

35 | 缓存问题：怎么解决缓存穿透、击穿和雪崩问题？

2023-9-8 邓明



你好，我是大明。今天我们再来聊一个缓存中的热门面试话题：怎么解决缓存穿透、击穿和雪崩问题。

这个问题之所以常见，是因为你在使用缓存的过程中一不小心就会遇到它们。比如说在前面的课程里面我给你讲过，如果缓存崩溃，那么大量请求就会落到数据库上，直接把数据库压垮。然而很多新手在刚接触缓存的时候完全意识不到这个问题，只有在出了线上故障之后才会考虑缓存崩溃的事情。因此，面试官就倾向于在面试的时候确认你是否会解决这一类的问题。

这一节课，我就带你综合分析这三种情况，以及其他可能出现的缓存问题。

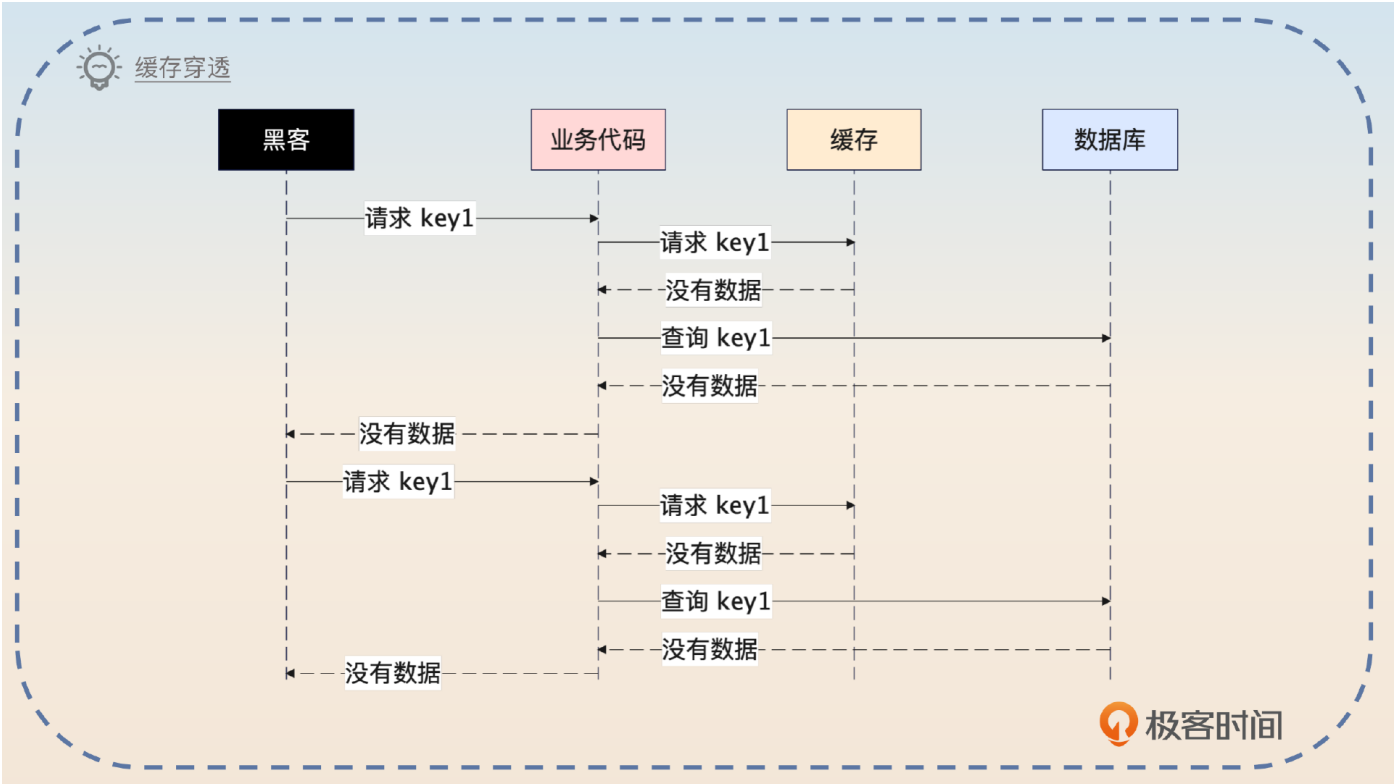
不得不说的是，缓存穿透、击穿和雪崩是三个很容易搞混的概念，尤其是缓存穿透和击穿。

