Journal of Computer Applications
计算机应用,2012,32(3):609 -613
ISSN 1001-9081
CODEN JYIIDU
2012-03-01
http://www.joca.cn

文章编号:1001 -9081(2012)03 -0609 -05
doi:10.3724/SP.J.1087.2012.00609

基于多核处理器的 L7-Filter 规则匹配改进算法

余　涛*,吴卫东
(武汉科技大学 计算机科学与技术学院,武汉 430065)
(*通信作者电子邮箱 yutao_2006@126.com)

摘　要:针对多核处理器的体系结构和网络数据流在时间上的局部性特点,提出了一种基于多核处理器的分链动态适应算法。该算法通过对网络数据流进行类型分类并根据网络数据流的时间局部性对规则链进行动态优化,从而有效减少了多核处理器下 L7-Filter 对网络数据流的匹配次数,显著提升了规则匹配效率。仿真实验结果表明:在网络数据包个数相同条件下,所提算法在性能上约有 7% 的提高。随着网络数据包个数的增加,性能优越性更加明显。

关键词:多核处理器;网络数据流;L7-Filter;时间局部性;数据包分类;动态优化
中图分类号:TP301.6;TP393.02　　**文献标志码**:A

Improved L7-Filter's pattern matching algorithm based on multi-core processors

YU Tao*, WU Wei-dong
(*College of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan Hubei* 430065, *China*)

Abstract: According to the architecture of multi-core processors and the temporal local characteristics of network data flow, a division and dynamic adaptation algorithm was proposed based on multi-core processors. Classifying network data flow by the type and optimizing chain of rules dynamically by the temporal locality of network flow, the count of the multi-core's L7-Filter matching network data flow were reduced effectively and the processing efficiency was improved dramatically. The simulation result shows that given the number of packets in the same conditions, the algorithm has about 7 percent improvement of the multi-core processing performance. With the increasing number of network packets, the performance superiority becomes more obvious.

Key words: multi-core processor; network data flow; L7-Filter; temporal locality; packet classification; dynamic optimization

0　引言

随着网络爆炸式的增长以及高速以太网的出现(如 10 GbE),网络流量也随之高速增长,这就对网络服务质量(Quality of Service, QoS)[1]提出了更高的要求。传统的数据包分类技术主要是基于数据包头部信息做出分类决策。然而,当前的许多网络应用会有意或无意利用头部信息隐藏真实的行为,如对等网(Peer-to-Peer, P2P)和超文本传输协议(HyperText Transfer Protocol, HTTP),它们会在连接建立的过程中通过动态分配端口号等方式来调整 IP 头部信息。

L7-Filter 是 Linux 上 QoS 框架的一个重要组件。它区别于传统的基于数据包头部信息的分类方法,采用了深度数据包检测(Deep Packet Inspection, DPI)[2]技术,是基于应用层数据来对数据包分类从而提高网络服务质量。单核心处理器下的 L7-Filter 无论是在处理性能还是硬件资源利用率上效果都很理想。

然而随着网络服务器中多核处理器的大规模部署[3],多核处理器下的 L7-Filter 并没有表现出应有的性能提升。针对这些问题,文献[4]提出基于接收端调节(Receive Side Scaling, RSS)[5]技术的核心绑定技术,很好地解决了多核处理器上因多核心计算迁移所产生的 Cache 一致性问题[6],提高了多核处理器下的 L7-Filter 数据包分类处理性能。但是,这种方法在具体到每一个匹配核心上采用的仍然是传统的线性搜索查找算法,时间复杂度为 $O(n)$,最坏情况下要匹配 $N \times d$ 次计算才可以得出分类结果(其中 N 为规则条数,d 为规则维数),这种方法在时间效率方面存在固有的缺陷。文献[7]提出的对规则集动态优化只对规则集本身进行探讨,但却没有充分考虑网络数据流本身的特性。文献[8]虽然介绍了一系列规则集优化算法来提高核心的处理性能,但存在实现复杂度较高等问题。

本文利用网络数据流在时间上的局部性特点,并结合多核处理器的特性,对网络数据包分类算法进行改进,提出了一种基于多核处理器的分链动态适应算法(简称 DDA 算法)。

1　L7-Filter

L7-Filter 是基于网络应用层数据对数据包分类的方法。核心思想是利用正则表达式搜索的方法[9]对目标数据包的应用层数据进行搜索并分类,然后将分类的结果交给系统上层为服务质量作进一步处理,如带宽分配等。多核处理器下的原始 L7-Filter 体系结构如图 1 所示。

