基于URL过滤与内容过滤的网络净化模型

李石君,李 洲,余 军,张 科
(武汉大学 计算机学院,湖北 武汉 430072)

摘 要:在信息时代里,信息爆炸似地增长着,而为数众多的不良信息充斥其中。针对这种现状,许多的公司与研究机构都提出了自己的过滤模型。文中在研究过去模型的基础上,提出了一种高效可靠的网络净化模型。该模型在运用WCCP协议的前提下,结合了URL过滤和内容过滤这两种过滤方法对网络信息进行过滤,改进了传统的URL过滤算法。

关键词:WCCP协议;URL过滤;内容过滤;缓存
中图分类号:TP393.08 **文献标识码**:A **文章编号**:1005-3751(2006)01-0005-03

A Model of Web Purifying Based on URL Filter and Content Filter

LI Shi-jun, LI Zhou, YU Jun, ZHANG Ke
(School of Computer, Wuhan University, Wuhan 430072, China)

Abstract: In the information age, information is developing rapidly, but the bad information is also mixed in it. Aiming at this state, lots of companies and research institutions proposed their own filter models. This paper proposes a highly efficient and trusted filter model based on the research of the old models. This model works on the protocol of WCCP. It filters the Web information by combining the models of the URL filter and the content filter and improves the traditional mechanisms of URL filter.

Key words: WCCP; URL filter; content filter; cache

0 引 言

作为一个巨大的全球性的信息中心,Internet正以惊人的速度发展壮大。人们可以通过Internet获取各种各样的信息和资源。然而,由于网络的开放性使得任何人可以获得几乎任何种类的信息,这不免让一些"不良信息"乘虚而入,如反动信息、不利于青少年健康的信息等等。国家教育部及信息安全部门对青少年健康上网问题都十分重视,一些公司和科研机构也对此类问题进行了富有成果的研究,提出了一些方法,但可靠性不高。

传统的模型或仅采用URL过滤,或仅采用内容过滤。单纯的URL过滤对于URL库的建立必须采用人工录入的方法,必定浪费大量的人力物力,而且对非法URL的覆盖程度也不高;而仅仅采用内容过滤则将导致大量冗余操作,效率低下。文中在对两种方法的优缺点进行比较研究的基础上,提出了一种基于WCCP(Web cache communication protocol)协议的双重过滤模型[1]。

1 WCCP对用户请求的初步处理

WCCP是CISCO公司提出的一种高速缓存技术协议[2],是路由器与缓存引擎(cache engine)之间的通信协议,旨在减少网络中大量的迂回传输。当用户提出浏览页面请求时,路由器首先把请求重定向(redirect)到缓存引擎,如果缓存引擎中已经存在用户请求页面的拷贝,则由缓存引擎直接将这个页面发送给用户,否则缓存引擎就到Web服务器上取得这个页面以及这个页面的所有对象,拷贝一份放到缓存引擎之后再将其发送给用户,这样就可以保证用户下次访问同样的内容时能够在缓存引擎中命中。用户对WCCP的这种运作机制是完全不知觉的,故它又被叫做透明缓存服务器,用户并不需要在浏览器上做任何的设定即可享受透明缓存服务器的服务。

以上是对WCCP原理的简述,而文中所要研究的网络过滤模型的第一层便基于此。当局域网的用户发送一个HTTP请求时(此处假定是基于WCCP V1,如果用的是WCCP V2的话,透明服务器提供的资料类型不仅是网页,还可以是RTSP、MMS等),路由器依据WCCP协议将其重定向到透明缓存服务器,如果缓存服务器中有该请求的备份,则用户的请求为合法请求(因为在缓存服务器中存在的页面都是经过后续的过滤机制过滤过的,所以从某种程度上,可以保证它们的合法性,而这也是与原始的WCCP一点区别之处,因为原始的WCCP并不要求缓存中存的都是合法信息),确定了请求为合法之后,则响应该请求并把页面发送给用户;如果此时无法处理,即不能确定该用户的请求是否为合法请求,则该请求将接受第一层过滤机制——URL过滤机制的检测。

