

之前遇到过的几个Redis相关面试题解析

来自： 小6互联网求职面试

 马听

2024年01月01日 11:44



扫码加入
查看更多优质内容

去年（哈哈）分享了 [\[Redis专场\] 分享几个之前遇到过的Redis相关面试题](#)

当时没写解析，这里就来分享一下自己的思路。

1 Redis主从复制原理

- a. 从服务器像主服务器发送slaveof 或者 replicaof的连接请求，主库通过bgsave fork子进程备份全量rdb，期间主服务器接收的写请求会缓存起来。备份完成之后将rdb，缓存的命令发给从服务器，从服务器丢弃现有数据，恢复rdb快照，恢复缓存命令。
- b. 复制期间，主服务器会发送写操作同步给从服务器
- c.增量同步，发生在主从断开，从又连主的情况下。主服务器维护back_log环形缓冲区，主服务器处理写请求时，同时会复制到该缓冲区。主从断开，从连上主之后，会发送主机id和偏移量与主进行对比，然后将对比得出的增量部分发送给从服务器。如果断开时的从服务器偏移量被覆盖掉，那只能进行全量同步了。

（主从复制原理这个问题的解析由星友“好起来了”分享）

2 Redis 如何做备份

常见的备份方式为RDB和AOF

RDB是创建数据快照，这个快照包括Redis里所有的数据

AOF是记录每次写入操作

对于 Redis 的备份，建议如下：

- 数据不能丢失时，可以采用RDB和 AOF 的混合使用
- 如果允许分钟级别的数据丢失，可以只使用 RDB，可以通过配置save参数的方式，或者定时任务去触发bgsave
- 如果有从实例，也优先考虑在从实例上进行备份。

3 Redis 数据淘汰策略有哪些？

当Redis内存使用达到设置的阈值时，Redis会挑选部分key删除，从而释放内存空间。
这个时候最大内存策略就会发挥作用。

Redis的8种最大内存策略：

- noeviction，达到内存限制时，写入操作会报错。不会删除Redis里现有的key
- allkeys-lru，保留最近使用的key；删除最近最少使用 (LRU) 的key
- allkeys-lfu，保留常用的key；删除最不常用 (LFU) 的key
- volatile-lru：删除最近最少使用的并且设置了过期时间的key
- volatile-lfu，删除最不常用的并且设置了过期时间的key
- allkeys-random，随机删除一些key，为添加的新数据腾出空间
- volatile-random，随机移除设置了过期时间的key
- volatile-ttl，删除设置了过期时间和最短剩余时间的key

4 Redis 怎么查找并优化 bigkey 和 hotkey？

找到Redis大key的方法：

- a、通过编写程序，采用扫描的方式找到大key
- b、通过rdbtools分析RDB文件找到大key
- c、通过Redis自带客户端找到大key：redis-cli --bigkeys

大key优化方式：

- a、有些 Bigkey 业务不需要使用了，因此可以考虑删除掉，但是要注意的是：如果直接 del，可能会阻塞 Redis 服务。大致有下面几种处理办法：
如果 key 类型为 string，则直接删除；

如果 key 类型为 hash、list、set、sorted set，使用 hscan 命令，每次获取部分（例如100个）field-value，再利用 hdel 删除每个 field；

Redis 在4.0 版本支持 lazy delete free 的模式，删除 Bigkey 不会阻塞 Redis。

b、控制大小或拆分Bigkey

c、更换数据库：其实有些场景下，使用 Redis 并不是最优的选择，比如长文本，如果放在 Redis 中，很可能就是一个 Bigkey，因此建议不要存入 Redis，用文档型数据库 MongoDB 代替或者直接缓存到 CDN 上。

找到热key的办法：

a、通过redis-faina（原理就是运行monitor，记录短时间执行的所有命令，并进行统计分析）

b、使用客户端工具：redis-cli --hotkeys

热key优化建议

a、拆分、迁移或复制热点key

b、使用读写分离

c、本地缓存加通知机制（将 Hotkey 放在业务端的本地缓存中，然后使用发布订阅机制保证业务端本地缓存与 Redis 数据一致。）

5 redis cluster 某个节点挂了，内部是怎么工作的？用的什么算法？

集群中的每个节点都会定期的向集群中其他节点发送 ping 消息，以此交换各个节点状态信息。

节点间会互相探测，如果节点1和节点2探测的时间超过node-timeout

则节点1会把节点2标记为主观下线

如果超过半数以上的节点认为节点2主观下线。就判断为节点客观下线

就开始选举从节点了

首先看从节点和主节点之间的断开时间

如果断开时间太久，就会取消选举资格。

具体时间是两个参数的乘积控制

cluster-node-timeout * cluster-slave-validity-factor

但是有一个特殊情况，比如cluster-slave-validity-factor设置为0，那不管主从断开多久，从都会参与选举。

在有选举资格的实例中，再判断偏移量，偏移量最大的slave，优先选为新主。

选举完成之后

这个新主节点执行slaveof no one变成master节点

再把故障节点负责的槽分配给这个新主节点

向集群中其他节点广播Pong消息，表明已经完成切换

6 redis cluster 某个槽的内存特别大，可能什么原因？

a、这个槽存在大key

b、使用了哈希标签，比如一个key是{user001}.name，另外一个key是{user001}.email，那么这两个key会分配在一个槽里面（只要{}中的内容一样）

7 Redis 之前的版本为什么是单线程？Redis6 多线程原理？是否会存在线程并发安全问题？

a、Redis 之前的版本为什么是单线程？

使用单线程使得Redis的开发和维护更简单

即使使用单线程也能并发处理多个客户端的请求，主要使用的是IO多路复用和非阻塞IO

对于Redis系统来说，主要的性能瓶颈是内存或者网络，并非CPU。

尽管多线程，可以增加吞吐量，但是多线程就涉及到共享某个资源的情况，就需要考虑锁。

b、Redis6 多线程原理

首先，当有客户端请求时，会和Redis实例建立Socket连接，并把Socket放入到全局等待队列中，随后，主线程通过轮询的方式把Socket连接分配给IO线程组。

主线程一旦把Socket分配给IO线程，就会进入阻塞状态，等待IO线程完成客户端请求读取和解析。因为有多多个IO线程并行处理，所以这个过程会很快

等到所有IO线程解析完请求，主线程会继续以单线程的方式执行这些命令操作。这样，也能保证命令操作的原子性。

当主线程执行完请求操作后，会把结果写入缓冲池，然后，主线程会阻塞，等待IO线程组把这些结果写回到Socket中。

也会有多个并发的IO线程来回写Socket，所以速度也很快，等到IO线程回写Socket完毕，主线程会清空全局队列，等待客户端的后续请求。

c、是否会存在线程并发安全问题？

可以理解为Redis 6.0是网络层的多线程，实际在Redis里面执行命令操作，还是主线程以单线程的方式去执行，也能保证命令的原子性。

8 Redis Cluster 内部是怎样找到对应的 key

写入key时，都是根据key做哈希运算定位到哪个槽的

但是有时候，要查询的key不在我们连接的节点上

在客户端和集群建立连接后，实例就会把哈希槽的分片信息发送给客户端。

而每个实例会把自己的哈希槽信息发送给集群内其他的实例，

这样，每个实例就会知道所有哈希槽和实例的对应关系了。