涉及的知识点蛮多的，可以看看：

- 《Redis 设计与实现 —— RDB》
- 《Redis 设计与实现 —— AOF》

Redis 有几种数据“过期”策略？

Redis 提供了 3 种数据过期策略：

- 被动删除：当读/写一个已经过期的key时，会触发惰性删除策略，直接删除掉这个过期 key 。
- 主动删除：由于惰性删除策略无法保证冷数据被及时删掉，所以Redis 会定期主动淘汰一批已过期的 key 。
- 主动删除：当前已用内存超过maxmemory限定时，触发主动清理策略，即 「数据“淘汰”策略」 。

在 Redis 中，同时使用了上述 3 种策略，即它们非互斥的。

想要进一步了解，可以看看 《关于 Redis 数据过期策略》 文章。

Redis 有哪几种数据“淘汰”策略？

- 1、 volatile-lru：从设置过期的数据集中淘汰最少使用的数据；
- 2、volatile-ttl：从设置过期的数据集中淘汰即将过期的数据（离过期时间最近）；
- 3、volatile-random：从设置过期的数据集中随机选取数据淘汰；
- 4、allkeys-lru：从所有 数据集中选取使用最少的数据；
- 5、allkeys-random：从所有数据集中任意选取数据淘汰；
- 6、no-envicition：不进行淘汰；。

具体的每种数据淘汰策略的定义，和如何选择讨论策略，可见 《Redis实战（二）内存淘汰机制》 。

Redis 回收进程如何工作的？

- 1.一个客户端运行了新的命令，添加了新的数据
- 2.Redis 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。
- 3.Redis 执行新命令……

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（跌宕起伏）。

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？

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。

聊聊 Redis 使用场景

- 1.缓存
  对于热点数据，缓存以后可能读取数十万次，因此，对于热点数据，缓存的价值非常大。例如，分类栏目更新频率不高，但是绝大多数的页面都需要访问这个数据，因此读取频率相当高，可以考虑基于 Redis 实现缓存。
- 会话缓存
  此外，还可以考虑使用 Redis 进行会话缓存。例如，将 web session 存放在 Redis 中。
- 时效性
  例如验证码只有60秒有效期，超过时间无法使用，或者基于 Oauth2 的 Token只能在5分钟内使用一次，超过时间也无法使用。
- 访问频率
  出于减轻服务器的压力或防止恶意的洪水攻击的考虑，需要控制访问频率，例如限制 IP 在一段时间的最大访问量。
- 计数器
  数据统计的需求非常普遍，通过原子递增保持计数。例如，应用数、资源数、点赞数、收藏数、分享数等。
- 消息队列
  Redis 能作为一个很好的消息队列来使用，依赖 List 类型利用 LPUSH 命令将数据添加到链表头部，通过BRPOP命令将元素从链表尾部取出。同时，市面上成熟的消息队列产品有很多，例如 RabbitMQ。因此，更加建议使用 RabbitMQ 作为消息中间件。
- 最新动态
  按照时间顺序排列的最新动态，也是一个很好的应用，可以使用 Sorted Set 类型的分数权重存储 Unix 时间戳进行排序。
- 在线飞机列表
  按照飞机编号、飞机架次使用sorted set进行处理

详细的介绍，可以看看如下文章：

- 《聊聊 Redis 使用场景》
- 《Redis 应用场景及实例》
- 《Redis 常见的应用场景解析》
- 《Redis 和 Memcached各有什么优缺点，主要的应用场景是什么样的？》