收稿日期:2005-04-23
基金项目:国家自然科学基金资助(60273072);国家"八六三"高新技术研究发展计划资助(2002AA4234502)
作者简介:李石君(1964—),男,湖南岳阳人,博士,副教授,研究方向为数据库技术、Web信息技术。

2 URL 过滤

2.1 URL 库的建立及缓存的初始化

在缓存引擎上,建立对 URL 进行过滤用的数据库,包括合法 URL 表(White List)和非法 URL 表(Black List)。White List 与 Black List 均包括 URL 名(uname)、访问次数(ucount)两个属性,同时分别对 White List 和 Black List 的 URL 名建立索引。

另外据 Internet 的统计表明,超过 80% 的客户经常访问的是 20% 的网站内容,即 URL 库中虽然存储了大量的数据,但最经常被人们使用的只是其中的一小部分,频繁地访问硬盘上的数据库会降低效率,因此在此处引进 cache 技术减少缓存服务器对服务器的访问[3]。

系统在启动的时候,自动根据管理员设置的缓冲区大小将 White List 中访问次数最多的部分与 Black List 中访问时间最近的部分从数据库中读到各自的缓冲区中,完成缓存的初始化。

2.2 实施过滤

过滤时,把从用户请求数据包中提取的 URL 与 White List 中的 uname 进行匹配,如果匹配成功,则说明用户的请求是合法的,但目前页面缓存中没有。这时由缓存服务器定向该请求到 Internet,Internet 将页面信息传输到透明缓存服务器上,而后透明缓存服务器按照 WCCP 协议的规定更新页面缓存,将信息发给用户,即用户的请求得到响应;如果匹配不成功,则进行进一步的非法 URL 匹配,把请求中的 URL 部分与 Black List 中的 uname 进行匹配检测,这时如果匹配成功了,说明用户所请求的信息不合法,在这种情况下,应返回给用户警告提示;而如果再次匹配失败,则说明用户的请求的 URL 在 URL 数据库中无法找到对应项,标记该 URL 为可疑的(suspicious),该请求为可疑请求,再进行下一步处理,同样也将该请求发送到 Internet。

2.3 缓冲区中 URL 匹配

在缓冲区中匹配 URL 的时候,如果采用从缓冲区头部至尾部逐个匹配的话,必然造成匹配时间过长,效率低下,于是在匹配 URL 的时候采用了散列匹配的方法。这里选用了 ELFHash 函数作为散列函数,这个函数源于 UNIX System V,号称"实际生活散列函数中的典型魔术",在效率和平衡性方面表现都比较好,也是在信息查询研究工作中的首选。

如果缓冲区中查找成功则用户的请求可以立即得到相应的响应,否则说明用户所要求的 URL 不在缓冲中,此时应对数据库中的 Black List 或 White List 进行检索,如果在数据库中找到了相应项目则也认为查找成功,并将相应项目添加到缓冲区中。

当缓冲区满的时候就要实施相应的缓冲区替换算法。对于 White List 或 Black List 对应的缓冲区均采用 LRU (least recently used)算法,将最近最久未被访问的 URL 给替换出去。整个 URL 缓存运作流程如图 1 所示。

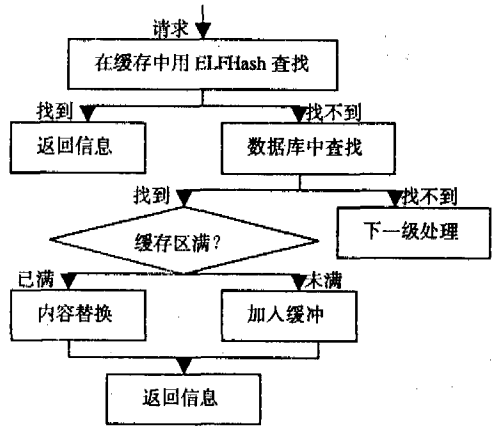

图 1 URL 缓存流程图

2.4 数据库的更新

在系统运行一段时间之后,URL 数据库里面的内容势必变得比较庞大,而且不少 URL 会失效(例如 White List 中的 URL 可能不再合法,或 White List 与 Black List 中 URL 所链接的页面已经不存在),此时管理员就应对数据库中的 URL 逐个访问,然后通过后面提到的内容过滤机制确定数据库中记录的 URL 信息是否依然有效,对无效信息予以删除。URL 过滤层总体过滤流程如图 2 所示。