文献[4]提出的核心绑定技术中采用了 RSS 技术。利用这种技术,在一个 CPU 核心上绑定预处理线程,如图 1 中的①。该线程主要做包括数据包在多处理器核心上的调度等工作。预处理核心在把数据包向匹配核心调度时的单位是以单个连接为单位的。系统其余的每个处理核心上都绑定一个数据包分类线程,如图 1 中的②。

这种利用 RSS 技术的方法很好地解决了由操作系统对负载强制调度及负载均衡调度带来的 CPU 核心间 Cache 拷

收稿日期:2011-08-30;**修回日期**:2011-11-16。　**基金项目**:湖北省教育厅科技项目(D20101105)。
作者简介:余涛(1985 -),男,湖北郧县人,硕士研究生,主要研究方向:多核处理器应用算法、计算机网络、嵌入式系统;　吴卫东(1964 -),男,湖北武汉人,副教授,博士,主要研究方向:多核处理器应用算法、计算机网络。

贝产生的对 L7-Filter 性能影响的问题。

然而,文献[4]的改进算法具体到每个匹配线程(核心)时,采用的是将规则以正则表达式的方式描述并将所有的规则组织成一条规则链表。每当对一个数据包做匹配操作时,就线性地从规则链头搜索到链尾,明显地存在着效率不高的弱点。另一方面,在数据包分类处理中也没有充分利用规则链本身的特性以排除肯定不符合的匹配操作,这样就会无故地增加核心的计算次数从而浪费 CPU 资源。特别是当网络流量比较大时,这两点不足更加明显。

通过分析传统的查找算法,可以归纳出它具有以下两点关键特征:1)待查找的数据有明确的值大小,从而确立与比对目标值的大小关系;2)比对目标之间存在着明确的值大小关系以建立查找模型,如二分查找树等。

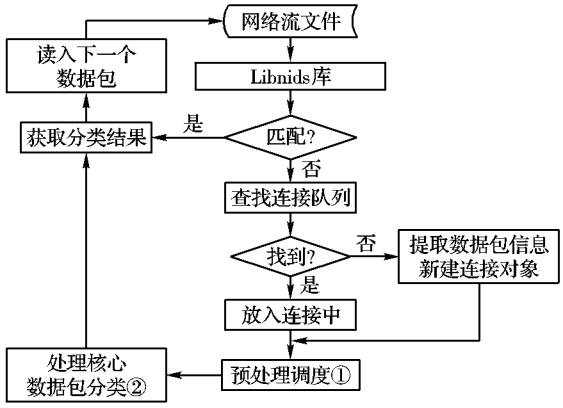

图 1 L7-Filter 数据流程

然而在 L7-Filter 模型中,规则采用的是正则表达式的,其本质上是一个字符串,数据包也是如此,这两者都不具备上述两个关键特征。基于这样的事实,决定了传统的查找算法已不再适合 L7-Filter 的模型。

2 DDA 算法模型

2.1 时间流指数

在 L7-Filter 体系结构中,对内核网络栈的数据包的调度是以连接为基本单位的,定义 1 给出了连接的定义。

定义 1 连接是由数据包的传输层协议、源地址、源端口、目的地址、目的端口组成的唯一字符串,其特性有:1)区别传统连接的概念,即不以 TCP 和 UDP 来划分是否为连接;2)连接不区分方向性,即源地址、源端口和目的地址、目的端口的地位是对等的,如图 2 所示。

