Redis可使用缓存：提高系统的响应速度，提升并发访问量,但是从内存中（redis读取数据）和从磁盘中（mysql读取数据）的响应速度是不在一个级别的。

缓存击穿

缓存击穿：请求要访问的数据，**缓存中没有但是数据库中有**。

原因：可能是缓存过期，或者一个线程正在构建缓存时,另一个线程正在访问这个缓存的数据.

解决：

1、**互斥锁**：一个线程构建缓存，其他线程在睡眠或者轮询——不符合高并发低时延

2、**后台续命**：后台开一个定时任务，**专门主动更新即将过期的数据**。快过期时再去数据库查询一次并重新放入缓存中。

3、**永不过期**：

缓存穿透

缓存穿透：缓存穿透是指用户查询数据，**在数据库没有，自然在缓存中也不会有**。这样就导致用户查询的时候，在缓存中找不到，每次都要去数据库再查询一遍，然后返回空（相当于进行了两次无用的查询）。这样请求就绕过缓存直接查数据库，这也是经常提的缓存命中率问题。

原因：多为**恶意请求**

解决：

1、**缓存空对象**：如果一个查询返回的数据为空（不管是数据不存在，还是系统故障），我们仍然把这个空结果进行缓存，但它的过期时间会很短，最长不超过五分钟。通过这个直接设置的默认值存放到缓存，这样第二次到缓存中获取就有值了，而不会继续访问数据库，这种办法最简单粗暴！

第一个问题:如果在某个时间，缓存为空的记录，在数据库里面有值了，你怎么办？

解决方法一:设置缓存的时候，同时设置一个过期时间，这样过期之后，就会重新去数据库查询最新的数据并缓存起来。

解决方法二:如果对实时性要求非常高的话，那就写数据库的时候，同时写缓存。这样可以保障实时性。

解决方法三:如果对实时性要求不是那么高，那就写数据库的时候给消息队列发一条数据，让消息队列再通知处理缓存的逻辑去数据库取出最新的数据。

对于恶意攻击，请求的时候key往往各不相同，且只请求一次，那你要把这些key都缓存起来的话，因为每个key都只请求一次，那还是每次都会请求数据库，没有保护到数据库呀？

2、**布隆过滤器**：将所有可能存在的数据哈希到一个足够大的bitmap中，一个一定不存在的数据会被这个bitmap拦截掉，从而避免了对底层存储系统的查询压力。

优点：占用空间少

缺点：结果是概率性的，不是确切的。所以它可以缓解数据库的压力，但不能挡住。

分布式环境下使用布隆过滤器需要用到redis，所以redis不仅可以用来做缓存，还可以用来做布隆过滤器。

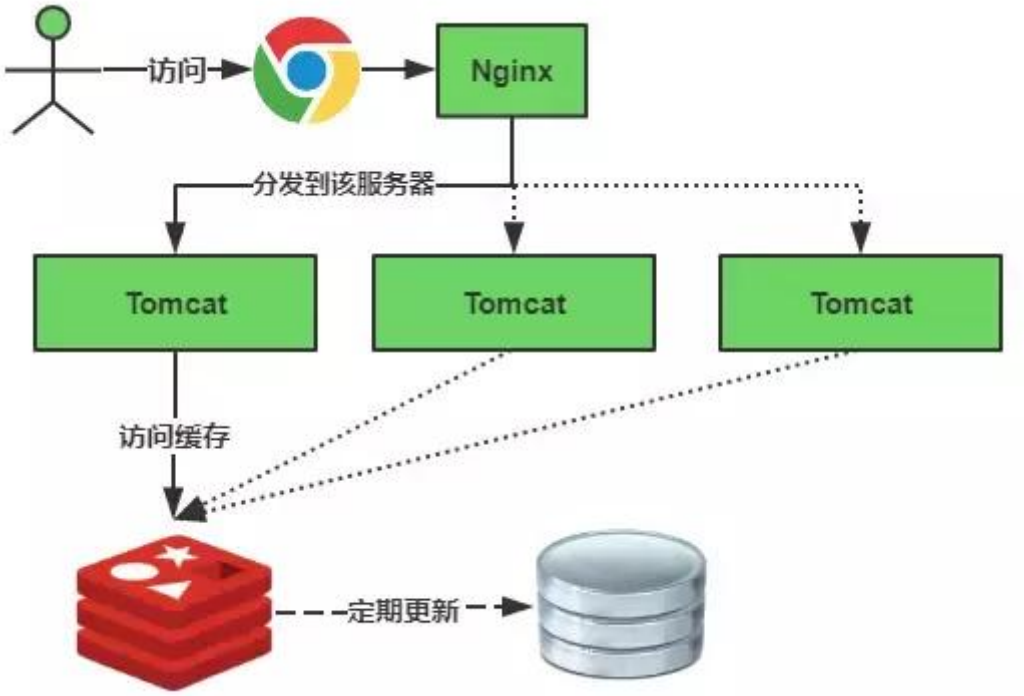
布隆过滤器直接返回数据库中是否有这条数据，如果没有直接返回，如果可能存在那就去查数据库存在缓存再return。

