第33卷 第14期 计 算 机 工 程 2007年7月
Vol.33 No.14 Computer Engineering July 2007
·软件技术与数据库· 文章编号：1000—3428(2007)14—0046—03 文献标识码：A 中图分类号：TP311

位并行多维数据包分类算法研究

王学光
(华东政法学院信息科学技术学院(松江大学园区)，上海 201620)

摘 要：位并行算法是一种快速的包分类算法，由于空间占用量过大，不能扩展到大规模规则库。该文从位并行算法出发，比较分析了它的两种改进算法，通过引入位图映射及元组空间的概念，提出了一种新的改进算法，在时间复杂度与空间复杂度上都较位并行算法有很大提高并具有很好的扩展性。在模拟环境下对算法进行了评测，给出了试验数据的分析结果。

关键词：数据包分类；位并行；位图映射；元组空间

Research on Bit-parallelism Based Multi-dimensional Packet Classification

WANG Xueguang
(School of Information Science and Technology, East China University of Politics and Law, Shanghai 201620)

【Abstract】 Bit-parallelism is a fast scheme for packet classification, but it scales poorly as the filter databases grow in size. By thinking of BV and ABV algorithm, taking the bit-parallelism scheme and adding the ideas of bitmap mapping and tuple space, this paper presents a new algorithm. The new algorithm reduces the complexities of both the time and storage and can scale well with the growth of the filter databases in size. It realizes the algorithm in a virtual environment and analyzes the experimental data.

【Key words】 packet classification; bit-parallelism; bitmap mapping; tuple space

对于多维搜索问题，简单的解决办法是将其分解为若干子问题，然后将各子问题的结果进行一定的相交操作，得出原问题的结果；自然的方法是将k维查询分解为k个一维查询，k个一维查询可并行进行，以减少总的查询时间。这就是基本位并行包分类算法的基本思想[1]。位并行算法最大的优点就是可以在多维上并行进行规则的查找，因此具有很快的执行速度；缺点是它只适合于小型规则库的匹配查找，不具有良好的扩展性，随着规则库的增长，位并行算法的内存消耗将急剧膨胀，在最坏情况下，位并行算法将占用$k*n*(2n+1)$的内存空间，空间复杂度为$O(n^2)$，同时它的查询时间($O(t_{RL}+n/w)$)也将以线性速度增长。

1 位并行包分类改进算法

最原始的位并行算法主要是解决多维范围匹配问题，BV(bit vector)[2]算法通过引入trie结构，使位并行算法同时适用于多维前缀匹配以及多维范围匹配，并使算法的实现变得更为直观，但是在时间与空间复杂度上，BV算法较位并行算法并没有什么改善。ABV(aggregated bit vector)[2]算法通过位图聚集的方式，改善了BV算法在平均情况下的执行时间，使其具有更好的扩展性。但在空间占用量方面ABV算法并未作出改进，为了存储额外的聚集位图，它需要的内存空间甚至比BV算法还要多，因此，在扩展性上还具有一定的局限性。位并行算法在以下方面还存在改进的必要性：

(1)算法数据结构所占内存空间是制约位并行算法扩展性的一个主要因素，因此应该考虑降低算法的空间复杂度；

(2)在保证算法空间复杂度降低的前提下，进一步降低算法的时间复杂度；

(3)位并行算法的更新速度太慢，因此，这也是一个需改进的方面。

通过观察发现，只要降低位并行算法中位图的位数，就可以同时减少算法的执行时间与占用的内存空间。因此，可以考虑对位图中的比特位进行一定形式的归并，位图归并以后，在每一维上要求仅对位图进行一次读取就可以取得匹配规则，这样才能保证空间复杂度与时间复杂度的改善。

2 位图映射及元组空间

2.1 位图映射

将原始位并行算法中的位图称为长位图，长位图中的每一比特代表的是规则库中的一条规则，假定它的位数为 n，考虑对规则数据库中的规则进行一定形式的归并，假如归并后产生 m 个集合，每个集合中包含一类具有相同属性的规则，这样就有 $m <= n$，甚至 $m << n$，在进行预处理时，重新生成新的位图，其中每一比特代表归并后的集合，而不是规则本身。将新位图称为短位图，这样就将单一位图的位数减少至 m。如图1所示。

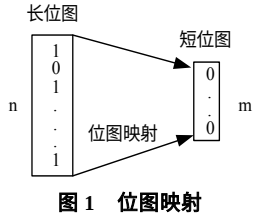

图1 位图映射

2.2 元组的概念

元组空间的概念由Srinivasan提出[3]：虽然在规则数据库

基金项目：上海市青年教师基金资助项目
作者简介：王学光(1975－)，男，博士、副教授，主研方向：计算机网络
收稿日期：2006-07-30 **E-mail：**wangxueguang@ecupl.edu.cn

