面试必备：布隆过滤器是什么？有什么用？

捡田螺的小男孩 4 days ago

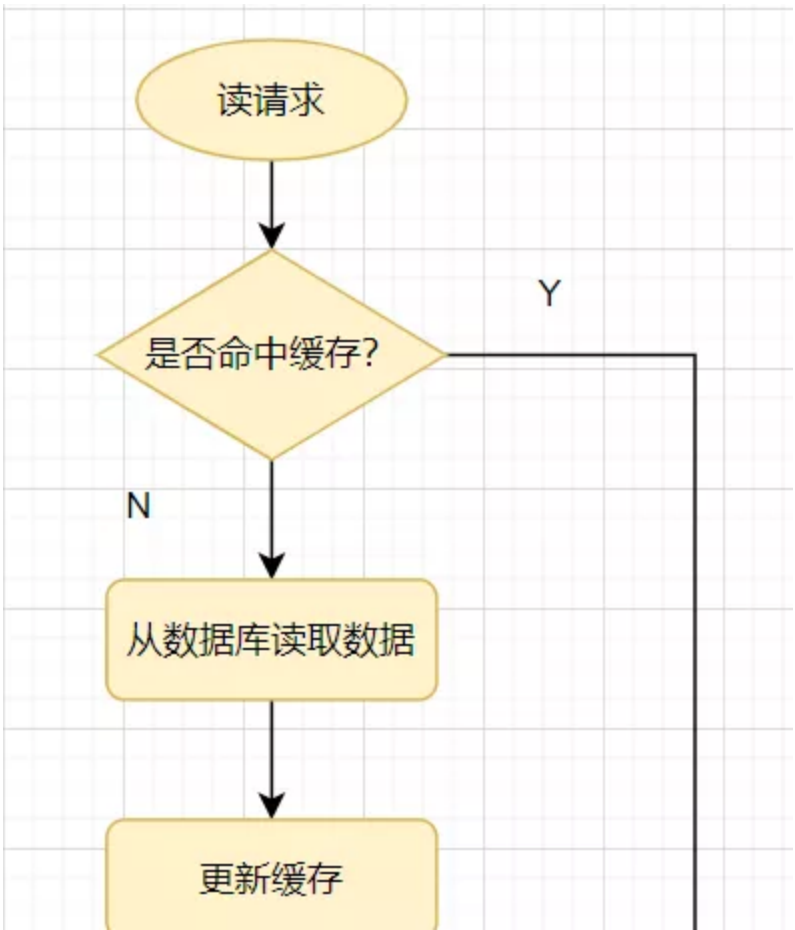
前言

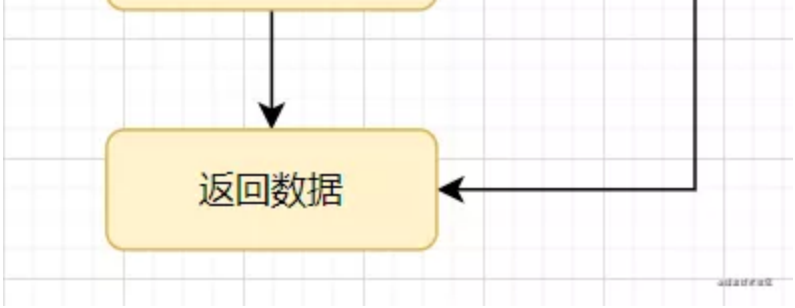
大家好，我是**程序员田螺**。今天我们来聊聊一道经典面试题，布隆过滤器是什么？有什么用？