3、提前做一些**“用户鉴权、参数校验”**

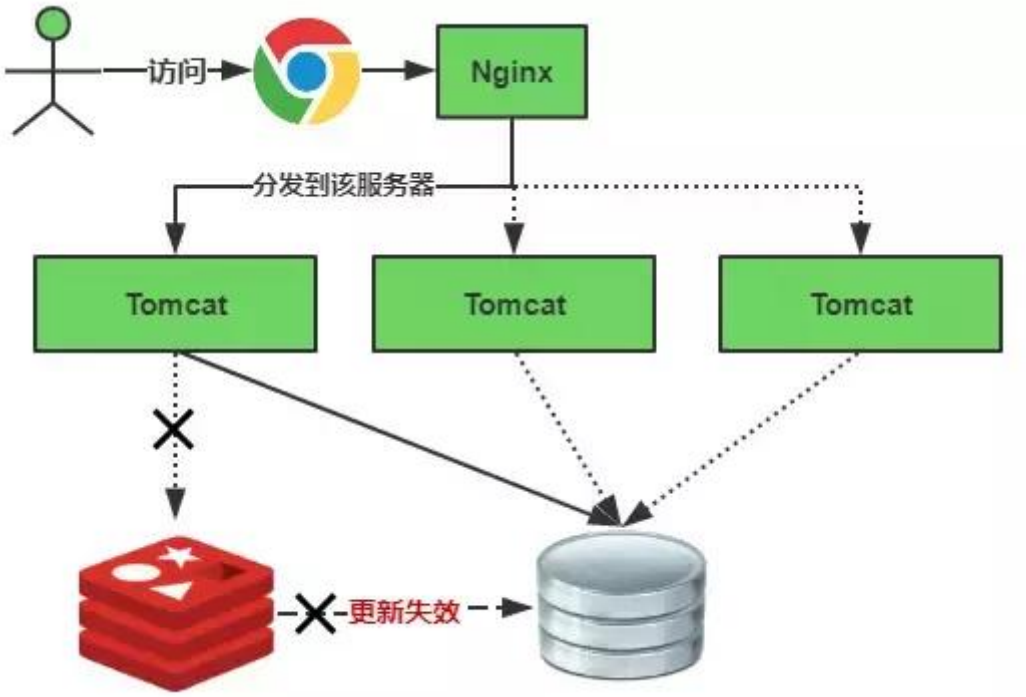
缓存雪崩

缓存雪崩：缓存中大多数的数据在同一时间到达过期时间，而查询数据量巨大，这时候，又是缓存中没有，数据库中有的情况了。请求都打到数据库上，引起数据库流量激增。

缓存正常从Redis中获取：



缓存失效瞬间：



与“缓存击穿”不同的是**缓存击穿指大量的请求并发查询同一条数据，缓存雪崩是不同数据都过期了**，导致这些数据在缓存中都差不到或者缓存服务直接挂掉了。

解决：

1、互斥锁：缓存雪崩可以看成多个缓存击穿, 互斥锁方案的思路就是如果从redis中没有获取到数据，就让一个线程去数据库查询数据，然后构建缓存，其他的线程就等着，过一段时间后再从redis中去获取。

2、“错峰”过期：在设置key过期时间的时候，在加上一个**短的随机过期时间**，这样就能避免大量缓存在同一时间过期，引起的缓存雪崩。

3、缓存集群：缓存服务挂掉的多数原因是单点应用。——引入redis集群，使用主从加哨兵。

4、限流器+本地缓存