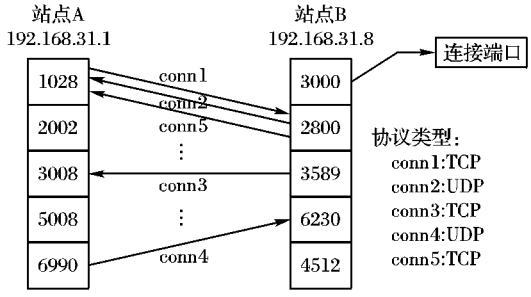

图 2 连接定义图示

在图 2 所示中,由定义 1 可知共有 5 个连接字符串:

conn1: TCP:192.168.31.1:1028 192.168.31.8:2800

conn2: UDP:192.168.31.1:2800 192.168.31.8:1028

conn3: TCP:192.168.31.1:3589 192.168.31.8:3008

conn4: UDP:192.168.31.1:6990 192.168.31.8:6230

conn5: TCP:192.168.31.1:2800 192.168.31.8:1028

根据定义 1 易知,conn1 和 conn5 是同一连接,由于 conn1 和 conn2 在协议上不同,所以不是同一连接,其他同理。

另外,通过对实际网络流量的观察以及对入侵检测数据集 MIT DARPA 2000[10] 的统计分析得出互联网流量在一定的时间内表现出一定的局部性。

这种局部性可以通过定义 1 连接的概念对网络数据流进行统计并计算其时间流(Time-Flow, T-F)指数以获得定量分析。定义 2 给出 T-F 指数的定义。

定义 2 T-F 指数。在单位时间 t 内,流经该网络设备接口 j 上的流量均值 f 与该设备上的路由条目数均值 ε 和该设备上活动端口平均数乘积的比值,T-F 指数可由式(1)计算:

$$F(j) = \frac{1}{tn}\sum_{j=0}^{n}\frac{f(j)}{\varepsilon(j) * p(j)} \tag{1}$$

其中:$f(j)$、$p(j)$ 和 $\varepsilon(j)$ 分别表示单位时间 t 内,网络设备 j 上流量均值、路由条目均值和该设备使用中活动端口均值;n 为该网络设备上物理网卡的总数。

通过文献[11]及 T-F 指数的计算易知,这种流量的局部性表现在:在一段时间范围内,适用于同一网络应用协议的流量比较集中,这些流量在数据包规则匹配上的表现就是频繁地命中某一小部分特定规则。

2.2 Linux 的 Slab 层技术及多处理器核心亲和性

多核心处理器的亲和性[12] 是指在处理器核心计算过程中,相关的任务尽可能地位于同一核心进行计算,这样就可以减小进程迁移使数据在核心间拷贝所带来的负载。实现进程固定于某一特定 CPU 的技术叫 CPU 硬亲和力。在 Linux 2.6 内核中,通过内核提供的接口可以很容易地把进程(或线程)固定在某个特定的处理器核心上运行。

在 Linux 内核实现中实现了一种叫作 Slab[13] 层的技术。该技术主要是为了减少数据对象频繁高速地分配和回收对系统性能的影响,其思想是让内核把不同对象分为不同的高速缓存组,每个组中都只包含一种对象,每种对象也仅只对应一个高速缓存组。利用内核所提供的任务与核心绑定接口将每个高速缓存组绑定到一个固定的核心上,这样就可以解决在多核处理器上操作系统对任务强制调度和负载均衡所带来的 Cache 一致性问题。

另外,由于受到对象数据缓存组大小的限制,网络中的一个数据包可能会被划分为几小段以连接的方式存放在对应的对象数据缓存组中,这样就使得对一个数据包的匹配分类可能需要进行几个阶段,最后的结果由这几次计算的结果综合得到。这时基于连接的网络数据包对象数据缓存组使得核心每次处理时都是处理来自同一连接的数据包(段),而核心绑定技术的思想又可以保证同一连接的数据包(段)都尽量只在一个处理核心上进行计算,通过这两个处理就很好地解决了在多核处理器上由于操作系统对任务强制调度和负载均衡所带来的 Cache 一致性问题。

本文中以式(1)计算的 T-F 指数和 Linux 下的 Slab 层技术为理论基础,结合多核心绑定技术为每一个核心绑定一个数据包匹配线程,而被固定的每个线程都对应一个数据资源链表,该链表即为 Slab 层技术中的对象数据缓存组,存放的是来自内核网络栈中钩子函数所发送过来的网络数据包。