缓存穿透

应对**缓存穿透**问题，我们可以使用**布隆过滤器**。我们先来回顾下缓存穿透知识点哈：

一个常见的缓存使用方式：读请求来了，先查下缓存，缓存有值命中，就直接返回；缓存没命中，就去查数据库，然后把数据库的值更新到缓存，再返回。





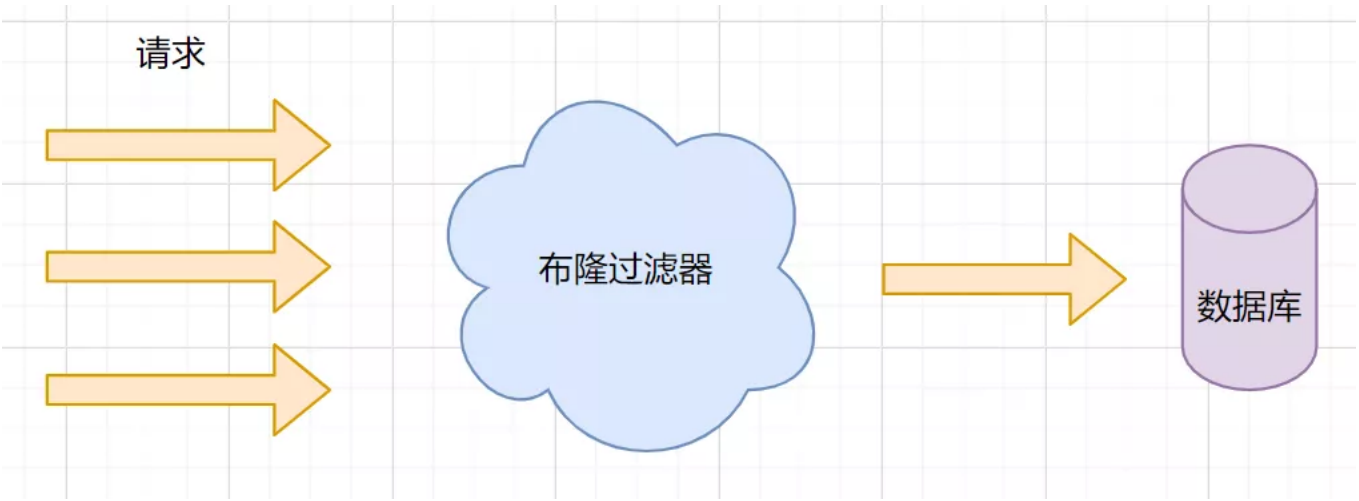
读取缓存

缓存穿透：指查询一个一定不存在的数据，由于缓存是不命中时需要从数据库查询，查不到数据则不写入缓存，这将导致这个不存在的数据每次请求都要到数据库去查询，进而给数据库带来压力。

假设我们需要查产品详情，有查询请求进来，我们先根据**产品Id**直接去缓存中查一下，没有的话，再去查下数据库。如果现在有**大量请求**进来，而且都在请求一个不存在的产品Id，那么这些请求都会怼到数据库，数据库压力一上来，可能就挂了。因此，我们可以在请求数据库层前，加个中间层，去缓解数据库压力嘛，如果，不存在的话，就不去查数据库啦。

大量数据判断是否存在

这个中间层，是不是用**HashMap**就好了呢？听起来不错嘛，HashMap时间复杂度可以达到O(1)，但是呢因为HashMap数据是在内存里面的，如果大量的数据远超出了服务器的内存呢，那就无法使用HashMap啦，可以使用**布隆过滤器**来做这个缓冲的事情。



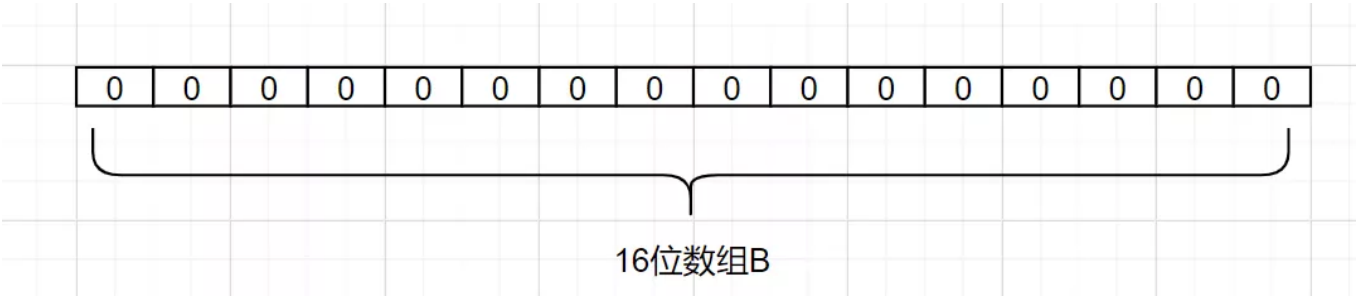
布隆过滤器是什么

布隆过滤器是一种占用空间很小的数据结构，它由一个很长的二进制向量和一组Hash映射函数组成，它用于检索一个元素是否在一个集合中，空间效率和查询时间都比一般的算法要好的多，缺点是有一定的误识别率和删除困难。

