2021 Redis高频面试题

1、什么是Redis？简述它的优缺点?

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，因此Redis可以用来实现很多有用的功能。

比方说用他的List来做FIFO双向链表，实现一个轻量级的高性 能消息队列服务，用他的Set可以做高性能的tag系统等等。

另外Redis也可以对存入的Key-Value设置expire时间，因此也可以被当作一 个功能加强版的memcached来用。 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。

2、Redis相比memcached有哪些优势?

(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

3、Redis支持哪几种数据类型?

String、List、Set、Sorted Set、hash

4、Redis主要消耗什么物理资源?

内存。

5、Redis的全称是什么?

Remote Dictionary Server。

6、Redis有哪几种数据淘汰策略?

noeviction:返回错误当内存限制达到并且客户端尝试执行会让更多内存被使用的命令（大部分的写入指令，但DEL和几个例外）

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。

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。

allkeys-random: 回收随机的键使得新添加的数据有空间存放。

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。

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。

7、Redis官方为什么不提供Windows版本?

因为目前Linux版本已经相当稳定，而且用户量很大，无需开发windows版本，反而会带来兼容性等问题。

8、一个字符串类型的值能存储最大容量是多少?

512M

9、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。如果不将数据放在内存中，磁盘I/O速度为严重影响redis的性能。

在内存越来越便宜的今天，redis将会越来越受欢迎。 如果设置了最大使用的内存，则数据已有记录数达到内存限值后不能继续插入新值。

10、Redis集群方案应该怎么做？都有哪些方案?

1.codis。

目前用的最多的集群方案，基本和twemproxy一致的效果，但它支持在 节点数量改变情况下，旧节点数据可恢复到新hash节点。

2.redis cluster3.0自带的集群，特点在于他的分布式算法不是一致性hash，而是hash槽的概念，以及自身支持节点设置从节点。具体看官方文档介绍。

3.在业务代码层实现，起几个毫无关联的redis实例，在代码层，对key 进行hash计算，然后去对应的redis实例操作数据。 这种方式对hash层代码要求比较高，考虑部分包括，节点失效后的替代算法方案，数据震荡后的自动脚本恢复，实例的监控，等等。

11、Redis集群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失败了，那么整个集群就会以为缺少5501-11000这个范围的槽而不可用。

12、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。

13、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Memcached）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的使用Redis来缓存会话的文档。甚至广为人知的商业平台Magento也提供Redis的插件。

（2）全页缓存（FPC）

除基本的会话token之外，Redis还提供很简便的FPC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。

此外，对WordPress的用户来说，Pantheon有一个非常好的插件 wp-redis，这个插件能帮助你以最快速度加载你曾浏览过的页面。

（3）队列

Reids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。

如果你快速的在Google中搜索“Redis queues”，你马上就能找到大量的开源项目，这些项目的目的就是利用Redis创建非常好的后端工具，以满足各种队列需求。例如，Celery有一个后台就是使用Redis作为broker，你可以从这里去查看。

（4）排行榜/计数器

Redis在内存中对数字进行递增或递减的操作实现的非常好。集合（Set）和有序集合（Sorted Set）也使得我们在执行这些操作的时候变的非常简单，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”，我们只需要像下面一样执行即可：

当然，这是假定你是根据你用户的分数做递增的排序。如果你想返回用户及用户的分数，你需要这样执行：

ZRANGE user_scores 0 10 WITHSCORES

Agora Games就是一个很好的例子，用Ruby实现的，它的排行榜就是使用Redis来存储数据的，你可以在这里看到。

（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非常多。我已看见人们在社交网络连接中使用，还可作为基于发布/订阅的脚本触发器，甚至用Redis的发布/订阅功能来建立聊天系统！

