第 6 章

大数据和空间限制

认识布隆过滤器

【题目】

不安全网页的黑名单包含 100 亿个黑名单网页，每个网页的 URL 最多占用 64B。现在想要实现一种网页过滤系统，可以根据网页的 URL 判断该网页是否在黑名单上，请设计该系统。

【要求】

1．该系统允许有万分之一以下的判断失误率。
2．使用的额外空间不要超过 30GB。

【难度】

尉 ★★☆☆

【解答】

如果把黑名单中所有的 URL 通过数据库或哈希表保存下来，就可以对每条 URL 进行查询，但是每个 URL 有 64B，数量是 100 亿个，所以至少需要 640GB 的空间，不满足要求 2。

如果面试者遇到网页黑名单系统、垃圾邮件过滤系统、爬虫的网址判重系统等题目，

又看到系统容忍一定程度的失误率，但是对空间要求比较严格，那么很可能是面试官希望面试者具备布隆过滤器的知识。一个布隆过滤器精确地代表一个集合，并可以精确判断一个元素是否在集合中。注意，只是精确代表和精确判断，到底有多精确呢？则完全在于你具体的设计，但想做到完全正确是不可能的。布隆过滤器的优势就在于使用很少的空间就可以将准确率做到很高的程度，该结构由 Burton Howard Bloom 于 1970 年提出。

首先介绍哈希函数（散列函数）的概念。哈希函数的输入域可以是非常大的范围，比如，任意一个字符串，但是输出域是固定的范围，假设为 S，并具有如下性质：

1．典型的哈希函数都有无限的输入值域。

2．当给哈希函数传入相同的输入值时，返回值一样。

3．当给哈希函数传入不同的输入值时，返回值可能一样，也可能不一样，这是当然的，因为输出域统一是 S，所以会有不同的输入值对应在 S 中的一个元素上。

4．最重要的性质是很多不同的输入值所得到的返回值会均匀地分布在 S 上。

第 1～3 点性质是哈希函数的基础，第 4 点性质是评价一个哈希函数优劣的关键，不同输入值所得到的所有返回值越均匀地分布在 S 上，哈希函数越优秀，并且这种均匀分布与输入值出现的规律无关。比如，"aaa1"、"aaa2"、"aaa3"三个输入值比较类似，但经过优秀的哈希函数计算后的结果应该相差非常大。读者只用记清哈希函数的性质即可，有兴趣的读者可以了解一些哈希函数经典的实现，比如 MD5 和 SHA1 算法，但了解这些算法的细节并不在准备代码面试的范围中。如果一个优秀的哈希函数能够做到很多不同的输入值所得到的返回值非常均匀地分布在 S 上，那么将所有的返回值对 m 取余（%m），可以认为所有的返回值也会均匀地分布在 0~m-1 的空间上。这是显而易见的，本书不再详述。

接下来介绍一下什么是布隆过滤器。假设有一个长度为 m 的 bit 类型的数组，即数组中的每一个位置只占一个 bit，如我们所知，每一个 bit 只有 0 和 1 两种状态，如图 6-1 所示。

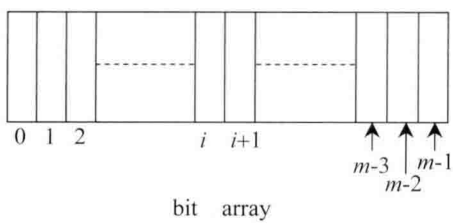

图 6-1

再假设一共有 k 个哈希函数，这些函数的输出域 S 都大于或等于 m，并且这些哈希函数都足够优秀，彼此之间也完全独立。那么对同一个输入对象（假设是一个字符串记为

URL)，经过 k 个哈希函数算出来的结果也是独立的，可能相同，也可能不同，但彼此独立。对算出来的每一个结果都对 m 取余（%m），然后在 bit array 上把相应的位置设置为 1（涂黑），如图 6-2 所示。

