Leaf-TCAM：一种并行IP路由查找方法及性能分析

朱国胜[1,3] 余少华[1,2,3] 戴锦友[1,2]
（华中科技大学计算机学院 武汉 430074）[1] （武汉邮电科学研究院 武汉 430074）[2]
（新一代光纤通信技术和网络国家重点实验室 武汉 430074）[3]

摘 要 分析了互联网路由表和路由更新的特征，提出了一种基于叶子节点进行路由表分区的并行IP路由查找方法Leaf-TCAM，分区子表按照流量特征在K个TCAM芯片中进行均衡分布。分析表明，该路由查找方法在引入$0.1*(K-1)$冗余的前提下具有$K-1$倍加速因子。该方法无需进行前缀扩展，90%以上的路由前缀无需排序，可以采用随机更新；同时还具有分区均匀、分区溢出代价小等特点，而功耗只有传统单片方案的12%。

关键词 路由查找，并行，Leaf-TCAM
中图法分类号 TP393.11 **文献标识码** A

Leaf-TCAM: A Parallel IP Address Lookup Method and Performance Analysis

ZHU Guo-sheng[1,3] YU Shao-hua[1,2,3] DAI Jin-you[1,2]
(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)[1]
(Wuhan Research Institution of Posts and Telecoms, Wuhan 430074, China)[2]
(State Key Laboratory for New Optical Communication Technologies and Networks, Wuhan 430074, China)[3]

Abstract Features of global routing table and prefix updates were analysized. A parallel IP address lookup scheme based on leaf nodes called Leaf-TCAM was proposed. The global routing table is partitioned into sub-tables and put into K independent TCAM chips. Our scheme can get speedup factor $K-1$ with $0.1*(K-1)$ redundant. Prefix expansion is not needed and over 90% prefix updates can be done in random mode. Our scheme can partition the routing table evenly and has low cost when the sub-table is overflow. The power cunsumption is only 12% of traditional single chip scheme.

Keywords IP address lookup, Parallel, Leaf-TCAM

IP路由查找面临巨大挑战，主要表现在：1）算法设计困难，路由查找需要在前缀值和长度二个维度进行查找以实现最长前缀匹配（Longest Prefix Matching：LPM）[1]；2）路由表庞大，目前已经超过26万条，每年增长约5万条[2]；3）路由更新频繁，最高每秒更新数千条[2]；4）接口速率越来越高，OC-768（40Gbps）已经商用，100Gbps以太网标准将在2010年发布。基于软件的路由查找方法如Unibit-Trie[1]，Multibit-Trie[1]，Tree-Bitmap[3]等需要多次内存操作才能完成路由查找，无法满足高速接口线速转发要求。基于三态内容寻址存储器TCAM（Tenary Cotent Addressable Memory）的路由查找可以在一个时钟周期内完成查找关键字（IP地址）和所有前缀表项的匹配，并且从多个匹配的结果中选择最长匹配前缀（存放地址最低的前缀）输出。目前TCAM时钟周期可以做到4ns[4]，但是TCAM路由查找存在功耗高，前缀排序存放导致更新困难等问题。

虽然晶体管集成度（容量）按照Moore定律递增，但是支撑查找的存储器存取时间每年只减少（速率增加）约7%[5]，因此存取技术的发展无法赶上高速接口线速转发的要求。基于TCAM的并行IP路由查找将路由表分配到K个并行的TCAM芯片中，不单纯采用减少每次查找所花的时间来解决路由查找瓶颈，而是通过并行的体系结构来突破存储器访问速度的限制，一方面单次查找无需和所有的表项作匹配便可大大降低每次查找的平均功耗，另一方面可以获得远高于以往的算法吞吐能力。基于TCAM的并行IP路由查找算法的关键在于路由表的分区方法、并行路由查找的负载均衡以及是否支持路由前缀的增量更新。

本文分析了路由表前缀分布特征，发现90%的路由前缀为无孩子叶子节点，只有10%的路由前缀为内部节点，所有叶子节点都不重叠，同时路由更新更多地体现在长前缀的增加或者删除，短的内部节点前缀相对稳定。提出了一种基于叶子节点进行路由表分区的并行IP路由查找方法Leaf-TCAM，分区子表按照流量特征在K个TCAM芯片中按照

