基于双数组有限状态机的URL访问控制算法

丁振国,赵大勇
DING Zhen-guo,ZHAO Da-yong

西安电子科技大学 计算机学院,西安 710071
College of Computer Science,Xidian University,Xi'an 710071,China
E-mail:zhdyvsswallow@163.com

DING Zhen-guo,ZHAO Da-yong.URL access control algorithm based on double-array finite state machine.Computer Engineering and Applications,2007,43(36):129-131.

Abstract: The Access Control Module in the Service Selection Gateway(SSG) is a scheme in which URL can be parsed from user request data packet,and access control and route selection can be finished according to the user access privilege.A model of improved Finite State Machine is first presented,and then described with Double-Array.Also an improved strategy that the node with most child nodes is processed firstly is presented.This algorithm not only improves search speed,but also needs a smaller space for data store than others and reduces the data sparseness.The proposed algorithm is applied to the Access Control Module.Experimental results show that this algorithm is of good efficiency.

Key words: Service Selection Gateway(SSC);URL;access control;Finite State Machine(FSM);double-array

摘 要: 业务选择网关(SSG)中的访问控制模块从用户请求数据包中解析出URL,并且根据用户的URL访问权限进行访问控制和路由选择。首先提出了改进的有限状态机模型,然后用双数组表示该有限状态机,并提出了优先处理分支结点较多的结点的优化策略。实验证明该算法不仅提高了查询速度,而且占用的存储空间也较少,进一步减少了数据的稀疏。最后将该算法应用在访问控制模块上,实践证明此算法可行、高效。

关键词: 业务选择网关;URL;访问控制;有限状态机;双数组
文章编号: 1002-8331(2007)36-0129-03 **文献标识码:** A **中图分类号:** TP393

1 引言

业务选择网关(以下简称SSG)是一种基于URL细粒度的内容选择的应用层网关,可以满足不同用户对内容的个性化需要[1]。其功能大致如下:用户将自己的服务请求提交给业务选择网关,它根据用户请求的数据包解析出URL,同时结合用户的访问信息和访问权限,对用户的访问进行认证,并对用户访问内容进行控制、分配以及计费,保证有权限的用户能够享受流畅的网络服务[2]。

业务选择网关是基于7层内容控制的网关软件,服务控制的基本单位是URL。业务选择网关从资源请求数据包中获取URL,根据访问控制列表判断请求用户是否具有该URL的权限:如果具有权限则允许资源访问;否则丢弃报文不允许用户访问。注册用户在登录后还可以进行资源定制和取消,因此访问控制算法在业务选择网关是非常重要的[3]。

本文所提出的访问控制是基于URL的细粒度的访问控制,主要是针对URL资源访问的控制,与传统的访问控制策略是有所区别的,这是业务选择网关的特定需求和功能所决定的。本文结合业务选择网关的特点提出了一种基于双数组有限状态机的高效适用的URL访问控制算法。

2 有限状态机简介

有限状态机FSM(Finite State Machine)是包含一组状态集(states)、一个起始状态(start state)、一组输入符号集(alphabet)、一个映射输入符号和当前状态到下一状态的转换函数(transition function)的计算模型。下面给出有限状态机的一般定义[4]。

定义1 $M=(Q,\Sigma,f,S,Z)$,其中,Q是有限状态集合,在任一确定的时刻有限状态机只能处于一个确定的状态;Σ是有限输入字符集合即字母表,在任一确定的时刻有限状态机只能接收一个确定的输入;f是状态转换函数,是在$Q\times\Sigma->Q$上的映像,$f(q_i,a)=q_j$在某一状态下给定输入后有限状态机将转入由状态迁移函数决定的一个新的状态;$S\in Q$是初始状态,有限状态机由此状态开始接收输入;$Z\in Q$是终结状态集合,又叫可接受状态或结束状态,有限状态机在达到终态后不再接收输入。有限状态机通常用图来表示,图中节点代表状态,有向弧表示迁移关系,输入标注在相应的关系弧旁边。

基金项目: 国家高技术研究发展计划(863)(the National High-Tech Research and Development Plan of China under Grant No.2004AA1Z2520);国家部委预研基金项目。

作者简介: 丁振国(1959-),男,博士,教授,博士生导师,主要研究领域为计算机网络与信息处理;赵大勇(1979-),男,硕士研究生,主要研究领域为计算机网络与信息处理。

