

大家好，很高兴我们可以继续学习交流Java高频面试题。从本小节开始，我们进入了第三个章节，主要介绍服务端的通用工具，即redis和kafka相关知识点。在我们的日常开发中，经常会需要使用缓存和消息队列。缓存的使用可以减少网络请求或者查询数据库的次数，有效提高访问速度。消息队列的使用可以实现生产者和消费者的解耦，实现异步通通过载保护等功能。

在当今的招聘需求中，即使是校园招聘也对缓存和消息队列有一定的考察要求，不要求大家有多好多深的掌握，但是基本的概念与原理还是需要了解的。可以说，缓存和消息队列的了解与熟悉，对于我们的面试来说是一个极大的加分项。

本小节就业界使用比较广的redis缓存技术进行一个简单的交流与学习，主要是针对面试中常见的redis知识点来展开交流，旨在让大家对redis的基本概念与原理有一定的了解与掌握。

## (1) redis有了解吗？

**答：redis (Remote Dictionary Server远程字典服务)**，是一款高性能的(key/value)分布式**内存数据库**，基于**内存运行**并支持**持久化**的NoSQL数据库。因为数据都在内存中，所以运行速度快。redis支持丰富的数据类型并且支持事务，事务中的所有命令会被序列化、按顺序执行，在执行的过程中不会被其他客户端发送来的命令打断。

**解析：**

这是对redis考察的一个试探性提问，应聘者可以根据自己的理解与掌握，先从整体上对redis进行一个阐述。针对**redis基础知识**的考察，面试官还可以选择继续追问如下的常见知识点。

### redis相比memcached有哪些优势？(掌握)

- 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。
- redis的速度比memcached快很多，并且redis支持数据的持久化。
- redis支持数据的备份，即master-slave模式的数据备份。
- 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自己构建了VM 管理机制。
- value大小不同，redis最大可以达到512MB，而memcache只有1MB。

### 面试官：“redis都支持哪些数据类型？应用场景有哪些？”(掌握)

redis支持五种数据类型作为其Value，redis的Key都是字符串类型的。

- **string**：redis 中字符串 value 最大可为512M。可以用来做一些计数功能的缓存（也是实际工作中最常见的）。
- **list**：简单的字符串列表，按照插入顺序排序，可以添加一个元素到列表的头部（左边）或者尾部（右边），其底层实现是一个链表。可以实现一个简单消息队列功能，做基于redis的分页功能等。
- **set**：是一个字符串类型的无序集合。可以用来进行全局去重等。
- **sorted set**：是一个字符串类型的有序集合，给每一个元素一个固定的**分数score**来保持顺序。可以用来做排行榜应用或者进行范围查找等。
- **hash**：键值对集合，是一个字符串类型的 Key和 Value 的映射表，也就是说其存储的Value是一个键值对（Key-Value）。可以用来存放一些具有特定结构的信息。

其实redis还支持三种特殊的数据类型，分别为BitMap、Geo和HyperLogLog，感兴趣的同学可以自行学习。

一般情况下，可以认为redis的支持的数据类型有上述五种，其底层数据结构包括：**简单动态字符串，链表，字典，跳表，整数集合以及压缩列表。**

9

## 面试官：“redis的配置文件有了解吗？”

下载并且解压redis之后，在解压目录下有个配置文件 **redis.windows.conf**（大家可以自行查阅），在配置文件中对redis进行了分模块配置，常用的模块如下：

- **NETWORK**：该模块可以配置一些redis服务器地址，端口以及超时时间等
- **GENERAL**：该模块可以对日志文件的路径和日志级别等进行配置
- **SNAPSHOTTING**：redis持久化配置信息等
- **REPLICATION**：redis集群配置等信息
- **MEMORY MANAGEMENT**：内存管理，包括数据过期删除策略信息的设置
- **APPEND ONLY MODE**：日志持久化方式信息设置