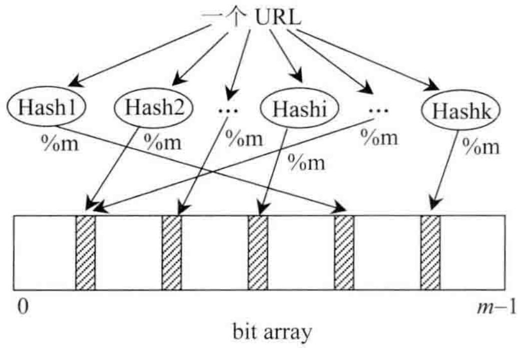

图 6-2

我们把 bit 类型的数组记为 bitMap。至此，一个输入对象对 bitMap 的影响过程就结束了，也就是 bitMap 中的一些位置会被涂黑。接下来按照该方法处理所有的输入对象，每个对象都可能把 bitMap 中的一些白位置涂黑，也可能遇到已经涂黑的位置，遇到已经涂黑的位置让其继续为黑即可。处理完所有的输入对象后，可能 bitMap 中已经有相当多的位置被涂黑。至此，一个布隆过滤器生成完毕，这个布隆过滤器代表之前所有输入对象组成的集合。

那么在检查阶段时，如何检查某一个对象是否是之前的某一个输入对象呢？假设一个对象为 a，想检查它是否是之前的输入对象，就把 a 通过 k 个哈希函数算出 k 个值，然后把 k 个值取余（%m），就得到在[0,m-1]范围上的 k 个值。接下来在 bitMap 上看这些位置是不是都为黑。如果有一个不为黑，说明 a 一定不在这个集合里。如果都为黑，说明 a 在这个集合里，但可能有误判。再解释具体一点，如果 a 的确是输入对象，那么在生成布隆过滤器时，bitMap 中相应的 k 个位置一定已经涂黑了，所以在检查阶段，a 一定不会被漏过，这个不会产生误判。会产生误判的是，a 明明不是输入对象，但如果在生成布隆过滤器的阶段因为输入对象过多，而 bitMap 过小，则会导致 bitMap 绝大多数的位置都已经变黑。那么在检查 a 时，可能 a 对应的 k 个位置都是黑的，从而错误地认为 a 是输入对象。通俗地说，布隆过滤器的失误类型是“宁可错杀三千，绝不放过一个”。

布隆过滤器到底该怎么实现？读者已经注意到，如果 bitMap 的大小 m 相比于输入对象的个数 n 过小，失误率会变大。接下来先介绍根据 n 的大小和我们想达到的失误率 p，如

何确定布隆过滤器的大小 m 和哈希函数的个数 k，最后是布隆过滤器的失误率分析。下面以本题为例来说明。

黑名单中样本的个数为 100 亿个，记为 n；失误率不能超过 0.01%，记为 p；每个样本的大小为 64B，这个信息不会影响布隆过滤器的大小，只和选择哈希函数有关，一般的哈希函数都可以接收 64B 的输入对象，所以使用布隆过滤器还有一个好处是不用顾忌单个样本的大小，它丝毫不能影响布隆过滤器的大小。

所以 n=100 亿，p=0.01%，布隆过滤器的大小 m 由以下公式确定：

$$m=-\frac{n\times\ln p}{(\ln 2)^2}$$

根据公式计算出 m=19.19n，向上取整为 20n，即需要 2000 亿个 bit，也就是 25GB。

哈希函数的个数由以下公式决定：

$$k=\ln 2\times\frac{m}{n}=0.7\times\frac{m}{n}$$

计算出哈希函数的个数为 k=14 个。

然后用 25GB 的 bitMap 再单独实现 14 个哈希函数，根据如上描述生成布隆过滤器即可。

因为我们在确定布隆过滤器大小的过程中选择了向上取整，所以还要用如下公式确定布隆过滤器真实的失误率为：

$$(1-e^{-\frac{nk}{m}})^k$$

根据这个公式算出真实的失误率为 0.006%，这是比 0.01%更低的失误率，哈希函数本身不占用什么空间，所以使用的空间就是 bitMap 的大小（即 25GB），服务器的内存都可以达到这个级别，所有要求达标。之后的判断阶段如上文的描述。