到稿日期：2009-06-26 返修日期：2009-08-31 本文受中国下一代互联网示范工程项目（CNGI-04-3-1D）和新一代光纤通信技术和网络国家重点实验资助。

朱国胜（1972－），男，博士生，高级工程师，CCF会员，主要研究方向为下一代互联网体系结构和高速网络系统设计，E-mail：zhugswh@163.com；余少华（1962－），男，博士，教授，博士生导师，CCF会员，主要研究方向为光因特网和下一代网络体系结构；戴锦友（1969－），男，博士，主要研究方向为分组城域网。

贪心方法进行均衡分布放置，每片 TCAM 同时保存 10%的内部节点。该路由查找方法在引入 0.1 * (K−1)冗余的前提下具有(K−1)倍加速因子。该方法无需进行前缀扩展，90%以上的路由更新采用随机更新，避免 TCAM 路由表项排序导致的更新困难；同时还具有分区均匀、分区溢出代价小等特点。采用 4 片 4ns 商用 TCAM 芯片，整个系统的包转发率为 300MPPS，可以满足 2 端口 100Gbps 以太网单板线卡最小包长线速转发要求。而功耗只有传统单片方案的 12%。

1 相关工作

TCAM 每个存储单元可以保存“0”、“1”和“*”(任意)三种状态从而支持最长前缀匹配。路由前缀需要按照前缀长度递减的次序依次存放(低地址区存放长前缀)，路由查找时输入关键字(待查 IP 地址)和所有表项进行并行匹配，存在多重匹配时返回最低地址位置的匹配表项，在一个 TCAM 时钟周期内可以完成一次路由查找，相比传统基于软件的路由查找需要多次内存操作来说具有相当的性能优势。但是如前所述，TCAM 的缺点也是非常明显的，在功耗方面，SRAM 每个比特的存放只需要 6 个晶体管，而 TCAM 每个比特的存放需要额外的 6 个晶体管存放掩码，另外还需要 4 个晶体管实现匹配逻辑，一个 18Mb 的 TCAM 芯片功耗为 15 瓦，存取每比特功耗约为 SRAM 的 150 倍[6]；在路由更新方面，为保证查找结果正确，前缀更新后必须保证按照前缀长度递减存放，因此插入长的前缀可能会导致短前缀的移动，最坏情况下需要移动 W 次，其中 W 为地址长度，对于 IPv4 来说 W 为 32，对于 IPv6 来说 W 为 128；另外，TCAM 在进行更新操作时查找操作被禁止，到达的 IP 分组会被缓存，缓存溢出会导致丢包。

针对 TCAM 增量更新问题，文献[7]提出了 PLO_OPT 和 CAO_OPT 算法，其中 PLO_OPT 在 TCAM 的中间位置预留空闲自由区，这样最多只需要移动 $W/2$ 次 TCAM 表项就可以实现一次更新操作。进一步分析发现排序约束只在相互重叠的前缀之间才是必要的，CAO_OPT 算法最多只需移动 $D/2$ 次 TCAM 表项就实现一次更新，D 为相互重叠的前缀的最大个数，目前 D 不超过 8。另外，从降低功耗的角度文献[8]提出了通过修剪(Pruning)和掩码扩展(Mask Expansion)来压缩路由表冗余信息，减少存放在 TCAM 中的表项个数。现有商用 TCAM 提供分区禁用(Partition Disable)机制，比如一个 512k 大小 36 比特宽度的 TCAM 芯片被分成 64 个块(Block)，每个 Block 的大小为 8k，TCAM 查询不对整个芯片进行而是根据查找关键字的特征选择对某些 Block 进行查询，这样可以大大降低每次查询的平均功耗。整个查找涉及特征匹配和相应的 TCAM Block 查询两个阶段，算法关键在于第一阶段的特征匹配也就是路由表的分区方法。文献[9]提出了 3 种路由表分区方法，分别为比特选择(Bit-Selection)、子树分割(Subtree-Split)和后序分割(Postorder-Split)。Bit-Selection 在第一阶段的特征判定采用简单的逻辑电路实现，Subtree-Split 和 Postorder-Split 的第一阶段需要采用被称为索引 TCAM 的 TCAM 芯片，Subtree-Split 和 Postorder-Split 算法需要辅助的 Unibit-Trie 结构。