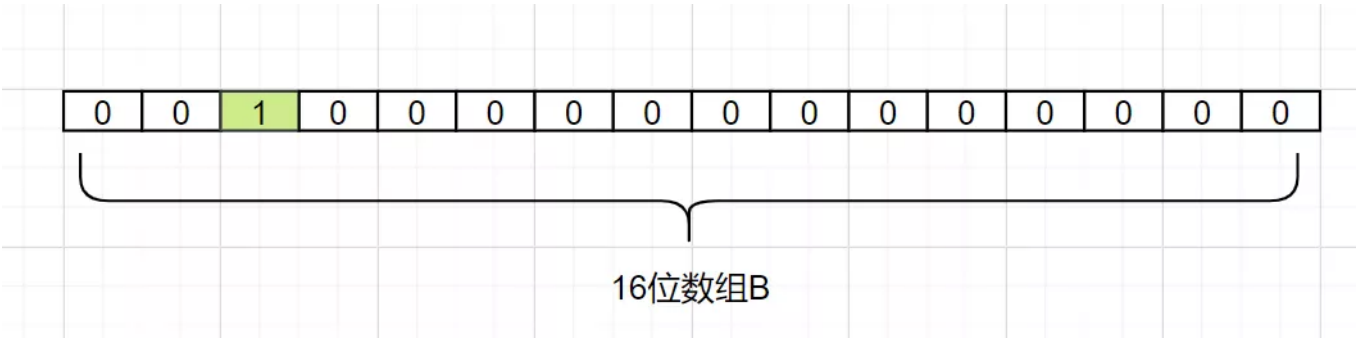
布隆过滤器原理是？假设我们有个集合A，A中有n个元素。利用**k个哈希散列**函数，将A中的每个元素**映射**到一个长度为a位的数组B中的不同位置上，这些位置上的二进制数均设置为1。如果待检查的元素，经过这k个

哈希散列函数的映射后，发现其k个位置上的二进制数**全部为1**，这个元素很可能属于集合A，反之，**一定不属于集合A。**

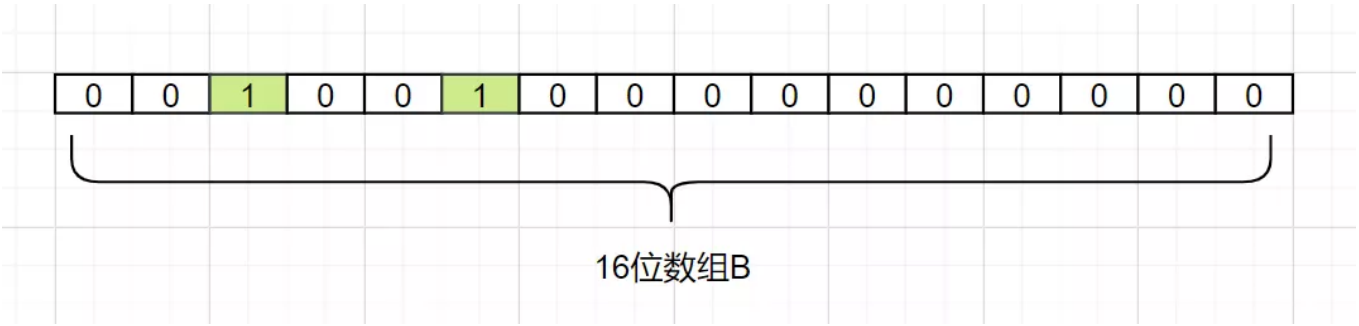
来看个简单例子吧，假设集合A有3个元素，分别为{**d1,d2,d3**}。有1个哈希函数，为**Hash1**。现在将A的每个元素映射到长度为16位数组B。



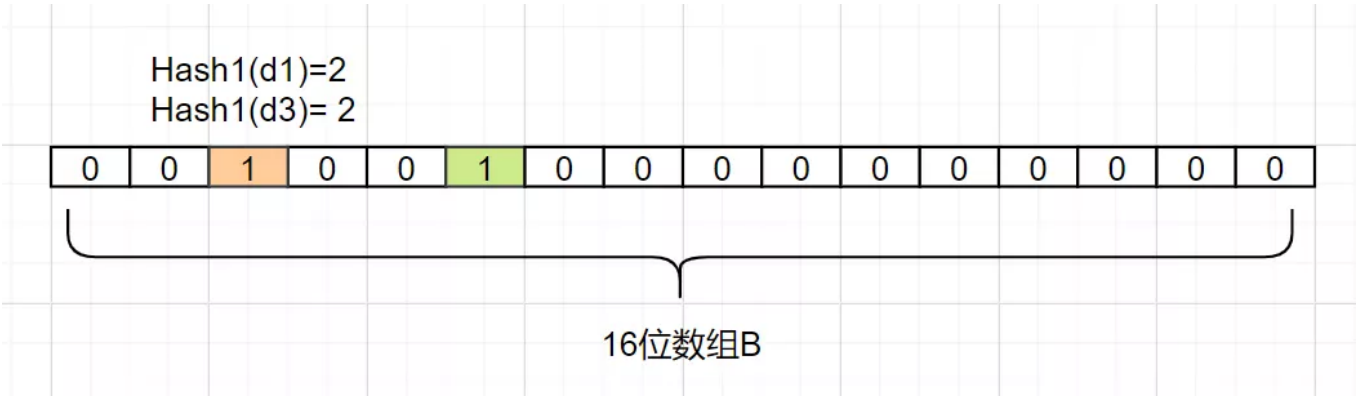
我们现在把d1映射过来，假设Hash1（d1）= 2，我们就把数组B中，下标为2的格子改成1，如下：



