Journal of Computer Applications
计算机应用,2013,33(9):2450 - 2454
ISSN 1001-9081
CODEN JYIIDU
2013-09-01
http://www.joca.cn

文章编号:1001-9081(2013)09-2450-05
doi:10.11772/j.issn.1001-9081.2013.09.2450

基于规则集划分的多决策树报文分类算法

马　腾*,陈庶樵,张校辉,田　乐
(国家数字交换系统工程技术研究中心,郑州 450002)
(*通信作者电子邮箱 123mateng321@163.com)

摘　要:为克服决策树算法处理高速网络、大容量规则集下的报文分类问题时内存使用量大的弊端,提出一种基于规则集划分的多决策树报文分类算法。在保证规则子集数量可控的前提下,采用启发式算法将规则集划分为有限个规则子集,最大限度分离交叠规则;提出两级级联决策树结构,降低决策树深度以减少规则查找时间。理论分析表明,该算法空间复杂度较传统单决策树算法大幅降低。仿真结果表明,该算法的内存使用量比目前空间性能最好的EffiCuts算法减少了30%,且维度可扩展性更好。

关键词:报文分类;规则集划分;多决策树;内存使用量;大容量规则集
中图分类号:TP393.0　　**文献标志码**:A

Multiple decision-tree packet classification algorithm based on rule set partitioning

MA Teng*, CHEN Shuqiao, ZHANG Xiaohui, TIAN Le
(*National Digital Switching System Engineering and Technology Research Center, Zhengzhou Henan 450002, China*)

Abstract: For solving the problem of decision-tree algorithms' too much memory usage when coping with packet classification under the circumstance of high rate network and large volume rule set, a multiple decision-tree algorithm based on rule set partitioning was proposed in this paper. On the condition of controlling the number of subsets, heuristics were used to partition the rule set into limited number of subsets, in which the overlapping rules had been separated. Cascading decision-tree structure was proposed to lower the depth and reduce search time. The theoretical analysis shows that space complexity has been reduced greatly compared to traditional single decision-tree algorithm. The simulation results demonstrate that the algorithm reduces memory usage about 30% and has better dimension scalability when being compared with EffiCuts, which has the best performance for memory usage so far.

Key words: packet classification; rule set partitioning; multiple decision-tree; memory usage; large volume rule set

0　引言

报文分类是网络应用领域的关键技术之一。目前业界的解决方案主要有两种:基于硬件的三态内容可寻址寄存器(Ternary Content Addressable Memory,TCAM)和基于随机存取存储器(Random Access Memory,RAM),它们均可以线速处理报文。随着链路带宽不断增加、网络应用日益多元化,分类规则集呈现出新的特点:容量增大、规则维数增多、范围规则大量出现,使得基于TCAM的多域报文分类算法举步维艰(TCAM不宜处理范围规则)[1-3],而运行于可编程门阵列(Field Programmable Gate Array,FPGA)+RAM架构的决策树算法在规则集容量、规则维数方面扩展性强,且适合处理范围规则,成为研究热点。

规则集中的规则在某些域相互交叠,使得这类算法在预处理阶段构建立决策树时,不可避免出现规则复制,带来严重的存储空间消耗。受限于高速存储器的容量,高速网络、大容量规则集下的报文分类算法必须解决内存消耗量大的问题。为此在预处理阶段需要对规则集进行合适的划分,使得规则子集内部的规则相互交叠的概率大幅降低,从而达到抑制规则复制、减少算法内存使用量的目的。划分规则集、建立多决策树结构的报文分类算法是本文研究的内容。

1　研究背景

报文分类可模型化为计算几何中的多维空间点定位问题,以二维规则集为例,如图1所示,规则R_1、R_2、R_3、R_4视为二维空间中的矩形(带优先级),报文头部提取到的关键字视为空间中的点P,则报文分类过程就是查询覆盖点P的优先级最高的矩形,再按照该矩形对应的规则策略处理与点P对应的报文。