如前所述，为克服存储器访问速度的限制，可以采用并行的体系结构，一方面查找被分发到 K 个并行 TCAM 芯片中，在提高系统整体查找性能的同时也减少了每次查找需要匹配的表项的数目以降低功耗；另一方面对于每个单芯片的查找依然可以利用前述的 Patrition Disable 机制来进一步降低功耗。通用的并行 TCAM 体系结构如图 1 所示。

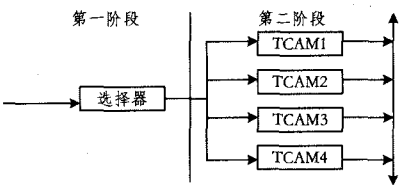

图 1 通用的并行 TCAM 体系结构

基于 TCAM 的并行 IP 路由查找算法的评价体现在如下 3 个方面：

1)分区方法：分区是否均匀，是否需要冗余存放，分区方法同时决定了第一阶段的选择器实现；

2)负载均衡：是否支持将到达的路由查找请求在 K 个芯片间进行均衡的分发；

3)查找性能：并行查找取得的加速因子及平均每次查找的能耗降低比例；

4)路由更新：路由表是否需要进行前缀扩展，是否支持增量更新，更新时分区溢出代价。

文献[10]采用 Bit-Selection 算法选择 10～13 比特将路由表分成 16 个 group，采用分区负载统计和冗余存放进行负载均衡。该算法无法保证分区的均匀，需要对 8 比特、9 比特前缀进行前缀扩展(Pefix Expansion)，不支持增量更新，特别是在分区溢出时需要重新选择比特位进行路由表重构。

文献[11,12]采用基于 IP 地址范围(Range-based)的分区算法来判断前缀 $P*$ 是否只属于某个范围$[a,b]$，但需要检查如下的条件是否成立：

$$(a<P0\cdots0)\ \text{AND}\ (P1\cdots1<b) \tag{1}$$

某些前缀可能属于多个范围，比如前缀 * 属于所有的范围，则需要在多个地方冗余存放。文献[11]提出用 Preorder Spliting 来确定分区点，采用 Unibit-Trie 辅助结构，从分区分界点到根路径上的所有前缀都需要在相邻的分区内冗余存放，理论上需要冗余存放 32 * (K−1)个前缀，Range-based 分区算法无需进行前缀扩展，可以进行均匀分区。文献[11]和文献[12]都采用文献[13]中所述的基于 TCAM 的前缀缓存进行负载均衡和提高查找性能。不同的是文献[11]的前缀缓存被分布在各并行 TCAM 芯片的某个 Block 里，无需额外的 TCAM 芯片，但是会导致报文乱序；而文献[12]需要 K 个额外的位于第一阶段的小的 TCAM 芯片，其实现复杂且成本较高。同时基于 TCAM 的前缀缓存为保证缓存正确性，内部节点不能进行缓存，存在不公平性问题。更新方面，Range-based 的分区算法支持增量更新，在分区溢出的情况下一个前缀的更新需要移动的最大的前缀个数为 K(同时需要保证 TCAM 的前缀长度排序)，这种情况发生在其它分区也溢出的情形下。

2 Leaf-TCAM 基本思想

2.1 路由表前缀分布特征

CIDR 的路由聚合虽然减缓了骨干网路由表的增长速率，但是随着连接的主机和网络的数量急剧增长，运营商独立地址(Provider Independent Addresses)和多穴(Multihoming)

站点的增多，骨干网路由表依然呈现高速增长态势，每年约增长5万条路由条目，目前已经超过26万条，为了进行路由表的有效分区，提取了文献[14]中APNIC，Oregon-IX和Route View2008年4月30日的路由表数据进行分析（所有骨干网路由表都具有类似的特性），如表1所列。

表1　3个骨干路由表统计（2008年4月30日）

位置	路由条目	叶子节点	非叶子节点	叶子节点比例
APNIC	254626	232210	22416	91.19%
Oregon-IX	261762	238097	23665	90.95%
Route View	261930	238269	23661	90.96%