我们现在把**d2**也映射过来，假设Hash1（d2）= 5，我们把数组B中，下标为5的格子也改成1，如下：



接着我们把**d3**也映射过来，假设Hash1（d3）也等于 2，它也是把下标为2的格子标1：

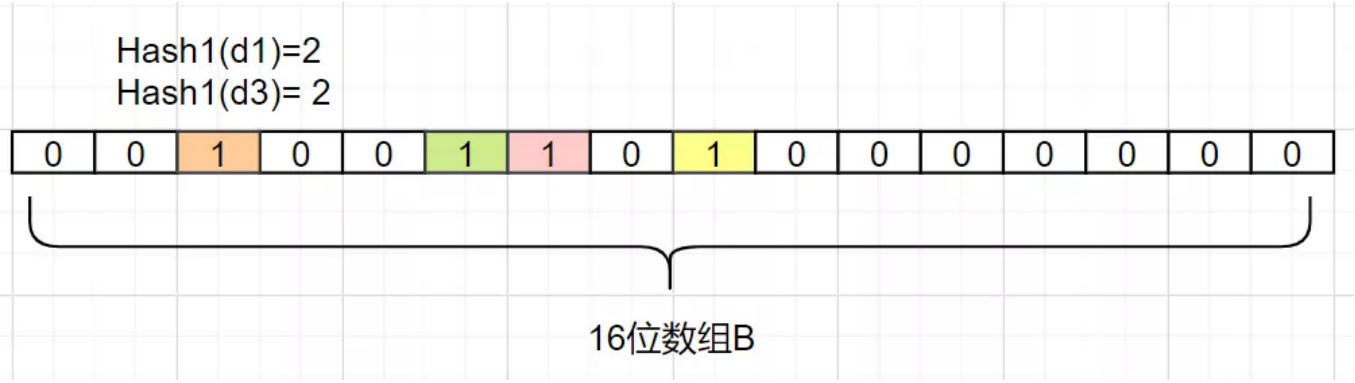


因此，我们要确认一个元素dn是否在集合A里，我们只要算出Hash1（dn）得到的索引下标，只要是0，那就表示这个元素**不在集合A**，如果索引下标是1呢？那该元素**可能**是A中的某一个元素。因为你看，d1和d3得到的下标值，都可能是1，还可能是其他别的数映射的，布隆过滤器是存在这个**缺点**的：会存在**hash碰撞**导致的假阳性，判断存在误差。

如何**减少这种误差**呢?

- 搞多几个哈希函数映射，降低哈希碰撞的概率
- 同时增加B数组的bit长度，可以增大hash函数生成的数据的范围，也可以降低哈希碰撞的概率

我们又增加一个Hash2**哈希映射**函数，假设Hash2（d1）=6,Hash2（d3）=8,它俩不就不冲突了嘛，如下：



即使**存在误差**，我们可以发现，布隆过滤器并**没有存放完整的数据**，它只是运用一系列哈希映射函数计算出位置，然后填充二进制向量。如果**数量很大的话**，布隆过滤器通过**极少的错误率，换取了存储空间的极大节省**，还是挺划算的。

目前布隆过滤器已经有相应实现的开源类库啦，如**Google的Guava类库**，Twitter的 Algebird 类库，信手拈来即可，或者基于Redis自带的Bitmaps自行实现设计也是可以的。

经典面试题：聊聊缓存击穿、缓存穿透、缓存雪奔

捡田螺的小男孩

经典面试题：聊聊Redis高可用

捡田螺的小男孩