如何使用 Redis 实现分布式锁？

- 1.方案一：set 指令

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。

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

几个重要事件：

- 1.默认等待锁的时间
- 2.默认重试间隔时间
- 3.锁超时时间

- 2.方案二：redlock

Redisson实现Redis分布式锁的N种姿势
Redlock：Redis分布式锁最牛逼的实现

- 3.对比 Zookeeper 分布式锁

> 从可靠性上来说，Zookeeper 分布式锁好于 Redis 分布式锁。
> 从性能上来说，Redis 分布式锁好于 Zookeeper 分布式锁。

如何使用 Redis 实现消息队列？

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。

- 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。

什么是 Redis Pipelining ？

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。

使用Redis的管道来实现一次性插入大量的数据

Redis 很早就支持管道（pipelining）技术，因此无论你运行的是什么版本，你都可以使用管道（pipelining）操作 Redis。

如何实现 Redis CAS 操作？

在 Redis 的事务中，WATCH 命令可用于提供CAS(check-and-set)功能。

假设我们通过 WATCH 命令在事务执行之前监控了多个 keys ，倘若在 WATCH 之后有任何 Key 的值发生了变化，EXEC 命令执行的事务都将被放弃，同时返回 nil 应答以通知调用者事务执行失败。

具体的示例，可以看看 《Redis 事务锁 CAS 实现以及深入误区》 。

Redis 集群都有哪些方案？

- 1、Redis Sentinel
- 2、Redis Cluster
- 3、Twemproxy
- 4、Codis
- 5、客户端分片

关于前四种，可以看看 《Redis 实战（四）集群机制》 这篇文章。

什么是 Redis 主从同步？

Redis 的主从同步(replication)机制，允许 Slave 从 Master 那里，通过网络传输拷贝到完整的数据备份，从而达到主从机制。

- 主数据库可以进行读写操作，当发生写操作的时候自动将数据同步到从数据库，而从数据库一般是只读的，并接收主数据库同步过来的数据。
- 一个主数据库可以有多个从数据库，而一个从数据库只能有一个主数据库。
- 第一次同步时，主节点做一次bgsave操作，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。

好处：
通过 Redis 的复制功，能可以很好的实现数据库的读写分离，提高服务器的负载能力。主数据库主要进行写操作，而从数据库负责读操作。

更多详细，可以看看 《Redis 主从架构》 。

如何使用 Redis Sentinel 实现高可用？

可以看看 《Redis哨兵集群实现高可用》

如果使用 Redis Cluster 实现高可用？

可以看看

- 《Redis 集群教程》 完整版
- 《Redis 集群模式的工作原理能说一下么？》 精简版

> 说说 Redis 哈希槽的概念？

Redis Cluster 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。

因为最大是 16384 个哈希槽，所以考虑 Redis 集群中的每个节点都能分配到一个哈希槽，所以最多支持 16384 个 Redis 节点。

> Redis Cluster 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 个复制节点。

所以，Redis Cluster 可以说是 Redis Sentinel带分片的加强版。也可以说：

- Redis Sentinel 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。
- 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。

Redis Cluster 方案什么情况下会导致整个集群不可用？

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 宕机了，那么整个集群就会以为缺少 5501-11000 这个范围的槽而不可用。

Redis Cluster 会有写操作丢失吗？为什么？

Redis 并不能保证数据的强一致性，而是【异步复制】，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。

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。

请说说生产环境中的 Redis 是怎么部署的？

- Redis Cluster，10 台机器，5 台机器部署了 redis 主实例，另外 5 台机器部署了 redis 的从实例，每个主实例挂了一个从实例，5 个节点对外提供读写服务，每个节点的读写高峰 qps 可能可以达到每秒 5 万，5 台机器最多是 25 万读写请求每秒。
- 机器是什么配置？32G 内存 + 8 核 CPU + 1T 磁盘，但是分配给 Redis 进程的是 10g 内存，一般线上生产环境，Redis 的内存尽量不要超过 10g，超过 10g 可能会有问题。那么，5 台机器对外提供读写，一共有 50g 内存。
- 因为每个主实例都挂了一个从实例，所以是高可用的，任何一个主实例宕机，都会自动故障迁移，Redis 从实例会自动变成主实例继续提供读写服务。
- 你往内存里写的是什么数据？每条数据的大小是多少？商品数据，每条数据是 10kb 。100 条数据是 1mb，10万条数据是1g。常驻内存的是 200 万条商品数据，占用内存是20g，仅仅不到总内存的50%。目前高峰期每秒就是 3500 左右的请求量。
- 其实大型的公司，会有基础架构的 team 负责缓存集群的运维。