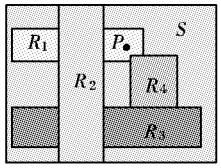


图1　报文分类问题的数学模型

理论研究[4]表明,维数$d>3$时,报文分类算法时间复杂度下限为$O(\log N)$(N为规则数目),此时空间复杂度为$O(N^d)$;空间复杂度下限为$O(N)$,此时时间复杂度为$O(\log^{d-1} N)$。N较大时,一般很难保证时间复杂度与空间复杂度同时满足要求。通常情况下都是根据实际应用环境,利

收稿日期:2013-03-25;**修回日期**:2013-05-25。　**基金项目**:国家863计划项目(2011AA01A103);国家973计划项目(2012CB315901,2012CB315906);国家科技支撑计划项目(2011BAH19B01)。

作者简介:马腾(1987 -),男,陕西西安人,硕士研究生,主要研究方向:报文分类;　陈庶樵(1973 -),男,河南郑州人,教授,博士,主要研究方向:网络安全;　张校辉(1979 -),男,河南洛阳人,讲师,博士,主要研究方向:异常流检测;　田乐(1987 -),男,陕西咸阳人,硕士研究生,主要研究方向:报文分类。

用规则集的统计特征提高算法效率。

2　相关工作

早期基于决策树的多域报文分类算法包括 HiCuts 算法[5]和 HyperCuts 算法[6],均折中考虑了分类速率与内存使用量,但它们采用均匀切分规则空间的启发式算法不可避免带来规则复制(见图 2 左图,R_2 在左侧子节点出现复制)。近年出现的 HyperSplit 算法[7]提出了沿投影端点非均匀切分规则空间的思路,减小了规则复制的概率,算法内存使用量大幅降低(见图 2 右图,没有规则复制),但 HyperSplit 对规则空间的二分处理导致决策树深度过大,算法基于 FPGA 实现的逻辑复杂度极高,使得系统无法提供满足需求的吞吐率。

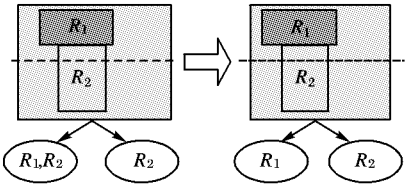


图 2　规则空间的均匀切分和沿投影端点的非均匀切分

文献[8]提出的 EffiCuts 算法引入了规则集划分与重构的思想,实现大规则与小规则分类处理,一定程度上分离了交叠的规则,算法空间性能较 HyperCuts 大为改进。美中不足是 EffiCuts 的维度扩展性差,当规则域数增多时(如基于 OpenFlow[9]的软件定义网络,路由设备需处理 12 个维度的报文分类),决策树数量剧增,导致算法基于 FPGA 实现的逻辑复杂度极高,系统无法提供满足现有网络需求的吞吐率。文献[10]中提出了一种划分规则集的有效方法,使得划分后每个规则子集内部的规则在某个维度上严格不交叠,但该算法产生的规则子集数量较多,导致需要建立过多的决策树,同样增加了硬件实现的逻辑复杂度。

本文提出的基于规则集划分的多决策树报文分类算法(Multiple Decision-tree packet classification algorithm based on Rule Set Partition, RSP-MD),主要基于对现有算法三个方面的改进:1)在规则子集数量可控的前提下,采用启发式算法逐步划分规则集,最大限度分离交叠的规则;2)提出两级级联的决策树结构,降低决策树深度;3)提高算法的维度扩展性,使之能应对高速网络中维数增加情况下的报文分类问题。

3　算法描述

本文算法包含两个部分:规则集划分算法和决策树构建算法。首先通过规则集划分算法将原始规则集划分为多个规则子集,然后对每个规则子集建立决策树,形成多决策树并行查找的系统架构。

3.1　相关定义

定义 1　报文分类。给定规则集 C 和 N 条规则 R_j, $1<j<N$, $C=\{R_1,R_2,\cdots,R_N\}$, R_j 由三部分组成:

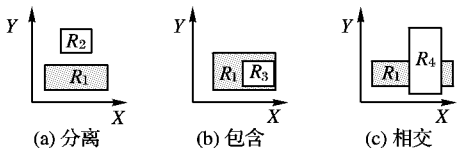
1) 范围表达式 $R_j[i]$, $1\leqslant i\leqslant d$,规则包含的 d 个字段;