从表1可以看出，路由表90%以上的节点为叶子节点，如果将10%的内部节点在K片TCAM芯片进行冗余存放，路由表分区只考虑叶子节点，则具有如下优势：

1）分区方法方面：可以采用简单的树遍历算法进行分区，分区简单且均匀，叶子节点之间不存在相互重叠和包含关系，无需冗余存放叶子节点，在TCAM Block中的放置也没有排序要求；

2）负载均衡方面：可以根据分区的负载大小在K片TCAM芯片中进行均衡放置；

3）查找性能方面：与和其他并行查找方案类似，可以获得的加速因子为$K-1$；具体见第4节的性能分析；

4）路由更新方面：无需进行前缀扩展，支持增量更新，通常情况下分区溢出时只需要进行一次表项的移动操作。

2.2　路由更新特征

链路状态和配置的改变都会促发路由更新消息，无状态BGP实现、配置错误和路由更新自同步特性[15]会触发更多的路由更新消息。路由更新消息会引起路由转发表的重新计算，导致TCAM表项的添加或者删除，与每秒数百万次路由查找相比，前缀更新频率要小得多，均值为每秒2.35次，但是峰值达每秒5千次[2]。由于TCAM表项的更新将导致路由查找被缓存，缓存会增加报文处理时延，缓存溢出还会导致丢包。

直观分析表明，覆盖大范围网络的短前缀（内部节点）会相对稳定，覆盖小范围网络的长前缀（叶子节点）更新会相对频繁。文献[16]指出路由更新主要由非知名前缀构成，4.5%的非知名前缀占据了50%的路由更新消息，而其流量仅占1.4%；对文献[2]的50个更新最频繁的前缀进行了分析，发现只有2个前缀为内部节点，96%的更新为叶子节点。采用Leaf-TCAM体系结构，90%的叶子节点的存放无需排序，超过90%的更新可以直接采用随机更新，对于小于10%内部节点的更新依然可以采用PLO_OPT或者CAO_OPT算法。

2.3　Leaf-TCAM体系结构

Leaf-TCAM并行路由查找也是采用二阶段TCAM体系结构，图2所示为4片Leaf-TCAM路由查找体系结构。内部节点被放置在TCAM高地址区（阴影部分），同时需要保持前缀长度降序排列，内部节点在每片TCAM中都进行存放。叶子节点被进行Range-based分区后根据其各自负载大小放置在各TCAM芯片的Block中，叶子节点在TCAM中的放置无需排序，唯一需要保证的是存放叶子节点的TCAM地址要小于存放内部节点的TCAM地址。路由查找时待查关键字IP地址首先在第一阶段进行Range匹配，得到相应的TCAM芯片号和该芯片内的Block号，查找请求被送往第二阶段相应的缓存进行排队，在每片TCAM芯片里面，存放内部节点的Block和第一阶段Range匹配得到的Block被使能，可以很容易证明，经过二阶段路由查找，正确的路由查找结果将返回。

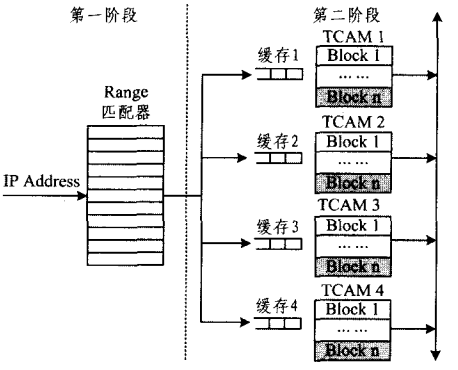

图2　Leaf-TCAM体系结构

3　Leaf-TCAM具体算法

3.1　路由表分区算法

假设N为路由表前缀个数，ϕ为叶子节点比例系数，K为并行TCAM芯片个数，每个TCAM的Block大小为b，每个TCAM可被分成n个Block，$S[j]$为前缀j的负载比例系数，要求满足的先决条件为：

$$(K*b*n)>N+(K-1)*(1-\phi)*N \tag{2}$$

条件(2)可以确保K个并行的TCAM芯片可以容纳N个路由前缀。分区算法可以采用简单的先序遍历算法，叶子节点被依次放入分区$D[i]$，$i\in[1,Ceiling(\phi*N/b)]$，内部节点被依次放入分区$D[i]$，$i\in[Ceiling(\phi*N/b)+1,Ceiling(\rho*N/b)+Ceiling((1-\rho)*N/b)]$，前缀$j$被放入相应的分区$i$时，该分区$i$对应的负载$L[i]$也被更新，分区结束后$L[i]$反映了放置在分区$i$中所有前缀的负载比例之和，该参数被下面的分区放置算法所使用，叶子节点分区的边界点同样被记录。