2.3 DDA 算法实现

基于 2.1 节的 T-F 指数和 2.2 节的 Slab 层技术的理论基础,给出基于 Linux 下的 Slab 技术和多处理器核心亲和性的

新 L7-Filter 体系结构如图 3 所示。

在图 3 所示的 DDA 算法核心中主要分为两个步骤:1)数据包在多核心上调度分配,见图 3 中的①;2)处理核心规则链动态适应,见图 3 中的②。

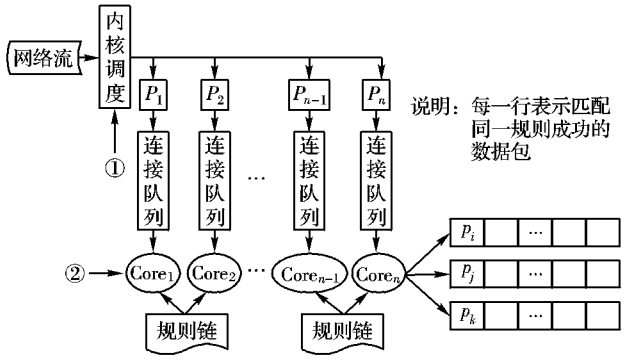

图 3　基于 DDA 算法的 L7-Filter 体系结构

2.3.1　多处理核心数据包分链(DIV)算法

在图 3 的第①步中,当有网络流过来时,通过内核钩子程序就可以很容易地以固定字节大小的形式获取网络流中的数据包。然而,不足在于文献[2]中并没有充分利用网络流中的数据包的运输层协议特性。因为规则链中的规则可以根据将要匹配成功的数据包的运输层协议对其进行分类,而网络流中的数据包也可以基于运输层特定性进行分类。经过这样的分类后这样就可以明显减少匹配次数。对数据包和规则分类前后如图 4 所示。

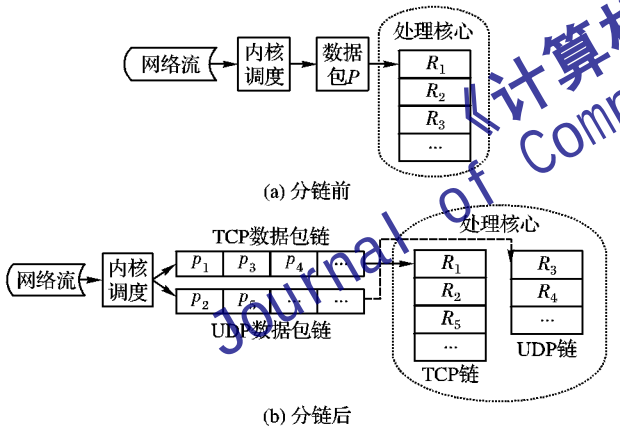

图 4　数据包及规则链分类

从图 4(a)可发现,从网络流来的数据包无论是 TCP 类还是 UDP 类都要经过处理器核心中规则链的所有规则处理,而规则链中 TCP 类规则不可能和 UDP 类数据包匹配成功,同理易知其他情况。这样就浪费了处理器的处理时间。

而在图 4(b)中,DIV 算法的核心思想就是利用 Libnids[14] 库的不同回调函数对数据包按照运输层协议进行分类调度,同时对处理器核心中的规则进行分类,这样就可以保证 TCP 类的数据包只和 TCP 规则链中的规则进行匹配计算,UDP 同理。

关于 DIV 算法的性能分析,本文作如下假设:在对数据包按运输层协议分类时,数据包 P_i 为 TCP 类数据包的概率为 P_{TCP},为 UDP 类数据包的概率为 P_{UDP}。图 4(a)的规则链长度为 N,图 4(b)中 TCP 规则链长度为 N_{TCP},UDP 规则链长度为 N_{UDP},则 N、N_{TCP} 和 N_{UDP} 三者关系为:

$$N_{TCP} + N_{UDP} = N$$

且有

$$P_{TCP} + P_{UDP} = 1$$

那么在最坏情况下,图 4(a)未使用 DIV 算法的匹配次数 E_{old} 为:

$$E_{old} = P_{TCP} * N + P_{UDP} * N = (P_{TCP} + P_{UDP}) * N = N$$

而在图 4(b)使用 DIV 算法后的匹配次数 E_{new} 为:

$$E_{new} = P_{TCP} * N_{TCP} + P_{UDP} * N_{UDP} \leqslant P_{TCP} * N + P_{UDP} * N \leqslant N = E_{old} \quad (2)$$