2) 优先级 $pri(R_j)$,当报文同时匹配多条规则时,用于确定最佳匹配规则;

3) 规则决策 $action(R_j)$,对报文的处理动作。

对收到的报文 $P=\{P_1,P_2,\cdots,P_d\}$,在 C 中查找到与之相匹配且优先级最高的规则 R_m,即 R_m 满足 $pri(R_m)>pri(R_j)$, $\forall j\neq m$, $1\leqslant j\leqslant N$, $1\leqslant m\leqslant N$,该过程称为报文分类。

定义 2　规则之间的关系。对于任意两条规则,它们之间可能的关系有如图 3 所示的 3 种。



(a) 分离　(b) 包含　(c) 相交

图 3　规则之间的关系示例

1)分离($R_i\cap R_j=\varnothing$),如图 3(a)中的 R_1 和 R_2。

2)包含($R_i\subseteq R_j$),如图 3(b)中的 R_1 和 R_3,这里 $R_3\subseteq R_1$。

3)相交($R_i\cap R_j\neq\varnothing$且两者不满足 2)中的包含关系),如图 3(c)中 R_1 和 R_4。

后文将包含和相交关系统一称为交叠。

定义 3　独立规则子集。记 I 为 C 的一个规则子集,即 $I\subseteq C$,若 I 满足条件:$\forall R_i,R_j\in I$, R_i 和 R_j 在维度 x 上满足定义 2 中的分离关系,称 I 是 C 的维度 x 上的独立规则子集;反之,称 I 为 C 的非独立规则子集。对 C 所有 x 维度上的独立规则子集,称包含规则数量最多的规则子集为该维度的最大独立规则子集。

3.2　规则集划分算法

在固定规则子集数量前提下,每一轮采用启发式算法选择维度,使得沿该维度划分能够最大限度降低规则的交叠程度,在该维度上提取出当前规则集的一个最大独立规则子集;然后从当前规则集中去除该独立规则子集,进行下一轮的划分,直至规则子集数量达到预先设定的值或当前规则集已经为空,划分完毕。

3.2.1　划分维度的选取

划分的目的是最大限度地降低规则子集内部规则的交叠程度。一般而言,规则在各个维度的交叠程度是不同的,因此,需要选择合适的维度进行规则集的划分。

可供选择的选取划分维度的启发式算法有:

1)包含范围区间数最多的维度。

将当前规则集投影到每个维度,在每个维度形成多个范围区间。设各维度的范围区间数量依次为 $L_i(i=1,2,\cdots,d)$,则选取的划分维度 x 为:

$$x=\underset{i}{\operatorname{argmax}}\ L_i \tag{1}$$

2)投影端点最多的维度。

同 1),投影后在每个维度形成多个投影端点。设各维度的投影端点数量依次为 $K_i(i=1,2,\cdots,d)$,则选取的划分维 x 度为:

$$x=\underset{i}{\operatorname{argmax}}\ K_i \tag{2}$$

经对算法性能的仿真对比(见第 4 章),本文选取 1)中的算法作为确定划分维度的启发式算法。

3.2.2　最大独立规则集的提取

文献[10]提供了一种在某个维度提取最大独立规则集的算法,并从理论上证明该算法得到的规则子集确为当前规则集的最大独立规则子集。本文算法与之类似,以 classifier 表示当前规则集,sub_classifier 表示最终得到的最大独立规则子集,最大独立规则集提取算法 IS_extract 实现的伪码如下所示:

```
//初始化最大独立规则集为空
initialize:   sub_classifier ← ∅;
//将 classifier 中的规则按照其 x 维度右端点的大小升序排列
classifier. sort( );
if( classifier  == ∅)
    return;
```

```
else
    //选取右端点最小的规则 temp_rule 放入 sub_classifier
    temp_rule ← find_min_right_rule( classifier);
    sub_classifier. push_back( temp_rule);
    //删除 classifier 中与 temp_rule 有交叠的规则
    for i := classifier. begin( ) to classifier. end( ) do
        if( is_overlapping( temp_rule, i))
            classifier. erase( i);
        end if
    end for
    goto line 3;                    //返回至 if( classifier == ∅) 处
end if
```