什么是 Redis 分区？

- Redis 分区是什么？
  分区是分割数据到多个Redis实例的处理过程，因此每个实例只保存key的一个子集。
- 分区的优势？
  - 通过利用多台计算机内存的和值，允许我们构造更大的数据库。
  - 通过多核和多台计算机，允许我们扩展计算能力；通过多台计算机和网络适配器，允许我们扩展网络带宽。
- 分区的不足？
  redis的一些特性在分区方面表现的不是很好：
  - 涉及多个key的操作通常是不被支持的。举例来说，当两个set映射到不同的redis实例上时，你就不能对这两个set执行交集操作。
  - 涉及多个key的redis事务不能使用。
  - 当使用分区时，数据处理较为复杂，比如你需要处理多个rdb/aof文件，并且从多个实例和主机备份持久化文件。
  - 增加或删除容量也比较复杂。redis集群大多数支持在运行时增加、删除节点的透明数据平衡的能力，但是类似于客户端分区、代理等其他系统则不支持这项特性。然而，一种叫做presharding的技术对此是有帮助的。
- 分区类型？
  Redis 有两种类型分区。 假设有4个Redis实例 R0，R1，R2，R3，和类似user:1，user:2这样的表示用户的多个key，对既定的key有多种不同方式来选择这个key存放在哪个实例中。也就是说，有不同的系统来映射某个key到某个Redis服务。
  - 范围分区:最简单的分区方式是按范围分区，就是映射一定范围的对象到特定的Redis实例。比如，ID从0到10000的用户会保存到例R0，ID从10001到20000的用户会保存到R1，以此类推。这种方式是可行的，并且在实际中使用，不足就是要有一个区间范围到实例的映射表。这个表要被管理，同时还需要各种对象的映射表，通常对Redis来说并非是好的方法。
  - 哈希分区(另外一种分区方法是hash分区。这对任何key都适用，也无需是object_name:这种形式，像下面描述的一样简单：)
    - 用一个hash函数将key转换为一个数字，比如使用crc32 hash函数。对key foobar执行crc32(foobar)会输出类似93024922的整数。
    - 对这个整数取模，将其转化为0-3之间的数字，就可以将这个整数映射到4个Redis实例中的一个了。93024922 % 4 = 2，就是说keyfoobar应该被存到R2实例中。注意：取模操作是取除的余数，通常在多种编程语言中用%操作符实现。

可能有胖友会懵逼，又是 Redis 主从复制，又是 Redis 分区，又是 Redis 集群。傻傻分不清啊！

- Redis 分区是一种模式，将数据分区到不同的 Redis 节点上，而 Redis 集群的 Redis Cluster、Twemproxy、Codis、客户端分片(不包括 Redis Sentinel) 这四种方案，是 Redis 分区的具体实现。
- Redis 每个分区，如果想要实现高可用，需要使用到 Redis 主从复制。

你知道有哪些 Redis 分区实现方案？

Redis 分区方案，主要分成两种类型：

- 客户端分区，就是在客户端就已经决定数据会被存储到哪个 Redis 节点或者从哪个 Redis节点读取。大多数客户端已经实现了客户端分区。
  - 案例：Redis Cluster 和客户端分区。
- 代理分区，意味着客户端将请求发送给代理，然后代理决定去哪个节点写数据或者读数据。代理根据分区规则决定请求哪些 Redis 实例，然后根据 Redis 的响应结果返回给客户端。
  - 案例：Twemproxy 和 Codis 。

查询路由(Query routing)的意思，是客户端随机地请求任意一个 Redis 实例，然后由 Redis 将请求转发给正确的 Redis 节点。Redis Cluster 实现了一种混合形式的查询路由，但并不是直接将请求从一个Redis 节点转发到另一个 Redis 节点，而是在客户端的帮助下直接 redirect 到正确的 Redis 节点。

