高性能网包分类理论与算法综述

亓亚烜[1),2)] 李 军[2),3)]

[1)](清华大学自动化系 北京 100084)

[2)](清华大学信息技术研究院 北京 100084)

[3)](清华信息科学与技术国家实验室(筹) 北京 100084)

摘 要 随着 IP 网络架构的不断演进以及网络业务和安全需求的不断增长,高性能网包分类在下一代交换机、路由器、防火墙等网络基础设备中有着越来越广的应用.网包分类算法作为高性能网包分类的核心技术,具有重要的研究价值和实践意义.文中从理论分析和算法设计两方面介绍了高性能网包分类的最新研究成果.在理论分析层面,依据计算几何理论对网包分类问题的数学解法及复杂度进行了归纳,总结了网包分类算法的理论依据及性能评价方法.在算法设计层面,对具有影响力的网包分类算法按照不同的研究方向进行了归类和介绍,并结合自身研究成果对不同类别的算法设计思路行了深入分析.作者在多核网络处理器平台以及 FPGA 平台上实现了几类具有代表性的网包分类算法,并通过真实的网络流量测试比较了不同类型算法在不同系统平台上的实际性能.最后,作者总结并展望了高性能网包分类的下一步发展方向.

关键词 网包分类;计算几何;算法;评测;多核;FPGA

中图法分类号 TP393 **DOI 号** 10.3724/SP.J.1016.2013.00408

Theoretical Analysis and Algorithm Design of High-Performance Packet Classification Algorithms

QI Ya-Xuan[1),2)] LI Jun[2),3)]

[1)](*Department of Automation, Tsinghua University, Beijing* 100084)

[2)](*Research Institute of Information Technology, Tsinghua University, Beijing* 100084)

[3)](*Tsinghua National Laboratory for Information Science and Technology, Beijing* 100084)

Abstract With the evolution of IP-based network architecture and the increase of network services, high-performance packet classification becomes a fundamental technology for high-speed network devices. In this survey, we introduce recent development of high-performance packet classification algorithms from theory to practice. First, we provide theoretical basis of the packet classification problem. Then we categorize and analyze existing packet classification algorithms according to different research directions. To evaluate system-level performance, we implement typical algorithms on both multi-core network processor and FPGA hardware platforms, and use real-life data sets for performance evaluation. Finally, we state our conclusion and discuss the future work on high-performance packet classification algorithms.

Keywords packet classification; computational geometry; algorithm; evaluation; multi-core; FPGA

收稿日期:2011-03-09;最终修改稿收到日期:2011-11-30.本课题得到清华信息科学与技术国家实验室(筹)学科交叉基金及北京市共建项目专项资助.**亓亚烜**,男,1979 年生,博士,主要研究方向为网络体系结构和网络安全.E-mail:yaxuan@tsinghua.edu.cn.**李 军**,男,1962 年生,博士,研究员,主要研究领域为网络体系结构和网络安全.

1 引　言

随着互联网架构的不断演进以及互联网应用的不断涌现，基于单一 IP 地址域的传统路由技术已经不能满足日益增长的网络业务和网络安全需求. 由于网包分类(packet classification)能够依据多域网包包头(packet header)信息对网络流量进行细粒度的分类，该技术已在路由器、安全网关及流量控制系统等各类网络设备中得到了广泛应用[1-4]. 与此同时，随着业务感知网络、数据中心网络以及软件定义网络等前沿网络技术的发展，高性能网包分类技术已成为下一代互联网发展和演进中的研究热点[5-8].

图1 描述了一个典型的基于IPv4 五元组(5-tuple)的网包分类系统. 该五元组包括网包包头中的源/目的 IP 地址域(各 32 bit)、源/目的传输层端口域(各 16 bit)以及传输层协议域(8 bit). 网包分类规则(如表 1 例)存储于网包分类系统中. 网包分类引擎根据输入网包的五元组信息与分类规则进行匹配. 网包分类系统将依据匹配规则的决策(action)对输入网包进行相应的处理，例如接受转发(ACCEPT)、拒绝转发(DENY)、重置连接(RESET)或丢弃网包(DROP).

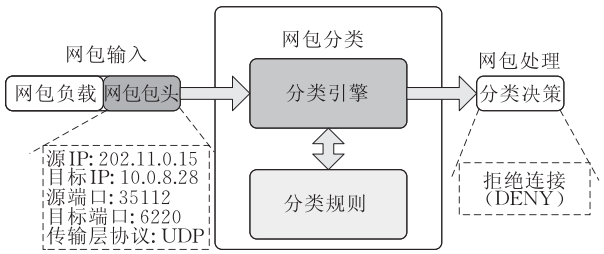

图 1　网包分类系统

表 1　网包分类规则示例

规则	目标 IP 地址	源 IP 地址	目标端口	源端口	L4 协议	优先级	决策
r_1	10.0.8.28/32	64.10.8.20/32	80	0～65535	TCP	1	RESET
r_2	10.0.8.28/32	0.0.0.0/0	53	0～65535	TCP	2	ACCEPT
r_3	10.0.0.0/16	202.11.0.15/32	0～65535	0～65535	UDP	3	DENY
r_4	10.0.0.0/16	0.0.0.0/0	0～65535	0～65535	TCP	4	DENY
r_5	10.1.3.20/32	0.0.0.0/0	80	6110～6112	UDP	11	ACCEPT
r_6	10.1.3.0/24	0.0.0.0/0	0～65535	1024～65535	ANY	12	ACCEPT
r_7	0.0.0.0/0	0.0.0.0/0	0～65535	0～65535	ANY	999	DROP

高性能网包分类算法作为网包分类设备的核心技术，一直受到学术界和工业界的广泛关注. 自 20 世纪 90 年代以来，网包分类算法的研究成果层出不穷. 然而，相关研究的总结工作却仅现于 21 世纪初的若干综述性文章[9-12]. 其后近十年间的优秀研究成果则缺乏系统的归纳和分析. 为了进一步推动高性能网包分类算法的研究和发展，本文从理论分析、算法优化和系统实现 3 个层面，结合自身的研究成果，对近年来的高性能网包分类算法进行了全面的分析和总结. 本文主要贡献包括：

(1) 总结网包分类算法的理论依据：理论依据对于高性能网包分类算法的设计具有指导性作用. 本文依据计算几何理论对网包分类问题的数学解法及复杂度进行了归纳，总结了网包分类算法的理论依据及性能评价方法.

(2) 归类并分析现有网包分类算法：高性能网包分类技术的核心在于软硬件算法的优化设计. 本文对具有影响力的网包分类算法按照空间分解、规则分组及硬件设计等不同的研究方向进行了归类，并结合自身研究成果对不同类别的算法设计方法进行了深入的比较和分析.

(3) 测试高性能网包分类算法性能：高性能网包分类算法的实际性能需要在真实系统上进行验证. 本文在多核网络处理器平台以及 FPGA 硬件平台上实现了几类具有代表性的网包分类算法，并通过真实的网络流量测试比较了不同类型算法在不同系统平台上的实际性能. 实验中所用的测试方法、测试数据以及自行开发的算法源代码将对外公开，以推进高性能网包分类算法及其相关研究的进一步发展.

本文第 2 节介绍网包分类算法的理论依据及复杂度分析；第 3 节归纳和比较近年来主流网包分类算法的优化方法；第 4 节考察典型网包分类算法实现于不同网络处理平台下的真实性能；第 5 节总结全文，并展望下一步的研究工作.

2 网包分类算法的理论依据

本节首先通过数学定义，将网包分类问题归结为计算几何领域中的点定位问题；然后介绍和分析点定位问题的多类数学解法及其理论复杂度；最后，

结合真实网包分类规则的统计特性总结网包分类算法性能的评估标准.

2.1 网包分类问题的数学描述

网包分类问题本质上是多域空间中的点定位问题(point location problem)[13-14]. 为了便于复杂度分析,首先介绍网包分类问题中 3 个基本概念:网包、搜索空间及分类规则:

网包 p(Packet). 网包 p 包含 d 个域的网包包头. 网包包头的各个域分别表示为 $p[1],p[2],\cdots,p[d]$,其中每个域的取值都是特定长度的比特串. 例如 32 比特的 IPv4 网络层 IP 地址,16 比特的传输层端口号等.

搜索空间 S(Search Space). 网包 p 在 d 维空间所有可能的取值构成搜索空间 S. S 的各个维度值域不同,对于 IPv4 五元组网包分类问题,$S=[0,2^{32}-1]\times[0,2^{32}-1]\times[0,2^{16}-1]\times[0,2^{16}-1]\times[0,2^{8}-1]$.

分类规则 r(Rule). 每个分类规则包含 3 个部分:各域范围表示(range expression)$r[1],r[2],\cdots,r[d]$、规则优先级(priority) $r.pri$ 和规则决策(action) $r.act$. 若网包 p 与规则 r 匹配(match),则 $\forall 1\leq i\leq d, p[i]\in r[i]$. 对于包含 n 个规则的规则集合 $R=\{r_1,r_2,\cdots,r_n\}$,p 可能与其中多个规则匹配①.