14、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。

15、Redis和Redisson有什么关系？

Redisson是一个高级的分布式协调Redis客服端，能帮助用户在分布式环境中轻松实现一些Java的对象(Bloom filter, BitSet, Set, SetMultimap, ScoredSortedSet, SortedSet, Map, ConcurrentMap, List, ListMultimap, Queue, BlockingQueue, Deque, BlockingDeque, Semaphore, Lock, ReadWriteLock, AtomicLong, CountDownLatch, Publish / Subscribe, HyperLogLog)。

16、Jedis与Redisson对比有什么优缺点？

Jedis是Redis的Java实现的客户端，其API提供了比较全面的Redis命令的支持；

Redisson实现了分布式和可扩展的Java数据结构，和Jedis相比，功能较为简单，不支持字符串操作，不支持排序、事务、管道、分区等Redis特性。Redisson的宗旨是促进使用者对Redis的关注分离，从而让使用者能够将精力更集中地放在处理业务逻辑上。

17、Redis如何设置密码及验证密码?

设置密码：config set requirepass 123456

授权密码：auth 123456

18、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。

19、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.

20、Redis集群会有写操作丢失吗？为什么?

Redis并不能保证数据的强一致性，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。

21、Redis集群之间是如何复制的?

异步复制

22、Redis集群最大节点个数是多少?

16384个。

23、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。

24、怎么测试Redis的连通性?

ping

25、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。

26、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。

27、Redis事务相关的命令有哪几个?

MULTI、EXEC、DISCARD、WATCH

28、Redis key的过期时间和永久有效分别怎么设置?

EXPIRE和PERSIST命令。

29、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。

30、Redis回收进程如何工作的?

一个客户端运行了新的命令，添加了新的数据。

Redi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。

一个新的命令被执行，等等。

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。

如果一个命令的结果导致大量内存被使用（例如很大的集合的交集保存到一个新的键），不用多久内存限制就会被这个内存使用量超越。

31、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。如果不将数据放在内存中，磁盘I/O速度为严重影响redis的性能。在内存越来越便宜的今天，redis将会越来越受欢迎。如果设置了最大使用的内存，则数据已有记录数达到内存限值后不能继续插入新值。

32、Redis常见的性能问题都有哪些？如何解决?

● 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。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，如果数据比较关键，某个Slave开启AOF备份数据，策略为每秒同步一次。

● 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。

● 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。

33、Redis最适合的场景有哪些?

● 会话缓存（Session Cache）

● 全页缓存（FPC）

● 队列

● 排行榜/计数器

● 发布/订阅

34、Memcache与Redis的区别都有哪些?

● 存储方式不同，Memcache是把数据全部存在内存中，数据不能超过内存的大小，断电后数据库会挂掉。Redis有部分存在硬盘上，这样能保证数据的持久性。

● 数据支持的类型不同memcahe对数据类型支持相对简单，redis有复杂的数据类型。

● 使用底层模型不同 它们之间底层实现方式以及与客户端之间通信的应用协议不一样。Redis直接自己构建了VM机制，因为一般的系统调用系统函数的话，会浪费一定的时间去移动和请求。

● 支持的value大小不一样redis最大可以达到1GB，而memcache只有1MB。

35、Redis有哪几种数据结构?

● String——字符串
String数据结构是简单的key-value类型，value不仅可以是String，也可以是数字（当数字类型用Long可以表示的时候encoding就是整型，其他都存储在sdshdr当做字符串）。

● Hash——字典
在Memcached中，我们经常将一些结构化的信息打包成hashmap，在客户端序列化后存储为一个字符串的值（一般是JSON格式），比如用户的昵称、年龄、性别、积分等。

● List——列表
List说白了就是链表（redis使用双端链表实现的List）

● Set——集合
Set就是一个集合，集合的概念就是一堆不重复值的组合。利用Redis提供的Set数据结构，可以存储一些集合性的数据。

● Sorted Set——有序集合
和Set相比，Sorted Set是将Set中的元素增加了一个权重参数score，使得集合中的元素能够按score进行有序排列，

● 带有权重的元素，比如一个游戏的用户得分排行榜

● 比较复杂的数据结构，一般用到的场景不算太多

36、Redis的持久化是什么?

