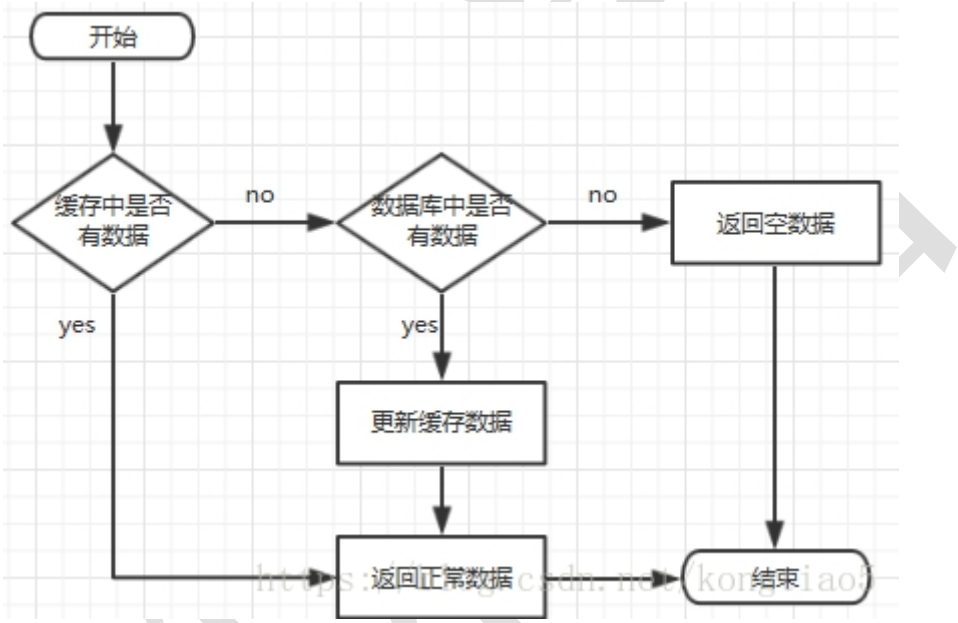
缓存穿透、缓存击穿、缓存雪崩区别和解决方案

本文链接：https://blog.csdn.net/kongtiao5/article/details/82771694

一、缓存处理流程

前台请求，后台先从缓存中取数据，取到直接返回结果，取不到时从数据库中取，数据库取到更新缓存，并返回结果，数据库也没取到，那直接返回空结果。



二、缓存穿透

原因：

缓存穿透是指缓存和数据库中都没有的数据，而用户不断发起请求，如发起为 id 为“-1”的数据或 id 为特别大不存在的数据。这时的用户很可能是攻击者，攻击会导致数据库压力过大。

解决方案

1. 接口层增加校验，如用户鉴权校验，id 做基础校验，id<=0 的直接拦截；

缺点：

虽然解决了非法调用并过滤了大部分不合法的查询条件，但是还是无法避免有查询不到数据的情况，这时候就需要 2。

2. 从缓存取不到的数据，在数据库中也没有取到，这时也可以将 key-value 对写为 key-null，缓存有效时间可以设置短点，如 30 秒（设置太长会导致正常情况也没法使用）。这样可以防止攻击用户反复用同一个 id 暴力攻击

三、缓存击穿

原因

缓存击穿是指缓存中没有但数据库中有的数据（一般是缓存时间到期，必定发生在第一次查询指定的数据时），这时由于并发用户特别多，同时读缓存没读到数据，又同时去数据库去取数据，引起数据库压力瞬间增大，造成过大压力

解决方案

1. 设置热点数据永远不过期。
2. 当缓存中没有指定数据，第一次并发请求相同的数据时，数据库压力会骤然增大。通过加互斥锁解决，互斥锁参考代码如下：

```java
public static String getData(String key) throws InterruptedException
{
    //从缓存读取数据
    String result = getDataFromRedis(key);
    //缓存中不存在数据
    if (result == null)
    {
        //去获取锁，获取成功，去数据库取数据
        if (reenLock.tryLock())
        {
            //从数据获取数据
            result = getDataFromMysql(key);
            //更新缓存数据
            if (result != null)
            {
                setDataToCache(key,result);
            }
            //释放锁
            reenLock.unlock();
        }
        //获取锁失败
        else
        {
            //暂停100ms再重新去获取数据
            Thread.sleep(100);
            result =  getData(key);
        }
    }
    return result;
}
```


说明：

1）缓存中有数据，直接走上述代码 13 行后就返回结果了

2）缓存中没有数据，第 1 个进入的线程，获取锁并从数据库去取数据，没释放锁之前，其他并行进入的线程会等待 100ms，再重新去缓存取数据。这样就防止都去数据库重复取数据，重复往缓存中更新数据情况出现。

3）当然这是简化处理，理论上如果能根据 key 值加锁就更好了，就是线程 A 从数据库取 key1 的数据并不妨碍线程 B 取 key2 的数据，上面代码明显做不到这点。

缺点：

首批并发查询由于有互斥锁，速度较慢，后面的查询速度比较快。可以通过预加载缓存来解决这个问题。

四、缓存雪崩

原因

缓存雪崩是指缓存中数据大批量到过期时间，而查询数据量巨大，引起数据库压力过大甚至 down 机。和缓存击穿不同的是，缓存击穿指并发查同一条数据，缓存雪崩是不同数据都过期了，很多数据都查不到从而查数据库。

解决方案：

1. 缓存数据的过期时间设置随机，防止同一时间大量数据过期现象发生。
2. 如果缓存数据库是分布式部署，将热点数据均匀分布在不同搞得缓存数据库中。
3. 设置热点数据永远不过期。

五、为什么要设置过期时间？

上文中的缓存击穿、缓存穿透，通过设置数据永久不过期就可以解决，但是数据不过期就意味着不会从数据库查询最新值，不会将最新值更新到缓存。这样的设置方法对于那些从头到尾不会发生改变的数据而言，是绝妙的，但是那些需要更新的数据就需要设置一个合适的过期时间。