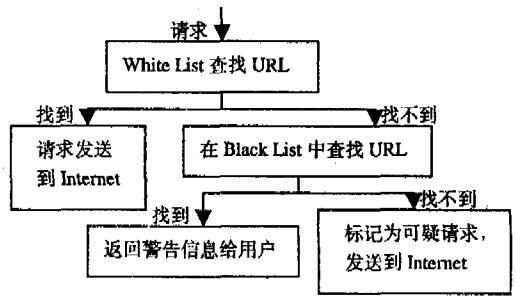

图 2 URL 过滤图

3 内容过滤

3.1 内容过滤的实施

经过 URL 过滤之后,如果用户的请求被标记为可疑请求,透明缓存服务器还是按照处理合法 URL 请求一样的方法来处理,把它定向到 Internet,只不过做了一点标记。对于标记过的请求,从 Internet 返回的信息是要经过最后的基于内容过滤操作的。内容过滤后,如发现信息是合法的,则返回给用户,同时页面缓存和 White list 也按一定规则刷新;但是如信息是非法的,则返回一个警告信息给用户,同时记录该非法 URL 到 Black List 中;如果请求的 URL 在 Internet 中不存在,则直接将该请求对应的"404 ERROR"页面返回给用户,不把该页面添加进缓存中。

对于内容过滤的具体操作采用图 3 所示的模型[4]。

首先用网络爬虫(Web Crawler)对 Internet 返回的页面进行分析,把从中抽取的图片和链接对应的页面存入临时数据库中,然后触发数据分析器对临时数据库中的内容进行分析得到该页面对应的特征向量,接着更新触发器对页面进行相应的处理(如果确认页面信息合法,则将页面

加入到页面缓存中,并把对应的 URL 加入到 White List 中,如果页面信息非法,则将其对应的 URL 加入到 Black List 中)。

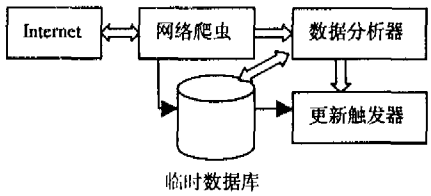

图 3　内容过滤模型

3.2　网络爬虫抽取页面特征

为了确定一个页面是否含有非法信息,首先必须提取它的关键信息,考虑到一些不健康网页仅仅就是由图片信息组成,几乎没有什么文本信息,所以既选择了图片,又选择了文本信息来作为网页的特征,同时还考虑到如果一个网页它所链接到的网页是非法网页的话,那么该网页也应该被视为非法的,所以也选取了链接信息作为网页的特征[5]。结合以上几点考虑,笔者总结了非法网页的特征,用向量表示如下:

Vow= (bSEW, nWD, nWDwS, nLINK, nLINKwS, nIMG, nIMGwS)

bSEW:表示是否含有非法信息

nWD: 当前页的总字数

nWDwS:表示非法信息的字数

nLINK:总的链接数

nLINKwS:链接到非法的网页的链接数

nIMG:总的图数

nIMGwS:非法图片数

有了这个特征向量之后,就可以利用网络爬虫来构造出每个网页的特征。首先分析网页中的链接情况,即确定 nLINK 和 nLINKwS 的值,为此必须遍历相关的链接。很明显,这是一个从底到顶的过程,基于此认识,用堆栈来构造数据结构,逐层分析链接情况。

在分析的过程中,先分解网页,删除各种网页标记,然后解析文本内容,得到文本信息,接着分析图片信息,以决定它们是否是非法信息,从而得到 nWD,nWDwS,nIMG,nIMGwS,根据这些信息逐层返回确定对应的链接是否为合法链接,得到页面的 nLINK,nLINKwS。最后基于以上信息,构造出特征向量。