RDB持久化：该机制可以在指定的时间间隔内生成数据集的时间点快照（point-in-time snapshot）。

AOF持久化：记录服务器执行的所有写操作命令，并在服务器启动时，通过重新执行这些命令来还原数据集。AOF文件中的命令全部以Redis协议的格式来保存，新命令会被追加到文件的末尾。Redis还可以在后台对AOF文件进行重写（rewrite），使得AOF文件的体积不会超出保存数据集状态所需的实际大小。

AOF和RDB的同时应用：当Redis重启时，它会优先使用AOF文件来还原数据集，因为AOF文件保存的数据集通常比RDB文件所保存的数据集更完整。

37、RDB的优缺点?

- **优点：**RDB是一个非常紧凑（compact）的文件，它保存了Redis在某个时间点上的数据集。这种文件非常适合用于进行备份：比如说，你可以在最近的24小时内，每小时备份一次RDB文件，并且在每个月的每一天，也备份一个RDB文件。这样的话，即使遇上问题，也可以随时将数据集还原到不同的版本。RDB非常适用于灾难恢复（disaster recovery）：它只有一个文件，并且内容都非常紧凑，可以（在加密后）将它传送到别的数据中心，或者亚马逊S3中。RDB可以最大化Redis的性能：父进程在保存RDB文件时唯一要做的就是fork出一个子进程，然后这个子进程就会处理接下来的所有保存工作，父进程无须执行任何磁盘I/O操作。RDB在恢复大数据集时的速度比AOF的恢复速度要快。

- **缺点**：如果你需要尽量避免在服务器故障时丢失数据，那么RDB不适合你。虽然Redis允许你设置不同的保存点（save point）来控制保存RDB文件的频率，但是，因为RDB文件需要保存整个数据集的状态，所以它并不是一个轻松的操作。因此你可能会至少5分钟才保存一次RDB文件。在这种情况下，一旦发生故障停机，你就可能会丢失好几分钟的数据。每次保存RDB的时候，Redis都要fork()出一个子进程，并由子进程来进行实际的持久化工作。在数据集比较庞大时，fork()可能会非常耗时，造成服务器在某某毫秒内停止处理客户端；如果数据集非常巨大，并且CPU时间非常紧张的话，那么这种停止时间甚至可能会长达整整一秒。

38、AOF的优缺点?

● 优点:

使用AOF持久化会让Redis变得非常耐久（much more durable）：你可以设置不同的fsync策略，比如无fsync，每秒钟一次fsync，或者每次执行写入命令时fsync。AOF的默认策略为每秒钟fsync一次，在这种配置下，Redis仍然可以保持良好的性能，并且就算发生故障停机，也最多只会丢失一秒钟的数据（fsync会在后台线程执行，所以主线程可以继续努力地处理命令请求）。AOF文件是一个只进行追加操作的日志文件（append onlylog），因此对AOF文件的写入不需要进行seek，即使日志因为某些原因而包含了未写入完整的命令（比如写入时磁盘已满，写入中途停机，等等），redis-check-aof工具也可以 轻易地修复这种问题。

Redis可以在AOF文件体积变得过大时，自动地在后台对AOF进行重写：重写后的新AOF文件包含了恢复当前数据集所需的最小命令集合。整个重写操作是绝对安全的，因为Redis在创建新AOF文件的过程中，会继续将命令追加到现有的AOF文件里面，即使重写过程中发生停机，现有的AOF文件也不会丢失。而一旦新AOF文件创建完毕，Redis就会从旧AOF文件切换到新AOF文件，并开始对新AOF文件进行追加操作。

● 缺点:

对于相同的数据集来说，AOF文件的体积通常要大于RDB文件的体积。根据所使用的fsync策略，AOF的速度可能会慢于RDB。在一般情况下，每秒fsync的性能依然非常高，而关闭fsync可以让AOF的速度和RDB一样快，即使在高负荷之下也是如此。不过在处理巨大的写入载入时，RDB可以提供更有保证的最大延迟时间（latency）。

AOF在过去曾经发生过这样的bug：因为个别命令的原因，导致AOF文件在重新载入时，无法将数据集恢复成保存时的原样。（举个例子，阻塞命令BRPOPLPUSH就曾经引起过这样的bug。）测试套件里为这种情况添加了测试：它们会自动生成随机的、复杂的数据集，并通过重新载入这些数据来确保一切正常。虽然这种bug在AOF文件中并不常见，但是对比来说，RDB几乎是不可能出现这种bug的。

39、简单说说缓存雪崩及解决方法

缓存雪崩我们可以简单的理解为：由于原有缓存失效，新缓存未到期间
(例如：我们设置缓存时采用了相同的过期时间，在同一时刻出现大面积的缓存过期)，所有原本应该访问缓存的请求都去查询数据库了，而对数据库CPU和内存造成巨大压力，严重的会造成数据库宕机。从而形成一系列连锁反应，造成整个系统崩溃。）

解决办法:
大多数系统设计者考虑用加锁（ 最多的解决方案）或者队列的方式保证来保证不会有大量的线程对数据库一次性进行读写，从而避免失效时大量的并发请求落到底层存储系统上。还有一个简单方案就时讲缓存失效时间分散开。

40、缓存穿透怎么导致的?

在高并发下查询key不存在的数据，会穿过缓去存查询数据库。导致数据库压力过大而宕机。
解决方法:

1. 对查询结果为空的情况也进行缓存，缓存时间（ttl）设置短一点，或者该key对应的数据insert了之后清理缓存。
   缺点：缓存太多空值占用了更多的空间
2. 使用布隆过滤器。在缓存之前在加一层布隆过滤器，在查询的时候先去布隆过滤器查询 key 是否存在，如果不存在就直接返回，存在再查缓存和DB。

布隆过滤器原理： 当一个元素被加入集合时，将这个元素通过n次Hash函数结果映射成一个数组中的n个点，把它们置为1。检索时，我们只要看看这些点是不是都是1就（大约）知道集合中有没有它了：如果这些点有任何一个0，则被检元素一定不在；如果都是1，则被检元素很可能在。总之布隆过滤器是一个很大二进制的位数组，数组里面只存0和1。

41、项目中有出现过缓存击穿，简单说说怎么回事?

缓存在某个时间点过期的时候，恰好在这个时间点对这个Key有大量的并发请求过来，这些请求发现缓存过期一般会从数据库中加载数据并回设到缓存，这个时候大并发的请求可能会瞬间把后端DB压垮

解决方案：

1. 用分布式锁控制访问的线程，使用redis的setnx互斥锁先进行判断，这样其他线程就处于等待状态，保证不会有大并发操作去操作数据库。
2. 不设超时时间，采用volatile-lru淘汰策略
   缺点：会造成写一致问题，当数据库数据发生更新时，缓存中的数据不会及时更新，这样会造成数据库中的数据与缓存中的数据的不一致，应用会从缓存中读取到脏数据。可采用延时双删策略处理。

42、遇到缓存一致性问题，你怎么解决的?

由于缓存和数据库不属于同一个数据源，本质上非原子操作，所以是无法保证强一致性的，只能去实现最终一致性。
解决方案：

1. 延时双删：先更新数据库同时删除缓存，等2秒后再删除一次缓存，等到读的时候在回写到缓存。
2. 利用工具(canal)将数据库的binlog日志采集发送到MQ中，然后通过ACK机制确认处理删除缓存

43、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。以 Java 为例，使用自带的 map 或者 guava 实现的是本地缓存，最主要的特点是轻量以及快速，生命周期随着 jvm 的销毁而结束，并且在多实例的情况下，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 或 memcached 之类的称为分布式缓存，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。缺点是需要保持 redis 或 memcached服务的高可用，整个程序架构上较为复杂。

44、如何选择合适的持久化方式?