布隆过滤器失误率分析。假设布隆过滤器中的 k 个哈希函数足够好且各自独立，每个输入对象都等概率地散列到 bitMap 中 m 个 bit 中的任意 k 个位置，且与其他元素被散列到哪儿无关。那么对某一个 bit 位来说，一个输入对象在被 k 个哈希函数散列后，这个位置依然没有被涂黑的概率为：

$$(1-\frac{1}{m})^k$$

经过 n 个输入对象后，这个位置依然没有被涂黑的概率为：

$$(1-\frac{1}{m})^{kn}$$

那么被涂黑的概率就为：

$$1-(1-\frac{1}{m})^{kn}$$

那么在检查阶段，检查 k 个位置都为黑的概率为：

$$(1-(1-\frac{1}{m})^{kn})^k=(1-(1-\frac{1}{m})^{-m\times\frac{-kn}{m}})^k$$

在 x->0 时，(1+x)^(1/x)->e。上面等式的右边可以认为 m 为很大的数，所以-1/m->0，所以化简为：

$$(1-(1-\frac{1}{m})^{-m\times\frac{-kn}{m}})^k\sim(1-e^{-\frac{nk}{m}})^k$$

有关布隆过滤器失误率的公式如上，上文最先提到的确定布隆过滤器大小 m 及其哈希函数的个数 k 的两个公式都是从这个公式出发才推出的，接下来展示一下推出的过程。首先我们分析一下，如果给定 m 和 n 的值，根据如上的失误率公式，k 取何值可使误判率最低？设误判率为 k 的函数为：

$$f(k)=(1-e-\frac{nk}{m})^k$$

设 $b=e^{n/m}$，则公式化简为：

$$f(k)=(1-b^{-k})^k$$

两边取对数得到：

$$\ln f(k)=k\times\ln(1-b^{-k})$$

两边对 k 求导：

$$\frac{1}{f(k)}\times f'(k)=\ln(1-b^{-k})+k\times\frac{1}{1-b^{-k}}\times(-b^{-k})\times\ln b\times(-1)$$

$$=\ln(1-b^{-k})+k\times\frac{b^{-k}\times\ln b}{1-b^{-k}}$$

对等号右边的部分求最值：

$$\ln(1-b^{-k})+k\times\frac{b^{-k}\times\ln b}{1-b^{-k}}=0$$

$$\Rightarrow(1-b^{-k})\times\ln(1-b^{-k})=-k\times b^{-k}\times\ln b$$

$$\Rightarrow(1-b^{-k})\times\ln(1-b^{-k})=b^{-k}\times\ln b^{-k}$$

$$\Rightarrow 1-b^{-k}=b^{-k}$$

$$\Rightarrow b^{-k}=\frac{1}{2}$$

$$\Rightarrow e^{-\frac{kn}{m}} = \frac{1}{2}$$

$$\Rightarrow \frac{kn}{m} = \ln 2$$

$$\Rightarrow k = \ln 2 \times \frac{m}{n} = 0.7 \times \frac{m}{n}$$

至此，我们得到了如何根据 m 与 n 的值得到最合适的哈希函数数量 k 的公式，把这个公式带回失误率公式，就得到了如何根据失误率 p 和样本数 n 来确定布隆过滤器大小 m 的公式。

布隆过滤器会有误报，对已经发现的误报样本可以通过建立白名单来防止误报。比如，已经发现“aaaaaa5”这个样本不在布隆过滤器中，但是每次计算后的结果都显示其在布隆过滤器中，那么就可以把这个样本加入到白名单中，以后就可以知道这个样本确实不在布隆过滤器中。

在此特别感谢本篇文章参考网文的作者 Allen Sun（http://www.cnblogs.com/allensun/archive/2011/02/16/1956532.html）。

只用 2GB 内存在 20 亿个整数中找到出现次数最多的数

【题目】

有一个包含 20 亿个全是 32 位整数的大文件，在其中找到出现次数最多的数。

【要求】

内存限制为 2GB。

【难度】

士 ★☆☆☆

【解答】

想要在很多整数中找到出现次数最多的数，通常的做法是使用哈希表对出现的每一个数做词频统计，哈希表的 key 是某一个整数，value 是这个数出现的次数。就本题来说，一共有 20 亿个数，哪怕只是一个数出现了 20 亿次，用 32 位的整数也可以表示其出现的次数