3.2.3 规则集划分算法描述

以 M 表示预先设定的规则子集数量。划分过程采用递归方式实现，最终将规则集划分为 M 个规则子集 $IS[0]$，$IS[1]$，$IS[2]$，…，$IS[M-1]$，算法实现的伪码如下：

```
//初始化: M 个规则子集均为空
    initialize:   i ← 0, sub_classifier[0] ← ∅;
    if((i < (M - 1)) && (classifier != ∅))
        //采用 3.2.1 节提出的启发式算法选择“最佳”维度 x
        x ← cal_dim_to_split( classifier);
        //提取当前规则集在 x 维度上的一个最大独立规则子集
        IS[i] ← IS_extract( classifier, x);
        //从当前规则集中去除独立规则子集
        classifier ← classifier - IS[i];
```

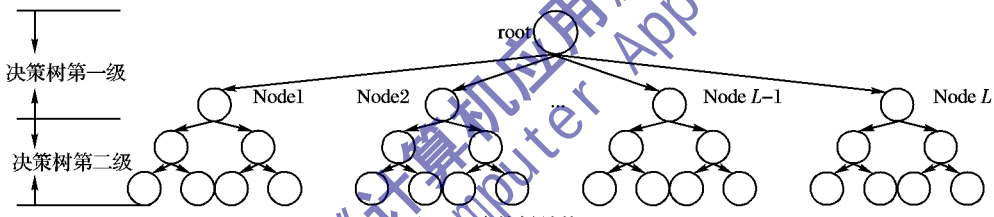


图 4 决策树结构

确定第一级决策树的子节点数量，一般取 2 的幂次，如 4，8，16，根据规则集划分算法的测试结果，当原始规则集规模为 100 000 时，划分后的少数规则子集容量大于 10 000，其余规则子集的容量仅为数千条，因此这里：对规则容量大于 10 000的子集，决策树第一级的子节点数量定为 16；对规则容量不足10 000的子集，决策树第一级的子节点数量定为 8。需要注意的是，两级决策树结构是针对前 $M-1$ 个规则子集设计的，它们均为独立规则集，对第 M 个规则子集（非独立规则子集），直接按照 hyperSplit 算法建立决策树即可。决策树构建算法的实现如下所示：

```
//处理前 M - 1 个规则子集, 建立两级决策树
for  i := 0 to M - 2  do
    //将规则子集中的规则按其 x 维度右端点的大小升序排列
    sub_classifier[i]. sort( );
    //确定第一级决策树的子节点数量
    if( sub_classifier[i]. size( ) > 10 * 1024)
        num_cuts := 16;
    else
        num_cuts := 8;
    end if
    //对各子节点对应的规则集按 HyperSplit 算法建立
    //第二级决策树
    num_rules ← sub_classifier[i]. size( ) / num_cuts;
    initialize  hs_classifier[num_cuts] ← {∅}
    for  j := 0 to num_cuts-1  do
        for  k := 0 to num_rules-1  do
            temp_rule = sub_classifier[i]. pop_up( );
            hs_classifier[j]. push_back( temp_rule);
        end for
        create_hypersplit_tree( hs_classifier[j]);
    end for
end for
//处理第 M 个规则子集, 按 HyperSplit 算法建立决策树
create_hypersplit_tree( sub_classifier[M - 1]);
```

3.2.2 规则查找

基于决策树的报文分类算法按如下过程完成规则查找：报文到来后，提取头部关键字，逐层遍历决策树，直至找到叶节点，再线性查找叶节点对应的少量规则，输出匹配规则索引。在多决策树的报文分类系统中，上述过程在所有决策树中并行执行，最后通过优先级判决器，输出优先级最高的规则索引，见图 5。