通过式(2)的分析可知,DIV 算法可以明显减少处理器核心的比较次数,特别是在 TCP 和 UDP 规则比较平均时。另外,无论是对数据包分类调度还是对规则链规则的分类都只是发生在系统初始化阶段,这部分工作对后续的规则匹配没有任何性能上的损失。

另外一个问题就是数据包在核心间的调度问题。数据包调度问题就是为数据包查找其所属连接队列的问题。对于一个系统而言,处理器核心数目较小且固定,当一个数据包从网络流中被调度出来后就提取数据包头部信息组成连接串(见定义 1)并在核心间顺序查找对应的连接对象缓存组。同时每个连接对象缓存组都维护着自己特有的连接字符串。当两者连接字符串相同时就将数据包放入该连接对象缓存组,这个连接对象缓存组是匹配线程的数据资源,匹配线程又和核心绑定在一起,这样就完成了数据包在多核处理器上的调度过程。一旦在所有的处理核心上都找不到这样的连接对象缓存组时,就在负载较少的核心上新建这样一个对象缓存组并附接到负载较少核心的对象缓存组上。

2.3.2　多处理核心规则链动态适应(MDYA)算法

从 2.2 节介绍的 Slab 层技术和核心亲和性可知,当利用核心绑定技术为多核处理器核心绑定一个数据包处理线程后,处理的过程如图 3 中的②。核心 Core1 和 Core2 共享一条规则链。通过具体分析每个处理器核心上数据包的分类过程发现:模式规则的匹配过程,本质上仍然是一个利用规则链线性查找对该数据包所属应用程序进行分类的过程,不同点在于这里采用了正则表达式的查找方式。

根据 T-F 指数的计算式(式(1))可知,网络数据流在一定的时间范围内表现出一定的局部性,即在一定时间内数据包隶属于同一网络应用协议(对应于规则链的规则)有较高的概率。所以 MDYA 算法的核心思想就是通过对规则进行基于规则权重因子的计算动态调整图 4 中(b)的规则链,规则权重因子较大的规则应尽量放在规则链的前面,这样就可以相对减少查找比较的次数。

MDYA 算法提到的规则权重因子的计算需考虑 3 种因素:1)规则命中频率 ρ,即规则近期被命中的概率(初始为 0);2)命中时效性 γ,即数据包命中规则距上次命中经历的时长(以未命中次数递增计,初始为 0);3)规则当前位置值 τ(初始值为以字典序构建规则树时获得的位置值,从 1 开始计)。规则权重因子计算公式如式(3)所示:

$$W(i) = \left\lceil \left(\frac{n}{\tau_i} - \gamma_i \right) * \rho_i \right\rceil \quad (3)$$

其中:规则当前位置值 τ 越靠前其位置值较越小,最小值为 1。当命中规则 i 时,ρ_i 按照式(4)进行修正:

$$\rho(i) = \left(\rho(i) + \frac{1}{n} \right) * F(j) \quad (4)$$

其中 $F(j)$ 为计算 T-F 指数。运用式(4)对 ρ_i 进行修正的原因是为了加快调整的速度,以使算法能尽可能快地产生效果。

基于规则权重因子的 MDYA 算法使得规则在链中局部

模糊有序。简化模型如图 5 所示，在 TCP 规则链中，规则 R_1 与规则 R_3 满足如下关系式：

$$| W(R_1) - W(R_3) | > \varepsilon$$

其中：$W(R_1)$ 是关于规则 R_1 的权重因子计算函数（如 2.3.2 节式(3) 所述），ε 是相关于当前流经网络设备的网络连接数的权值。当相邻两条规则满足上述关系式，就预测 R_1 在当前具有更高的访问权重，继而将该规则向前移动 1 位。

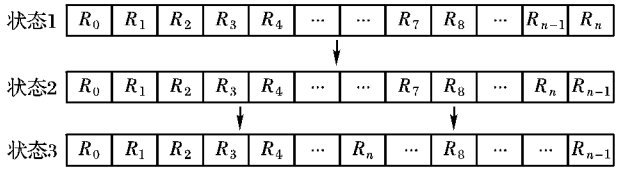

图 5 规则学习模型

在简化模型中，为了降低分析难度，假设 ε 的值取 1，权重因子计算函数只计算访问频数。如简化模型中所述，在前 N 次数据包模式匹配操作中对 N 个规则的每一个规则 R_i 都访问了一次，这时每个规则的权重因子都为 1。假设在接下来的 $N/2$ 次规则匹配中都对规则 R_n 匹配成功，这使得规则 R_n 的权重因子变为 $N/2$。由于这里 ε 的值为 1，而过去的 $N/2$ 次规则匹配操作中，其余的规则无一次命中，所以它们的权重因子都仍为 1，这样就使得每次匹配成功后规则 R_n 在规则链中的位置相对向前移动一位。