基于上述定义,分类规则 r 对应于搜索空间 S 中的一个超长方体(hyper-rectangle),而网包 p 则对应于 S 中的一个点. 当 p 落入 r 所表示的超长方体中时,p 即与 r 匹配. 为了便于理解,图 2 给出了一个二维($d=2$)网包分类问题的示例. 其中搜索空间 $S=[0,3]_X\times[0,3]_Y$,网包 p 的点坐标为 $(p[x]=3,p[y]=3)$,规则集合 $R=\{r_1,r_2,\cdots,r_5\}$ 如表 2 所示. 由于 p 落入 r_4 和 r_5 对应的矩形(2 维超长方体)中,因此 p 与 r_4 和 r_5 匹配. 若考虑匹配优先级,则由于 r_4 的优先级高于 r_5,网包分类系统执行 $r_4.act$(DENY).

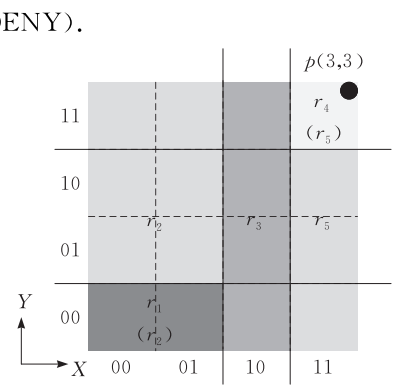

图 2 二维网包分类问题示例

表 2 二维网包分类规则示例

规则	X 域	Y 域	优先级	决策
r_1	[00,01]	[00,00]	1	DENY
r_2	[00,01]	[00,11]	2	ACCEPT
r_3	[10,10]	[00,11]	3	ACCEPT
r_4	[11,11]	[11,11]	4	DENY
r_5	[11,11]	[00,11]	5	ACCEPT

2.2 网包分类问题的数学解法

对于空间点定位问题,线性查找(linear search)是最简单的解法. 通过将输入网包 p 与所有规则逐一匹配,即可得到所有匹配规则. 对于 d 维空间的 n 个规则,线性查找的时间复杂度为 $\Theta(d\times n)$,空间复杂度为 $\Theta(n)$. 由于查找时间随规则增加而呈线性增长,线性查找仅适用于小规模的规则集合.

在计算几何领域,多域空间点定位问题存在多种数学解法. 每种解法具有不同的时间和空间复杂度,下面将分别进行介绍.

2.2.1 规则投影区间查找算法

规则投影区间查找算法(下文简称区间查找算法)源自 Overmars 等人提出的高维区域树(Hierarchical segment Tree,本文简称为 H-Tree)算法[13-14].

算法 1. H-Tree 算法.

当 $d=1$ 时,由于 n 个规则的端点(end point)在一维空间中最多构成 $2n+1$ 个连续区间(segment),因此可以构建一个空间复杂度为 $\Theta(n)$的平衡二分查找树(balanced binary search tree)来进行查找,查找时间为 $\Theta(\log n)$.

当 $d>1$ 时,首先依据规则集合在第 d 维的投影区间构造平衡二分查找树 T_d. 树的每一个节点 v 对应一个区间 I_v,其中 I_v 表示 v 的所有子节点构成的连续区间. 对于满足

(1) 在 d 维上的投影区间完全包含 I_v;

(2) 在 d 维上的投影区间不完全包含 v 的父节点 v' 对应的区间 $I_{v'}$;

的规则子集 R_v,用同样的方式在另外 $d-1$ 个维度上依次构造平衡二分查找树 $T_{v,d-1}$,直到最后一个维度即可.

H-Tree 算法结合 Chazelle 等人[15]提出的分散叠层(fractional cascading)技术,能够以 $\Theta(\log^{d-1}n)$ 的时间复杂度和 $\Theta(n\times\log^{d-1}n)$ 的空间复杂度解决点定位问题[13].

虽然 H-Tree 具有良好的空间复杂度,但时间复杂度依然过高(优于线性查找算法). 下面介绍降低时间复杂度的 Set-pruning Segment Tree(本文简称为 S-Tree)算法[16-17].

① 对于防火墙等应用,通常执行匹配规则中优先级最高的那条规则所包含的规则决策.

算法 2.　S-Tree 算法.

当 $d=1$ 时，S-Tree 和 H-Tree 算法相同. 由于 n 个规则的端点(end point)在一维空间中最多构成 $2n+1$ 个连续区间，因此可以构建一个空间复杂度为 $\Theta(n)$ 的平衡二分查找树来进行查找，查找时间为 $\Theta(\log n)$.

当 $d>1$ 时，S-Tree 采用规则复制的方法避免回溯查找. 首先依据规则集合在第 d 维的投影区间构造平衡二分查找树 T_d. T_d 的每一个叶节点 v 对应一个区间 I_v，其中 I_v 不包含任何子区间. 对于规则子集 $R_v=\{r_i \mid r_i \cap I_v \neq \varnothing, 1\leq i\leq n\}$，依次为其构造子树 $T_{v,d-1}$ 即可.

与 H-Tree 相比，S-Tree 的时间复杂度降低到了 $\Theta(d\times\log n)$，但依据定理 1，S-Tree 的空间复杂度将增至 $\Theta(n^d)$.

定理 1.　d 域空间中的 n 个规则至多可构成 $(2n+1)^d$ 个相互不重叠的超长方体.

证明.

首先证明 $d=1$ 时命题成立. 一维空间中的规则退化为线段，即数轴上的一个区间.

当 $n=1$ 时，一个规则构成一个区间，命题成立.

假设 $k(k\geq 1)$ 个规则可构成 $2k+1$ 个相互不重叠的区间. 那么当 $n=k+1$ 时，第 $k+1$ 个规则的两个端点最多落入 $2k+1$ 个不重叠的区间中的两个不同的区间中，并将这两个区间划分为四个区间，因此最多增加两个不重叠区间. 又由 $2k+1+2=2(k+1)+1$，所以命题对于 $n=k+1$ 亦成立，即一维情况下命题成立.

多域情况下，即当 $d>1$ 时，由于每个域上最多有 $2n+1$ 个不重叠区间，经过交叉相乘，d 个域上最多出现 $(2n+1)^d$ 个不重叠的超立方体，由此命题得证.　　证毕.

2.2.2　网包搜索空间分解算法

与区间查找算法不同，网包搜索空间分解算法(下文简称空间分解算法)通过对空间的均匀切分构建网包分类的数据结构. 空间分解算法源于路由查找算法，通常使用 trie 结构对空间进行逐级等分. 这里首先讨论规则均为最长前缀匹配的情况，后面会讨论一般情况. 下面首先介绍基于 Hierarchical Trie(本文简称 H-Trie)的空间分解算法[13].

算法 3.　H-Trie 算法.

当 $d=1$ 时，构建一个 W 深度的 trie 结构，每一个规则均存储于 tire 的一个节点上，此 trie 结构空间复杂度为 $\Theta(n\times W)$，时间复杂度为 $\Theta(W)$.

当 $d>1$ 时，首先用 $\{r_1[d], r_2[d], \cdots, r_n[d]\}$ 构造 trie 结构 T_d. 然后为每一个不同的前缀 P_i(T_d 中不同的节点)所对应的规则子集 R_i 构建 $d-1$ 维度上 trie 结构 $T_{i,d-1}$. 依次类推，直到第一个维度即可.

H-Trie 在搜索过程中需要回溯查找(back tracking search)，因此时间复杂度为 $\Theta(W^d)$. 由于 H-Trie 中规则集合只存储一次，因此算法的空间复杂度为 $\Theta(n\times W)$. 注意，此时的空间复杂度只对应前缀匹配的规则，而本文讨论的网包分类问题是基于范围匹配的，因此这里讨论范围到前缀转换问题(range-to-prefix). 首先给出定理 2.

定理 2.　一个 $[0,2^W]$ 区间中的范围可以用至多 $2(W-1)$ 个前缀表示[18].

证明.　给出存在性证明.

首先考虑当范围为 $[0,b]$ 时至多需要多少个前缀表示.

当范围为 $[0,b]$，其中 $2^{W-1}<b\leq 2^{W-1}$ 时，首先记录表示区间 $[0,2^{W-1})$ 的前缀 P_0，然后将区间 $[2^{W-1},2^W]$ 等分，生成两个新区间 $[2^{W-1},2^{W-1}+2^{W-2})$ 和 $[2^{W-1}+2^{W-2},2^W]$，则 b 必然落入两个区间中的一个，如果 $2^{W-1}+2^{W-2}<b\leq 2^W$，则记录表示区间 $[2^{W-1},2^{W-1}+2^{W-2})$ 的前缀 P_1；依次不断等分 b 所在子区间，并当 b 落入右半区间时记录左半区间的前缀 P_i；经过至多 $W-1$ 次等分，区间 $[2^{W-1},2^W]$ 将不可再分，由于整个过程中最多记录下 W 个前缀($W-1$ 次等分得到的前缀和 P_0)，而这些前缀恰好可以表达范围 $[0,b]$. 因此，范围 $[0,b]$ 需要至多 W 个前缀表示.