## (2) redis是单线程的吗？为什么执行速度这么快？（重点掌握）

**答：**redis是单线程的，redis的单线程是指网络请求模块使用了一个线程，所以不需考虑并发安全性。但是对于需要依赖多个操作的复合操作来说，还是需要锁的，而且有可能是分布式锁。

**那么单线程的redis为什么执行速度如此之快？**

- 基于内存实现，完全内存计算
- 单线程操作，避免了线程上下文切换操作
- **多路I/O复用的线程模型**，实现了一个线程监控多个IO流，及时响应请求
- redis对外部的依赖比较少，属于轻量级内存数据库

**解析：**

redis的线程模型**多路I/O复用机制**是一个比较重要并且常见的考察点。目前支持I/O多路复用的系统调用有**select**，**pselect**，**poll**，**epoll**等函数。I/O多路复用就是通过一种机制**一个进程可以监视多个描述符**，一旦某个描述符就绪或者写就绪，其能够通知应用程序进行相应的读写操作。

**多路I/O复用机制**与多进程和多线程技术相比系统开销小，系统不必创建进程/线程，也不必维护这些进程/线程，从而大大减小了系统的开销。

**关于常见函数的特点如下所示：**

- **select函数：**
  - 会修改传入的参数数组，这个对于一个需要调用很多次的函数，是非常不友好的。
  - 有最大监听连接数1024个的限制
  - 如果任何一个sock(I/O stream)出现了数据，select没有返回具体是哪个返回了数据，需要采用轮询的方式去遍历获取
  - 线程不安全（当你在一个线程中已经监听该socket，另一个线程想要将该socket关闭，则结果会不可预知）
  - “If a file descriptor being monitored by select() is closed in another thread, the result is unspecified”
- **poll函数：**
  - 去掉了1024的限制（使用链表搞定）

- 不再修改传入的参数数组
- 依然是线程不安全的
- **epoll函数:**
  - epoll 不仅返回socket组里面数据，还可以确定 具体哪个socket有数据
  - 线程安全

### 题目总结:

限于篇幅，在这个题目中，我们仅仅是简单对线程模型多路I/O复用机制做了一个阐述与介绍，有能力的同学可以继续研究其具体实现机制，这个题目是面试中的一个很好的加分项（如果大家可以阐述清楚）

## (3) 使用redis可能出现的问题:

答：在这里我们主要介绍Redis可能出现的四个问题，如下所示：

### 缓存雪崩:

**举例：**缓存同一时间大面积的失效，这个时候又来的一波请求都到数据库上，导致数据库连接异常。

**解决办法：**可以给缓存设置不同的缓存时间，更新数据使用互斥锁或者通过双缓存在避免缓存雪崩。

### 缓存击穿:

**举例：**redis中存储的是热点数据，当高并发请求访问redis中热点数据的时候，如果redis中的数据过期了，会造成缓存击穿的现象，请求都打到了数据库上。

**解决办法：**使用互斥锁，只让一个请求去load DB，成功之后重新写缓存，其余请求没有获取到互斥锁，可以尝试重新获取缓存中的数据。。

### 缓存穿透:

**举例：**故意的去请求缓存中不存在的数据，导致请求都打到了数据库上，导致数据库异常。

**解决办法：**可以使用互斥锁或者无论是否取到结果都将结果存入缓存，还可以使用有效的机制（比如布隆过滤器）来拦截不合法的key值等。

### 数据库和缓存的双写一致性问题:

在高并发请求下很容易导致数据不一致的问题，如果你的业务需要保证数据的强一致性，那么建议不要使用缓存。在数据库中和缓存数据的删除或者写入过程中，如果有失败的情况，会导致数据的不一致。

### 解决办法:

- **双删延时的解决办法。**可以先删除缓存数据，然后再更新数据库数据，最后再隔固定的时间再次删除缓存。
- **更新数据库产生的binlog订阅（使用canal）。**将有变化的key记录下来，并且尝试去不断的去删除缓存（如果上次删除缓存失败）