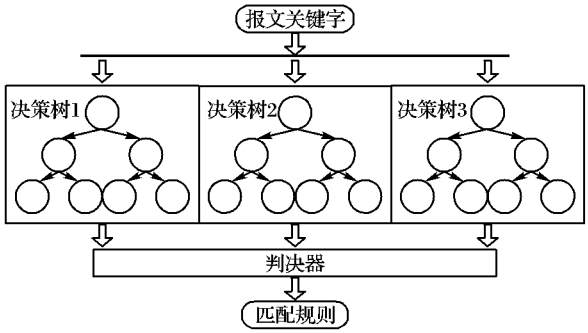


图 5 多决策树报文分类算法的查找过程

3.2.3 规则更新

为提供足够吞吐率，一般基于决策树的报文分类算法通

```
    i ← i + 1;
else
    //最后剩余的规则放入第 M 个规则子集
    IS[i] ← classifier;
    M←i;
end if
```

3.3 基于多决策树的报文分类算法

基于多决策树的报文分类算法包括 3 个部分：决策树构建、规则查找和规则更新。

3.3.1 决策树构建

规则集划分完成后，对每个规则子集分别建立决策树。传统的决策树算法从根节点开始，利用选择切分维度和切分点的启发式算法，连续切分多维规则空间，直至节点对应规则子集包含的规则数量不大于预先设定的门限，该节点为决策树的一个叶节点。

本文对传统决策树结构进行改进，提出两级决策树级联算法[11]：1）对每棵决策树的根节点，由于其对应规则集中的规则在维度 x 上互不交叠，为降低决策树高度，保持规则数量在子节点分布均衡，采取沿 x 维度对规则等数量多点切分的方法；2）对每棵决策树根节点的子节点，以其作为根节点，按照 HyperSplit 算法建立第二级决策树，原因在于 HyperSplit 沿规则投影点切分规则空间的方法大大减少了算法内存使用量。决策树结构见图 4。

过将树形结构在 FPGA 上映射为多级深度流水线实现,而规则的增量更新通过在流水线中插入 Write Bubble 实现[12-13]。首先离线计算每级流水线待更新的节点内容,然后启动更新过程,插入 wtite bubble。每一个 write bubble 分配一个标识符,当写使能位被置位后,write bubble 用新的节点内容更新 FPGA 片上相应的双端口 BRAM[14]。文献[15]中规则的增量更新过程类似查找过程,一定时间后会破坏决策树的优化结构。文献[16]中提到了一种支持大批量规则更新的算法 Back Buffer,但作者没有给出具体的实现过程。

3.4　算法复杂性分析

根据文献[17]的结论,基于决策树的报文分类算法最坏情况下的时间复杂度是 $O(d)$,空间复杂度是 $O(N^d)$,这里 d ($d>3$)为规则的维数,N 为规则集的容量。将规则集划分为 M 个子集,M 个子集的规则容量依次是 $N_1,N_2,\cdots,N_M$,则算法的空间复杂度变为 $O({N_1}^d+{N_2}^d+\cdots+{N_M}^d)$,这里将对算法空间复杂度的下限作深入分析。

定理 1　对于函数 $f(x)=x^d, x\in(0,+\infty)$,取 $\lambda_i\in\mathbf{R}$, $0\leqslant\lambda_i\leqslant 1$ 且 $\sum_{i=1}^{M}\lambda_i=1(i=1,2,\cdots,M)$,则如下不等式成立:

$$f(x)=x^d\geqslant\sum_{i=1}^{N}\lambda_i^d x^d\geqslant x^d/M^{d-1} \tag{3}$$

式(3)右边不等式等号成立的条件是 $\lambda_1=\lambda_2=\cdots=\lambda_M=1/M$。

证明　先证左边的不等式,有已知条件有

$$f(x)=f\left(\sum_{i=1}^{M}\lambda_i x\right)=\left(\sum_{i=1}^{M}\lambda_i x\right)^d \tag{4}$$

注意到 $x\in(0,+\infty)$,省去式(4)右边表达式中的交叉项,得到

$$f(x)=x^d=\left(\sum_{i=1}^{M}\lambda_i x\right)^d\geqslant\sum_{i=1}^{M}\lambda_i^d x^d \tag{5}$$