再考虑 $[0,2^W]$ 空间中的范围 $[a,b]$，通过将 $[0,2^W]$ 二等分，范围 $[a,b]$ 最多被分为两个范围，并且这样的两个范围通过平移和镜像，都可以等价的转化为 $[0,2^{W-1}]$ 空间中以 0 为起点的范围. 根据上面的证明，这两个范围都可以用至多 $W-1$ 个前缀表示，因此范围 $[a,b]$ 可用至多 $2(W-1)$ 个范围表示.　　证毕.

根据定理 2 可知，一个范围匹配的规则在 d 个域上最多可以转换为 $(2(W-1))^d$ 个前缀匹配的规则. 因此，对于范围匹配，H-Trie 的空间复杂度为 $\Theta(n\times W^d)$. 当 $d>1$ 时，最后两个维度的查找可以使用 Srinivasan 等人[18]提出的 Grid-of-Trie 结构进一步减少查找时间. 此时 H-Trie 的时间和空间复杂度可以改进为 $\Theta(W^{d-1})$ 及 $\Theta(n\times W^{d-1})$. 由于 H-Trie 的时间复杂度过高，可使用 Set-pruning Trie(本文简称 S-Trie)算法，通过规则复制来降低时间复杂度[9].

算法 4. S-Trie 算法.

当 $d=1$ 时,S-Trie 算法与 H-Trie 相同. 构建一个 W 深度的 trie 结构,每一个规则均存储于 tire 的一个节点上,此 trie 结构空间复杂度为 $\Theta(n\times W)$,时间复杂度为 $\Theta(W)$.

当 $d>1$ 时,首先对 $\{r_1[d],r_2[d],\cdots,r_n[d]\}$ 构造 trie 结构 T_d. 然后对于每一个叶节点 v,将包含 v 代表的前缀 P_v 的所有规则子集 R_v 构建 $d-1$ 维度上 trie 结构 $T_{v,d-1}$. 依次类推,直到第一个维度即可.

由于 S-Trie 的查找过程不需要回溯,因此时间复杂度为 $\Theta(d\times W)$,空间复杂度为 $\Theta(n^d\times dW)$. 由此可见,H-Trie 和 S-Trie 分别是对空间和时间性能的折中,但两者的时间性能在 n 较大时均优于线性查找算法.

2.2.3 数学解法总结

多域点定位问题的各类数学解法总结于表 3. 这些数学解法为网包分类算法设计提供了理论依据. 从该表可知,允许回溯查找的算法具有较好的空间特性,而允许规则复制的算法则具有较快的查找速率. 由于查找速率决定网包分类系统的吞吐率,因此典型的网包分类算法大多采用了允许规则复制的方法. 关于查找的策略,区间查找算法和空间分解算法均有各自的优势,并广泛用于不同的网包分类算法. 空间分解算法虽然不需要存储规则投影点,并可利用多比特 trie 进一步加快查找速率,但是需要考虑范围到前缀匹配,因此最坏情况下存储空间为投影区间二分法的 $\Theta(W^d)$ 倍(参见定理 2). 关于网包分类算法时间和空间性能折中的进一步分析可以参阅文献[19].

表 3 多域正交区域点定位问题的算法比较

维度	复杂度	回溯查找		规则复制	
		空间分解	区间查找	空间分解	区间查找
		H-Trie	H-Tree	S-Trie	S-Tree
$d=1$	时间	$\Theta(W)$	$\Theta(\log n)$	$\Theta(W)$	$\Theta(\log n)$
	空间	$\Theta(n\times W)$	$\Theta(n\times\log n)$	$\Theta(n\times W)$	$\Theta(n\times\log n)$
$d>1$	时间	$\Theta(W^{d-1})$	$\Theta(\log^{d-1}n)$	$\Theta(d\times W)$	$\Theta(d\times\log n)$
	空间	$\Theta(n\times W^{d-1})$	$\Theta(n\times\log^{d-1}n)$	$\Theta(n^d\times dW)$	$\Theta(n^d\times d\log n)$

2.3 网包分类算法的评价方法

由上述算法分析可知,网包分类问题的各类数学解法均具有较高的时间或空间复杂度,即在最坏情况下无法同时满足查找速率和存储空间的两方面要求. 幸运的是,在实际的网络应用中网包分类问题往往不会达到理论上的最坏情况[20]. 当前的网包分类算法设计大多通过引入规则集合的特征来提高分类速率、降低内存使用.

Gupta 等人[20]通过对大量实际的(real-life)规则集合的研究,总结并归纳出一系列规则集合特征:

(1)实际规则集合中规则数目不会太多,一般从几十条规则到数千条规则. 规则数目不多可能是由于网络应用本身规模的限制,也可能是基于当前路由器处理能力的考虑.

(2)规则在协议域通常只有很少的几个取值. 绝大多数规则集合中只出现 TCP 和 UDP 两种传输层协议. 个别规则集合中可能涉及 ICMP、IGMP 和 GRE 等协议.

(3)传输层端口号域取值范围很广,采用范围前缀转换很可能是非常低效的.

(4)与同一个网包匹配的规则通常少于 5 个,最多出现过 10 个①.

(5)同一规则集合中的多个规则往往在某些域上具有相同的设置.

(6)规则集合中所有规则在单一域的不同取值的个数通常远小于规则个数.

(7)规则集合出现的重叠个数远远小于理论上界.

另外,对于不同应用下的规则集合,也会出现各自不同的统计特性. 例如在 WUSTL(华盛顿大学圣路易斯分校)公开数据集中,核心路由器上的访问控制规则(ACL)、防火墙的安全策略(FW)以及 Linux 网关的 iptables 规则(IPC)就有相当大的差异[21]. Qi 等人[17]通过统计这些规则集合中各个维度上的投影区间个数说明了网包分类问题的实际复杂性.

从表 4 中的统计结果可以看出:

(1)同一类型规则集合在不同维度上的统计特性不同.

(2)不同种类的规则集合的统计特性不同.

(3)所有规则集合的实际复杂度均远小于理论复杂度.

解决多域网包分类问题需要设计高效的分类算法. 一般来说,评价一个算法的好坏,需要从 3 个方面进行综合考虑:

(1)分类速率. 网包分类系统通常需要满足线速(wire-speed)处理速度. 对于实际的网包分类系统,通常使用系统吞吐率(throughput)来评价分类速率. 例如要满足 100 Gbps 的网络带宽,网包分类系统需要每秒钟处理 150 M 个 64 字节的网包. 在算

① 此处结论结合本文实验进行了部分修正. Gupta 等人在文献[19]的原文中为 7 个.

表 4　网包分类规则复杂性比较

规则集*	规则数	各域非重叠区间理论最大值	目标IP地址域非重叠区间实际值	源IP地址域非重叠区间实际值	目标端口域非重叠区间实际值	源端口域非重叠区间实际值	非重叠超长方体理论最大值	非重叠超长方体实际值
FW1-100	92	185	45	19	48	20	1.17×10^{9}	8.21×10^{5}
FW1-1K	791	1583	314	221	75	23	6.28×10^{12}	1.20×10^{8}
FW1-10K	9311	18623	13901	7270	77	22	1.20×10^{17}	1.71×10^{11}
ACL1-100	98	197	120	48	70	1	1.51×10^{9}	4.03×10^{5}
ACL1-1K	916	1833	559	143	165	1	1.13×10^{13}	1.32×10^{7}
ACL1-10K	9603	19207	1325	7921	181	1	1.36×10^{17}	1.90×10^{9}
IPC1-100	99	199	119	118	26	26	1.57×10^{9}	9.49×10^{6}
IPC1-1K	938	1877	796	559	78	49	1.24×10^{13}	1.70×10^{9}
IPC1-10K	9037	18075	4604	2377	94	59	1.07×10^{17}	6.07×10^{10}

注：*ACL1、FW1、IPC1 分别为 3 类不同的规则集合，ACL1-1K 表示包含约 1000 条规则的 ACL 规则集合[21].

法分类速率的评价中，考虑到处理单元(CPU)的计算速度比外围存储设备(DRAM)的访问速度快得多，因此当计算量大小在可接受的范围内时，通常使用内存访问次数(memory access times)来评价一个算法的分类速率.

(2) 内存使用. 网包分类算法的内存使用不仅仅指存放规则集合本身所占用的存储空间，还包括算法建立的用于查找的数据结构存储空间. 考虑到网包分类系统的内存空间有限，网包分类算法应尽可能压缩其数据结构以支持更多的分类规则. 另外，有时候还必须考虑算法预处理过程中的内存使用. 例如在递归生成多级 trie 的数据结构过程中，有时需要大型的堆栈支持. 如果系统无法满足此内存需求，那么即使最终数据结构较小也无法实现.

(3) 预处理时间. 由于网包分类规则并非固定不变，网包分类算法需要依据规则更新生成新的数据结构. 本文讨论的算法的预处理时间指依据新的规则集合重新生成网包分类数据结构的全部时间.