递推易得：经过 $N/2$ 次学习，规则 R_n 的位置就由规则链的最后一位移到了中间位置，同时这样的移位学习操作的结果是在每一步中进行，学习的结果立即就能被后面的规则匹配操作所利用。随着网络数据包的不断计算，规则链最终会逐步有序，最后在一定的时间内达到稳定。

另外，在多核处理器调度算法设计[15] 中，并发与临界区加/解锁对数据的一致性和系统性能起着至关重要的作用。MDYA 算法中，为了解决因每个处理器核心独享规则树带来的空间复杂度急剧增加（可能从 1 增加到 n）的问题，采用每两个核心共享一棵规则树的方法。

MDYA 算法核心思想中，采用的是将当前规则与其前面相邻一位（非全局）规则的权重因子进行比较（局部模糊有序）调整，同时每两个核心共享一条规则链，这样使得学习后的规则能立即在多处理器核心间进行共享优化结果并相互学习。MDYA 算法的可行性在于：1）局部模糊有序操作的时间代价远远小于精确有序。就平均时间性能最佳的快速排序而言，每做一次规则匹配操作就意味着要消耗掉 $O(n \log n)$ 的时间用在规则精确排序上，这对于网络数据包规则匹配操作本身而言，是很大的时间消耗；2）计算出来的规则精确有序仅仅是对过去数据包计算出的 T-F 指数的一种体现，该指数对未来的数据包匹配有一定的指导意义，但这种意义并不是绝对的；3）在多核处理器中，若采用精确有序的排序操作将会使得对临界区资源加/解锁的频率变得异常频繁，这将严重影响多核处理器的处理性能。

在上述简化模型中，定义临界区为调整规则位置段，对临界区资源的访问采用 GetLock 和 ReleaseLock 这样的加/解锁的方式实现共享读和互斥写。

DDA 算法正是结合 DIV 算法和 MDYA 算法。DDA 算法可明显提升多核处理器的数据包匹配性能。下面给出 DDA 算法的流程。

算法 基于多核处理器的分链动态适应 DDA 算法。
输入 1）网络数据流；2）规则库。
输出 分类的数据包。

```
1)  从规则库中读取所有规则并按规则对应的运输层协议进行分类组成不同的规则链
2)  从网络数据流获取一个数据包 Package
3)  利用 Libnids 库接口获取数据包的运输层协议标志 P
4)  根据数据包的运输层协议标志 P 获取对应规则分链树的规则链 pp
5)  while(GetR(pp))
    // GetR(RL):从规则链 RL 中获取位置 pp 处规则字符串 r
6)      Re = Match(r,p);          //利用正则表达式对数据包 P 做正则匹配
7)      if(Re)                    //Re 非 0,表示匹配成功
8)          if(W(r) > W(r-1))
                //利用式(3)计算当前和上一条规则的权重因子
9)              GetLock(lock);            //获取独享锁
10)             CPoistion(r,r-1);
                //调整规则 r 和 r-1 的相对位置
11)             ReleaseLock(lock);        //释放独享锁
12)         设置数据包的标记值,转 2)
13)     pp++,转 5)
14) 丢弃数据包 Package,并转 2)
```

3 算法性能分析

3.1 实验环境配置

软件环境为 Linux 2.6.32 内核，GCC 版本是 4.3.2；CPU 是 Intel Xeon 系列四核心 CPUX3350，主频为2.66 GHz。实验中仿真网络数据流量采用的是入侵检测项目 MIT DARPA 数据集[10]。优化算法时间的测量采用的是在代码中硬编码（嵌入代码）的方法获得处理时间。