式(3)左边的不等式得证,再证式(3)右边的不等式。

需要用到幂平均不等式[18]。设 p 是非零实数,定义实数 $x_1,x_2,\cdots,x_M$ 指数为 p 的幂平均为

$$M_p(x_1,x_2,\cdots,x_M)=\left(\frac{1}{M}\sum_{i=1}^{M}x_i^p\right)^{1/p} \tag{6}$$

幂平均不等式可表述为:p、q 均为非零实数,如果 $p<q$,则

$$M_p(x_1,x_2,\cdots,x_M)\leqslant M_q(x_1,x_2,\cdots,x_M) \tag{7}$$

当且仅当 $x_1=x_2=\cdots=x_M$ 时,等号成立。

在幂平均不等式中,令 $p=1,q=d,x_i=\lambda_i x$,就有

$$\frac{1}{M}\sum_{i=1}^{M}\lambda_i x\leqslant\left(\frac{1}{M}\sum_{i=1}^{M}(\lambda_i x)^d\right)^{1/d} \tag{8}$$

化简式(8),得

$$\sum_{i=1}^{N}\lambda_i^d x^d\geqslant\frac{x^d}{M^{d-1}} \tag{9}$$

由幂平均不等式等号成立的条件,当且仅当 $\lambda_1=\lambda_2=\cdots=\lambda_M=1/M$ 时,式(9)取等号,从而式(3)右边不等式成立。　　证毕。

在上述定理中,令 $x=N,\lambda_i=N_i/N$,可以得到如下的结论:经过规则集划分,基于决策树的报文分类算法的空间复杂度大大降低,且当各规则子集的容量相等时,算法空间复杂度最小,为 $O(N^d/M^{d-1})$,仅是原来的 $1/M^{d-1}$。

4　性能仿真

规则集的生成工具采用 ClassBench[19],它目前被广泛采用为规则集生成和测试平台。ClassBench 可以产生三种类型的规则集供仿真使用:ACL(Access Control Lists)、FW(FireWall)和 IPC(IP Chains),三种大的类型又分别包含有 5,5,2 共 12 种子类型。所有规则都由 IPv4 五元组构成:32 bit 源 IP 地址、32 bit 目的 IP 地址、16 bit 源端口号、16 bit 目的端口号、8 bit 协议号。测试环境为 Intel Core i3 2.4 GHz/1.92 GB 微机,Linux 操作系统(Ubuntu 5.10),算法用 C++语言实现(其中 EffiCuts 仿真代码由原作者提供)。

4.1　两种维度选取算法的对比

固定规则子集的数量为 6,规则集的容量均为 10 000,表 4 给出了各种类型规则集下 3.2.1 节选取划分维度的两种启发式算法的测试结果。

表 4　两种启发式算法规则集划分结果对比

规则集类型	算法	#1	#2	#3	#4	#5	#6
ACL1	算法 1	4 276	1 840	929	507	337	1714
	算法 2	4 276	1 840	929	507	337	1 714
FW1	算法 1	6 950	2 260	37	26	21	17
	算法 2	6 950	2 260	26	37	21	17
IPC1	算法 1	1 684	972	708	1 023	527	4 123
	算法 2	1 684	972	708	1 023	527	4 123

表 4 中算法 1 指代基于范围区间数的维度选取算法,算法 2 指代基于投影端点数的维度选取算法,"#"号下方的数据为该规则子集包含的规则数。对比可见两种算法对各种规则集划分的结果几乎是一样的(表 2 中只有 FW1 型规则的划分结果略有不同),而两者的运行时间长短不同(如图 6 所示,相关含义同表 4)。

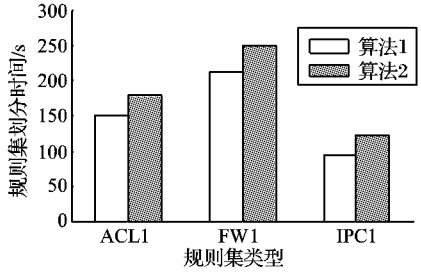


图 6　两种算法的规则集划分时间对比

各种规则集下算法 1 的运行时间约为算法 2 的 80%,原因在于每一次选取维度时,算法 1 需要的比较次数更少(一个范围区间对应两个投影端点)。鉴于此,本文选取算法 1 作为规则集划分的维度选取算法。