Redis 有哪些重要的健康指标？

推荐阅读 《Redis 几个重要的健康指标》

如何提高 Redis 命中率？

推荐阅读 《如何提高缓存命中率（Redis）》 。

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？

Redis 可以处理多达 2^32 的 keys ，任何 list、set、和 sorted set 都可以放 2^32 个元素。

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？

使用 keys 指令可以扫出指定模式的 key 列表。

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。

Redis 常见的性能问题都有哪些？如何解决？

- 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。
  - 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，所以 Master 最好不要写内存快照。
  - Master AOF 持久化，如果不重写 AOF 文件，这个持久化方式对性能的影响是最小的，但是 AOF 文件会不断增大，AOF 文件过大会影响 Master 重启的恢复速度。
  - 所以，Master 最好不要做任何持久化工作，包括内存快照和 AOF 日志文件，特别是不要启用内存快照做持久化。如果数据比较Slave 开启AOF备份数据，策略为每秒同步一次。
- 2.Master 调用 BGREWRITEAOF 重写 AOF 文件，AOF 在重写的时候会占大量的 CPU 和内存资源，导致服务 load 过高，出现短暂服务暂停现象。
  - TODO 怎么解决？
- 3.尽量避免在压力很大的主库上增加从库。
  - TODO 怎么解决？
- 4、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，也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。
- 5、Redis 主从复制的性能问题，为了主从复制的速度和连接的稳定性，Slave 和 Master 最好在同一个局域网内。

Redis的key是如何寻址的?

- 1.redis 中的每一个数据库，都由一个 redisDb 的结构存储。
  - redisDb.id存储着redis数据库以整数表示的号码。
  - redisDb.dict 存储着该库所有的键值对数据。
  - redisDb.expires 保存着每一个键的过期时间。
- 2.当redis 服务器初始化时，会预先分配 16 个数据库（该数量可以通过配置文件配置），所有数据库保存到结构 redisServer 的一个成员 redisServer.db 数组中。当我们选择数据库 select number 时，程序直接通过 redisServer.db[number] 来切换数据库。有时候当程序需要知道自己是在哪个数据库时，直接读取 redisDb.id 即可
- redis 的字典使用哈希表作为其底层实现。dict 类型使用的两个指向哈希表的指针，其中 0 号哈希表（ht[0]）主要用于存储数据库的所有键值，而1号哈希表主要用于程序对 0 号哈希表进行 rehash 时使用，rehash 一般是在添加新值时会触发，这里不做过多的赘述。所以redis 中查找一个key，其实就是对进行该dict 结构中的 ht[0] 进行查找操作。
- 4.既然是哈希，那么我们知道就会有哈希碰撞，那么当多个键哈希之后为同一个值怎么办呢？redis采取链表的方式来存储多个哈希碰撞的键。也就是说，当根据key的哈希值找到该列表后，如果列表的长度大于1，那么我们需要遍历该链表来找到我们所查找的key。当然，一般情况下链表长度都为是1，所以时间复杂度可看作o(1)

> 当redis 拿到一个key 时，如果找到该key的位置

- 1、当拿到一个key后， redis 先判断当前库的0号哈希表是否为空，即：if (dict->ht[0].size == 0)。如果为true直接返回NULL。
- 2、判断该0号哈希表是否需要rehash，因为如果在进行rehash，那么两个表中者有可能存储该key。如果正在进行rehash，将调用一次_dictRehashStep方法，_dictRehashStep 用于对数据库字典、以及哈希键的字典进行被动 rehash，这里不作赘述。
- 3、计算哈希表，根据当前字典与key进行哈希值的计算。
- 4、根据哈希值与当前字典计算哈希表的索引值。
- 5、根据索引值在哈希表中取出链表，遍历该链表找到key的位置。一般情况，该链表长度为1。
- 6、当 ht[0] 查找完了之后，再进行了次rehash判断，如果未在rehashing，则直接结束，否则对ht[1]重复345步骤。 到此我们就找到了key在内存中的位置了。

如何保证数据库与redis缓存一致的

先淘汰缓存中的数据，然后更新数据库