3.2 算法执行时间分析

表 1 为实验结果统计表。在本次实验设计中，以 8×10^4

表 1 DDA 和原始算法执行时间比较 s

实验序号	8×10^4 个数据包		16×10^4 个数据包		24×10^4 个数据包		32×10^4 个数据包		40×10^4 个数据包	
	原始算法	DDA	原始算法	DDA	原始算法	DDA	原始算法	DDA	原始算法	DDA
1	7.595	7.487	16.354	15.273	20.415	19.107	25.473	23.337	29.137	27.233
2	7.386	7.352	16.222	15.309	20.315	19.105	25.346	23.229	29.179	27.128
3	7.402	7.444	16.246	15.367	20.366	19.126	25.227	23.309	29.154	27.240
4	7.617	7.243	16.206	15.247	20.663	19.138	25.335	23.350	29.120	27.231
5	7.414	7.407	16.224	15.323	20.549	19.135	25.520	23.374	29.200	27.212
6	7.603	7.229	16.683	15.385	20.627	19.124	25.416	23.294	29.198	27.163
7	7.623	7.380	16.240	15.317	20.459	19.135	25.414	23.327	29.168	27.168
8	7.498	7.410	16.284	15.333	20.438	19.103	25.615	23.381	29.163	27.249
9	7.590	7.453	16.608	15.340	20.446	19.115	25.378	23.359	29.121	27.231
10	7.485	7.351	16.364	15.375	20.659	19.110	25.214	23.373	29.152	27.227
平均	7.521	7.375	16.343	15.326	20.493	19.119	25.393	23.333	29.159	27.208

个数据包为单位,分了 5 组实验,每组实验进行 10 次,表 1 记录了 10 次实验测得的多核处理器的处理时间,最后对每组的统计结果求平均值。

3.2.1　CPU 执行效率分析

图 6 是 CPU 处理时间直方图,从中可看出:当数据包个数为 8×10^4 时,减少时间几乎为 0,而随着数据包个数的增加,在处理时间上均有减少,而且减少的幅度是和数据包个数的增加成正比的。

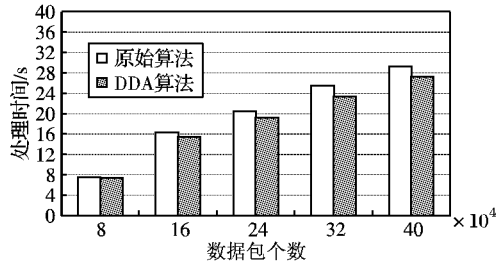

图 6　CPU 处理时间直方图

3.2.2　算法性能提升分析

算法改进前后节约时间及提升百分比如表 2 所示。

表 2　DDA 算法性能提升统计表

数据包个数	节约时间/s	提升百分比/%
8×10^4	0.146	2.0
16×10^4	1.017	6.2
24×10^4	1.374	6.7
32×10^4	2.060	8.2
40×10^4	1.948	6.7

从表 2 可看出:当数据包个数增加到 16×10^4 以后,这时无论是节约的绝对时间还是提升的百分比,都有明显增加。当数据包个数达到 40×10^4 时,这是算法性能的提升率趋向于稳定。通过上面的分析可知,整个算法经历了规则初始学习(性能提升很少)、规则学习逐步完成(性能提升明显)、规则学习逐步稳定(性能提升稳定)的过程。

4　结语

本文对 L7-Filter 的工作机制做了简要介绍,并说明了文献[4]对传统单核服务器上 L7-Filter 的改进方法及存在的不足。通过规则所采用的运输层协议对规则链进行分类,具体到每条规则链上利用网络数据流的流量统计模型与动态适应相结合的方法使得规则局部有序。在实验论证的基础上,证明了这种改进的模式匹配算法在 CPU 利用率和数据包的处理性能上的优越性。同时,由于整个规则的学习需要大规模网络流量的支持,所以本文算法在网络流量较小的情况下性能不理想,甚至较以往有一定的损耗,这部分损耗主要是核心间为了互斥并发访问规则链所带来的加/解锁操作。另外,为了降低核心间互斥并发访问规则链所带来的加/减锁操作影响,需要使每两个核心共享一条规则,这相对于改进前全局公用一条规则链而言,增加了算法的空间复杂度。以上两点的不足,也正是以后研究的一个重点。

参考文献:

[1] 林闯,王元卓,任丰原. 新一代网络 QoS 研究[J]. 计算机学报,2008,31(9):1525 - 1535.