中的规则数目很多，但是规则中各字段长度的组合很少。

考虑一个规则数据库 FD，其中包含 n 条规则，每条规则有 k 个字段。

元组是一个长为 k 的向量。比如[8,16,8,0,16]是一个长为 5 的元组，或称为 5 维元组，元组的维数与规则的维数是一样的，因为元组的一个分量对应于规则的一个字段。

称规则 F 属于元组 T 是指：F 第 i 个字段的前缀长度或范围长度等于元组 T 的第 i 个分量的值 $T[i]$。比如 $F_1=(01*,111*)$ 和 $F_2=(11*,010*)$ 这两条规则都属于元组[2,3]，而规则 $F_3=(0010*,11*)$ 则属于元组[4,2]。

可见，元组就是规则中各字段长度的组合，而且元组与规则是一对多的关系，一个元组对应多条规则，在实际数据库中这个比例是很大的。表 1 为两维前缀规则集，表 2 中给出了其对应的元组集。

表 1　规则集示例

Rules	Field1	Field2
F_1	00*	00*
F_2	00*	01*
F_3	0*	0*
F_4	1*	10*
F_5	1*	0*
F_6	11*	0*

表 2　元组表

Tuples	Rules
T_1 [2,2]	F_1，F_2
T_2[1,1]	F_3，F_5
T_3[1,2]	F_4
T_4[2,1]	F_6

2.3　元组优先级定义

对于一个数据包来说，它能够匹配的元组可能有多个，因此，有必要对元组的优先级进行一个定义，以保证最后查找结果的唯一性，即查找到元组为优先级最高的元组。一般来说，前缀越长或者范围越短，那么，相应的规则优先级就越高。

以 k 维元组 $T_1=[T_{1,1},T_{1,2},...,T_{1,k}]$ 和 $T_2=[T_{2,1},T_{2,2},...,T_{2,k}]$ 为例，其中，$T_{1,i}$ 和 $T_{2,i}$ 分别为 T_1 和 T_2 的第 i 个分量。令 i 从第一维开始，如果 $T_{1,i}>T_{2,i}$，那么 $priority(T_1)>priority(T_2)$，如果 $T_{1,i}<T_{2,i}$，那么 $priority(T_1)<priority(T_2)$，如果 $T_{1,i}=T_{2,i}$，那么对 $T_{1,i+1}$ 和 $T_{2,i+1}$ 重复这两步操作，直到判断出 T_1 和 T_2 的优先级为止。由于 T_1 和 T_2 的各字段中必有一个字段不相同，因此它们的优先级是确定的。

元组优先级定义的思想与最长匹配问题的思想是相符的。规则前缀越长，优先级越高，对应到元组中就是相应字段值越大，优先级越高。同样对于规则的范围字段来说，由于元组是用嵌套等级来定义范围的长度，因此范围越小，它的嵌套等级就越高，按照对元组的优先级定义方式，相应元组的优先级也越高。如果两个元组前面若干字段值相等，无法决定优先级，那么就以此类推，由后面的字段值对优先级进行确定。按照这个定义，可以得出表 2 中所示元组的优先级序列为 $T_1>T_4>T_3>T_2$。

3　多维 TSBP 算法

TSBP（tuple space based on bit-parallelism）算法分两部分完成：预处理过程和查询过程。预处理过程将规则的各字段投影到相应的坐标轴上，形成若干不相交区间，每一区间指派一个位图，用来指示哪些元组所含的规则落在这一区间中，可以将这些位图视为这一算法的中间数据结构。在查找过程中，首先在各维上通过并行搜索，查找出数据包各字段所在的区间，由此得到在每一维上的元组位图，对所有位图做“与”操作，得到数据包所属的最高优先级的元组，然后在元组对应的哈希表中通过一次内存访问取得与数据包相匹配的规则。

3.1　预处理过程

给定规则数据库 FD，假定它有 n 条 k 维规则，分别以 $F_1...F_n$ 表示，$F_i(j)$ 表示规则 F_i 的第 j 维字段的值。TSBP 算法的预处理过程如下：

根据元组空间的定义求出 n 条规则所对应的元组集合，假定共有 m 个元组，$1\le m\le n$，元组以 $T_1...T_m$ 表示。元组 T_i 所包含的规则存储在哈希表 $Hashtable(T_i)$ 中。