缓存穿透

缓存穿透是指**数据既不在缓存中，也不在数据库中。**

最常见的场景就是有攻击者伪造了大量的请求，请求某个不存在的数据。这会造成两个后果。

缓存里面没有对应的数据，所以查询会落到数据库上。
数据库也没有数据，所以没有办法回写缓存，下一次请求同样的数据，请求还是会落到数据库上。

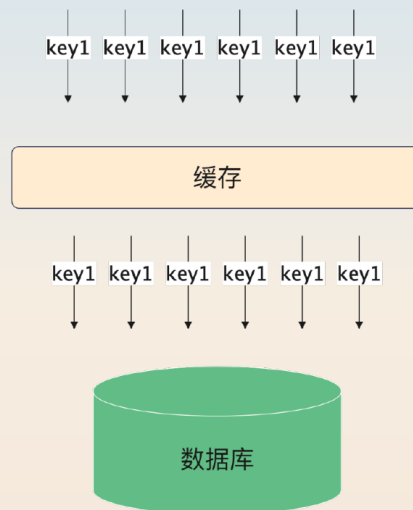


如果你没有在服务层面上采用熔断、限流之类的措施，那么数据库就可能崩溃。

缓存击穿

缓存击穿是指**数据不在缓存中**，导致请求落到了数据库上。注意，数据这个时候在数据库中是存在的，所以我们可以预计，查询到数据库中的数据之后就会回写缓存。

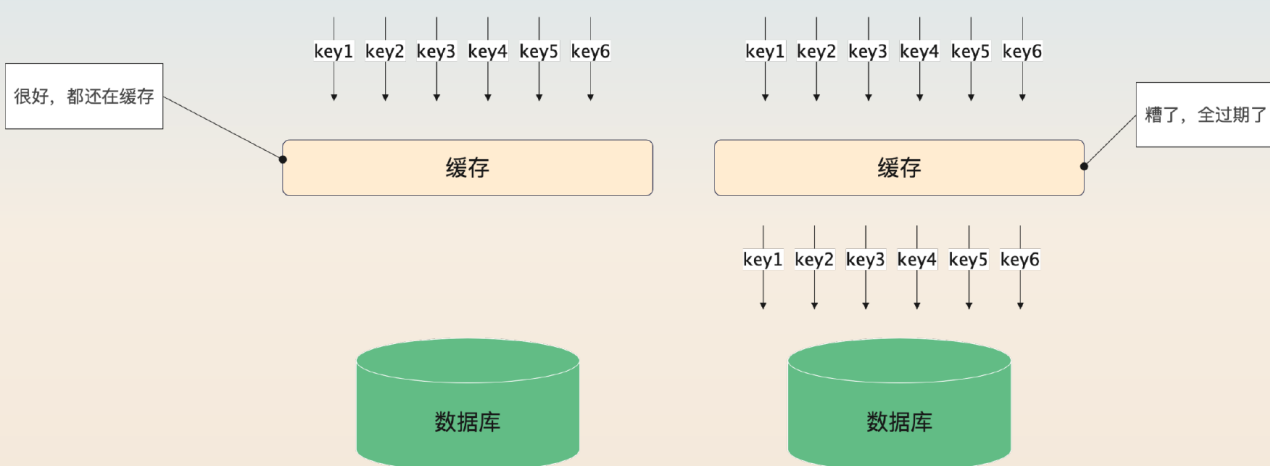
这看起来并没有问题。但如果请求的是热点数据呢？比如说同一时刻有几百个人请求某个大博主的数据，这些请求都没有命中缓存，那么几百个查询请求都会落到数据库上。



因此，如果请求的数据并不是什么热点数据，那么击穿也没有什么问题，它就是普通的缓存未命中而已。

缓存雪崩

缓存雪崩是指缓存里大量数据在同一时刻过期，导致请求都落到了数据库上。



缓存雪崩基本上都是因为一次性加载了很多数据到缓存中，并且都设置为同一个过期时间。比如说在应用启动的时候，提前从数据库里查询数据，然后放到缓存里面，这样这一批数据

就会在同一时刻过期。又比如榜单数据计算好了之后加载到缓存里，都是同一个过期时间，导致这一批榜单数据同一时间过期。

面试准备

在面试前，你需要在公司里面收集一些信息。

比如：

你用的 Redis 是如何部署的？用的是 Redis Cluster 还是 Redis Sentinel？

你们公司有没有出现过 Redis 崩溃的问题？如果有，是什么原因引发的？

公司有没有出现过Redis连不上的问题？如果有，后面有没有使用什么方案来容错？

你有没有遇到过缓存穿透、击穿或者雪崩等问题？如果有，你当时是怎么解决的，有没有可以改进的点？

公司有没有保护数据库的措施？比如说防止缓存失效，导致数据库不堪重负直接崩溃。

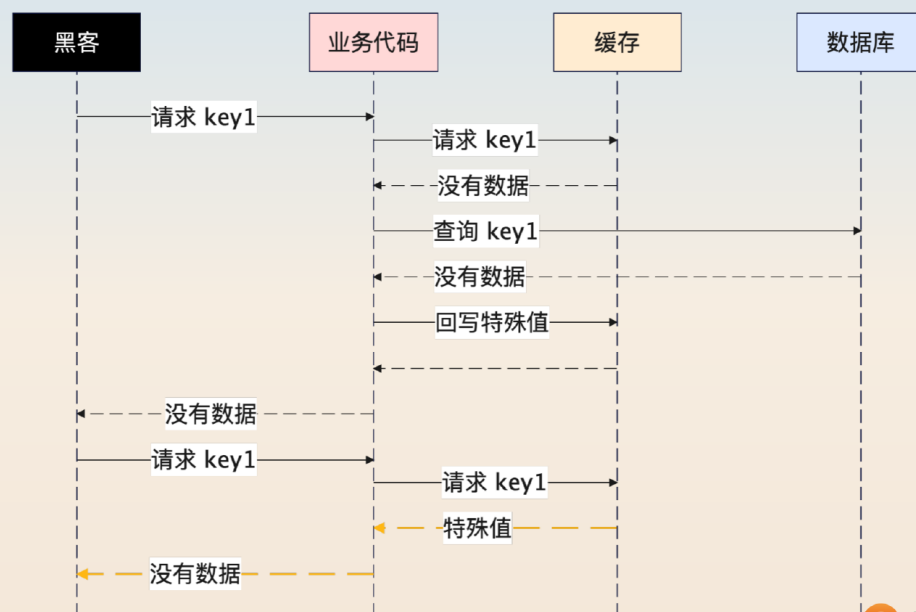
你可以在简历中或者自我介绍的时候强调自己解决过很多缓存问题，包括缓存穿透、击穿、雪崩等问题。面试官多半会对这些点感兴趣，那么你就可以趁机把你前面准备的各种案例说出来。同时，你也可以考虑把各种缓存问题融合到自己的高可用方案里面，作为其中的一个环节。

解决缓存穿透

缓存穿透是因为数据本身不存在而引起的，所以我们就想办法在确认数据不存在之后，避免下一次查询再次落到数据库上。这有两种解决思路。

回写特殊值

第一种思路是在缓存未命中，而且数据库里也没有的情况下，往缓存里写入一个特殊的值。这个值就是标记数据不存在，那么下一次查询请求过来的时候，看到这个特殊值，就知道没有必要再去数据库里查询了。



你可以直接介绍这个方案。

第一种思路是回写特殊值。也就是在第一次查询发现数据库里都没有数据的时候，直接写入一个特殊值。那么下一次查询过来的时候，看到缓存里的特殊值，就知道没有数据，这时候直接返回就可以了。在这种设计之下，数据库只需要支撑住第一次请求就可以。

但是这个方案也是有缺点的。

如果攻击者每次都用不同的且都不存在的 key 来请求数据，那么这种措施毫无效果。并且，因为要回写特殊值，那么这些不存在的 key 都会有特殊值，浪费了 Redis 的内存。这可能会进一步引起另外一个问题，就是 Redis 在内存不足，执行淘汰的时候，把其他有用的数据淘汰掉。