3.2　分区放置算法

为避免某块TCAM芯片对应的缓存队列满而其它缓存队列空闲的情况出现，需要将分区算法得到的各分区$D[i]$根据其负载在K个TCAM芯片进行均衡存放。图3所示为支持负载均衡的分区放置算法Loadbalance_Placing。

```
Loadbalance_Placing(D,L,K)
{
  对于叶子节点分区，根据L[i]大小对D[1..m]进行从大到小排序得到D'[1..m];
  L'[1..K]=0;//记录各芯片的负载大小
  for(i=1;i<=m;i++){
    if(D'[i]为叶子节点分区){
      D'[i]放入L'[1..K]最小的TCAM芯片j的空闲Block;
      L'[j]+=L[i];
    }
  }
  所有内部节点分区D'[i]放置于所有TCAM芯片的高地址区;
}
```

图3　支持负载均衡的分区放置算法

该算法首先对存放叶子节点分区根据负载大小进行降序

排列，然后按照贪心方法将排序后分区表 $D'[i]$ 中的叶子分区依次放入负载最小的 TCAM 芯片，内部节点分区在所有 TCAM 芯片中同时存放。

3.3 路由查找和更新算法

如 2.3 节所述，Leaf-TCAM 路由查找算法非常简单，第一阶段的范围匹配得到 TCAM 芯片号和 Block 号，TCAM 芯片号可以作为硬件片选信号，而 Block 号和已知的内部节点 Block 号可以用于 TCAM Block 使能的掩码计算。

路由更新需要区分内部节点和叶子节点，对于内部节点可以采用文献[7]中提出的 PLO_OPT 或者 CAO_OPT 算法，对于叶子节点则直接采用随机插入和删除算法。

4 性能分析和比较

我们采用 K 个服务时间固定的 $M/D/1$ 排队系统模型来分析 Leaf-TCAM 的性能。假定分组到达服从泊松分布，单位时间业务强度为 λ，TCAM 查找服务速率为 μ，查找服务时间 $T_s=1/\mu$，单片 TCAM 查找单元单位时间内的查找量为：

$$\rho=\lambda/(\mu*K) \tag{3}$$

每个缓存里面排队等候的分组个数为：

$$l_q=\rho^2/(2*(1-\rho)) \tag{4}$$

分组在系统内的平均逗留时间为：

$$W_s=\rho*T_s/(2*(1-\rho))+T_s \tag{5}$$

按照文献[15]的要求，ρ 要小于 1，取 $\lambda/\mu=K-1$，$\rho=(K-1)/K$，K 等于 4 时：

$$\rho=3/4,T=K*\rho=3,l_q=9/8,W_s=5*T_s/2 \tag{6}$$

从式(6)可以看出，系统获得的加速因子 T 为 3，缓存队列长度 l_q 很短，需要增加少量等待时延。以 2 端口 100Gbps 以太网单板线卡为例，最小以太网长包长度为 672bits(包括前导码、帧间隔)，要求包转发率为 298MPPS，采用 4 片 4ns 商用 TCAM 芯片，$W_s=5*T_s/2=10$ns，整个系统的包转发率为 300MPPS，可以满足 2 端口 100Gbps 以太网接口最小包长线速转发要求。

功耗方面，每次 TCAM 的路由查找功耗和需要匹配的 TCAM 条目成正比，假设路由总条目 $N=260$k，Leaf-TCAM 的 Block 大小为 8k，每次只需要匹配一个 Block 的叶子节点，则每次查找 Leaf-TCAM 的功耗和单片 TCAM 功耗之比为：

$$P_e=(0.1*260k+8k)/260k=0.12 \tag{7}$$

每次查找的平均功耗约为单片方案的 0.12，大大降低了路由查找的功耗。

表 2 对 Leaf-TCAM 并行路由查找和文献[10-12]中的并行路由查找算法进行了比较。

表 2 基于 TCAM 的并行 IP 路由查找算法比较($K=4$)

