雪崩问题

Key的过期淘汰机制

Redis可以对存储在Redis中的缓存数据设置过期时间，比如我们获取的短信验证码一般十分钟过期，我们这时候就需要在验证码存进Redis时添加一个key的过期时间，但是这里有一个需要格外注意的问题就是：并非key过期时间到了就一定会被Redis给删除。

定期删除

Redis 默认是每隔 100ms 就随机抽取一些设置了过期时间的 Key，检查其是否过期，如果过期就删除。为什么是随机抽取而不是检查所有key？因为你如果设置的key成千上万，每100毫秒都将所有存在的key检查一遍，会给CPU带来比较大的压力。

惰性删除

定期删除由于是随机抽取可能会导致很多过期 Key 到了过期时间并没有被删除。所以用户在从缓存获取数据的时候，redis会检查这个key是否过期了，如果过期就删除这个key。这时候就会在查询的时候将过期key从缓存中清除。

内存淘汰机制

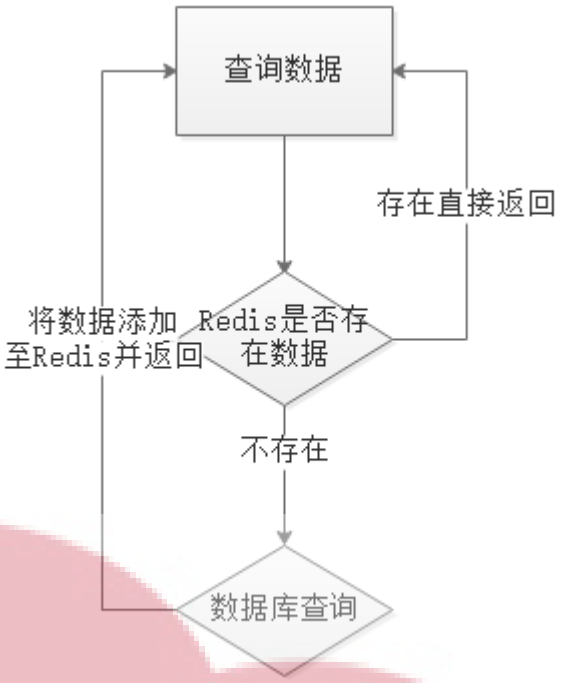
仅仅使用定期删除 + 惰性删除机制还是会留下一个严重的隐患：如果定期删除留下了很多已经过期的key，而且用户长时间都没有使用过这些过期key，导致过期key无法被惰性删除，从而导致过期key一直堆积在内存里，最终造成Redis内存块被消耗殆尽。那这个问题如何解决呢？这个时候Redis内存淘汰机制应运而生了。Redis内存淘汰机制提供了6种数据淘汰策略：

- `volatile-lru`：从已设置过期时间的数据集中挑选最近最少使用的数据淘汰。
- `volatile-ttl`：从已设置过期时间的数据集中挑选将要过期的数据淘汰。
- `volatile-random`：从已设置过期时间的数据集中任意选择数据淘汰。
- `allkeys-lru`：当内存不足以容纳新写入数据时移除最近最少使用的key。
- `allkeys-random`：从数据集中任意选择数据淘汰。
- `no-enviction（默认）`：当内存不足以容纳新写入数据时，新写入操作会报错。

一般情况下，推荐使用`volatile-lru`策略，对于配置信息等重要数据，不应该设置过期时间，这样Redis就永远不会淘汰这些重要数据。对于一般数据可以添加一个缓存时间，当数据失效则请求会从DB中获取并重新存入Redis中。

缓存击穿

首先我们来看下请求是如何取到数据的：当接收到用户请求，首先先尝试从Redis缓存中获取到数据，如果缓存中能取到数据则直接返回结果，当缓存中不存在数据时从DB获取数据，如果数据库成功取到数据，则更新Redis，然后返回数据



定义：高并发的情况下，某个热门key突然过期，导致大量请求在Redis未找到缓存数据，进而全部去访问DB请求数据，引起DB压力瞬间增大。

解决方案：缓存击穿的情况下一般不容易造成DB的宕机，只是会造成对DB的周期性压力。对缓存击穿的解决方案一般可以这样：

- Redis中的数据不设置过期时间，然后在缓存的对象上添加一个属性标识过期时间，每次获取到数据时，校验对象中的过期时间属性，如果数据即将过期，则异步发起一个线程主动更新缓存中的数据。但是这种方案可能会导致有些请求会拿到过期的值，就得看业务能否可以接受，
- 如果要求数据必须是新数据，则最好的方案则为热点数据设置为永不过期，然后加一个互斥锁保证缓存的单线程写。