从算法设计的角度来看，目前的大多数算法都比较重视分类速率和内存使用两方面的性能，与之对应的是算法的时间性能和空间性能. 两个因素在算法设计中往往互相制约，通常在满足某一方面性能的情况下，尽可能优化另一方面的性能. 从算法实现的角度来看，网包分类算法必须要兼顾多种系统平台的硬件约束，能够自适应部署于各类平台并最大程度利用系统资源以满足网包分类设备的性能需求. 因此，对于一个出色的网包分类算法其性能并不仅仅体现在理论分析中，还必须考虑到算法实现、运行环境、软硬件平台和特殊需求等多种问题. 只有这样的算法才真正具有研究意义和现实的价值.

2.4　小　结

本节总结了网包分类算法的理论依据，并结合实际规则分析给出了算法评价的常用方法. 从计算几何中的复杂度分析表明，不存在某种通用的算法适用于所有的多域网包分类问题. H-Trie 和 H-Tree 算法具有较好的空间特性，但需要耗时的回溯查找；S-Trie 和 S-Tree 算法通过规则复制大幅降低了搜索时间，但代价是指数级的内存增长.

幸运的是在实际应用中极少会遇到理论中的最坏的情况. 实际应用中的网包分类问题往往具有各类结构和统计特性. 将这些特性应用到网包分类算法的设计中去，通常可以得到“足够快”的分类速率，同时满足“足够少”的内存使用. “足够快”和“足够少”在这里表明了一种权衡(tradeoff)的思想，是算法时间性能和空间性能的折中. 算法设计的最终目标就是在理论依据之上，充分挖掘规则特性并考虑系统约束，寻求最优的权衡.

3　网包分类算法

自 20 世纪 90 年代末至今，网包分类问题的相关研究不断发展，出现了一大批优秀的网包分类算法. 本节依据网包分类算法的研究方向，按照空间分解、规则分组和硬件设计 3 个方向，对常见的网包分类算法进行归类和介绍. 与此同时，我们结合自身的研究成果，对各类网包分类算法的研究思路进行了深入的分析和比较.

3.1　空间分解算法

空间分解是一种分而治之的策略. 使用空间分解的网包分类算法将原始搜索空间 S 分解为 m 个子空间，且满足如下约束：

$$S=S_1\cup S_2\cup\cdots\cup S_m \text{ 且}$$

$$\forall 1\le i,j\le m, \text{有 } S_i\cap S_j=\varnothing.$$

每个子空间对应的规则子集合满足

$$R=R_1\cup R_2\cup\cdots\cup R_m,$$

$$\forall r\in R_k, 1\le k\le m, \text{有 } S_k\subseteq r.$$

由此可见,经过空间分解后,落入每个子空间的网包具有唯一的规则子集与之匹配.因此只要确定子空间归属即可完成网包分类.由第 2 节理论分析可知,空间分解的数学解法分为搜索空间二分法和投影区间二分法两大类.现有绝大多数网包分类算法均可归结为上述方法的拓展.下面我们以搜索空间均分和投影区间分解为线索,分别介绍不同算法的设计思想和优化方法.

3.1.1 搜索空间均分算法

搜索空间均分算法主要包括基于 Trie 结构的各类网包分类算法,如文献[22-33].该类算法的基本思路是:通过对搜索空间及其子空间进行迭代的均匀切分(cut),将原始搜索空间愈分愈细,直到落入每个子空间中的所有网包都有唯一的规则子集与之匹配.

基于搜索空间分解的算法通常使用拓展的 S-Trie 结构,即将单比特 S-Trie 拓展为可变宽的多比特 S-Trie(variable-stride multi-bit Set-pruning Trie).该结构的根节点(root node)对原始搜索空间进行第 1 级均匀切分;第 1 级的内部节点(internal node)对原始空间进行第 2 级均匀切分.每一个叶节点(leaf node)包含与该子空间所有点都匹配的规则子集.该类算法的特点是每级切分满足

$$S=S_1\cup S_2\cup\cdots\cup S_m\text{且}$$
$$|S_1|=|S_2|=\cdots=|S_m|.$$

该类算法的理论复杂度与 S-Trie 相同,实际时间性能由树深度(depth)决定,实际空间性能则取决于节点数目和节点大小.因此,基于搜索空间均分的网包分类算法主要在如何提高搜索空间分解效率上进行探索和研究.

Gupta 等人[22] 提出的 HiCuts(Hierarchical Intelligent Cuttings)是最早使用启发式算法来提高搜索空间分解效率的网包分类算法.HiCuts 利用基于多比特 Trie 的决策树(decision tree)结构对搜索空间进行多级均匀切分.在决策树的每一个内部节点,HiCuts 使用可分离判别函数选取当前最优的切分维度,并使用内存空间约束函数确定当前空间的切分次数.若当前子空间所包含的规则个数小于指定阈值(binth),HiCuts 算法生成一个叶节点,并将所含规则存储在叶节点中进行线性查找.Singh 等人提出的 HyperCuts 算法将 HiCuts 算法中的单域空间切分拓展到多域中去,在每个内部节点从多个维度上对当前搜索空间进行划分[29].由于多域切分以指数级别提高了每个节点的空间分解次数,因此 HyperCuts 算法构建的决策树深度大大减少,从而有效提高了算法的时间性能.

近年来的研究指出,将 HiCuts 和 HyperCuts 直接用于真实网包分类系统中存在诸多缺陷[24].首先,由于启发式算法每级切分次数不同,导致决策树深度不确定,进而使得系统吞吐率无法保障.其次,由于每个内部节点都使用指针数组或指针矩阵连接子节点,在切分次数增长的情况下,指针结构的存储空间随之增长.在真实系统中,过高的内存需求将导致网包分类算法难以利用有限的高速内存资源实现快速的网包分类.此外,由于 HiCuts 和 HyperCuts 每个节点的大小不一致,使得查找过程中每次访存(memory access)所读取的字节数以及相应的节点处理过程不一致.这种不一致使内存分配单元变得复杂,并降低硬件处理单元的效率.为解决这些问题,学术界提出了一系列新的改进算法[24-25,31].本文以 AggreCuts[25] 算法为例,具体说明提高算法性能的一般思路.

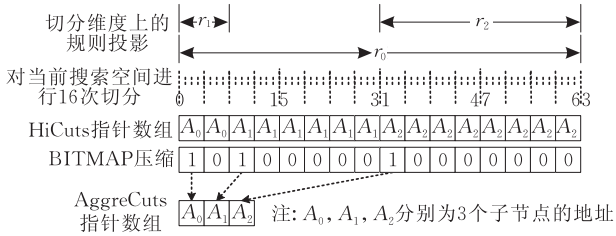

图 3 AggreCuts 中的 BITMAP 压缩

AggreCuts 的核心思想是利用 BITMAP(比特串)来压缩 HiCuts 算法中的指针数组,从而有效提高空间分解效率.如图 3 所示,当前搜索空间被切分为 16 个子空间.由于第 1～2 个子空间包含相同规则子集$\{r_0,r_1\}$,按照 HiCuts 算法,为这 2 个子空间生成一个新的子节点(存储地址为 A_0),并使用 2 个指针建立 2 个子空间到该子节点的映射关系.同理,第 3～8 个子空间(包含$\{r_0\}$)生成第 2 个子节点(地址 A_1),第 9～16 个子空间(包含$\{r_0,r_2\}$)生成第 3 个子节点.HiCuts 算法使用包含 16 个指针的指针数组建立 16 个子空间到 3 个子节点的映射关系.考虑到实际应用中指针数组的冗余性,AggreCuts 使用 16 比特的 BITMAP 对图 3 中的 16 个指针进行了压缩.BITMAP 的设置为:第 1 个比特为 1;若第 i ($i>1$)个指针跟第 $i-1$ 个不同,则该比特为 1,否则为 0.与 BITMAP 相对应的是一个压缩指针数组,该数组中顺序存储 BITMAP 中 1 位置所对应的指针(即原始指针数组中所有与前一指针不相同的指针).在查找过程中,落入第 j 个子空间的网包,只需要以 BITMAP 中前 j 个比特相加的和为索引读取

压缩指针数组中的相应指针即可.

AggreCuts 的时间和空间性能见图 4 和图 5. 由图 4 可知，AggreCuts 的决策树深度(与内存访问次数呈线性关系)不到 HiCuts 的 1/5，而且不随规则个数增加而变化. 由图 5 可知，AggreCuts 比 HiCuts 减少了 1～2 个数量级的内存使用. AggreCuts 的局限性在于需要进行 BITMAP 运算. 但随着处理器技术的发展，高性能多核网络处理器(如 Intel IXP2800 Cavium OCTEON3860)均提供高效的硬件 BITMAP 运算指令，因此 AggreCuts 可以在大多数高性能网络处理平台上得到广泛应用.