这时候你就可以引出下一个点了，考虑使用布隆过滤器。

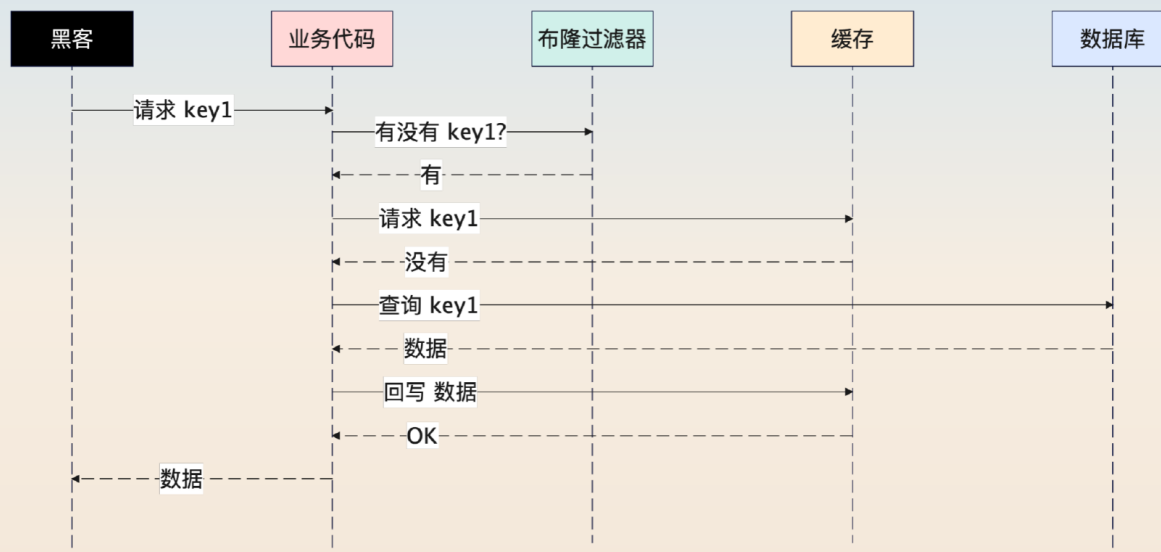
布隆过滤器

既然缓存穿透是因为数据不存在，那么提前用布隆过滤器判断一下不就可以了嘛？

正常请求一个 key 的流程如下：



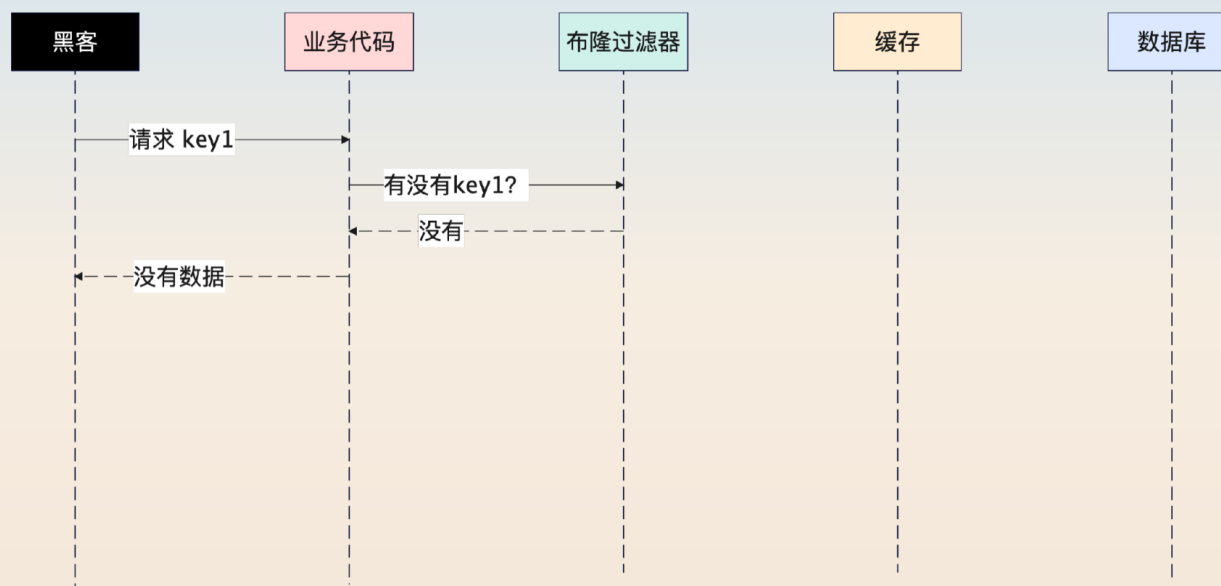
布隆过滤器正常流程



如果请求一个 key 不存在，那么布隆过滤器会直接说数据不存在，那么就没必要继续往下查询了。



布隆过滤器 key 不存在



你先简要介绍这个流程。

使用布隆过滤器的流程是业务代码收到请求之后，要先问一下布隆过滤器有没有这个 key。如果说没有，那就不用继续往后执行了。如果布隆过滤器说有

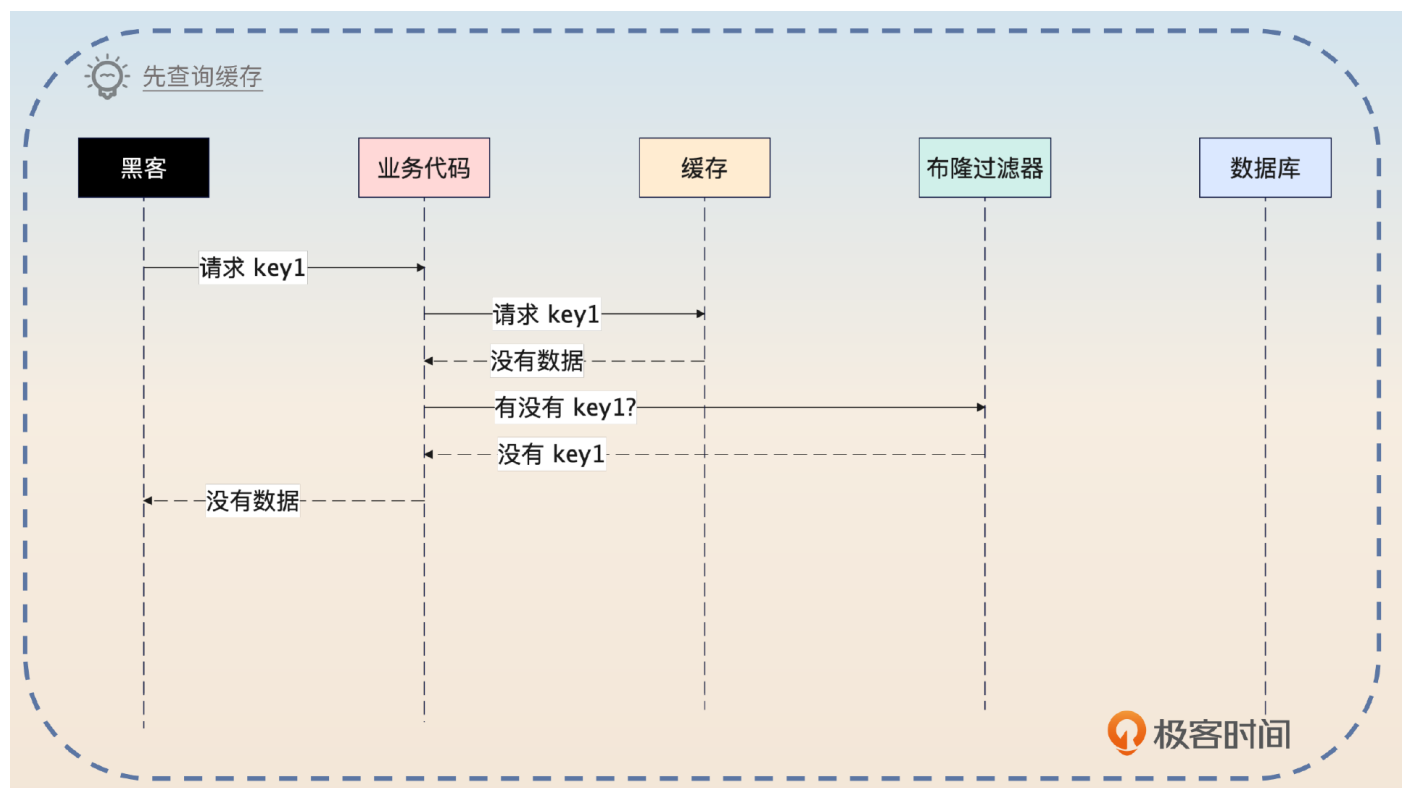
那么就继续往后执行，去查询缓存和数据库，并且在查询到了数据的时候，回写到缓存里面。

