一种基于 Bloom-filter 表项压缩的 TCAM 业务识别算法

陈正虎* 兰巨龙 黄万伟 李玉峰
(国家数字交换系统工程技术研究中心 郑州 450002)

摘 要：在三态内容寻址存储器(Ternary Content Addressable Memory, TCAM)表项宽度和存储容量约束下，该文提出一种基于匹配表项压缩的 BF-TCAM 算法，采用 Bloom-Filter(BF)对匹配关键字进行单字节编码压缩关键字长度，解决了匹配吞吐率低和存储空间不足问题。针对 BF 在表项压缩过程带来的冲突率上升问题，引入向量存储空间策略，利用向量存储空间实现多个哈希函数映射，相对于比特向量策略，有利于降低匹配冲突率。测试实验表明，相对于传统的 TCAM 匹配算法，BF-TCAM 算法不但提高了匹配吞吐率和存储空间利用率，同时可有效降低 BF 压缩产生的冲突率。

关键词：三态内容寻址存储器(TCAM)；Bloom 滤波器(BF)；模式匹配
中图分类号：TP393 **文献标识码**：A **文章编号**：1009-5896(2011)09-2212-07
DOI: 10.3724/SP.J.1146.2011.00058

A TCAM Service Identification Algorithm Based on Access Compression Using Bloom-filter

Chen Zheng-hu Lan Ju-long Huang Wan-wei Li Yu-feng
(*National Digital Switching System Engineering & Technological R & D Center*, *Zhengzhou* 450002, *China*)

Abstract: Within the confines of access width and storage capacity in Ternary Content Addressable Memory (TCAM) chip, pattern matching algorithm using TCAM has low throughput and limited memory. This paper proposes BF-TCAM algorithm based on access compression by single byte coding using Bloom-Filter (BF) algorithm, increases the throughput and improves the utilization efficiency of storage space. BF algorithm will bring collision, which can be reduced by using vector memory instead of bit vector in BF-TCAM algorithm. The experimentation shows that the BF-TCAM algorithm proposed in this paper can improve the matching throughput and storage space utilization, and at the same time, effectively reduce the conflict ratio resulting from BF compression.

Key words: Ternary Content Addressable Memory (TCAM); Bloom-Filter (BF); Pattern matching

1 引言

互联网上新型的业务层出不穷，P2P 和流媒体类业务泛滥成灾，DDOS 等异常流量攻击日益繁多，以及不健康的虚假、恶意信息发布日益猖獗，对互联网的公共和个人安全造成了威胁。为净化网络环境，需要对互联网上承载的不同业务进行准确地识别并有区分地处理。当前骨干网络的链路带宽已达到了 40 Gbps，而现有的业务识别系统的吞吐率在 10 Gbps 左右，远远不能满足需要。因此迫切的需要一种高速的业务识别机制，对数据包的五元组以及负载作深度的报文检测[1]，满足链路的需求。

对于模式匹配算法，按照实现途径有基于软件和基于硬件两种实现方式。传统的基于软件的算法有 Knuth-Morris-Pratt(KMP) 算法[2]，Boyer-Moore(BM)算法[3]，Aho-Corasick(AC) [4]算法以及 Commentz-Walter(CW) 算法[5]，改进型的 Aho-Corasick-Boyer-Moore(ACBM) 算法[6]和 setwise Boyer-Moore-Horspool(BMH)算法[7]在空间和时间损耗上要优于传统的算法，但基于软件的算法由于依靠处理器来完成，是一种时分复用的串行结构，随着模式集的增加，此类算法的空间复杂度也随之增加。目前基于软件实现的算法吞吐率一般小于 1 Gbps[8]，难以满足 40 Gbps 网络线速的需求。

目前高速的模式匹配算法主要分为基于硬件实现的 Bloom-Filter(BF)算法和基于三态内容寻址存储器 (Ternary Content Addressable Memory, TCAM)的相关算法[9]。Bloom[10]在 70 年代提出了一种高效的哈希结构 BF，突破了传统的哈希函数映射

2011-01-18 收到，2011-05-03 改回
国家 863 计划重大专项，高可信业务管控系统基金(2009AA01A346)资助课题
*通信作者：陈正虎 chenzhenghu@sina.cn