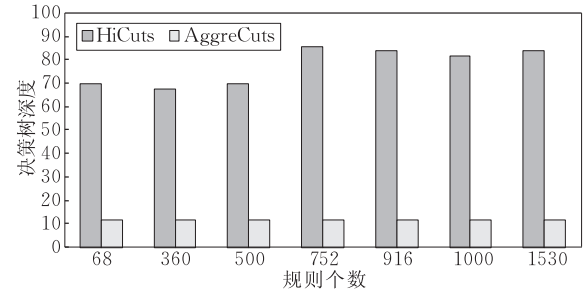

图 4　AggreCuts 算法的决策树深度

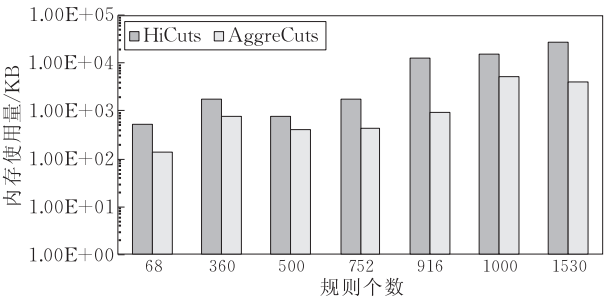

图 5　AggreCuts 算法的内存使用

3.1.2　投影区间分解算法

投影区间分解算法包括文献[16，18，20]、文献[17]和文献[34-37]. 该类算法的基本思路是：将网包分类规则投影在各个域上，每个域上相邻的两个投影点构成一个投影区间. 分类过程中首先确定网包在各个域上属于哪个投影区间(子空间)，然后再通过子空间求交的方法完成最终匹配. 该类算法的特点是每级切分尽可能满足子空间的规则数相等：

$$S=S_1\cup S_2\cup\cdots\cup S_m\text{且}$$
$$|R_1|=|R_2|=\cdots=|R_m|.$$

依据第 2 节理论分析，基于规则投影区间分解的算法的对应于点定位中的 S-Tree 算法，两者具有相同理论复杂度. 实际应用中，投影区间分解算法对 S-Tree 算法进行了拓展. 一方面利用启发式方法改进区间查找的数据结构提高分类速率，另一方面利用规则冗余特性通过迭代求交降低内存使用.

最早的基于投影区间分解的算法是 Srinivasan 等人提出的 Cross-Producting 算法[16]. 该算法采用最长前缀匹配进行各域上的投影区间查找，并使用一个 d 维的表结构(cross-producting table)完成空间求交. 由于最长前缀匹配查找效率较二分查找低($\Theta(W)$对比 $\Theta(\log n)$)，且单一的 d 维求交表无法消除空间冗余，该算法仅适用于较小的规则集合[9]. Gupta 等人[20]提出的 RFC(Recursive Flow Classification)算法利用数组结构存储各域的投影区间，将每个维度上 $\Theta(W)$的最长前缀查找时间提高为 $\Theta(1)$(代价是每个域的存储空间变为 $\Theta(2^W)$). RFC 同时采用多级求交表进行迭代求交，一定程度减少了内存的占用. Xu 等人[35]提出的 HSM(Hierarchical Space Mapping)算法采用了单域上的多分查找改善 RFC 算法的数组存储问题，并提供支持 IPv6 的 128 位地址查找. 虽然 RFC 和 HSM 算法在实际网包分类系统中得到了应用，但对于大规模的数据规则集合其空间性能并不理想. 基于投影区间分解的最新研究成通过启发式维度选择和迭代二分查找等方法，有效提高了算法的空间性能[16-17,36]. 下面以 HyperSplit 算法为例，介绍投影区间分解算法的研究思路.

HyperSplit 使用启发式算法构建 S-Tree. 在生成每个内部节点的过程中，HyperSplit 依据当前规则集合(而非分类规则全集)在各个维度上的投影点分布情况选择最具可分离性的维度，然后用平行于坐标轴的超平面将当前维度上的投影区间二分. 其中维度选择依据为

$$\min_{1\le i\le d}\frac{1}{m^i}\sum_{j=1}^{m^i}|R_j^i|,$$

其中 m^i 为第 i 维度的区间个数.

多级内部节点对投影区间的不断二分，直到当前子空间为某个规则子集完全包含，此时生成叶节点. 图 6 和图 7 比较了 HSM 和 HyperSplit 两种算法的不同分解策略. 可以看出，HSM 算法将当搜索空间分解为 9 个子空间，而 HyperSplit 算法仅分解 5 个子空间. 从搜索过程来看，HSM 算法需要最多 4 次二分查找，而 HyperSplit 算法只需要 3 次. HyperSplit 算法的时空性能由图 8 和图 9 给出. 由图中数据可知，对于不同的规则集合，HyperSplit 算法的内存访问次数比 HSM 算法平均少 30%，内存使用则减少 1～2 个数量级. 依据文献[17，38]，HyperSplit 算法在多核网络处理器和 FPGA 硬件平台上分别达到了 10 Gbps 和 100 Gbps 吞吐率.

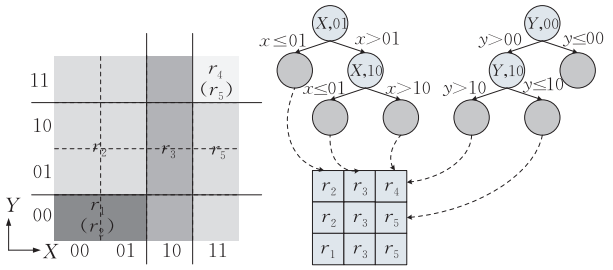

图 6 HSM 算法示例

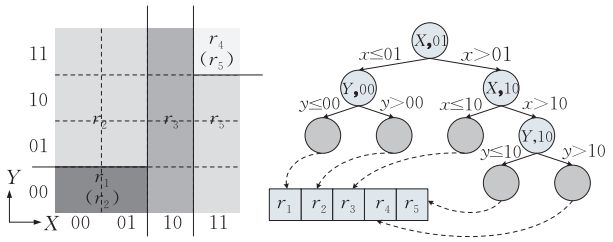

图 7 HyperSplit 算法示例

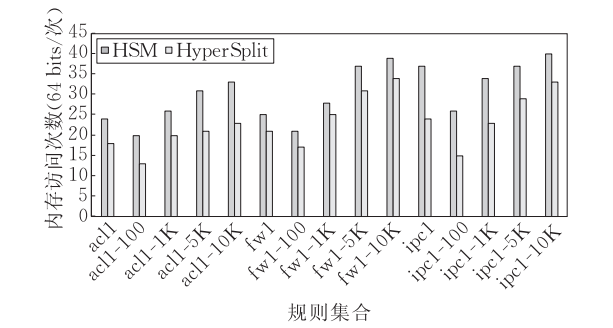

图 8 HyperSplit 算法内存访问次数

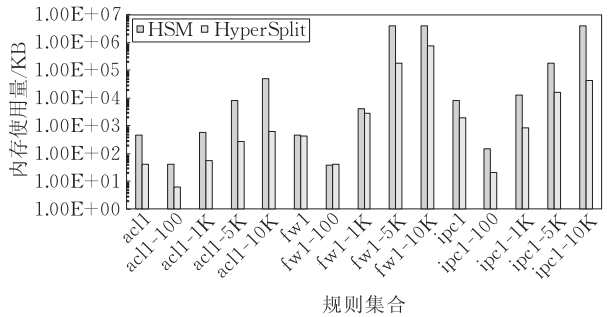

图 9 HyperSplit 算法内存使用

3.2 规则分组优化算法

由复杂度分析可知,随着规模集合的不断增长,基于空间分解的算法的内存需求呈指数级增加.近年来学术界提出了一系列基于规则分组的网包分类算法[24,39-42],有效降低了现有算法的存储空间需求.规则分组算法将规则集合按照一定的启发信息分为若干规则子集,然后对每个规则子集进行逐一或并行查找.规则分组算法是另一种分而治之的策略.此类算法将原始规则集合预先划分为 m 个规则子集,即

$$R=R_1\cup R_2\cup\cdots\cup R_m\text{且}$$
$$\forall 1\leq i,j\leq m,\text{有 } R_i\cap R_j=\varnothing.$$

规则分组算法通常与上一小节介绍的空间分解算法相结合,即首先进行规则分组,然后对每个规则子集使用空间分解算法进行查找.下面,我们分别介绍基于结构特性和组合优化的两类主流的规则分组算法.

3.2.1 基于结构特性的分组算法

基于规则结构特性的分组算法依据规则的结构特性将规则全集分为若干子集,使得同一子集内的规则具有良好的可分离性(separablility)[24],从而降低整体的存储需求.由于结构特性法需要对多个子集进行逐一或并行查找,如何在保证任意子集具备良好可分离性的前提下控制规则子集的个数,成为当前研究的重点.

基于结构特性的规则分组算法最具代表性的是 Vamanan 等人[24]提出的 EffiCuts.作者在论文中指出,HiCuts 和 HyperCuts 算法中大量的内存消耗的主要原因之一便是用单一的决策树处理相互重叠且结构各异的规则全集.为此,EffiCuts 算法的第一步便是基于结构特性的规则分组.其分组依据为规则在各个维度上投影区间的取值范围.若一个规则在某个域上的投影区间大于某一阈值(largeness fraction),则认为该规则的投影区间在该域上是"大"的;反之则为"小".根据每一个规则在各域上的结构特性,EffiCuts 将原始规则集合划分为 26 个(对于 IPv4 的五元组规则)规则子集,其中任一子集均满足作者定义的可分离性要求.