对于得到的向量,可以借助于数学上的向量空间的相关知识来判断,比如可以先给定一个非法的标准特征向量值和一个合法的标准特征向量值,再比较待测特征向量与标准特征向量在向量空间的距离及远近来进行聚类或分类,以确定是合法信息还是非法信息。

算法伪代码如下:

```
//把特征向量视为欧氏空间的一个点,设合法样本点为 Ca,非法样本点为 Cb,待分析样本点为 Cx
CrawlWeb (url Init_URLs) { //Init_URLs 为该 URL 链接的子 URL
For each url in Init_URLs
将 url 的所有链接存入数组 LNKs;
If url 不是叶结点 URL 并且
  LNKs 中至少有一项的特征向量 vow 未被算出
Then { UrlStack.Push(url) ; //UrlStack 为存放 URL 的栈
CrawlWeb(LNKs) ; }
Else If GetVect(url)得到的子 vow 中的 SEW = false
  Then
  vow 的 nLINKwS + + ;
End For
While (UrlStack 不为空)
UrlStack.Pop(url) ;
If GetVect(url)得到的子 vow 中的 bSEW = false
Then
  vow 的 nLINKwS + + ;
End While
Return; }
GetVect (url url) { //计算 url 对应的特征向量
分解网页并删除网页标记;
计算出对应 vow 中的 nWD,nLINK,nIMG;
算出非法词汇的数量,存入对应 vow 中的 nWDwS;
将图片送入图形分析器,如果非法则 vow 中 nIMGwS + + ;
通过得到的 vow 算出 Cx;
求出 Cx 到 Ca 的欧式距离 Da;
求出 Cx 到 Cb 的欧式距离 Db;
If Da< = Db Then
{ vow 中的 bSEW 为 true;
将对应的 url 链接加入 white list 中;}
Else { vow 中的 bSEW 为 false;
将对应的 url 链接加入 black list;}
Return vow; }
```

4　总体模型

最后对各个阶段的模型进行分析,得到总体模型(如图 4 所示)。

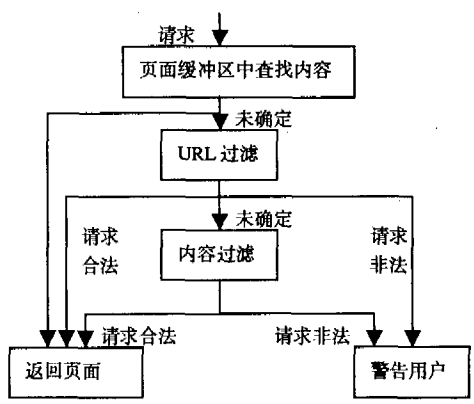

图 4　总体模型

此外对于缓存服务器上工作的各个服务进程应该适当分配优先级,使 CPU 的大部分工作时间用于处理页面缓存上,而将小部分时间用于请求过滤,从而使用户的合理请求可以得到尽快的响应。

(下转第 16 页)

```
DepartMent dep=DepartMent.find(ID);
dep.Name="保密室";
dep.DepartPepleNum=22;
dep.Update();
// 删除数据
dep.delete(ID);
```

这样的代码里面没有任何一句 SQL 语句,达到了对数据层透明,提高软件开发效率的目标。Web 端调用 Simple Query 模块实现简单查询方式如图 2 所示。

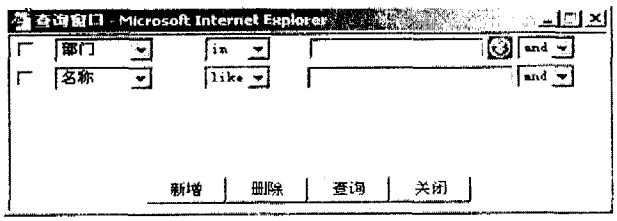

图 2 简单查询方式

根据用户对逻辑条件的选择,构造出 sql 语句直接送入数据库执行来实现简单的条件查询。通过采用 MY ORM Layer,使得当数据库中的表结构发生变化时,开发人员只需要修改对应的 XML 映射文件而不必改动代码,提高了系统的应用性能及可维护性;并且不必要为了一些简单的操作编写代码,节约了开发成本和时间。