然后你要记得介绍假阳性的问题。

但是布隆过滤器本身存在假阳性的问题，所以当攻击者请求一个不存在的 key 的时候，布隆过滤器可能会返回数据存在的假阳性响应。在这种情况下，业务代码依旧会去查询缓存和数据库。不过这个不需要担心，因为假阳性的概率是很低的。假如说假阳性概率是万分之一，那么就算攻击的并发有百万，也只有 100 个查询请求会落到数据库上，这一点查询请求就是毛毛雨了。

这时候，你可以补充一个变种，也就是先查询缓存，缓存中没有数据的时候，再去问布隆过滤器。

也可以考虑先查询缓存，当缓存没有数据的时候，再去查询布隆过滤器。如果布隆过滤器说有数据，再去查询数据库。



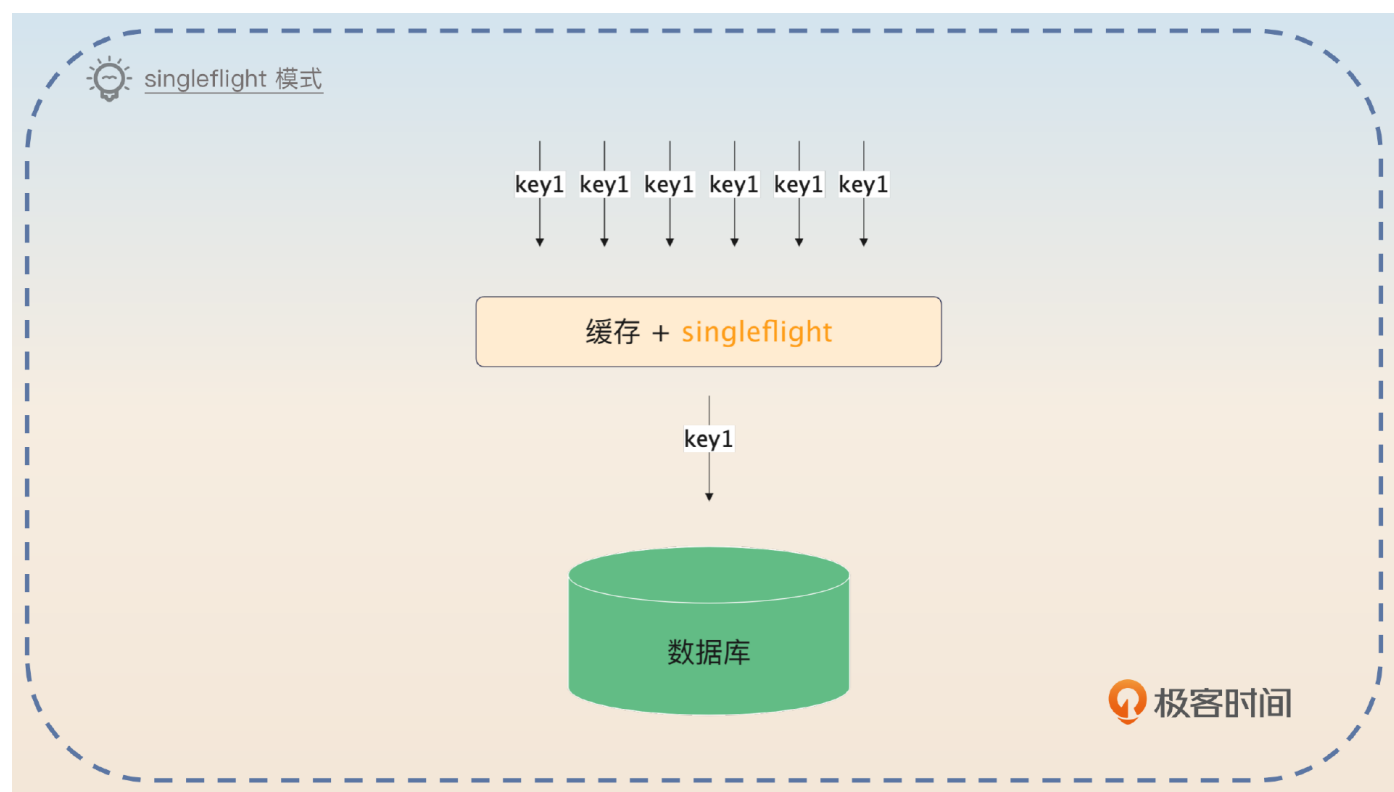
然后你对比两者。

这两种模式没有太大的差别。先查询布隆过滤器，保护效果会更好，也就是提前挡住了非法请求。而先查询缓存，对正常请求更加友好，因为正常请求大概率命中缓存，直接返回数据，也就不用查询布隆过滤器了。