## (4) redis的持久化方式有哪些？（重点掌握）

**答：**在redis中的介绍中，我们知道了redis不光是一个基于内存的缓存，同时还支持数据的持久化。在redis的配置文件中模块介绍中，我们可以设置redis的持久化方式。**redis的持久化方式有两种，即RDB和AOF的方式**，分别介绍如下：

9

## RDB（快照方式 snapshotting）（全量持久化）：

将当前内存中的数据快照写入磁盘，实现数据的持久化，恢复时可以将快照重新载入内存。

### 触发方式：

- **自动触发：**在配置文件中，可以配置执行了多少次save就自动触发自动持久化。
- **手动触发：**通过bgsave命令，在后台异步进行生成快照的操作，同时还可以响应客户端的请求。通过redis进程fork操作创建子进程，生成的快照由子进程负责，客户端请求只会在fork阶段被阻塞。

**快照恢复：**将备份文件 (dump.rdb) 移动到 redis 安装目录并启动服务，redis会自动加载快照文件数据到内存。但是，redis 服务器在载入 RDB 文件期间，会一直处于阻塞状态，直到载入工作完成为止。

### 优缺点分析：

- RDB持久化方式存在数据的丢失，因为其没有办法实现实时持久化。因为bgsave每次运行都要执行fork操作创建子进程，属于重量级操作，频繁执行成本过高，会影响系统性能。自动触发也存在丢失部分数据的情况。
- 在恢复大数据集时候，RDB方式相对于AOF要快。

## AOF（append-only-file）（增量持久化）：

在 redis配置文件的 APPEND ONLY MODE 中，可以设置AOF持久化。通过记录redis服务器所执行的写命令来记录数据库状态。恢复时可以将AOF文件载入内存，并且可以通过**redis-check-aof --fix** 进行修复AOF文件。

### AOF日志重写：

- AOF文件会随着服务器运行的时间越来越大，可以通过AOF重写来控制AOF文件的大小。
- AOF重写会首先读取数据库中现有的键值对状态，然后根据类型使用一条命令来替代前面对键值对操作的多条命令。
- 使用命令 bgrewriteaof 来实现AOF重写

### AOF重写缓存区：

redis 是单线程工作，当AOF文件较大时重写时间会比较长，在重写 AOF 期间，redis将长时间无法处理客户端请求。为了解决这个问题，可以将 AOF 重写程序放到子进程中执行，好处如下：

- 子进程进行 AOF 重写期间，服务器进程（父进程）可以继续处理其它客户端请求。
- 子进程带有父进程的数据副本，使用子进程而不是线程，可以在避免使用锁的情况下，保证数据的安全性。

### 子进程中AOF重写导致的问题：

- 子进程在进行 AOF 重写期间，服务器进程依然可以处理其它客户端请求，这就会导致数据库状态已经发生了改变，使得当前数据库数据状态和重写后的 AOF 文件中的数据不一致。
- 也就是出现了AOF文件和数据库中数据不一致的问题。

**数据状态不一致解决办法：**

- redis 服务器设置了一个 AOF 重写缓冲区。这个缓冲区在创建子进程后开始使用，当redis服务器执行一个客户端的写请求命令，之后将这个写命令也发送到 AOF 重写缓冲区。
- 当子进程完成 AOF 日志重写之后，给父进程发送信号，父进程接收此信号后，将 AOF 重写缓冲区的内容写到新的 AOF 文件中，保持数据的一致性。

9

**优缺点分析：**

- AOF文件可以做到秒级持久化，使用追加写的方式来写入，可读性强并且可以使用命令进行文件修复。
- 相比于RDB文件，同样数据下AOF文件体积要大。在redis负载较高时，秒级更新AOF文件会影响性能

**持久化策略选择：**

- AOF更安全，可将数据及时同步到文件中，但需要较多的磁盘IO，AOF文件尺寸较大，文件内容恢复相对较慢也更加完整。
- RDB持久化，安全性较差，它是正常时期数据备份及 master-slave数据同步的最佳手段，文件尺寸较小并且恢复速度较快。