3 ORM 组件的未来

随着 ORM 组件的数据缓存、延迟加载、SQL 语句优化及批量执行等技术的成熟,它的性能和直接访问数据库方式的性能差距将越来越小。并且随着计算机硬件性能的快速提高,性能将不再是 ORM 组件的发展瓶颈。未来的 ORM 组件将向着和 VS.NET 等开发环境集成,通过图形化界面定义类结构、映射关系、自动生成相关的实体类和 XML 映射文件等方向发展,并且类似 XPO 一样支持根据 class 直接生成数据库 schema 功能的 ORM 组件将成为主流。这类 Class→DB 的 ORM 组件将能和 UML 等建模工具集成,根据建模工具设计的类结构自动生成数据库表结构。这就不仅将程序开发者从繁琐的数据库设计中解脱出来,而且还从业务逻辑代码的编写中解脱出来,使他们能真正以面向对象的层次设计软件而无需关心底层的实现细节[5]。

参考文献:

[1] 飞 鹰. 什么是 O/R Mapping? [EB/OL]. http://blog.aspcool.com/tim/archive/2004/09/17/1109.aspx., 2004-09-17.

[2] Ambler S W. Robust Persistence Layer For Relational Databases [EB/OL]. http://www.ambysoft.com/persistenceLayer.pdf,1999-10-09.

[3] Fowler M. 企业应用架构模式[M]. 王怀民,周 斌,译. 北京:机械工业出版社,2004.

[4] Open Source nhibernate development group. What is NHibernate? [EB/OL]. http://nhibernate.sourceforge.net/, 2003-02-11.

[5] Progame. 功夫、Persistore 及其它[EB/OL]. http://www.cnblogs.com/progame/archive/2004/12/26/82133.html, 2004-12-26.

(上接第 7 页)

5 结束语

文中采用的过滤模型由于建立在 WCCP 框架中的缓存服务器上,所以充分利用了 WCCP 的优点——对于用户的重复请求利用缓冲机制予以处理,同时信息过滤可以在用户完全不知觉的情况下进行。

该模型在过滤信息上具有以下 3 个特点。首先是采用了两层过滤方式,传统的 URL 过滤方式需要专人维护非法 URL 表,而且对非法 URL 的覆盖率也不高,传统的内容过滤方式对每次用户请求后从 Internet 上得到的页面都要进行一遍内容过滤,拖延了请求响应时间,而文中提出的过滤模型避免了 URL 过滤与内容过滤的缺点,同时也实现了 URL 过滤的高效率和内容过滤的全面性,具有较好的性能;其次在 URL 过滤上,传统的 URL 过滤方法只是将用户请求中的 URL 与数据库中的非法 URL 表中的记录进行简单的比较,效率偏低,而该模型结合了缓存机制及合法 URL 和非法 URL 双表过滤机制,提高了 URL 匹配的速度,进一步优化了 URL 过滤性能;最后对于内容过滤该模型也进行了改进,针对传统内容过滤大多进行的是文本过滤,对信息的过滤不够完全,该模型采用了网络爬虫分析网页确定该网页的向量特征的方法,对网页合法性的确定更加合理。

参考文献:

[1] Sripada S, Reiter E, Hunter J, et al. A Two-stage Model for Content Determination[A]. In Proceedings of the 8th ACL-EWNLG'2001[C]. Toulouse, France: [s.n.], 2001. 3-10.

[2] Sublime Solutions Pty Ltd. Squid Cache Transparency and WCCP Version 1.1 [EB/OL]. http://www.sublime.com.au/squid-wccp. 2001.

[3] Wessels, Duane. Web 缓存(英文影印版)[M]. 北京: 清华大学出版社, 2002.

[4] Hammami M, Chahir Y, Chen Liming. WebGuard: Web Based Adult Content Detection and Filtering System [A]. In Proceedings of the IEEE/WIC International conference on web Intelligence (WI'03)[C]. Halifax, Canada: [s.n.], 2003. 574-578.

[5] Oard D W, Marchionini G. A Conceptual Framework for Text Filtering[Z]. Technical article CS-TR-3643. Maryland: University of Maryland, 1996.