对每一规则 $F_i(1\le i\le n)$，将其第 $j(1\le j\le k)$ 维字段 $F_i(j)$ 投影到 j 轴上，与位并行算法相同，这步操作在 j 轴上最多形成 $2n+1$ 个不相交的区间。

为 j 轴上的每一个区间 $e(1\le e\le 2n+1)$，指派一个 m 位的位图 B_e，位图中的每一位代表一个元组，按元组的优先级对这些比特位进行排序。当且仅当存在规则 $F\in T_i$ 且 F 在 j 轴上的投影覆盖了 e，才将 B_e 中对应于 T_i 的比特位置为 1，否则置为 0。

3.2　数据包查找过程

假设数据包 $P(E_1,E_2,...,E_k)$ 到达，那么规则的查询过程如下：

(1)与位并行算法类似，首先在每一维 j 上要通过二分查找或其他查找算法求出 E_j 所在的区间，并由此得出该数据包在第 j 维上所对应的位图 B_j。

(2)对于所有的位图 $B_1...B_k$，做“与”操作，得到位图 $B=\cap_{j=1}^{k}B_j$，B 中第 1 位为 1 的比特对应的元组即为数据包 P 所属的优先级最高的元组 T。

(3)在 T 所对应的哈希表 $Hashtable(T)$ 中，通过一次内存访问得到与 P 相匹配的规则 r。

以表 1 所示的规则集为例，首先将 F_1 与 F_2 中的前缀转换为范围，然后将它们分别投影到横轴与纵轴上，从而得到图 2 所示的投影。元组的优先级按前面讲到的方法进行确定，在位图中优先级高的元组对应的比特位在前，顺序依次为 1、4、3、2。

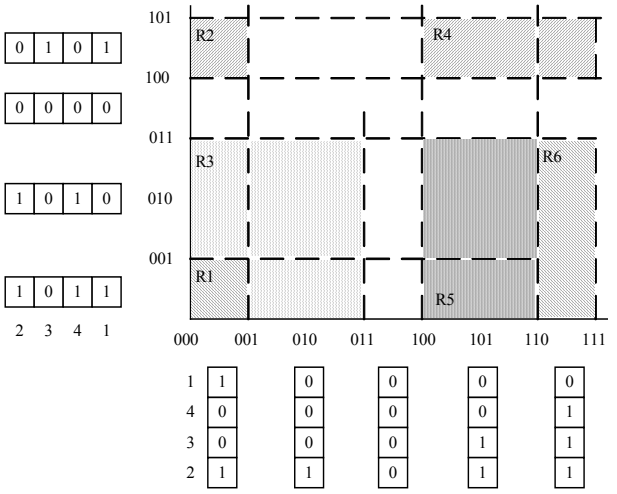

图 2　TSBP 多维算法预处理后产生的投影

3.3　算法复杂度分析

从 TSBP 算法的预处理过程来看，它与位并行算法的主要差别在于位图中各位所代表的不是规则，而是元组。假定规则共有 n 条，对应的元组个数为 m，在通常情况下 $m<<n$。这样就达到了降低位图占用空间的目的。

n 条规则在坐标轴上的投影最多将产生 $2n+1$ 个互不覆盖的区间，为每一区间指派一个 m 位的位图，这样所有位图占用的空间为 $k*(2n+1)*m$，空间复杂度为 $O(nm)$。最坏情况下（$m=n$），TSBP 算法的空间复杂度为 $O(n^2)$，与位并行算法相同；但在平均情况下 $m<<n$，TSBP 算法的空间复杂度会大大降低。

位图的位数降低了，读取一个位图所需要的内存访问量也会随之降低。读取所有的位图共需要进行 $\lceil k*m/w \rceil$ 次内存访问，其中，w 为一次内存访问读取的字长。与位并行相同，各维上对位图的读取可以并行执行，另外，为了查找到相匹配的规则，还需要从相应的哈希表中读取规则，这需要一次内存访问，可以忽略不计，因此总的时间复杂度为 $O(t_{RL}+m/w)$。在最坏情况下（$m=n$），这个时间复杂度与位并行是相同的，甚至还要多一次读取哈希表的内存访问量，但是在平均情况下，$m/w<<n/w$，因此在执行时间上，TSBP 算法较位并行算法也有极大改进。