1、一般来说， 如果想达到足以媲美PostgreSQL的数据安全性，你应该同时使用两种持久化功能。在这种情况下，当 Redis 重启的时候会优先载入AOF文件来恢复原始的数据，因为在通常情况下AOF文件保存的数据集要比RDB文件保存的数据集要完整。

2、如果你非常关心你的数据， 但仍然可以承受数分钟以内的数据丢失，那么你可以只使用RDB持久化。

3、有很多用户都只使用AOF持久化，但并不推荐这种方式，因为定时生成RDB快照（snapshot）非常便于进行数据库备份， 并且 RDB 恢复数据集的速度也要比AOF恢复的速度要快，除此之外，使用RDB还可以避免AOF程序的bug。

4、如果你只希望你的数据在服务器运行的时候存在，你也可以不使用任何持久化方式。

45、Redis持久化数据和缓存怎么做扩容?

1、如果Redis被当做缓存使用，使用一致性哈希实现动态扩容缩容。

2、如果Redis被当做一个持久化存储使用，必须使用固定的keys-to-nodes映射关系，节点的数量一旦确定不能变化。否则的话(即Redis节点需要动态变化的情况），必须使用可以在运行时进行数据再平衡的一套系统，而当前只有Redis集群可以做到这样。

46、Redis的内存淘汰策略有哪些?

Redis的内存淘汰策略是指在Redis的用于缓存的内存不足时，怎么处理需要新写入且需要申请额外空间的数据。

1、全局的键空间选择性移除

- 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。

2、设置过期时间的键空间选择性移除

- 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。

47、简单描述下Redis线程模型

Redis基于Reactor模式开发了网络事件处理器，这个处理器被称为文件事件处理器（file event handler）。它的组成结构为4部分：多个套接字、IO多路复用程序、文件事件分派器、事件处理器。因为文件事件分派器队列的消费是单线程的，所以Redis才叫单线程模型。

- 文件事件处理器使用 I/O 多路复用（multiplexing）程序来同时监听多个套接字， 并根据套接字目前执行的任务来为套接字关联不同的事件处理器。
- 当被监听的套接字准备好执行连接应答（accept）、读取（read）、写入（write）、关闭（close）等操作时， 与操作相对应的文件事件就会产生， 这时文件事件处理器就会调用套接字之前关联好的事件处理器来处理这些事件。

虽然文件事件处理器以单线程方式运行， 但通过使用 I/O 多路复用程序来监听多个套接字， 文件事件处理器既实现了高性能的网络通信模型， 又可以很好地与 redis 服务器中其他同样以单线程方式运行的模块进行对接， 这保持了 Redis 内部单线程设计的简单性。

48、Redis事务其他实现方式?

1、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

2、基于中间标记变量，通过另外的标记变量来标识事务是否执行完成，读取数据时先读取该标记变量判断是否事务执行完成。但这样会需要额外写代码实现，比较繁琐

49、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/s。

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。

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。

50、 如何解决 Redis 的并发竞争 Key 问题?

所谓 Redis 的并发竞争 Key 的问题也就是多个系统同时对一个 key 进行操作，但是最后执行的顺序和我们期望的顺序不同，这样也就导致了结果的不同!

推荐一种方案：分布式锁（zookeeper 和 redis 都可以实现分布式锁）。（如果不存在 Redis 的并发竞争 Key 问题，不要使用分布式锁，这样会影响性能）

基于zookeeper临时有序节点可以实现的分布式锁。大致思想为：每个客户端对某个方法加锁时，在zookeeper上的与该方法对应的指定节点的目录下，生成一个唯一的瞬时有序节点。判断是否获取锁的方式很简单，只需要判断有序节点中序号最小的一个。当释放锁的时候，只需将这个瞬时节点删除即可。同时，其可以避免服务宕机导致的锁无法释放，而产生的死锁问题。完成业务流程后，删除对应的子节点释放锁。

在实践中，当然是从以可靠性为主。所以首推Zookeeper。

51、 什么是 RedLock?