规则分组完成后,EffiCuts 使用 HyperCuts 算法对每个规则子集进行分类查找.为了控制规则子集的个数,EffiCuts 算法还使用了选择性树合并(selective tree merging)的策略,将规则简单的和规模较小的子集进行了选择性的合并,最终生成 5～6 个规则子集.作者通过实验证实,在内存访问次数相同的情况下,EffiCuts 比 HyperCuts 减少 57 倍的内存使用.

3.2.2 基于组合优化的分组算法

从优化的角度来看,规则分组是一个带约束的组合优化问题(combinatorial optimization).其约束为子集个数,而优化目标则为整体内存空间.由于规则特性法是确定性的方法,因此通常只能得到这个优化问题的局部最优解.对于一般的组合优化问题,随机搜索方法如模拟退火、禁忌搜索、遗传算法等能够得到全局最优解.

基于组合优化法的最新研究成果为 Fong 等人[42]提出的 ParaSplit 算法. 该算法使用模拟退火进行优化，在确定规则子集个数的约束下搜索最优的规则分组. ParaSplit 算法的目标函数为各个子集在 HyperSplit 算法下的内存使用总和. 模拟退火过程的初始解为通过对规则全集随机分组获取. 搜索过程中，每一个新解都由通过随机增减或交换任意两个子集间的规则获取. 新解是否被接受依据 Metropolis 准则. 若目标函数差(总内存增量)：

$$\Delta D<0,$$

则接受，否则以概率

$$P_{\text{accept}}=\mathrm{e}^{-\Delta D/T}$$

接受，其中 T 为退火温度.

ParaSplit 通过选择合理的初始温度，使得算法能在有限的迭代次数内满足收敛条件. 若使用结构特性法求取初始解，ParaSplit 能够以更快速度的收敛. 从图 10 可以看出，ParaSplit 的内存使用比 HyperSplit 少 1～3 个数量级.

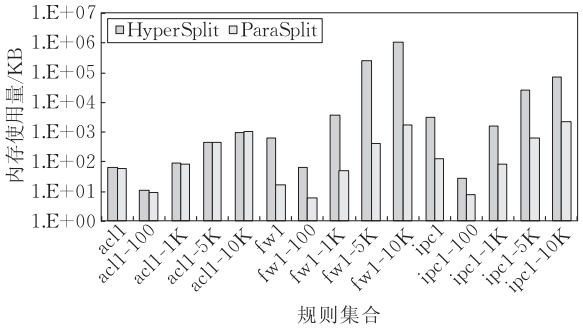

图 10　ParaSplit 算法的内存使用

3.3　本节总结

基于计算几何的空间分解算法是网包分类问题的研究基础. 采用不同的空间分解方法可以获取不同的时间和空间性能折中(time-space tradeoff)，以满足各类网包分类应用的需求. 空间分解算法结合启发式方法，可以利用真实规则集合的冗余性进一步提高分类速率，降低内存使用. 在算法优化的研究中，规则分组算法采用分而治之的策略通过对复杂规则集合的分解提高处理效率.

除本节介绍的网包分类算法之外，还有基于 Hash 方法[43-48]和基于 Ternary CAM 的网包分类算法[49-56]. 这些方法通常需要专用的硬件(如 ASIC)或存储设备(TCAM)来实现，而本文重点讨论基于可编程的多核系统和 FPGA 平台的网包分类算法，因此这里不再详述.

4　网包分类系统实现与测试

现有的网包分类研究，大多集中于网包分类算法的设计及其软件仿真实验，仅有少量的并不全面的真实系统评测[57-61]. 为了客观真实地反映网包分类系统的性能，我们将第 3 节中介绍的一系列网包分类算法实现于高性能多核处理器平台及 FPGA 硬件仿真平台上，并利用大量真实规则进行了全面测试.

4.1　基于多核平台的网包分类系统

测试采用 Cavium OCTEON3860 多核处理器. 该处理器包含 16 个运行在 500 MHz 的 MIPS 核心，8 个千兆 RGMII 的网络接口. 存储体系包括 32 K 字节的每核独享 L1 缓存，1 M 字节的共享 L2 缓存以及 2 G 字节的 DDR2 主存. 在开发编程工作中，我们使用了基于 Cavium SDK version 1.5 的 Simple Executive 模式以保证最大程度的发挥处理器性能.

基于 OCTEON3860 多核处理器平台的测试结果见图 11 和图 12. 我们实现了 HiCuts、HSM 和 HyperSplit 3 种算法. 从图 11 的测试可以看出，随着处理器核心数的增加，3 种算法的吞吐率增长都接近线性. 当使用全部 16 个核的时候，HyperSplit 算法达到了 6.4 Gbps 吞吐率(测试使用 64 字节网包)，其性能是 HSM 的 2 倍，HiCuts 的 4 倍. 图 12 是开启全部 16 个核的情况下，不同大小网包的吞吐测试结果. 由图可见，HyperSplit 算法在网包包长大于或等于 128 字节时，即可达到 100%线速(8 Gbps)，而 HSM 和 HiCuts 分别在 256 字节和 1024 字节才达到线速处理. 由此可见，HyperSplit 算法可以在多核系统平台上得到高效实现，提供接近线速的网包处理能力. 相比 HiCuts 算法，HyperSplit 算法的优势主要在于处理每个网包的内存访问次数少，因此处理速度快. 相比 HSM 算法，HyperSplit 算法的优势在于内存使用少，使得多核系统的缓存能够得到充分利用，从而降低每次访存的延迟，最终提高了整体处理速度.

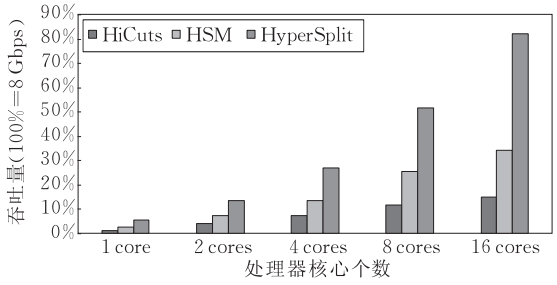

图 11　网包分类算法在多核处理器上的性能(64 字节网包)

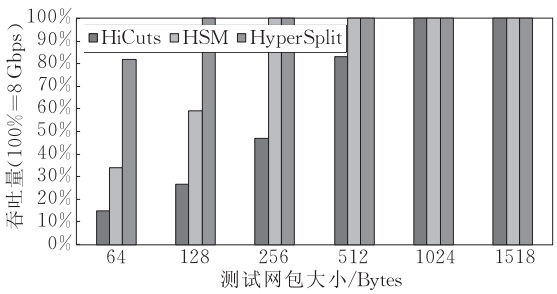

图 12 网包分类算法在多核处理器上的性能(16 核)

4.2 基于 FPGA 平台的网包分类系统

参考已有的基于 FPGA 的网包分类算法实现[62-68],在测试中我们使用了 Xilinx Virtex-6(model: XC6VSX475T)芯片. 该芯片的计算资源为 37440 可编程逻辑单元(configurable logic blocks). 包含 7640 K 比特分布式存储单元(Distributed RAM)以及 38304 K 比特块存储单元(Block RAMs). 所有实验结果使用 Xillinx 的 ISE 仿真平台获取.

从表 5 的数据比较可以看出,FPGA-Split 算法在单一 FPGA 平台上可以达到 142 Gbps 的吞吐率(测试使用 64 字节网包). FPGA 的最高可用时钟频率会随着分类规则个数增加而降低. 这是由于规则数增多导致内存使用增多,从而增加了 FPGA 内部互联的复杂性,进而导致电路延迟增加及时钟频率下降. 从逻辑和存储单元使用率上可见,当前商用 FPGA 芯片已经可以支持大规模高性能的网包分类实现.

表 5 FPGA-Split 性能分析(64 字节网包)

规则个数	FPGA 最高频率/MHz	最大吞吐/Gbps	逻辑单元使用率	存储单元使用率
100	139.1	142	444/37440	10/516
1000	134.0	137	602/37440	18/516
10000	115.4	118	747/37440	103/516

表 6 比较了多种算法在不同平台上的进行 64 字节网包分类的吞吐率. 由该表可知,基于多核处理器的网包分类可以达到 10 Gbps 吞吐率,而基于 FPGA 的硬件实现可以达到 100 Gbps 或更高的性能. 不同的处理器平台适用于不同的网络应用,其开发难度、系统功耗、可拓展性也各有不同. 多核平台可以有效用于多业务的处理,如包含网包分类功能的安全网关实现等. 硬件系统则更适用于高性能网络交换设备,为其提供 100 Gbps 以上的网包分类功能.

