链滴

浅谈布隆过滤器

作者：vcjmhg
原文链接：https://ld246.com/article/1569563764121
来源网站：链滴

首先在开始之前我们先考虑一个问题，如何快速判断某 URL 是否在 20 亿的网址 URL 集合中?

如果想判断一个元素是不是在一个集合里，一般思路的是将集合中所有元素保存起来，然后进行遍历较。链表、树、散列表（又叫哈希表，Hash table）等等数据结构都可以用于解决该类问题。但是随集合中元素的增加（比如说是20亿个url），我们需要的存储空间越来越大。同时检索速度也越来越。可能很多人首先想到的会是使用 HashSet，因为 HashSet基于HashMap，理论上时间复杂度为：O1)。达到了快速的目的，但是空间复杂度呢？字符串通过Hash得到一个Integer的值，Integer占4个节，那20亿个url字符串理论上需要：20亿*4/1024/1024/104=7.45G的内存，造成庞大的空间开销，显然这种解决方案是不合理的。但是针对这种问题我们该如解决哪？这便引出了我们这一节的主题布隆过滤器。

何为布隆过滤器

按照维基百科定义

布隆过滤器（英语：Bloom Filter）是1970年由布隆提出的。它实际上是一个很长的二进制向量和一列随机映射函数。布隆过滤器可以用于检索一个元素是否在一个集合中。它的优点是空间效率和查询间都远远超过一般的算法，缺点是有一定的误识别率和删除困难。

布隆过滤器的原理

布隆过滤器的原理是，当一个元素被加入集合时，通过K个散列函数将这个元素映射成一个位数组中的个点，把它们置为1。检索时，我们只要看看这些点是不是都是1就（**大约**）知道集合中有没有它了：果这些点有任何一个0，则被检元素一定不在；如果都是1，则被检元素很可能在。这就是布隆过滤器基本思想。

当然只有抽象的原理说明是很难让人理解的，下边我们结合例子对算法的执行流程进行说明。

首先在对算法描述之前我们先给定一个**前提**即：

我们此处使用的HASH算法使用的HASH值是Integer类型的：由于Integer.MAX_VALUE=2147483647，意思也就是任何一个URL的哈希值都会在0-2147483647之间。

第一步：对source数组进行预处理，我们可以定义2147483647长度的bit数组命名为source，用来

储hash之后产生的所有可能值。存储这个byte数组系统只需要：2147483647/8/1024/1024=256M。们分别通过N个HASH函数对这20亿个URL字符串进行处理得到20亿条N个不同的HASH值(此处我们之为结果对)，然后针对每个结果对(value_1,value_2 ... value_n)，我们分别将source的N个位置(source[value_1],source[value_2]...source[value_n])置为1，分别对20亿个结果对做如上的处理，我们便到的一个处理过的source数组，整个过程的处理逻辑如下图所示。

至此整个算法的预处理阶段算式完成。

第二步：url匹配当我们获得一个需要判断的url字符串时我们通过HASH_1、HASH_2...HASH_n等N HASH函数进行处理得到N个整数（value_1,value_2..value_n）分别判断source[value_i]的值是否是，如果是则判断下source[value_i+1]的值是否是1，以此类推，直到将N个HASH值判断完毕。中间何一个位置的值不是1，我们便可以得出结论，**待判断的url不在url集合中**，如果N个HASH判断完毕均满足条件，**我们便可以认为待判断的url在URL集合中**。整个过程如下图所示:

优点

1. 相比于其它的数据结构，布隆过滤器在空间和时间方面都有巨大的优势。布隆过滤器存储空间和插/查询时间都是常数O(K)。

2. 散列函数相互之间没有关系，方便由硬件并行实现。布隆过滤器不需要存储元素本身，在某些对保要求非常严格的场合有优势。

3. 布隆过滤器可以表示全集，其它任何数据结构都不能

缺点

整个过程并不难理解，但值得注意的是使用该算法判断目标字符串是由一定误算率的。一方面随着存数据元素的规模越来越越大，误算率也会随之增加。另一方面误算率还和使用的HASH的个数和种类关，一般来说选用的HASH函数越多，误算率就会越小。

应用场景

1、黑名单 2、URL去重 3、单词拼写检查 4、Key-Value缓存系统的Key校验 5、ID校验，比如订单统查询某个订单ID是否存在，如果不存在就直接返回。

使用方法

在实际开发中Guava框架提供了布隆过滤器的具体实现：BloomFilter，使得开发不用再自己写一套法的实现。

1. 创建布隆过滤器并且传入最大集合的个数，以及期望的误报率

Funnel:起到管道的作用将各种流和数据类型处理为BoomFilter可处理的类型

expectedInsertions:要处理的数据规模

expectedInsertion：期望的误报率

```
BloomFilter<String> filter=BloomFilter.create(Funnels.stringFunnel(Charset.defaultCharset()),
00,0.01);
```

2.初始化urls集合（相当于得到source数组）

```
//插入url集合String
urls[]={"vcjmhg.top","example.com","vcjmhg.com","vcjmhg.cn","baidu.com","google.com"};
```

原文链接：浅谈布隆过滤器

```
for(int i=0;i<urls.length;i++){    filter.put(urls[i]);}
```

3.输入测试url进行测试

```
//我们添加了七个元素，并且定义了最大字符数量为500，因此我们过滤器会产生十分准确的结果//
试vcjmhg.top是否在urls集合中,以及其准确率
System.out.println("vcjmhg.top是否在urls集合中"+filter.mightContain("vcjmhg.top"));
System.out.println("vcjmg.top的测试准确率:"+filter.expectedFpp());
//测试bing.com是否在urls集合中
System.out.println("bing.com是否在urls集合中:"+filter.mightContain("bing.com"));
System.out.println("bing.com此次的测试准确率:"+filter.expectedFpp());
```

运行结果：

与其他技术的对比与扩展

回顾布隆过滤器的原理我们发现布隆过滤器的原理与位运算的算法和题目很相似，我们可以尝试和位算相关的算法结合进行学习。

代码地址

博客代码地址