**解析：**

持久化策略是redis相关知识点中一个相当常见的面试考察点，主要是两种持久化方式的比较和选择，希望大家可以有效掌握与理解。

**(5) redis数据的过期回收策略与内存淘汰机制：**

答：redis中的数据过期回收策略使用了定期删除和惰性删除相结合的方式。

- **定期删除：**

redis会每隔一定的时间去抽查一定量的数据判断其是否过期，过期则进行删除。

- **惰性删除：**

在获取一个key的时候，redis会检查这个key是否已经过期，若过期，则会进行删除操作。

**内存淘汰机制：**

在配置文件中，我们可以对内存淘汰机制进行配置。当内存使用达到最大值时，redis可以使用的**清除策略**如下：

- **volatile-lru**：利用LRU算法移除设置过过期时间的key (LRU:最近使用 Least Recently Used )
- **allkeys-lru**：利用LRU算法移除任何key
- **volatile-random**：移除设置过过期时间的随机key
- **allkeys-random**：移除随机key
- **volatile-ttl**：移除即将过期的key(minor TTL)
- **noeviction**：不移除任何key，只是返回一个写错误，默认选项

**(6) redis的主从复制机制：**

**答：**当项目比较大的时候，我们可以使用**主从架构Master/Slave机制**，Master 以写为主，Slave 以读为主，Master 主节点更新后根据配置，自动同步到从机Slave 节点。

主从复制的原理包括**旧版同步**和**命令传播**，主从复制的代价就是系统复制较重的时候会导致主从延迟，并且根据CAP理论，无法同时保证服务可用性和数据一致性。

9

**解析：**

主从复制属于较为复杂的知识点，限于篇幅，我们就不展开进行介绍了，有能力和兴趣的同学可以自行去学习其相关原理。在这里我们来简单看下分布式系统中著名的CAP理论吧。

**什么是CAP理论？**

**CAP理论是指** 当网络分区发生时，一致性和可用性不可能同时保证。

- **C: Consistent 一致性**
- **A: Availability 可用性**
- **P: Partition tolerance 分区容忍度**
- **网络分区：**分布式系统的节点往往都是分布在不同的机器上进行网络隔离开的，这意味着必然会有网络断开的风险，网络断开也就意味着发生了网络分区。
- **最终一致性：**Redis可以保证最终一致性，从节点会努力追赶主节点，最终从节点的状态会和主节点的状态将保持一致。

## (7) redis对事务支持：

**答：** redis对事务的支持主要可以概括如下：

- **隔离性：**redis 是单进程序的程序，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。所以redis的事务支持隔离性。
- **redis会将一个事务中的所有命令序列化**，然后按顺序执行。**redis不可能在一个事务的执行过程中插入执行另一个客户端发出的请求**。可以保证Redis将这些命令作为一个单独的隔离操作执行。

redis操作事务的相关命令如下所示：

- **MULTI：**标记一个事务块的开始。
- **EXEC：**执行所有事务块内的命令。
- **DISCARD：**取消事务，放弃执行事务块内的所有命令。
- **UNWATCH：**取消 WATCH 命令对所有 key 的监视。
- **WATCH key [key ...]：**监视一个(或多个) key，如果在事务执行之前这个(或这些) key 被其他命令所改动，那么事务将被打断。

需要注意的是**redis的事务不支持回滚操作**，redis以 MULTI 开始一个事务，然后将多个命令入队到事务中，最后由 EXEC 命令触发事务，一并执行事务中的所有命令。只有当被调用的redis命令有语法错误时，这条命令才会执行失败，或者对某个键执行不符合其数据类型的操作，但是应该在将命令入队列的时候就应该并且能够发现这些问题，所以redis的事务不支持进行回滚操作。

