漏桶算法在反垃圾邮件系统中的应用

Application of Leaky Bucket in Anti – spam System

蓝炳伟　（海军 92057 部队 524037）

摘　要： 电子邮件是 Internet 应用最广的服务：通过网络电子邮件系统，您可以用非常低廉的价格，以非常快速的方式，与世界上任何一个角落的网络用户联络。但是，垃圾邮件也随着互联网的不断发展而大量增长，垃圾邮件的危害现在已经深入人心。本文主要介绍漏桶算法在反垃圾邮件系统中的应用，从而有效分析异常行为，以利于及时生成垃圾邮件过滤规则。

关键词： 反垃圾邮件系统　漏桶算法　垃圾邮件

1　引言

现代网络技术的高速发展，特别是 Internet 的日益普及，推动了各种网络应用服务的发展。作为网络应用最为重要的应用之一，E – mail 服务跨越了地域及空间的制约因素，为广大的使用者创造了良好的条件。同时产生的垃圾邮件也给互联网以及广大的使用者带来了很大的影响，这种影响不仅仅是人们需要花费时间来处理垃圾邮件、占用系统资源等，同时也带来了很多的安全问题。本文主要介绍漏桶算法在反垃圾邮件系统中的应用，从而及时生成垃圾邮件的过滤规则。

2　漏桶算法概述

2.1　令牌桶的简述

令牌桶是一种常用的流量控制技术。令牌桶本身没有丢弃和优先级策略，其原理如下：

（1）令牌以一定的速率放入桶中。

（2）每个令牌允许源发送一定数量的比特。

（3）发送一个包，流量调节器就要从桶中删除与包大小相等的令牌数。

（4）如果没有足够的令牌发送包，这个包就会等待直到有足够的令牌（在整形器的情况下）或者包被丢弃，也有可能被标记更低的 DSCP（在策略者的情况下）。

（5）桶有特定的容量，如果桶已经满了，新加入的令牌就会被丢弃。因此，在任何时候，源发送到网络上的最大突发数据量与桶的大小成比例。令牌桶允许突发，但是不能超过限制。

2.2　漏桶算法概述及实现方法

漏桶算法是修改后的令牌桶算法，和实际的令牌桶算法是不相同的。漏桶算法的基本思路类似于令牌桶，所不同的是，每发生一种事件（用户发一封信），则将此种事件对应的桶内增加“水”，与此同时，所有的桶按照一定的速率漏水，当桶内“水”到达指定的高度时，则判定某种行为超过限制。

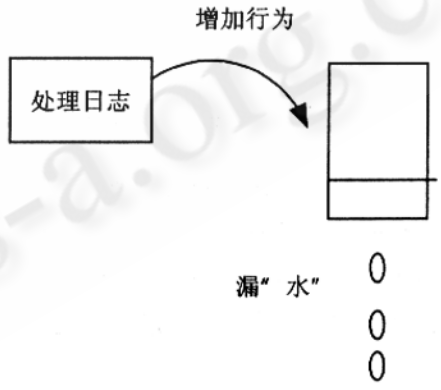

图 1　漏洞算法简单示意图

漏桶算法实现时主要有以下几个问题需要解决：

（1）在“漏水”时需要处理大量的“桶”，特别是在水泄漏光的时候，为了避免桶占用内存空间，需要释放掉所有空的桶，如何用最小代价来处理漏水是需要解决的。

（2）系统运行中会维护大量的桶，这些桶不停的产生消灭，会导致内存出现碎片，使得内存分配速度大大下降，从而严重影响这个系统的性能。

(3) 系统中维护的桶占用大量内存,如何使桶尽可能少占用内存。

针对以上问题,采用一些比较特殊的设计来解决以上问题。

首先是漏水处理,如果采用标准的漏水桶,桶内存放水量的设计,每次漏水时都需要扫描全部的桶,将水量减少,在桶数目多的时候,处理代价很大,注意到在这个应用下,桶的深度不会很大,一般在几千的量级,远远小于桶的数目。因此采用类似下图的设计。

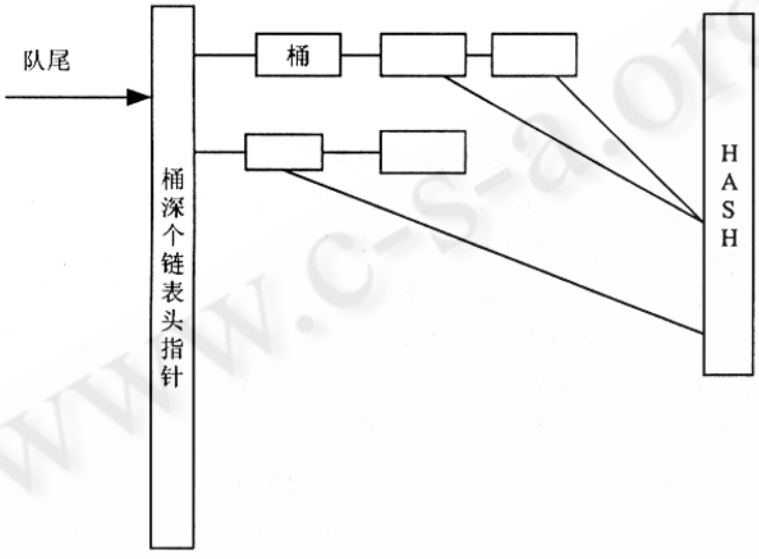

图 2　漏桶实现设计

在这个实现中,桶内不存放桶内水深,预先开启桶深个链表,用一个指针表明当时的最低水位位置,桶所在链表的位置与最低水位链表的差就表明了它当时的水位。这样在漏水时,只要移动最低水位链表指针,就达到了全部水桶漏水的目的。