表 6 多种平台性能对比(64 字节网包)

算法/平台	最大吞吐率/Gbps
基于 FPGA 的 FPGA-Split 算法	142.0
基于 OCTEON3860 的 HyperSplit 算法	6.4
基于 IXP2850 的 AggreCuts 算法	10.0

5 结论和展望

本文从理论分析、算法设计和系统实现三方面介绍和分析了近年来高性能网包分类算法的研究成果. 首先利用计算几何领域中的相关数学方法,阐明了网包分类问题的理论方法及复杂度分析,指出不同类型的网包分类算法普遍遵循的理论依据. 接下来,对具有影响力的网包分类算法按照不同的研究方向进行了归类,同时结合自身研究成果对不同类别算法的设计思路行了深入分析. 最后,将几类具有代表性的网包分类算法分别实现于多核网络处理器平台以及 FPGA 硬件平台上,并通过实际测试比较了不同类型算法在不同系统平台上的真实性能. 总而言之,理论分析为网包分类算法研究指明了方向. 但数学方法的复杂度并不能代表实际应用的情况. 因此在算法设计中,往往利用启发式方法、数据结构压缩方法、并行处理方法来消除规则冗余,提高分类速率. 此外,不同的网包分类算法实现于不同的软硬件平台时,需要依据平台的计算、存储、总线等特性进行优化,否则无法有效利用系统资源以获取最优的网包分类性能.

随着云计算数据中心、移动互联网、大规模物联网等新兴网络的出现,网包分类算法的研究将面临新的挑战. 首先是性能的需求,支持百万规则的具备 T 比特吞吐率的网包分类算法将成为研究热点,这类算法可以满足未来骨干网络和数据中心网络的带宽. 其次是业务的需求,具备网络业务识别能力的应用层网包分类算法将有重要研究价值,这类算法将为网络融合提供多业务的管控机制及服务质量保证. 最后,网包分类算法的芯片化研究也具有重要意义,具备高速网包分类功能的网络处理器将有力推动下一代互联网设备的发展.

参 考 文 献

[1] Varghese G. Network Algorithmics: An Interdisciplinary Approach to Designing Fast Networked Devices. New York: Morgan Kaufmann Publishers, 2005

[2] Chao J, Liu B. High Performance Switches and Routers. New York: Wiley, 2007

[3] Lin Chuang, Shan Zhi-Guang, Ren Feng-Yuan. Quality of Service of Computer Networks. Beijing: Tsinghua University Press, 2004(in Chinese)
(林闯, 单志广, 任丰原. 计算机网络的服务质量(QoS). 北京: 清华大学出版社, 2004)

[4] Xu Ke, Wu Jian-Ping, Xu Ming-Wei. Advanced Computer Networks: Architecture, Protocol Mechanism, Algorithm Design and Router Technology. 2nd Edition. Beijing: China Machine Press, 2009(in Chinese)
(徐恪，吴建平，徐明伟. 高等计算机网络：体系结构、协议机制、算法设计与路由器技术. 第 2 版. 北京：机械工业出版社，2009)

[5] Casado M, Freedman M J, Pettit J, Luo J, McKeown N, Shenker S. Ethane: Taking control of the enterprise//Proceedings of the ACM SIGCOMM. New York, USA, 2007: 1-12

[6] Joseph D, Tavakoli A, Stoica I. A policy-aware switching layer for data centers//Proceedings of the ACM SIGCOMM. Seattle, USA, 2008: 51-62

[7] Koponen T, Casado M, Gude N, Stribling J, Poutievski L, Zhu M, Ramanathan R, Iwata Y, Inoue H, Hama T, Shenker S. Onix: A distributed control platform for large-scale production networks//Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10). Vancouver, Canada, 2010: 351-364

[8] Popa L, Egi N, Ratnasamy S, Stoica I. Building extensible networks with rule-based forwarding//Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation(OSDI). Vancouver, Canada, 2010: 379-392

[9] Gupta P, McKewon N. Algorithms for packet classification. IEEE/ACM Transactions on Network, 2001, 15(2): 24-32

[10] Taylor D E. Survey and taxonomy of packet classification techniques. ACM Computer Survey, 2005, 37(3): 238-275

[11] Tian Li-Qin, Lin Chuang. Study and application of packet classification. Journal of Computer Research and Development, 2003, 40(6): 765-775(in Chinese)
(田立勤，林闯. IP 报文分类技术的研究及其应用. 计算机研究与发展，2003，40(6)：765-775)

[12] Xu Ke, Xu Ming-Wei, Wu Jian-Ping, Yu Zhong-Chao. Study of IP classification technology: A survey. Mini-micro Systems, 2002, 23(7): 773-779(in Chinese)
(徐恪，徐明伟，吴建平，喻中超. IP 分类技术研究综述. 小型微型计算机系统，2002，23(7)：773-779)

[13] Overmars M, Stappen A. Range searching and point location among fat objects. Journal of Algorithms, 1996, 21(3): 629-656

[14] Deng Jun-Hui. Computational Geometry Algorithms and Applications. 2nd Edition. Beijing: Tshinghua University Press, 2005(in Chinese)
(邓俊辉(译著). 计算几何——算法与应用. 第 2 版. 北京：清华大学出版社，2005)

[15] Chazelle B, Guibas L. Fractional cascading I: A data structuring technique. Algorithmica, 1986, 1: 133-162

[16] Chang Y, Lin Y, Lin C. Grid of segment trees for packet classification//Proceedings of the IEEE Advanced Information Networking and Applications. Perth, Australia, 2010: 1144-1149

[17] Qi Yaxuan, Xu Lianghong, Yang Baohua, Xue Yibo, Li Jun. Packet classification algorithms: From theory to practice//Proceedings of the IEEE INFOCOM. Rio, Brazil, 2009: 648-656

[18] Srinivasan V, Varghese G, Suri S, Waldvogel M. Fast and scalable layer four switching//Proceedings of the ACM SIGCOMM. Vancouver, Canada, 1998: 191-202

[19] Feldmann A, Muthukrishnan S. Tradeoffs for packet classification//Proceedings of IEEE INFOCOM. Tel-Aviv, Israel, 2000: 1193-1202

[20] Gupta P, McKeown N. Packet classification on multiple fields//Proceedings of the ACM SIGCOMM. Cambridge, Massachusetts, USA, 1999: 147-160

[21] Taylor D, Turner J. ClassBench: A packet classification benchmark. IEEE/ACM Transactions on Network, 2007, 15(3): 499-511

[22] Gupta P, McKeown N. Classifying packets with hierarchical intelligent cuttings. IEEE Micro, 2000, 20(1): 34-41

[23] Singh S, Baboescu F, Varghese G, Wang J. Packet classification using multidimensional cutting//Proceedings of the ACM SIGCOMM. Karlsruhe, Germany, 2003: 213-224

[24] Vamanan B, Voskuilen G, Vijaykumar T. EffiCuts: Optimizing packet classification for memory and throughput//Proceedings of the ACM SIGCOMM. New Delhi, India, 2010: 207-218

[25] Qi Yaxuan, Xu Bo, He Fei, Yang Baohua, Yu Jianming, Li Jun. Towards high-performance flow-level packet processing on multi-core network processors//Proceedings of the ACM/IEEE ANCS. Orland, Florida, USA, 2007: 17-26

[26] Baboescu F, Varghese G. Scalable packet classification//Proceedings of the ACM SIGCOMM. San Diego, California, USA, 2001: 199-210

[27] Baboescu F, Singh S, Varghese G. Packet classification for core routers: Is there an alternative to CAMs//Proceedings of the IEEE INFOCOM. San Francisco, California, USA, 2003: 53-63

[28] Pao D, Liu C. Parallel tree search: An algorithmic approach for multi-field packet classification. Computer Communications, 2007, 30(2): 302-314

[29] Lu W, Sahni S. Packet classification using space-efficient pipelined multibit tries. IEEE Transactions on Computers, 2008, 57(5): 591-605

[30] Zheng B, Lin C, Peng X. AM-Trie: An OC-192 parallel multidimensional packet classification algorithm//Proceedings of the IMSCCS. Hangzhou, Zhejiang, China, 2006: 377-384

[31] Zhang Yan-Jun, Chen You, Guo Li, Cheng Xue-Qi. A recursive packet classification algorithm based on decision tree. Journal of Beijing University of Posts and Telecommunications, 2006, 29(2): 45-48(in Chinese)
(张艳军，陈友，郭莉，程学旗. 基于决策树的递归包分类算法. 北京邮电大学学报，2006，29(2)：45-48)

[32] Zhang Shu-Zhuang, Luo Hao, Fang Bin-Xing. A parallel packet classification algorithm with real-time incremental update. Journal of Computer Research and Development, 2010, 47(11): 1903-1910(in Chinese)

(张树壮，罗浩，方滨兴. 一种支持实时增量更新的并行包分类算法. 计算机研究与发展，2010，47(11)：1903-1910)

[33] Woo T. A modular approach to packet classification：Algorithms and results//Proceedings of the IEEE INFOCOM. Tel-Aviv，Israel，2000：1213-1222

[34] Lakshman T，Stiliadis D. High-speed policy-based packet forwarding using efficient multi-dimensional range matching//Proceedings of the ACM SIGCOMM. Vancouver，Canada，1998：203-214

[35] Xu Bo，Jiang Dongyi，Li Jun. HSM：A fast packet classification algorithm//Proceedings of the International Conference on Advanced Information Networking and Applications (AINA). Taipei，China，2005：987-992