和元素存储的方式，用以表达庞大的数据集提高查找效率，缩短查找时间，节约存储资源，但该算法存在一定的冲突概率。文献[11]改进了 BF 算法，提出了并行的 BF 的硬件实现算法，大大提高了查找速率。利用 TCAM 实现的高速模式匹配算法，主要是围绕 TCAM 芯片的宽输入和高速并行查找的特点，对其查表方式做进一步的优化。TCAM 的缺点是容量小，在处理范围匹配和多模式匹配时，如五元组中源、目的 IP 和源、目的端口均须做范围匹配时，需将一条策略拆分成多条表项来匹配，这样便需要通过一种高效的表项管理机制缓解存储空间的压力。代表算法有 DRES(Dynamic Range Encoding Scheme) 算法[12]和 RM-TCAM(Range- Matching TCAM)算法[13]。DRES 算法利用额外的 RAM 进行复杂的范围编码，从而节约存储空间。RM-TCAM 算法通过修改 TCAM 内部构造，来实现一个高效的范围匹配单元(Range-Matching Cell, RMC)，从而在不用外在存储器的情况下大大节约了存储空间。但当模式集中存储的模式过长时，不仅需要更长的匹配周期，还要占据更多的存储空间，从而影响 TCAM 算法的性能。

针对 TCAM 芯片在处理长模式时吞吐率和存储空间受限的问题，本文提出了一种将 BF 算法和 TCAM 查找方式相结合的算法 —— BF-TCAM 算法，该算法在查询的每个字节设置一个 BF 引擎，通过对每个字节中可能出现的模式进行预编码以及对于查找字符串的寻码，实现了当模式集的长度、位置、内容不固定时表项压缩，从而减少了 TCAM 查找关键字的处理时间，提高了系统的吞吐率。实际测试表明，在关键字最长为 50 byte，每个字节的编码长度为 4 bit 时，该算法以冲突概率为 3.64×10^{-16} 的代价将单片 TCAM 的吞吐率提高到 28.8 Gbps，应用两片并行的 TCAM 可满足 40 Gbps 线速的需求。

2　BF 算法和 TCAM 工作方式

2.1 BF 算法

BF是一种把模式集经多个哈希函数映射并存储其结果的数据结构，通过这种结构把模式集转换为一种特殊的字符串。该算法分为编程(programm)和查询(query)两个阶段。工作原理如下：在编程阶段，首先选取哈希函数集 $H=\{h_1,h_2,\cdots,h_n\}$ 并定义长为 m bit的向量，哈希函数的结果指向 m bit向量中的任意一位。然后将模式集中的每个模式经 k 个哈希函数得到 k 个值，并将比特向量的相应位置置为1；若已为1，则不再处理，如图1(a)所示。在查询阶段，算法将待检测的字符串经过同一哈希函数集 H 得到 k 个结果，若对应的 k 个位置全为 1，则表示该模式串以一定的概率属于模式集；否则，若有一位为 0 则表示肯定不属于模式集。因此，BF算法会产生假阳性而不会产生假阴性。如图1(b)中的关键字经哈希函数H_1得到的向量的对应位置为0，表示此关键字不属于规则集。

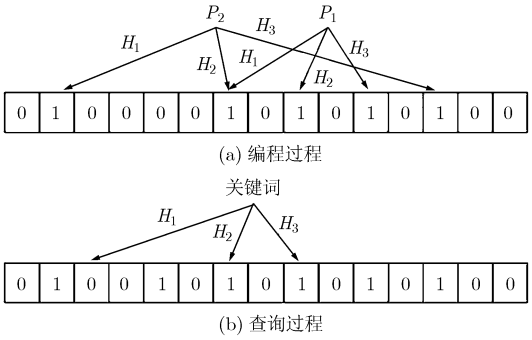

图1 BF原理

在应用BF引擎实现关键字的查询时，首先为每种长度设置一个BF，多个BF构成一个引擎，查询时每种长度的BF并行工作，实现了变长模式集的匹配。如图2(a)所示，BF引擎分别为2 byte到w byte设置一个BF，每个BF存储本字节长度的模式，并将命中结果统一输出给分析模块。在分析模块中，将命中的关键字与存储的模式进行比较，从而消除假阳性的存在，实现模式集的精确匹配。为提高算法的吞吐率，一般采用多个BF引擎并行工作的模式，具体实现如图2(b)所示。由于各个BF引擎独立的工作，因此图示的算法在4个引擎并行化实现时可以将一次查询的滑动窗口由1 byte提高到4 byte，从而提高算法的吞吐率。但在硬件实现时受限于芯片的缓存资源，文献[11]采用4个并行的BF引擎只实现了2.46 Gbps的吞吐率。

2.2 TCAM 的工作原理

TCAM 是一款并行查找芯片，它是一种基于内容查询的存储器，因其具有查询速度快、匹配时间固定等优点，在报文分类和 IP 查找中具有广泛的应用。TCAM 的表项有 0, 1 和 X(无关项)3 种逻辑态。查表输入的关键字与每条表项的掩码组成序偶来进行匹配，查表命中后返回优先级最高的命中表项所对应的地址。