在执行加水操作时,先通过 HASH 查找到相应的桶,将其从现在的链表上摘下,判断是否溢出,溢出则执行回叫函数,然后清空,否则链接到加水之后位置代表的链表上,在执行漏水操作时,将最低水位位置链表上的全部桶释放。同时将被释放的桶从 hash 链表中摘除。

其次要处理的问题是内存碎片问题,解决内存碎片的标准方法是开内存池,即预先分配一整块内存,划分为若干块,用一个链表将这些空闲块组织起来,在需要分配时不调用 malloc 分配而是从预先分配的内存中取出一块,当此块内存被释放时,将此块内存放回空闲块链表。

可以看出,使用内存池可以解决内存碎片问题,此外,系统的 malloc 调用为了避免多 CPU 操作冲突,内部会加锁,因此性能很差,通过使用内存池,只要保证通过内存池分配和释放的操作都在同一线程内进行,就可以不必加锁,从而提高性能。

但是内存池也存在缺点,第一是内存池分配的内存块大小一般是固定大小的,如果要分配不同大小的内存块,就需要开多个内存池,而且各个内存池中的内存不能互相使用。如果内存池预先分配得太大,就会造成浪费,考虑到一般线上系统的情况应当是稳定在一个程度上,因此需要经过试验来确定比较合适的内存池初始参数。

最后一个比较重要的问题是桶的内存占用问题。为了节省内存,将桶节点和 hash 节点合并为一个数据结构,目前每个桶节点本身的内存占用是 20 字节,设桶总数为 N,实际分配的桶数为 n,桶深为 M,桶 hash 大小为 H,平均字串长度为 L。则一组桶占用的内存空间为 $(20+L)*n+20N+4H+20M$。

反垃圾邮件系统中一共存在 8 组桶,这 8 组桶涉及的字串(邮件地址或者 IP)有很多是相同的,为了节省内存,字串采用共享式的方式,维护一个引用计数。对于相同的字符串,并不重复分配内存,而是增加它的引用计数,被释放时减少引用计数,当引用计数为 0 时才真正释放内存。这样对于 SENDER 发件人相关的桶来说,可以减少 80% 的内存使用量。

2.3 漏桶算法参数计算

通常对行为数量的限制标准是在时间 T 内最多产生 M 个行为,而漏桶算法的可配置的参数则是桶深 K 格水,每个行为增加 dk 格水,每 dt 时间漏掉 dm 格水,它们之间的关系满足下式:

$$M\frac{dk}{T}-\frac{dm}{dt}=\frac{K}{T} \quad (1)$$

在目前的漏桶实现中有如下限制:dm 总为 1,另外要求 dt 和 dk 为整数。增大 dt 可以减少漏水操作的次

数,从而减少处理计算量,但是会使得 K 增大增加内存占用,增大 dk 会增大 K,但对处理没有其他影响,因此 dk 取 1 即可。得到:

$$K = M - \frac{T}{dt} \quad (2)$$

此式中 T/dt 即为 T 时间段内,漏水操作的执行次数。

调节 M,T,dt 可以控制检测粒度,一定程度上控制处理计算量和内存使用。

3 漏桶算法在反垃圾邮件系统中的研究应用

反垃圾邮件系统中设置了 8 组桶,桶的分组如下表,当邮件系统接收一封邮件时,就向相应的桶加一滴水,当漏桶溢出时,这样就可以产生一条过滤规则,比如当某个发送者(abc@ hotmail. com)的 OKSENDER 桶溢出时,则可以判断这个用户在规定的时间里发送成功邮件太频繁,从而可以认为有发垃圾邮件的嫌疑,这样就可以把发送者 abc@ hotmail. com 加入黑名单中禁封一段时间了。其它桶溢出也一样的道理,这样就有效的阻止了频繁发送垃圾邮件的可能了。

桶分组	说明
OKSENDER	发送信件超量,封禁 SENDER
CBIP	内容被过滤,封禁 IP
UNSENDER	未知用户,封禁 SENDER
UNIP	未知用户,封禁 IP
RBSENDER	用户信箱被保护,封禁 SENDER
CBSENDER	内容被过滤,封禁 SENDER
SBSENDER	SENDER 被封禁,追加封禁 SENDER
OKIP	发送信件超量,封禁 IP

4 总结

本文简要介绍了漏桶算法,并详细阐述了在反邮件系统中的如何应用漏桶算法,很好地解决了反垃圾邮件系统中检测超量问题,及时有效地生成垃圾邮件的过滤规则。

参考文献

1 布卢姆,开放源码邮件系统安全,杜鹏 译,北京:人民邮电出版社,2002.4.

2 http://www.openspf.org/