## (8) Redis中其余重要知识点：

**答：**这篇文章，我们主要针对常见的基础知识点做了一个简单的介绍与学习，有能力或者有需求的同学可以继续深入学习如下的重要知识点：

- Redis实现分布式锁，如何加锁以及解锁
- Redis集群中的哨兵模式，主从复制等

## 总结：

本小节中我们主要针对面试中常见的redis相关知识点进行了交流与学习，对于需要重点掌握的知识点已经做了标记，希望大家可以重点掌握。下一小节中，我们将对Kafka进行简单的交流学习。

限于作者水平，文章中难免会有不妥之处。大家在学习过程中遇到我没有表达清楚或者表述有误的地方，欢迎随时在文章下边指出，我会及时关注，随时改正。另外，大家有任何话题都可以在下边留言，我们一起交流探讨。

讨论

评论



刘畅201904211606816

1#

打卡

发表于 2020-01-06 22:23:13

赞(0) 回复(0)



柳杰201905011049420

2#

看懂了吗

发表于 2020-01-15 17:43:10

赞(0) 回复(0)



牧水s N

3#

打卡

发表于 2020-02-13 18:22:49

赞(0) 回复(0)




越努力，越幸运~ N

4#

滴滴，打卡

发表于 2020-02-20 21:44:48

赞(0) 回复(0)



牛客874981128号

5#

为什么感觉好多知识点都没讲？ 处理并发呢，分布式锁呢

发表于 2020-02-28 17:14:29

赞(0) 回复(1)

我是祖国的花朵 N 作者： 有啥想了解的，欢迎在评论区交流。

2020-02-28 17:19:12

赞(0) 回复(0)

请输入你的观点

回复



我是祖国的花朵 N 作者

6#

关于分布式锁，为了使得加锁操作具有原子性，不可以使用多条命令来完成，我们可以使用带多个参数的set命令来完成，如下所示：`jedis.set(String key, String value, String nxxx, String expx, int time)`

第一个为key，我们使用key来当锁，因为key是唯一的。

第二个为value，我们传的是requestId，通过给value赋值为requestId，我们就知道这把锁是哪个请求加的了，在解锁的时候就可以有依据。

第三个为nxxx，这个参数我们填的是NX，意思是SET IF NOT EXIST，即当key不存在时，我们进行set操作；若key已经存在，则不做任何操作；

第四个为expx，这个参数我们传的是PX，意思是我们要给这个key加一个过期的设置，具体时间由第五个参数决定。

第五个为time，与第四个参数相呼应，代表key的过期时间。

发表于 2020-02-29 21:23:54

赞(4) 回复(0)

9



奋斗中的战斗机

7#

请问老师：redis数据丢失怎么办？

发表于 2020-03-12 15:07:04

赞(0) 回复(1)

我是祖国的花朵 (作者)： 你好，如果没有做持久化，那么重启是会导致数据丢失的。如果接受不了任何数据的丢失，我觉得可以每次更新都写aof来搞定，但是这样性能会受损。

2020-03-15 17:54:07

赞(0) 回复(0)

请输入你的观点

回复



四月hope

8#

请问有关于操作系统相关更新的打算吗？这方面比较欠缺，网上又太杂，特别是内核态和用户态如何转换，网上的感觉不太对，看书又看不太懂

发表于 2020-04-12 15:06:56

赞(0) 回复(1)

我是祖国的花朵 (作者)： 你好，按照我对校招和社招的了解，这块知识点考察是很少的，算不上高频知识点，说白了，研究操作系统还不如研究两个算法题目的有价值。所以，操作系统相关的知识点暂时没有更新的打算哈~

2020-04-12 21:13:45

赞(0) 回复(0)

请输入你的观点

回复



DGQ\_00


9#

老师，能不能推荐一本redis的书？不知道看哪本比较好

发表于 2020-04-19 14:37:51

赞(0) 回复(1)



我是祖国的花朵  作者： 黄健宏老师的redis设计与实现!!! 建议深入研究，虽然redis版本较低，但是绝对经典，好书。看了这个之后，再去看看别的redis新版本特性。

2020-04-21 11:07:28

赞(0) 回复(0)

请输入你的观点

9

回复