TCAM 可以实现高速的并行查找，但它昂贵的存储空间和复杂的表项管理机制限制了此芯片的进一步应用。当查询的关键字较长时，如业务识别时需对一组报文的五元组和部分负载进行查找，一次

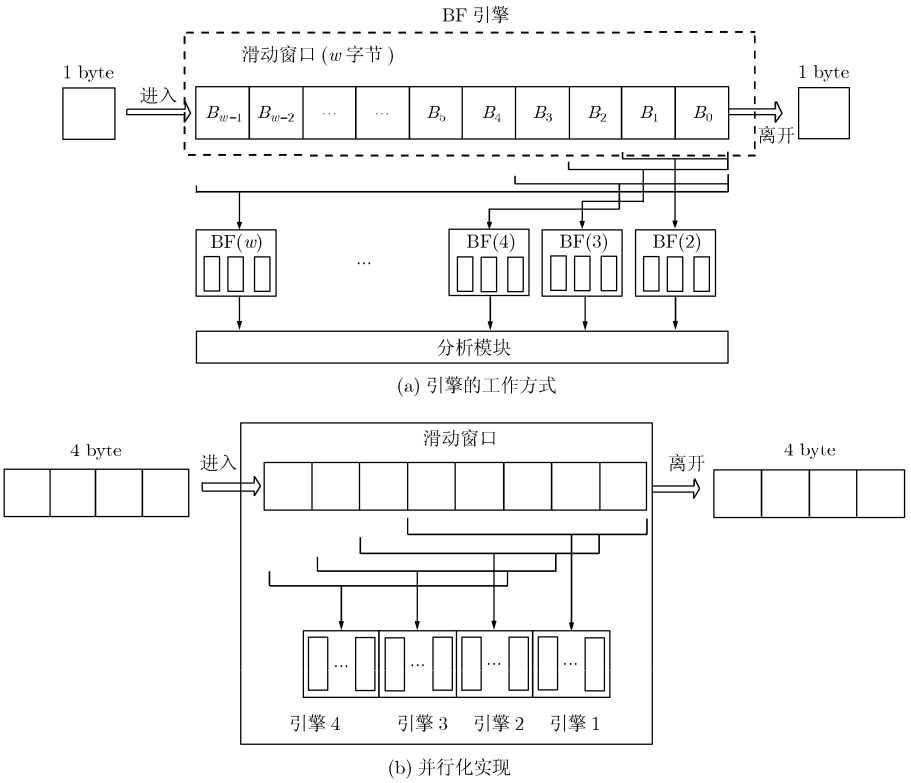

图 2 BF 引擎工作方式及并行化实现

查询需要占用多个 TCAM 的匹配周期，而 TCAM 稀少的存储空间往往不能支持过多的模式集。受 TCAM 器件本身的制约，在应用 TCAM 进行查找时的最大吞吐率等于工作频率和输入宽度的乘积，无法进一步增加。

3 BF-TCAM 算法的流程及性能分析

3.1 BF-TCAM 算法的流程

针对于 TCAM 芯片在处理长模式时吞吐率和存储空间受限的问题，本文将 BF 算法和 TCAM 查找方式相结合，提出了 BF-TCAM 算法。该算法在查询的每个字节设置一个 BF 引擎，通过对每个字节中可能出现的模式进行预编码以及对于查找字符串的寻码，实现了当模式集的长度、位置、内容不固定时表项压缩，从而可以快速、高效地处理关键字，实现吞吐率的提高。如图 3 所示，算法为每个字节提供一个 BF 引擎，查找时每个引擎并行化工作，并将每个字节的处理结果一起输入到 TCAM 中进行查找。在每个 BF 引擎中，应用向量存储空间代替比特向量，在每个地址上存储模式集的编码。

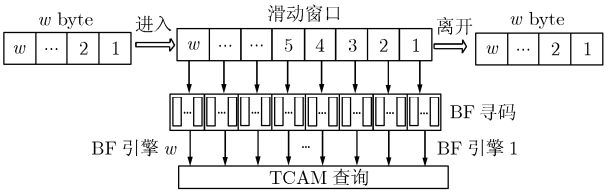

图 3 BF-TCAM 算法的实现方式

算法分为预编码和查找两个阶段。在预编码阶段，算法首先对每个字节上可能出现的模式进行编码，并将编码结果存储在哈希函数集的映射结果所对应的向量存储空间上，值得一提的是，为提高算法的通用性，对于不参加匹配的无关项 X 也进行编码。在查找阶段，用相同的哈希函数集来处理关键字，并读取映射地址上存储的编码，此过程称为寻码。然后将每个 BF 引擎的处理结果一起输入到 TCAM 中进行查找。若关键字的某一字节经哈希函数映射的地址上未存储模式集的编码，则送出的结果为全 0，因此，在编码阶段需将全 0 的编码预留。