[2] KUMAR S, TURNER J, WILLIAMS J. Advanced algorithms for fast and scalable deep packet inspection[C]// Proceedings of the 2006 ACM/IEEE Symposium on Architecture for Networking And Communications Systems. New York: ACM Press, 2006: 81 - 92.

[3] 曹折波,李青. 多核处理器并行编程模型的研究与设计[J]. 计算机工程与设计,2010,31(13):2999 - 3003.

[4] GUO DANHUA, LIAO GUANGDENG, BHUAN L N. A scalable multithreaded L7-Filter design for multi-core servers [C]// Proceedings of the 4th ACM/IEEE Symposium on Architecture for Networking and Communications Systems. New York: ACM Press, 2008: 60 - 68.

[5] Windows Hardware Development Center. Receive Side Scaling(RSS) [EB/OL]. [2011-10-20]. http://msdn.microsoft.com/en-us/windows/hardware/gg463253.aspx.

[6] 所光,杨学军. 多核处理机系统 Cache 管理技术研究现状[J]. 计算机工程与科学,2010,32(7):65 - 68.

[7] VALDVOGEL M. Multi-dimensional prefix matching using line search [C]// Proceedings of the 25th Annual IEEE Conference on Local Computer Networks. Washington, DC: IEEE Computer Society, 2000: 200 - 207.

[8] HAMED H, AL-SHAER E. On autonomic optimization of firewall policy organization [J]. Journal of High Speed Networks, 2006, 15(3): 209 - 227.

[9] 丁晶,陈晓岚,吴萍. 基于正则表达式的深度包检测算法[J]. 计算机应用,2007,27(9):2184 - 2193.

[10] MIT DARPA intrusion detection data sets[EB/OL]. [2010-10-10]. http://www.ll.mit.edu/IST/ideval/data/2000/2000_data_index.html.

[11] 杨赞,杨林,王宝林,等. 依据流统计特性的文分类规则动态优化[J]. 计算机应用研究,2011,28(5):1878 - 1882.

[12] (美)约翰逊,(美)威曾格,(美)普拉瓦提. Linux 服务器性能调整[M]. 韩智文,译. 北京:清华大学出版社,2004:23 - 24.

[13] (美) LOVE R. Linux 内核设计与实现[M]. 3 版. 陈莉君,康华,译. 北京:机械工业出版社,2011:143 - 148.

[14] Libnids[CP/OL]. [2010-10-10]. http://libnids.sourceforge.net/.

[15] 徐卫志,宋凤龙,刘志勇,等. 众核处理器片上同步机制和评估方法研究[J]. 计算机学报,2010,33(10):1777 - 1787.

(上接第 605 页)

[3] BAKER T P. An analysis of fixed-priority schedulability on a multiprocessor [J]. Real-Time Systems, 2006, 32(1/2): 49 - 71.

[4] BERTOGNA M . Real - time scheduling analysis for multiprocessor platforms [D]. Pisa: Scuola Superiore Sant'Anna, 2008.

[5] BERTOGNA M, CIRINEI M, LIPARI G. Schedulability analysis of global scheduling algorithms on multiprocessor platforms [J]. IEEE Transactions on Parallel and Distributed Systems, 2009, 20(4): 553 - 566.

[6] BARUAH S. Techniques for mutliprocessor global schedulability analysis [C]// Proceedings of the 28th IEEE International Real-Time Systems Symposium. Washington, DC: IEEE Computer Society, 2007: 119 - 128.

[7] BAKER T P, CIRINEI M. A necessary and sometimes sufficient condition for the feasibility of sets of sporadic hard-deadline tasks [C]// Proceedings of the 27th IEEE International Real-Time Systems Symposium. Piscataway, NJ: IEEE Press, 2006: 178 - 190.

[8] CIRINEI M, BAKER T P. EDZL scheduling analysis[J]. Real-Time Systems, 2008, 40(3): 264 - 289.

[9] BARUAH S, GOOSSENS J. Deadline monotonic scheduling on uniform multiprocessors [C]// Principles of Distributed Systems. Berlin: Springer-Verlag, 2008: 89 - 104.

[10] 石林勇,晏立. 多处理器全局单调比率的可调度性分析[J]. 计算机应用,2010,30(10):2735 - 2737.