4.2　EffiCuts 与 RSP-MD 内存使用量对比

固定叶节点包含的最大规则数量为 16,每个规则集的命名规则为"规则类型_规则数量",如 FW_10K 为 10 000 条防火墙安全策略规则,RSP-MD 算法与 EffiCuts 算法每条规则平均内存使用量对比如图 7 所示,共比较了 6 种不同类型、容量规则集下两种算法的平均内存使用量。

由图 7 可见,各种场景下 RSP-MD 算法较 EffiCuts 算法的内存使用量有明显改善,尤其是规则集容量比较大的时候,如 100 000,用不同类型的规则集测试时,其内存使用量均不超过 EffiCuts 的 70%。这主要得益于采用启发式算法划分规则集和 HyperSplit 算法沿规则边界非均匀切分搜索空间的结果。

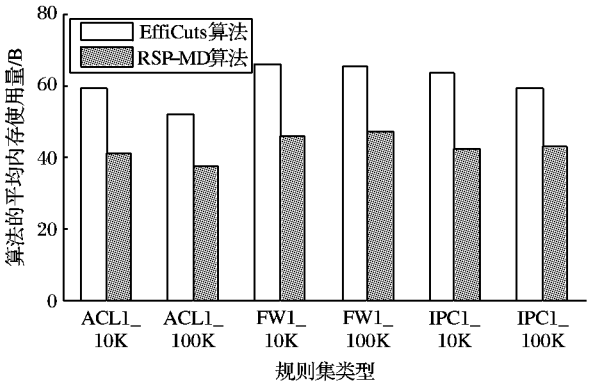


图 7 两种算法的内存使用量对比

4.3 EffiCuts 与 RSP-MD 决策树高度对比

针对 HyperSplit 算法决策树高度大的问题，本文提出了两级决策树级联算法。降低决策树的高度主要是为了降低算法硬件实现的逻辑复杂度。EffiCuts 与 RSP-MD 算法的决策树高度对比如图 8 所示。

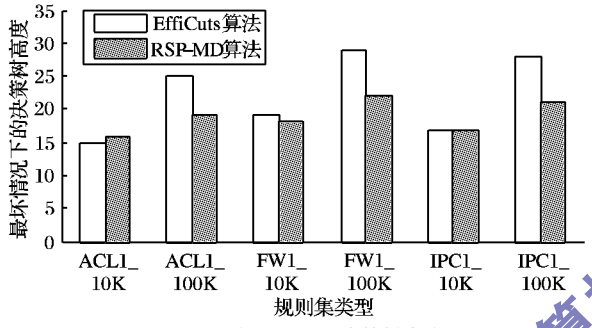


图 8 EffiCuts 与 RSP-MD 决策树高度对比

由图 8 可见，用不同类型的规则集测试，当规则集容量比较大时，如 100 000，RSP-MD 算法较 EffiCuts 算法，决策树高度略有降低，而规则集容量较小时，两者的决策树高度几乎相等。

4.4 EffiCuts 与 RSP-MD 算法运行时间对比

对比算法的预处理时间发现，RSP-MD 算法的预处理时间比 EffiCuts 有小幅增长（约 10%），如图 9 所示，测试中用到的规则集容量均为 100 000。

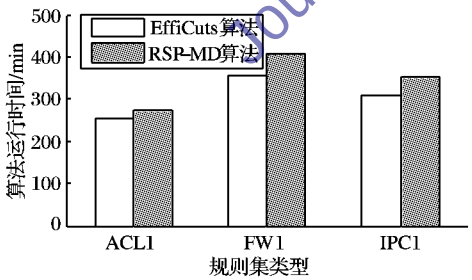


图 9 EffiCuts 与 RSP-MD 的运行时间对比

5 结语

基于决策树的报文分类算法是当前的研究热点，本文提出的 RSP-MD 算法和最为经典的 EffiCuts 算法相比，性能上有如下提升：1）规则子集数量可控，采用启发式算法划分规则集，最大限度分离交叠的规则，规则集容量为 100 000 时内存使用量降低到不到 EffiCuts 的 70%；2）两级级联的决策树结构，和 EffiCuts 有相当的决策树高度，有效减少算法查找时间；3）算法的维度扩展性好，能应对高速网络中维数增加情况下的报文分类问题。下一步工作是算法的硬件实现，即将算法构建的决策树映射成多级深度流水线，运行于 FPGA 平台。