3 访问控制算法的设计与实现

3.1 访问控制机制的总体概述

业务选择网关只允许登录用户与网关进行三次握手建立TCP连接。在建立连接之后,用户进行资源访问请求,网关获取访问资源的URL,根据用户的资源访问权限表判断登录用户是否具有该URL资源访问权限。

业务选择网关访问控制列表包括两个部分:登陆用户表和用户的资源权限表。登陆用户表主要用来保存登陆用户的IP地址,作为登陆用户判断的标识;资源权限列表保存了对应用户所拥有的URL权限,每一个登录用户都具有一个权限列表。访问控制列表的结构如图1所示。

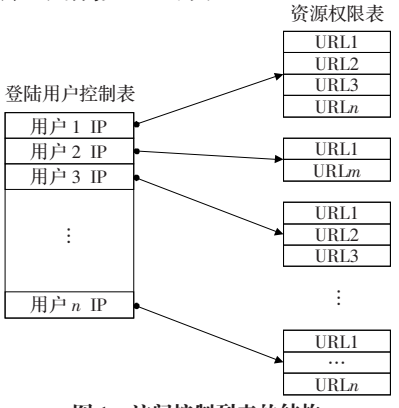

图1 访问控制列表的结构

随着用户数量增多和其资源权限表的增大,快速查找登录用户是否具有某个URL资源访问权限就会成为系统性能的瓶颈。

3.2 访问控制算法的设计

3.2.1 业务选择网关URL匹配规则及其特点

业务选择网关URL的匹配规则如下:(1)最短前缀匹配。如:用户具有权限的URL为:http://xidian.edu.cn/note,那么表示用户拥有访问/note文件目录下所有资源的权限,比如用户请求资源http://xidian.edu.cn/note/t1.rar,查找到用户具有权限的URL为http://xidian.edu.cn/note,则查找成功,因为http://ne.xidian.edu.cn/note/t1.rar http://ne.xidian.edu.cn/note,所以则允许资源访问;(2)不区分大小写,所以需将所有的字符转化为小写字符,然后再进行匹配查找。

3.2.2 构造改进的有限状态机

由于业务选择网关的URL匹配的特殊性,经典的多模式字符串匹配算法并不适合于应用到这里,所以要设计适合于业务选择网关的多模式字符串匹配算法。本文借鉴AC(Aho-Corasick)算法的思想,提出了一种基于改进的有限状态机的URL资源访问控制算法。运用用户所有权限的URL关键字来构建一个有限状态机,然后使用该有限状态机来处理需要查询的文本串(URL)。这个算法包含两个部分:第一部分使用所有的URL来构建一个确定的有限状态自动机;第二部分把待查询的URL作为状态机的输入,进行匹配查询。

假设$k=\{url_1,url_2,url_3,\cdots,url_n\}$是一批有限的某个用户权限资源表的所有URL关键字的字符串,并且给定*url*是将要调用的一个URL文本字符串。本算法所要解决的问题是如何快速查找出*url*在集合k中是否存在其最短前缀的子字符串。这个有限状态机是由一批状态所构成的,每个状态用一个数来描述。状态机将以持续流的方式读入*url*中的每个字符。有限状态机的行为只有两个函数:跳转函数*goto*和输出函数*output*。下图2表示了一个按关键词字符串{*http*,*html*,*www*}所建立的有限状态机。

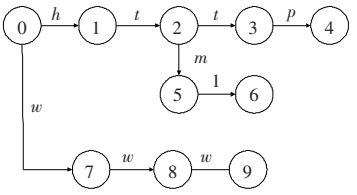

图2 有限状态机构造图

一个状态(通常是0)被指示为开始状态。在图中这些状态为0,1,…,9。*goto*函数g是当前状态和输入字符构成的一个数据状态。图中直接描述了这个*goto*函数。例如从0-1边上的记号h表示$g(0,h)=1$。没有箭头指示的输入表示失败。因而$g(2,\alpha)=fail$,α表示t和m以外的任意符号。当需要被处理的文本串到来的时候,文本串逐次通过状态机,每输入一个字符后,状态机都会对应一个状态输出。

如果s是当前状态机的状态,α是状态机当前输入的一个字符,若$g(s,\alpha)=r$,则状态机执行*goto*转换,状态机将进入r状态,并且*url*字符串中的下一个符号变成当前的符号。*output*函数表示匹配到该状态时是否匹配成功:如果值为1则表示匹配成功;如果为0则表示匹配失败。此状态机的*output*函数值如下:

表1 状态机的输出函数表

i	1	2	3	4	5	6	7	8	9
$output(i)$	0	0	0	1	0	1	0	0	1

这里可以看出:当退出匹配时,只有状态机处于状态4、6和9的时候才是匹配成功,处于其他状态时都是匹配失败。这里状态机退出匹配的条件有两种:一种是当跳转函数$g(s,\alpha)=fail$时;另外一种是当输入匹配的*url*处于字符串末尾(即再无字符读入状态机)时。当状态机退出匹配时,如果状态机处于s状态,那么就要判定一下$output(s)$的值:如果为1,则表示匹配成功,即*url*在k集合中能找到其前缀的子字符串,具有访问资源的权限;如果为0则表示匹配失败,不存在其前缀的子字符串,也就没有访问资源的权限。

3.2.3 双数组表示有限状态机

传统上的有限状态机一般用转换表方式来实现,表的列代表自动机的不同状态,行代表转换变量,但是对于URL查询来说,转换表的问题是数据稀疏导致严重的空间浪费,其空间复杂度为$O(n^2)$。为了让有限状态机的实现算法在占用空间较少的同时还要保证查询的效率,采用双数组(Double-Array)[5,6]来表示有限状态机。以图2为例来说明双数组状态机的构造和查询算法。

双数组(Double-Array)由两个整数数组构成,一个是*base*[];另一个是*check*[]。*base*数组中的每一个元素相当于状态机的一个结点,其值作为状态转移的基值。*check*相当于当前状态的前一状态,相当与校验值,用来检验该状态是否存在。设数组下标为i,如果*base*[i]、*check*[i]均为0,表示该位置为空;如果*base*[i]为负值,表示该状态为合法的URL。对于从状态s到状态t的一个转移,必须满足:$check[base[s]+c]=s$,$base[s]+c=t$,其中c是输入变量。

首先对出现的所有的字符进行编码:h-1,w-2,t-3,m-4,p-5,l-6,然后在此基础上构建双数组,先遍历一遍状态机,找

到所有的第一个字符为 h 和 w 字符串，h 和 w 的序列码分别为 1 和 2，找到下标为 1 和 2 的元素，令 $check[1]=0$，$check[2]=0$，$base[1]=1$，$base[2]=2$。初次的构建结果如表 2 所示。

然后做第二次遍历，找到 h 开头的词的第二个字符为 t，这个字符的编码为 3，因而需要找到对 h 状态的 $base$ 值进行修改，找到一个值 $B>2$，使得 $base[B+3]=0$，$check[B+3]=0$，即保证在 $base$ 基础上，加后续字符的序列码所跳转到的后状态为空。这里很容易得到 $B=3$，于是令 h 的 $base$ 值 $base[1]=3$，并将 t 放入双数组的状态 5 中，并令 $check[5]=1$，即表示这个状态的前一状态为 1。对 w 做同样的处理。得到的数组如表 3。

表 2　双数组状态机经历初次遍历构建结果

下标	1	2	3	4	5	6	7	8	9	10	11
Base	1	2	0	0	0	0	0	0	0	0	0
Check	0	0	0	0	0	0	0	0	0	0	0

表 3　双数组状态机第二次遍历后构建结果

下标	1	2	3	4	5	6	7	8	9	10	11
Base	3	3	0	0	0	0	0	0	0	0	0
Check	0	0	0	0	2	1	0	0	0	0	0

根据上述原理，依次类推，经历四次遍历，可以将所有的字符串作为不同的状态放入数组中，完成数组的构建。最后，还要再遍历一次状态机，修改 $base$ 值的符号，用负的 $base$ 值表示该位置为匹配成功位置，即用户具有该状态的 URL 访问权限。如果状态 i 对应的某个字符串，那么令 $Base[i]=(-1)\times Base[i]$。得到双数组如下：

表 4　双数组状态机最终构建结果

下标	1	2	3	4	5	6	7	8	9	10	11
Base	3	3	0	0	7	4	5	5	−9	−10	−11
Check	0	0	0	0	2	1	6	6	5	7	8
前缀	*h*	*w*			*ww*	*ht*	*htt*	*htm*	*www*	*http*	*html*