方法	分区方法	冗余	增量更新	负载均衡
文献[10]	不均匀	25%	不支持	支持
文献[11]	均匀	Cache	部分支持	支持(乱序)
文献[12]	均匀	Cache	部分支持	支持(附加硬件)
Leaf-TCAM	均匀	30%	支持(随机)	支持

从表 2 可以看出，文献[10]采用 Bit-Selection 算法进行分区，无法保证均匀分区，需要进行前缀扩展，不支持增量更新。文献[11,12]均采用基于 IP 地址范围(Range-based)的分区算法，可以保证均匀分区和部分增量更新。文献[11]基于 Prefix Cache 的负载均衡会导致报文乱序。文献[12]需要额外的 TCAM 硬件来协助实现负载均衡。Leaf-TCAM 也是采用基于 IP 地址范围(Range-based)的分区算法，可以保证分区均匀，支持增量更新和负载均衡，付出的代价是 30%($K=4$)的内部节点冗余存放。

结束语 本文分析了路由表前缀分布特征和前缀更新特征，提出了一种基于叶子节点进行路由表分区的并行 TCAM 路由查找方法，分区子表按照流量特征在 K 个 TCAM 芯片中进行均衡分布。分析表明该路由查找方法在引入 $0.1*(K-1)$ 冗余的前提下，具有 $K-1$ 倍加速因子。该方法无需进行前缀扩展，90%以上的路由无需排序，可以采用随机更新；同时还具有分区均匀、分区溢出代价小等特点，采用 4 片 4ns 商用 TCAM 芯片，整个系统的包转发率为 300MPPS，可以满足 2 端口 100Gbps 以太网单板线卡最小包长线速转发要求。而功耗只有传统单片方案的 12%。

参 考 文 献

[1] Ruiz-Sanchez M A, Biersack E W, Dabbous W. Survey and taxonomy of ip address lookup algorithms[J]. IEEE Network, 2001, 15(2): 8-23

[2] Huston G. BGP Reports[OL]. http://bgp.potaroo.net/index-bgp.html

[3] Eatherton W, Varghese G, Dittia Z. Tree Bitmap: Hardware/Software Ip Lookups with Incremental Updates[J]. ACM SIGCOMM Computer Communication Review, 2004, 34(2)

[4] IDT[OL]. http://www.idt.com/products/

[5] Hennessy J L, Patterson D A. Computer Architecture: A Quantitative Approach(3rd edition)[M]. Beijing China: The China Machine Press, 2000: 390-391

[6] Micron Technology Inc. Harmony TCAM 1 Mb and 2Mb. Datasheet, January 2003

[7] Shah D, Gupta P. Fast incremental updates on ternary-CAMs for routing lookups and packet classification[C]// Proc. Hot Interconnects 8. Aug. 2000: 145-153

[8] Liu H. Routing Table Compaction in Ternary CAM[J]. IEEE Micro, 2002, 22(1): 58-64

[9] Narlikar F Z G, Basu A. CoolCAMs: Power-Efficient TCAMs for Forwarding Engines[C]// IEEE INFOCOM. April 2003

[10] Zheng Kai, Hu Chengchen, Liu Hongbin, et al. An ultra-high throughput and power efficient TCAM-based IP lookup engine [C]// INFOCOM2004. Hong Kong, China, 2004

[11] Lin D, Zhang Y, Hu C, et al. Route table partitioning and load balancing for parallel searching with TCAMs[C]// Proc. IPDPS'07. 2007: 1-10

[12] Akhbarizadeh M, Nourani M, Panigrahy R, et al. A TCAM-Based Parallel Architecture for High-Speed Packet Forwarding [J]. IEEE Trans. Computers, 2007, 56(1): 58-72

[13] Akhbarizadeh M J, Nourani M. Efficient Prefix Cache for Network Processors[C]// High Performance Interconnects 2004. Stanford University, USA, Aug. 2004

[14] Smith P. BGP Routing Table Analysis[OL]. http://thyme.apnic.net/

[15] Labovitz C, Malan R, Jahanian F. Internet routing stability[J]. IEEE/ACM Trans. Networking, October 1998: 515-528

[16] Rexford J, Wang J, Xiao Z, et al. Bgp routing stability of popular destinations[C]// Proc. Internet Measurement Workshop. November 2002