Redis 官方站提出了一种权威的基于 Redis 实现分布式锁的方式名叫 Redlock，此种方式比原先的单节点的方法更安全。它可以保证以下特性：

1. 安全特性：互斥访问，即永远只有一个 client 能拿到锁
2. 避免死锁：最终 client 都可能拿到锁，不会出现死锁的情况，即使原本锁住某资源的 client crash 了或者出现了网络分区
3. 容错性：只要大部分 Redis 节点存活就可以正常提供服务

52、什么时候需要缓存降级?

当访问量剧增、服务出现问题（如响应时间慢或不响应）或非核心服务影响到核心流程的性能时，仍然需要保证服务还是可用的，即使是有损服务。系统可以根据一些关键数据进行自动降级，也可以配置开关实现人工降级。

缓存降级的最终目的是保证核心服务可用，即使是有损的。而且有些服务是无法降级的（如加入购物车、结算）。

在进行降级之前要对系统进行梳理，看看系统是不是可以丢卒保帅；从而梳理出哪些必须誓死保护，哪些可降级；比如可以参考日志级别设置预案：

1. 一般：比如有些服务偶尔因为网络抖动或者服务正在上线而超时，可以自动降级；
2. 警告：有些服务在一段时间内成功率有波动（如在95~100%之间），可以自动降级或人工降级，并发送告警；

3. 错误：比如可用率低于90%，或者数据库连接池被打爆了，或者访问量突然猛增到系统能承受的最大阀值，此时可以根据情况自动降级或者人工降级；
4. 严重错误：比如因为特殊原因数据错误了，此时需要紧急人工降级。

服务降级的目的，是为了防止Redis服务故障，导致数据库跟着一起发生雪崩问题。因此，对于不重要的缓存数据，可以采取服务降级策略，例如一个比较常见的做法就是，Redis出现问题，不去数据库查询，而是直接返回默认值给用户。

53、如何保证缓存与数据库双写时的数据一致性?

你只要用缓存，就可能会涉及到缓存与数据库双存储双写，你只要是双写，就一定会有数据一致性的问题，那么你如何解决一致性问题?

一般来说，就是如果你的系统不是严格要求缓存+数据库必须一致性的话，缓存可以稍微的跟数据库偶尔有不一致的情况，最好不要做这个方案，读请求和写请求串行化，串到一个内存队列里去，这样就可以保证一定不会出现不一致的情况

串行化之后，就会导致系统的吞吐量会大幅度的降低，用比正常情况下多几倍的机器去支撑线上的一个请求。

还有一种方式就是可能会暂时产生不一致的情况，但是发生的几率特别小，就是先更新数据库，然后再删除缓存。

问题场景	描述	解决
先写缓存，再写数据库，缓存写成功，数据库写失败	缓存写成功，但写数据库失败或者响应延迟，则下次读取（并发读）缓存时，就出现脏读	这个写缓存的方式，本身就是错误的，需要改为先写数据库，把旧缓存置为失效；读取数据的时候，如果缓存不存在，则读取数据库再写缓存
先写数据库，再写缓存，数据库写成功，缓存写失败	写数据库成功，但写缓存失败，则下次读取（并发读）缓存时，则读不到数据	缓存使用时，假如读缓存失败，先读数据库，再回写缓存的方式实现
需要缓存异步刷新	指数据库操作和写缓存不在一个操作步骤中，比如在分布式场景下，无法做到同时写缓存或需要异步刷新（补救措施）时候	确定哪些数据适合此类场景，根据经验值确定合理的数据不一致时间，用户数据刷新的时间间隔