[36] Xu Bo，Qi Yaxuan，He Fei，Zhou Zongwei，Xue Yibo，Li Jun. Fast path session creation on network processors//Proceedings of the ICDCS. Beijing，China，2008：573-580

[37] Liu Duo，Hua Bei，Hu Xianghui，Tang Xinan. High-performance packet classification algorithm for many-core and multithreaded network processor//Proceedings of the IEEE International Conference on Compilers，Architecture，and Synthesis for Embedded Systems (CASES). Seoul，Korea，2006：334-344

[38] Qi Yaxuan，Fong Jeffrey，Jiang Weirong，Xu Bo，Li Jun，Prasanna V. Multi-dimensional packet classification on FPGA：100Gbps and beyond//Proceedings of the Internation Confererence on Field-Programmable Technology (FPT). Beijing，China，2010：241-248

[39] Lu H，Pan M. Partition filter set for power-efficient packet classification//Proceedings of the IEEE GLOBECOM. San Francisco，California，USA，2006：1-5

[40] Zheng K，Liang Z，Ge Y. Gear up the classifier：Scalable packet classification optimization framework via rule set preprocessing//Proceedings of the ISCC. Pula-Cagliari，Sardinia，Italy，2006：814-819

[41] Jiang W，Prasanna V，Yamagaki N. Decision forest：A scalable architecture for flexible flow matching on FPGA//Proceedings of the FPL. Milano，Italy，2010：394-399

[42] Fong Jeffrey，Qi Yaxuan，Li Jun，Jiang Weirong. ParaSplit：An FPGA-optimized packet classification engine. THU-NSLAB Technical Report，2011

[43] Srinivasan V，Suri S，Varghese G. Packet classification using tuple space search//Proceedings of the ACM SIGCOMM. Cambridge，Massachusetts，USA，1999：135-146

[44] Song H，Turner J，Dharmapurikar S. Packet classification using coarse-grained tuple spaces//Proceedings of the ANCS. San Jose，California，USA，2006：41-50

[45] Dharmapurikar S，Song H，Turner J，Lockwood J. Fast packet classification using bloom filters//Proceedings of the ANCS. San Jose，California，USA，2006：61-70

[46] Yu H，Mahapatra R. A memory-efficient hashing by multi-predicate bloom filters for packet classification//Proceedings of the IEEE INFOCOM. Phoenix，AZ，USA，2008：1795-1803

[47] Pus V，Korenek J. Fast and scalable packet classification using perfect hash functions//Proceedings of the ACM/SIGDA FPGA. Monterey，California，USA，2009：229-236

[48] Priya A，Lim H. Hierarchical packet classification using a bloom filter and rule-priority tries. Computer Communications (COMCOM)，2010，33(10)：1215-1226

[49] Spitznagel E，Taylor D，Turner J. Packet classification using extended TCAMs//Proceedings of the ICNP. Atlanta，Georgia，USA，2003：120-131

[50] Lakshminarayanan K，Rangarajan A，Venkatachary S. Algorithms for advanced packet classification with ternary CAMs//Proceedings of the ACM SIGCOMM. Philadelphia，PA，USA，2005：193-204

[51] Zheng K，Che H，Wang Z，Liu B. TCAM-based distributed parallel packet classification algorithm with range-matching solution//Proceedings of the IEEE INFOCOM. Miami，FL，USA，2005：293-303

[52] Liu A，Meiners C，Torng E. TCAM Razor：A systematic approach towards minimizing packet classifiers in TCAMs. IEEE Transactions on Networking，2007，18(2)：490-500

[53] Meiners C，Liu A，Torng E. Topological transformation approaches to optimizing TCAM-based packet classification systems//Proceedings of the ACM SIGMETRICS. Seattle，Washington，USA，2009：73-84

[54] Meiners C，Liu A，Torng E. Bit Weaving：A Non-prefix approach to compressing packet classifiers in TCAMs//Proceedings of the ICNP. Princeton，NJ，USA，2009：93-102

[55] Pao D，Li Y，Zhou P. Efficient packet classification using TCAMs. Computer Networks，2006，50(18)：3523-3535

[56] Sun Y，Kim M. Bidirectional range extension for TCAM-based packet classification//Proceedings of the Networking. Chennai，India，2010：351-361

[57] Kounavis M，Kumar A，Vin H，Yavatkar R，Campbell A. Directions in packet classification for network processors//Proceedings of the Network Processor Workshop. Anaheim，California，USA，2003：10-22

[58] Cohen E，Lund C. Packet classification in large ISPs：Design and evaluation of decision tree classifiers//Proceedings of the ACM SIGMETRICS. Banff，Alberta，Canada，2005：73-84

[59] Srinivasan D，Feng W. Performance analysis of multi-dimensional packet classification on programmable network processors//Proceedings of the LCN. Tampa，Florida，2004：360-367

[60] Sahasranaman V，Buddhikot M. Comparative evaluation of software implementation of Layer-4 packet classification schemes//Proceedings of the ICNP. Riverside，California，USA，2001：220-228

[61] Li Yun-Hui. Research and design of packet classification system based on network processor [M. S. dissertation]. Beijing Jiaotong University，Beijing，2008(in Chinese)
(李昀晖. 基于网络处理器的流分类系统研究与设计[硕士学位论文]. 北京交通大学，北京，2008)

[62] Song H，Lockwood J. Efficient packet classification for network intrusion detection using FPGA//Proceedings of the

ACM/SIGDA FPGA. Monterey, California, USA, 2005: 238-245

[63] Jiang W, Prasanna V. Large-scale wire-speed packet classification on FPGAs//Proceedings of the ACM/SIGDA FPGA. Monterey, California, USA, 2009: 219-228

[64] Jiang W, Prasanna V. Field-split parallel architecture for high performance multi-match packet classification using FPGAs//Proceedings of the SPAA. Calgary, Canada, 2009: 188-196

[65] Xu Y, Liu Z, Zhang Z, Chao J. An ultra high throughput and memory efficient pipeline architecture for multi-match packet classification without TCAMs//Proceedings of the ANCS. Princeton, NJ, USA, 2009: 189-198

[66] Zhang T, Wang Y, Zhang L, Yang Y. High throughput architecture for packet classification using FPGA//Proceedings of the ANCS. Princeton, NJ, USA, 2009: 62-63

[67] Yan T, Wang Y. Design of packet classification co-processor with FPGA//Proceedings of the ESA. Mallorca, Spain, 2005: 275-275

[68] Basu A, Narlikar G. Fast incremental updates for pipelined forwarding engines//Proceedings of the IEEE INFOCOM. San Francisco, California, USA, 2003: 64-74

QI Ya-Xuan, born in 1979, Ph. D.. His research interests include network algorithms and architecture.

LI Jun, born in 1962, Ph. D., professor. His research interests include network security and architecture.

Background

With the evolution of Internet architecture and the flourishing of Internet applications, traditional IP forwarding is not capable of meeting the increasing demands for network services and security. For example, multimedia applications require good quality of services, enterprise network needs secure access control, and datacenter networks require isolation of multiple tenants. Because multi-dimensional packet classification can distinguish network traffic in fine grain, it has been widely used in modern routers, firewalls and traffic management systems.

Studies on multi-dimensional packet classification algorithms include theoretical analysis, algorithm design and system implementation. Theoretical analysis is the basis of design and evaluation of multi-dimensional packet classification algorithms. It specifies the worst-case bounds of temporal and spatial performance. In computational geometry, packet classification is equivalent to the point location problem; therefore, it can be solved by recursively binary partition using either segment trees or space decomposition algorithms.

Although theoretical analysis shows that both of the algorithms have very high temporal and spatial complexity, fortunately, real-life packet classification rule sets have structural redundancies that can be leveraged by heuristics to achieve good performance in practical applications. Segment tree based algorithms, such as RFC, HSM and HyperSplit, partition the search space recursively using end-point selection heuristics. Space decomposition algorithms, such as HiCuts, HyperCuts and AggreCuts, partition the search space by heuristic controlled uniform cuttings.

In practice, both of these heuristic-based algorithms are widely used in existing routers and firewalls. However, algorithms based on different heuristics may have different performance. Therefore, performance evaluation of packet classification algorithms is important to real-life applications. Generally, most important performance metrics are search speed, memory usage, and preprocessing time. Experimental results on a large number of rule sets are necessary for objective algorithm evaluation. To efficiently implement multi-dimensional packet classification algorithms on hardware platforms, such as multi-core network processors and FPGA chips, hardware limitation needs to be carefully considered. Algorithms to be deployed in hardware platforms need to be optimized by I/O bus bandwidth, on-chip memory size and parallelism schemes.

In this survey, we introduce recent advances of multi-dimensional packet classification algorithms from theory to practice. First, we provide theoretical basis of the packet classification problem. Then we categorize and analyze existing packet classification algorithms according to different research directions. To evaluate system-level performance, we also implement typical algorithms on both multi-core network processor and FPGA hardware platforms, and use real-life data sets for objective performance test. Finally, we state our conclusion and discuss the future work on high-performance multi-dimensional packet classification algorithms.