参考文献：

[1] LAKSHMINARAYANAN K, RANGARAJAN A, VENKATACHARY S. Algorithms for advanced packet classification with ternary CAMs[C]// Proceedings of the 2005 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. New York: ACM Press, 2005: 193-204.

[2] LIU A, MEINERS C, TORNG E. TCAM Razor: a systematic approach towards minimizing packet classifiers in TCAMs[J]. IEEE/ACM Transactions on Networking, 2010, 18(2): 490-500.

[3] MEINERS C, LIU A, TORNG E. Bit weaving: a non-prefix approach to compressing packet classifiers in TCAMs[J]. IEEE/ACM Transactions on Networking, 2012, 20(2): 488-500.

[4] OVERMARS M, STAPPEN A. Range searching and point location among fat objects[J]. Journal of Algorithms, 1996, 21(3): 240-253.

[5] GUPTA P, McKEOWN N. Classifying packets with hierarchical intelligent cuttings[J]. IEEE Micro, 2000, 20(1): 34-41.

[6] SINGH S, BABOESCU F, VARGHESE G, *et al*. Packet classification using multidimensional cutting[C]// Proceedings of the 2003 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. New York: ACM Press, 2003: 213-224.

[7] QI Y X, XU L H, YANG B H, *et al*. Packet classification algorithms: from theory to practice[C]// Proceedings of IEEE INFOCOM 2009. Piscataway, NJ: IEEE Press, 2009: 648-656.

[8] VAMANAN B, VOSKUILEN G, VIJAYKUMAR T. EffiCuts: optimizing packet classification for memory and throughput[J]. ACM SIGCOMM Computer Communication Review, 2010, 40(4): 207-218.

[9] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, *et al*. OpenFlow: enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[10] SUN X H, SAHNI S K, ZHAO Y Q Q. Packet classification consuming small amount of memory[J]. IEEE/ACM Transactions on Networking, 2005, 13(5): 1135-1145.

[11] VAMANAN B, VIJAYKUMAR T. TreeCAM: decoupling updates and lookups in packet classification[C]// Proceedings of the 2005 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. New York: ACM Press, 2011: Article No. 27.

[12] BASU A, NARLIKAR G. Fast incremental updates for pipelined forwarding engines[C]// Proceedings of INFOCOM 2003: the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. Piscataway, NJ: IEEE Press, 2003: 64-74.

[13] JIANG W, PRASANNA K. Scalable packet classification on FPGAs[J]. IEEE Transactions on Very Large Scale Integration Systems, 2012, 20(9): 1668-1680.

[14] Virtex-6 FPGA[EB/OL]. [2013-01-10]. http://www.xilinx.com/products/virtex6/.

[15] 陈兵，潘宇科，丁秋林. 一种采用启发式分割点计算的包分类算法[J]. 电子与信息学报，2009，31(7)：1594-1599.

[16] FONG J, WANG X, QI Y, *et al*. ParaSplit: a scalable architecture on FPGA for terabit packet classification[C]// Proceedings of 2012 the 20th IEEE Annual Symposium on High-Performance Interconnects. Piscataway, NJ: IEEE Press, 2012: 1-8.

[17] GUPTA P, MCKEOWN N. Algorithms for packet classification[J]. IEEE Network, 2001, 15(2): 24-32.

[18] 匡继昌. 常用不等式[M]. 3 版. 济南：山东科学技术出版社，2004：38-39.

[19] TAYLOR D, TURNER J. ClassBench: a packet classification benchmark[J]. IEEE/ACM Transactions on Networking, 2007, 15(3): 499-511.