其中 *www*、*http*、*html* 的 $base$ 值为负的状态表示合法的 URL，而其他的状态都是中间状态。用上述方法生成的双数组，将每个前缀字符串都视为一个状态，每个状态对应于数组的一个下标。查询时相当于从一个状态找到另一个状态。例如查询 *www*，先根据 w 的序列码 2，找到状态 w 的下标为 2，再根据第二个 w 的序列码 2 找到 *ww* 的下标 $base[2]+2=5$，同时根据 $check[5]=2$，表明 *ww* 是某个 URL 的一部分，可以继续查询，然后再找到 *www* 的下标为 9，此时 $base[9]<0$，$check[9]=5$，表明 *www* 在其 URL 权限表中，查询完毕。

为了描述方便，本文先假设 *url* 是待匹配的字符串。查询算法的语言描述如下：

```
char[ ] ur;     //待匹配的 url
int r=0;        //r 为当前状态
//逐次输入 url 的每个字符
for(i=0; i<url.strlen( ); i++)
{
    int b=code;     //code 为 url[i]的编码
    int d=base[r]+b;
 //如果状态转移失败则退出匹配
  if(check[d]! =r)
        break;
 else
        r=d; //改变当前状态;
}
if(base[r]<0)   //表示匹配成功
    return ture;
else            //表示匹配失败
    return false;
```

3.2.4　增加空间利用率的策略

考虑到在双数组状态机算法中，每个节点在数组中的位置都是由其父结点也就是上一状态的 $base$ 值决定。而一个结点的 $base$ 值的确定也取决于数组的当前空闲位置以及该结点的直接子节点。一个结点的直接子结点越多，该结点在找 $base$ 值时所遇到的冲突也就越多。因此优先处理分支较多的结点，有利于减少冲突，避免数组增长过大，减少数据稀疏。这里采用优先遍历大分支的广度优先遍历的策略来构建双数组，即在广度优先遍历的基础上，在同一层次上的分支采取直接子结点多的优先遍历。对于图 2 的遍历顺序为 1–7–2–8–3–5–4–6–9，这里 2 的直接子结点比 8 的多，所以先遍历 2 结点。

3.3　算法的复杂度分析

双数组状态机构造完成之后，查询实质上就是将待匹配的字符串的各字符分别转换为相应的序列码，然后作几次加法，即可查到相应的字符串。该算法的查询避免了字符串比较、copy 运算等步骤，直接进行数值计算和数组读取，因而时间上比其他查询算法都要快，查询效率是极高的。其查询的复杂度为 $O(n)$，其中 n 为 $min(url$ 的长度，集合中 url 最短前缀的长度$)$。

4　算法的性能测试和结果分析

4.1　测试环境

CPU：Intel Pentium 2.4 GHz，内存：512 M，操作系统：RedHat9。

4.2　测试数据和结果分析

为了测试算法的实际性能，业务选择网关系统选择分别含有 1 000、5 000、10 000 和 20 000 个 URL 的四组用户的资源权限表，分别测试了查询所有 URL 的匹配时间。统计了 Wu-Manber、AC(Aho-Corasick)、Hflp 哈希散列[7]和本算法的查找时间见表 5，通过表 5 可以更清楚地看出各种算法之间的关系。

表 5　处理时间　　s

URL 数	Wu-Manber	AC	Hflp 哈希	本算法
1 000	0.14	0.26	0.094	0.09
5 000	0.68	1.29	0.470	0.41
10 000	1.79	2.65	1.040	0.86
20 000	3.65	5.41	2.130	1.75

由表 5 可以看出基于双数组的有限状态机的 URL 查找算法比其他算法查询速度都要快，经典的多模式字符串匹配[8]算法 Wu-Manber 和 Aho-Corasick 算法对本文的 URL 查找其查询效率比较低；Hflp 是比较流行的 URL 散列函数，URL 查询效率也没有本算法高，而且 URL 的数量越多，Hflp 哈希的冲突就越多，而且平均查找长度也就越长，本算法的优势就也越明显。

另外在实验中还记录了测试数据为 10 000 个 URL 时转换表和未优化的双数组实现的有限状态及采用优化策略的双数组所占用的空间大小：转换表实现的占用空间为 4.194 M，而未加入优化策略的双数组和加入优化策略的生成数组的长度分别为 152 657 和 131 008，这样它们各自占用的空间为 152 657×2×4=1 221 256(bytes)≈1.221 M 和 131 008×2×4=1 048 064(bytes)≈1.048 M。从实验数据中可以得出加入优化策略的双数组占用的空间仅为转换表的 29%，也比未加入优化策略的双数组减少 14%的空间。所以本算法无论是从时间复杂度还是从空间复杂度来看都是比较适合应用到业务选择网关中。

(下转 140 页)