不过如果布隆过滤器也是在 Redis 的基础上实现的，两者就基本上没什么区别了。

解决缓存击穿

解决缓存击穿是很容易的，只需要用到我们在缓存模式里面提到的 singleflight 模式。也就是说，就算是一个热点数据，当几百个请求缓存未命中的时候，在 singleflight 模式之下，也只有一个请求会真的去查询数据，剩下的都在等着这个请求查询回来的结果。



解决缓存雪崩

缓存雪崩也很容易解决。在前置知识里我提到过之所以有雪崩，就是因为一次性加载了一大批数据放到了缓存里面，并且设置了同样的过期时间。

那么解决思路自然就有两个，一个是不允许一次性加载一大批数据到缓存，而这显然不现实，因为批量加载属于业务要求；另外一个思路就是设置不同的过期时间。

最简单的思路，就是在过期时间的基础上加一个偏移量。

要解决缓存雪崩，就是在数据批量加载到缓存的场景中在过期时间的基础上加上一个随机量。比如说，我预计这一批数据的过期时间是 15 分钟。那么我就在设置每一条数据的过期时间的时候，在 15 分钟的基础上加上一个 0~180 秒的偏移量。那么这一批数据就不会在同一时刻过期，也就不存在缓存雪崩的问题了。

这时候，面试官可能会问你这个偏移量的范围怎么确定。比如说在你的回答里面，你说的是 0~180 秒的偏移量，那么 0~10 秒的偏移量行不行？

这时候你要抓住核心，**偏移量要跟过期时间成正比，不能过低或者过高。**

这个随机偏移量应该和过期时间成正比。比如说如果过期时间是 15 分钟，那么随机偏移量在 0~180 秒都可以。如果数据量不多，那么 0~60 秒也可以。而如果过期时间很长，比如说 4 个小时，也可以把偏移量控制在 0~10 分钟。如果过期时间很短，比如说只有 10s，这个时候偏移量就只能在 0~3 秒内了。

当然，偏移量这个东西，除了随机生成，也可以有别的算法。比如说第一条数据加上 1 秒偏移量，第二条数据加上 2 秒，以此类推。

限流

此外你还要注意一点，就是引发缓存穿透、击穿和雪崩等问题的一个关键是**有很多请求落到了数据库上**。所以一个最简单的办法就是限制住这些请求，不让它们落到数据库上。所以，限流就可以用在这些场景中。