4 算法评测

本文在一个虚拟环境下对 TSBP 算法以及位并行算法进行了评测，开发工具为 VC++6.0，CPU 为 PIII700，内存 256MB。

评测是针对二维前缀匹配进行的，规则库中的规则由程序随机生成。评测目的主要是为了对 TSBP 算法与位并行算法的查找速度与占用内存量进行定量评测与对比。

4.1 评测内容

评测主要是针对以下几个方面进行：

(1)规则库中元组数目与规则数目的比例。

(2)数据包处理速度，单位为数据包/s，即每秒钟处理的数据包的个数。

(3)算法数据结构占用内存量，单位 KB。

4.2 评测结果

分别取规则数目为 100、300、500、1 000、2 000、3 000 的 6 组规则库样本，由 TSBP 算法和基本位并行算法分别对它们进行预处理，由此得到算法数据结构占用的内存量以及元组数目与规则数目的比例。预处理结束后，向数据包处理模块模拟发送数据包，统计出算法对数据包的处理速度。

表 3 给出了算法数据结构占用内存空间及数据包处理速度的评测结果。

表 3 内存占用及数据包处理速度评测结果

前缀库规模/条	中间数据结构占用内存/KB		数据包处理速度/（$\times 10^5$数据包/s）	
	TSBP 算法	基本位并行算法	TSBP 算法	基本位并行算法
100	32	36	7.6	7.8
300	118	203	4.5	4.4
500	626	773	3.4	3.0
1 000	1 932	2 923	2.4	1.4
2 000	5 031	11 207	1.9	0.8
3 000	8 223	24 621	1.6	0.6

基于位并行技术算法的并行性主要是依靠硬件来实现的，比如数据包在每一维上对应的坐标轴区间的查找、每一区间对应的位图值的读取以及位图相“与”操作的实现，都是基于硬件并行实现的。在 TSBP 算法评测中，主要是在软件环境下通过多线程机制对算法的并行性进行模拟，因此，考虑到软硬件在执行速度上有较大差异，这里的测试结果与算法的实际硬件执行相比在数据包处理速度上必定会有所下降，算法硬件执行时的数据包处理速度会比这里给出的值要快几个数量级。从给出的 TSBP 算法与基本位并行算法对数据包处理速度的横向对比可以看出，在采用相同算法思想的硬件实现中，TSBP 算法在速度上比基本位并行算法占有优势。

4.3 评测数据分析

从表 3 中可以看出，TSBP 算法在空间占用量上比基本位并行算法要少得多。在实验结果中，随着规则库规模的增加，基本位并行算法占用的内存量急剧增加，在 3 000 条规则的环境下，基本位并行算法占用的内存量已经达到 24MB，是同等条件下 TSBP 算法的占用内存量的将近 3 倍，而且比 2 000 条规则时的基本位并行算法占用内存量多很多，内存占用量起伏比较大。与此相比较，TSBP 算法随着规则库规模的增大，内存占用量的增长要缓慢得多。

从数据包的处理速度上看，在规则较少的时候，TSBP 算法与位并行算法对数据包的处理速度是基本持平的。起先，基本位并行算法的速度可能会优于 TSBP 算法，但是随着规则的增加，由于读取位图的位数减少，TSBP 算法的在处理速度上明显比基本位并行算法要快，而且随着规则数目的增加，这种处理速度上的优势会越来越明显。

5 结束语

本文以位并行算法为出发点，比较和分析了 BV 和 ABV 两种算法，在此基础上，通过引入位图映射与元组空间的概念，提出了一种新的多维数据包分类算法，其使得位并行算法性能指标有了很大提高。在算法的空间复杂度上，算法的内存空间占用量在 $O(nm)$ 的级别上，其中，$m<<n$，m 是元组的数量，n 是规则数量。另外，在算法的时间复杂度上，算法也较位并行算法有了很大改进，即并未因算法的内存空间复杂度的降低而引起在时间复杂度的加大。可见，改进算法比位并行算法具有更好执行效能。更重要的是改进算法的扩展性很好，可用于大规模分类规则库中。

参考文献

1 Lakshman T V, Stiliadis D. High Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching[C]// Proceedings of ACM SIGCOMM′98. 1998.

2 Baboescu F, Varghese G. Scalable Packet Classification[C]// Proceedings of ACM SIGCOMM′01. 2001.

3 Srinivasan V, Suri S, Varghese G. Packet Classification Using Tuple Space Search[C]//Proceedings of ACM SIGCOMM′99. 1999.

4 Tan Mingfeng, Lu Zexin, Gao Lei. Packet Classification Algorithm Using Multiple Subspace Intersecting[C]//Proceedings of the 3rd International Conference on Networking and Mobile Computing. 2005: 1083-1093.

5 Han Xiaofei, Wang Xueguang, Yang Mingfu. A Survey on Bit-parallelism Based Packet Classification[J]. Journal of East China University of Science and Technology, 2003, 29(5).