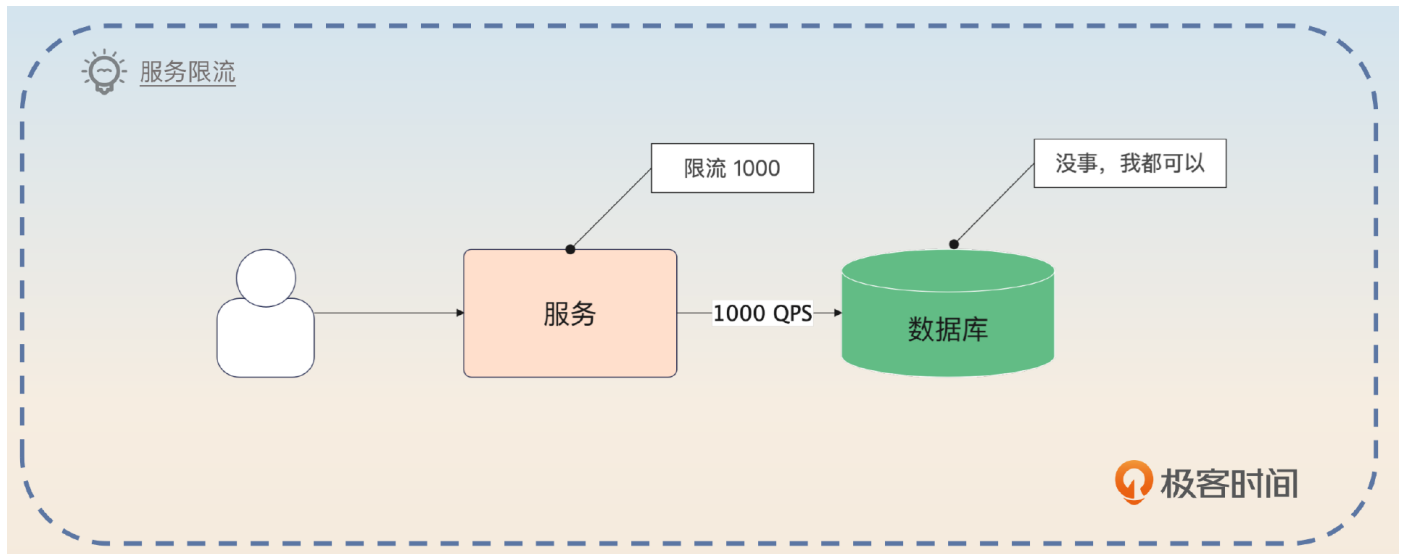
你可以回答限流的基本思路。

缓存穿透和击穿只有在高并发下才会成为一个问题，所以一个很自然的想法就是使用限流。限流可以考虑在两个地方使用：服务层面和数据库层面。

而在服务层面上限流，是有一个隐含假设的，你可以强调一下这个假设，就是**数据库撑得住**。

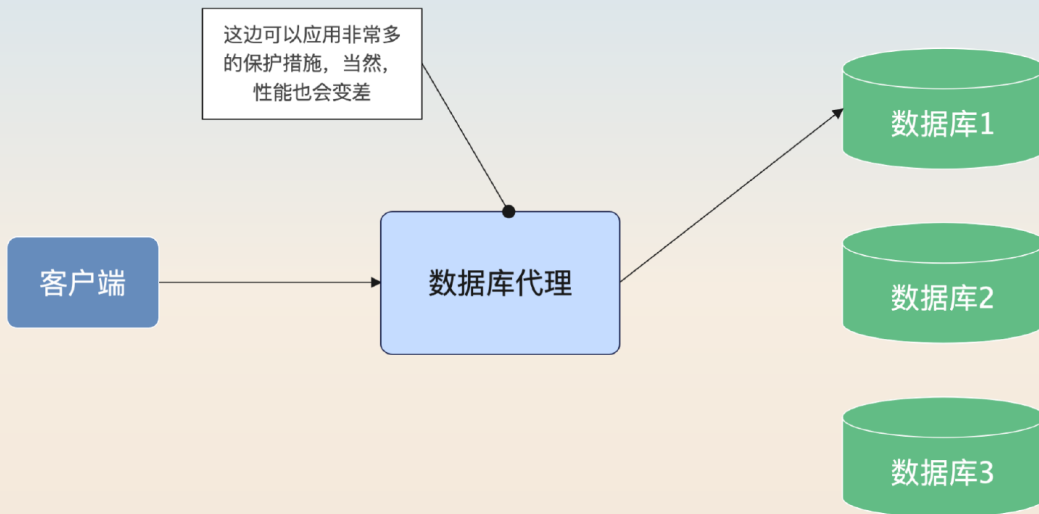
在服务层面上限流的时候，要保证只要 QPS 没有超过这个数值，就算所有的请求都没有命中缓存，直接落到了数据库上，数据库也能撑住这个流量。但是有

些时候也难以保证。或者说大多数人在考虑限流阈值，包括使用压测来确定限流阈值的时候，都已经把命中缓存这种情况考虑进去了。所以需要进一步考虑数据库层面的限流了。



我在降级那节课提到，为了防止数据库崩溃，最好在数据库访问上加一些限流措施。显然，就算数据库没有崩溃，这个限流还是可以保护数据库免遭大流量的冲击。

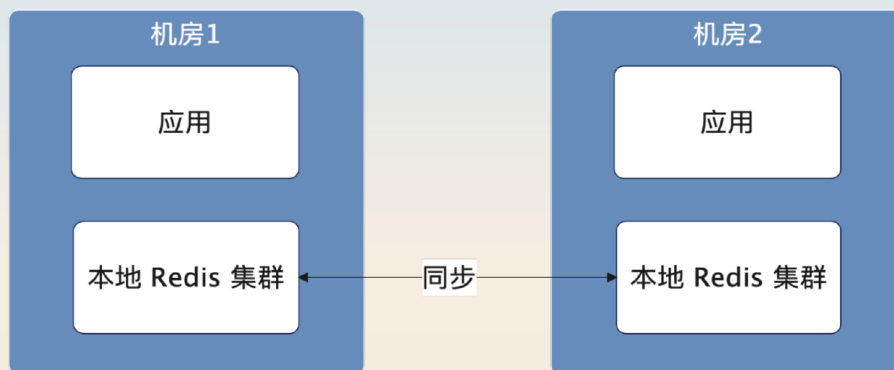
数据库层面的限流总的来说是必不可少的。不管是缓存崩溃，还是穿透或者击穿，限流都能保护住数据库。如果使用了数据库代理，并且这些代理支持限流，那么就可以直接在代理上做限流。如果没有使用代理或者代理不支持，那么就可以考虑在 ORM 上做。



亮点方案

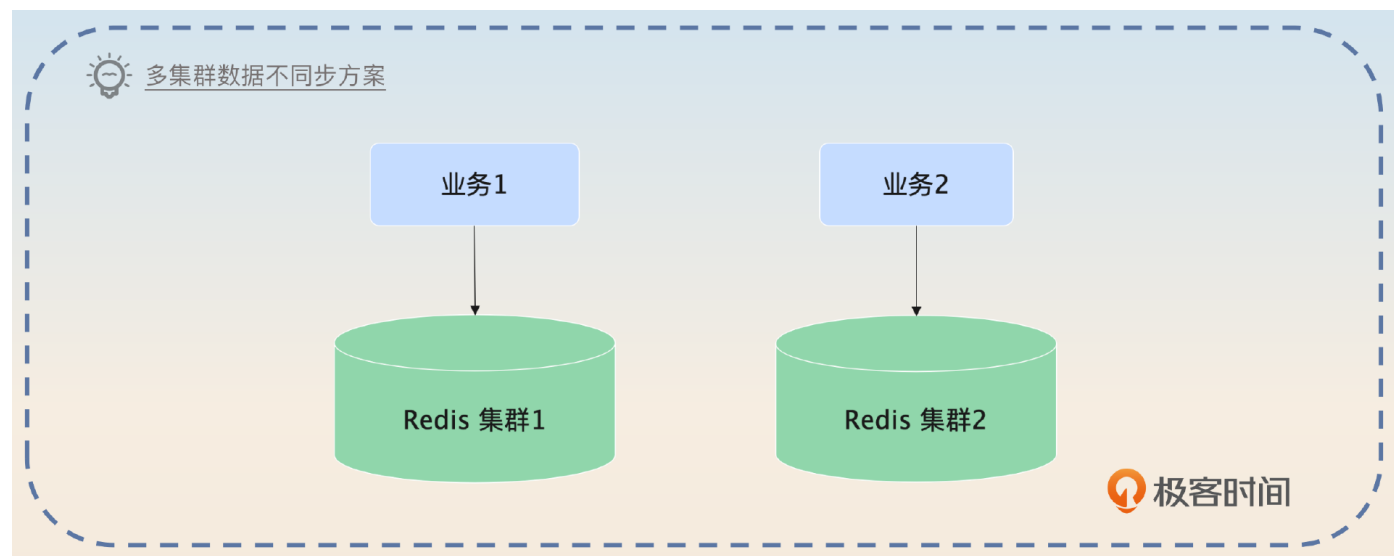
到这里，你已经知道不管是缓存穿透、击穿还是雪崩，归根结底就是请求都落到了数据库上。除了这三个异常，Redis 本身也有可能崩溃，又或者因为网络问题连不上这个集群。那么今天我给你准备的亮点方案——**集群互为备份**，就可以很好地解决这个问题。

很多大厂会用一些异地多活的方案，就是使用两个 Redis 集群，然后两个集群之间要保持数据同步。那么其中一个 Redis 集群崩溃的时候，就可以用另外一个 Redis 集群。



但是，这个方案太高端了，不在大厂的话很难接触到。所以我给你准备一个比较低端但是更加容易落地的方案。

这个方案的思路还是用两个或者多个 Redis 集群，但是你不会让这些集群之间保持数据同步。比如说你可以在两个云服务厂商上购买两个不同的 Redis 服务，然后尽可能让核心业务访问不同的集群。



那么你可以这样介绍这个思路。

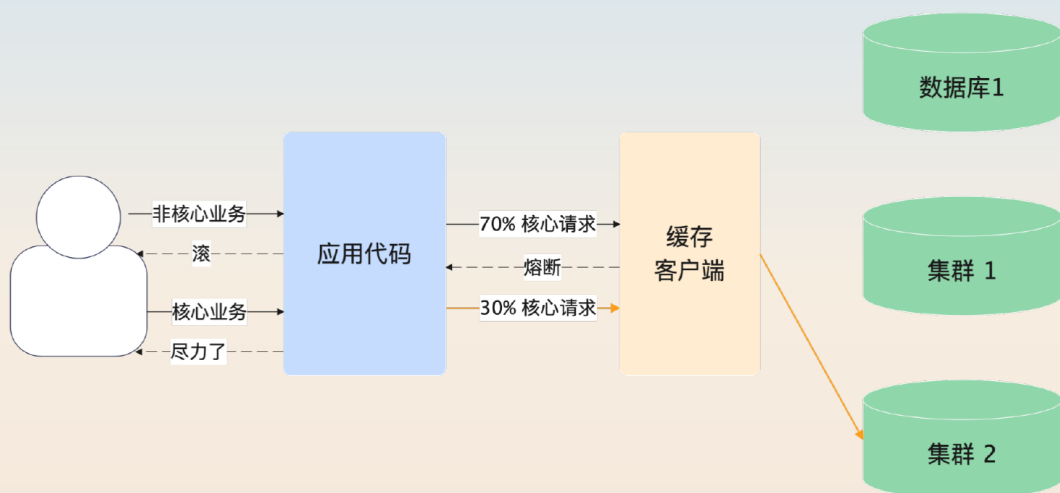
假设说我有两个业务，那么我准备两个 Redis 集群，业务1 主要用集群 1，集群 2 作为备份。业务 2 主要使用集群 2，集群 1 作为备份。

具体思路是这样的：

第一，业务 1 会和集群 1 保持心跳。当发现连不上 Redis 之后，就可以执行容错方案，这个时候业务 1 会保持和集群 1 的心跳。

第二，触发容错之后，业务 1 根据流量价值分成两部分。对于非核心业务来说，直接触发熔断，不会查询集群 2，也不会查询数据库，这是舍小保大。对于核心业务来说，按照预先设置的流量比例，查询集群 2，并回查数据库，其余请求一样熔断。如果当前流量比例查询集群 2 没有引起任何的问题，数据库也没有问题，那么就增大流量比例。

第三，当集群 1 重新恢复心跳之后，业务 1 还是逐步把集群 2 上的流量转发到集群 1 上。



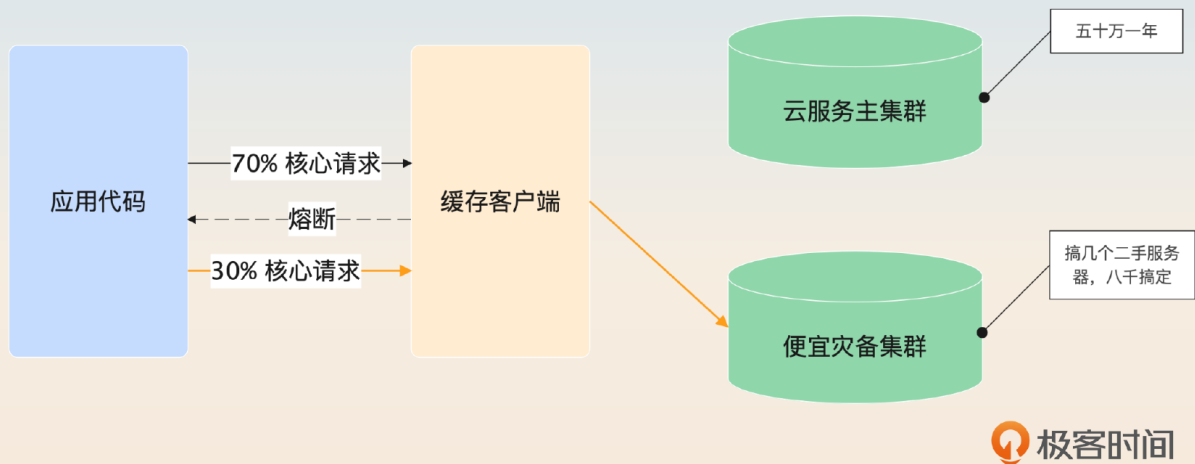
你可以进一步总结这个思路的要点，就是**渐进式**。

在触发容错之后，没有立刻把全部流量转发到集群 2 上，是因为担心集群 2 会撑不住，所以要逐步转发流量，每次转发之后发现没问题就可以调大比例。转发回集群 1 也是这样，因为和集群 1 刚恢复通信的时候，集群 1 上面什么数据都没有，而这个时候集群 2 还能用，所以不着急立刻转发回来，先小规模流量重建集群 1 上的数据。

那么为什么互为备份可行呢？因为正常 Redis 集群都有很大的余量，在遇到问题的时候互为备份顶一下就可以。当技术人员发现问题之后，会紧急采购新的 Redis 服务，或者部署新的集群接替集群 1。所以集群 2 大概率会在高负载状态下运行一段时间。

如果你面试的是比较小型的公司，对成本比较敏感，你就可以补充一个变种，关键词是凑合用便宜货。

如果觉得两个 Redis 集群服务太贵，那么也有一个低成本方案，公司可以自己部署一个小规模的 Redis 集群，甚至单机 Redis 作为所有业务的备份。这个 Redis 集群不要求高可用，对它的唯一要求就是撑住我线上集群的核心流量一段时间就可以。毕竟，不管这个备份集群有多差，都比完全没有要好。



这里还有一些细节问题。第一个是最开始按多少比例转发到集群 2 上比较合适，答案是这个初始流量的大小就是你业务数据库能撑住的流量大小。因为一开始转发到集群 2 上的流量，肯定都是缓存未命中的，也就是要回查数据库，所以数据库决定了这个初始流量。

第二个问题是后续流量要怎么放开。答案是你可以选择，按照比例增长、线性增长都可以。但是原则是要保守，因为你万一把集群 2 弄崩溃了，业务损失就更大了。

第三个问题就是怎么判定 Redis 已经崩溃了，或者恢复过来了？这个问题你在微服务部分已经多次遇到了，思路都是一样的，这里我就不重复了。

面试思路总结

今天这节课我给你介绍了缓存穿透、击穿和雪崩三个基本概念，你要注意它们之间的区别，尤其是含义非常接近的穿透和击穿两个概念，穿透是完全没有数据，而击穿是缓存里没有数据。

后面我也提出了具体的解决方案。

解决缓存穿透：回写特殊值和布隆过滤器两个方案

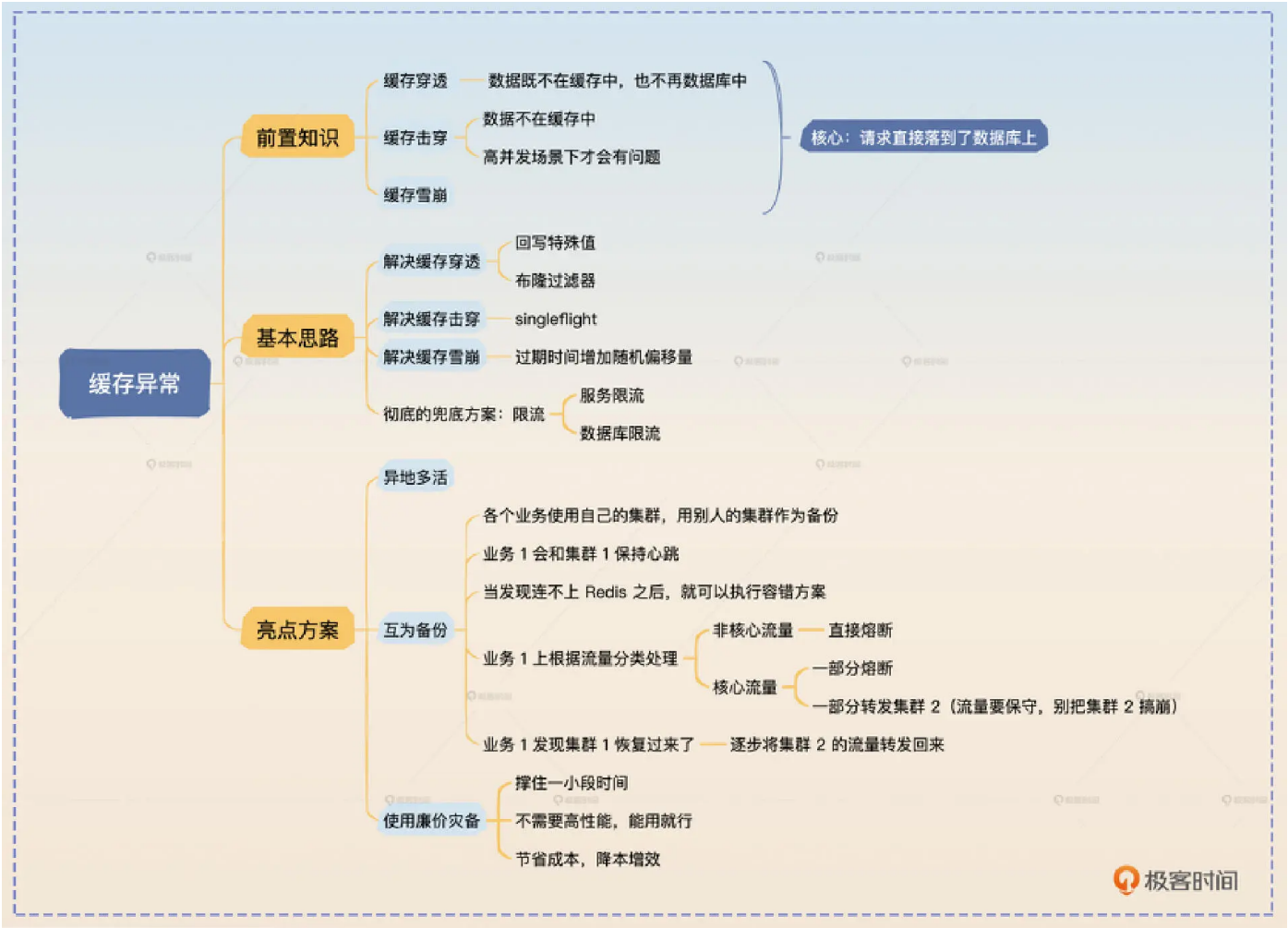
解决缓存击穿：singleflight 模式

解决缓存雪崩：过期时间增加随机偏移量

还有一个限流，能够彻底保护住数据库，对于穿透、击穿和雪崩等问题都有效果。

最后我给出了一个互为备份的集群容错方案，并且提供了一个经济适用版的变种。这一次的亮点方案稍微有点不同，是一个低端一点的方案。但是它更加符合大多数人的实际情况，在实践中落地也不是很难，只需要改造一下缓存客户端代码，加上探活和流量调度的逻辑就可以了。

如果你在大厂工作过，那么你可以考虑深入了解你们公司异地多活方案的细节。就算你没有实际部署实施过，但是面试的时候还是可以拿出来聊一下，凸显一下你的知识面。



思考题

最后，请你来思考两个问题。

能不能考虑在 Redis 崩溃之后启用本地缓存来作为容灾？如果可以，你准备怎么在面试中介绍这个方案？

除了我这里列举的方案，你还有没有别的思路可以解决缓存穿透、击穿和雪崩的问题？

欢迎你把思考后的答案分享到评论区，也欢迎你把这节课的内容分享给其他朋友，我们下节课再见！

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 34 | 缓存一致性问题：高并发服务如何保证缓存一致性？

下一篇 36 | Redis 单线程：为什么 Redis 用单线程而 Memcached 用多线程？

精选留言 3