预编码阶段：

步骤 1 将向量存储空间初始化；

步骤 2 对每个字节上可能出现的模式独立地进行编码。算法在编码时为无关项预留一个编码，方便查找时对于 TCAM 掩码的设置，同时将全 0 的编码保留，用来处理非模式集的关键字；

步骤 3 将单字节的编码结果存储在本字节对应的向量存储空间中；

步骤 4 将每个模式的编码结果存储在 TCAM 中。

查找阶段：

步骤 1　将待查找的关键字以字节为单位进行划分，并将其送入单字节的 BF 引擎中；

步骤 2　单字节的字符串经相同的哈希函数集映射后，读取其对应的地址上存储的编码，非模式集的字符串，读到的编码结果为 0；

步骤 3　将每个字节的编码结果一起送入 TCAM 中进行查找；

步骤 4　根据 TCAM 输出的命中地址，在 SRAM(Static Random Access Memory)中读出对于该数据包进行操作的策略。

算法的流程图如图 4 所示。

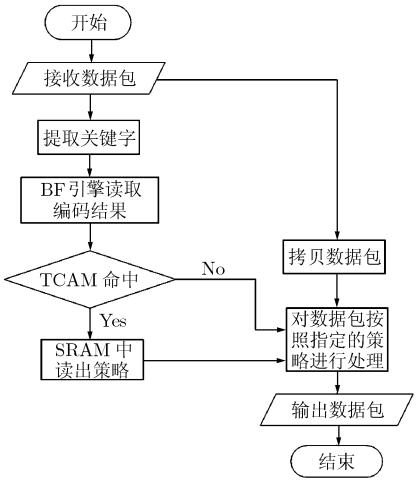

图 4 BF-TCAM 算法的流程图

该算法与传统的 BF 算法不同之处在于算法中的 BF 引擎只针对单字节进行处理，同时引擎中用向量存储空间取代比特向量并存储模式集的编码，寻码结束后应用 TCAM 进行查找。通过这样的处理方式，可以以极低的冲突概率将 TCAM 查找的关键字进行压缩，缩短 TCAM 处理每个数据包的时间，从而提高吞吐率。

3.2 算法的性能分析

为对 BF-TCAM 算法做出定量的分析，本文作表 1 所示的定义。

在第 i 个字节的 BF 引擎中，假设 $k_i n_i < m_i$，且模式集 S_i 中的每个哈希函数都是完全随机的。当集合 N_i 中所有的元素都被 k 个哈希函数映射到 m 位的比特向量时，这个向量中第 j 位仍为 0 的概率为

$$p_{ij} = (1-1/m_i)^{k_i n_i} \approx e^{-k_i n_i / m_i} \tag{1}$$

由于本算法的误判率只有假阳性而没有假阴性，所以对于不是模式集的关键字经 k 个哈希函数映射后，算法误认为是模式集中的概率为

$$f_{i\_\text{conflict}} = (1-P_{ij})^k \approx (1-e^{-k_i n_i / m_i})^{k_i} \tag{2}$$

表 1 本文所用的符号及含义

符号名称	表示的含义
R	模式的长度
r_i	第 i 个字节编码后的宽度
n_i	第 i 个字节上模式集的数目
S_i	第 i 个字节上的模式集
k_i	第 i 个 BF 中采用的哈希函数的个数
H_i	第 i 个字节上的哈希函数集
m_i	第 i 个 BF 中采用的比特向量存储空间的长度
p_{ij}	第 i 个 BF 中的比特向量中第 j 位为 0 的概率
f_{i_conflict}	第 i 个 BF 的误判概率
F_{conflict}	算法总的误判概率
F	算法正确识别的概率
T	算法实现的吞吐率

由式(2)可以看出，BF 引擎的误判率 f_{conflict} 与模式集的数目、哈希算法的数目以及比特向量的宽度有关。当 n, k 固定， m 增大时，误判率 f_{conflict} 随之降低；m, k 固定， n 减小时，误判率 f_{conflict} 随之降低。当 m, n 固定时，可以通过调整哈希算法的数目来降低误判率，通过对式(2)求导并对其做凹凸性分析，可以得出当 $k=(m/n)\ln 2$ 时 f_{conflict} 取最小值：

$$f_{\text{conflict\_min}} = (1/2)^k = (1/2)^{(m/n)\ln 2} \tag{3}$$

由式(2)，式(3)可知，在一个 BF 中，对于固定的模式集，选择的哈希函数的个数与比特向量的宽度有关。例如当 m/n =50 时，哈希函数的最优个数 $k \approx 50 \times 0.7=35$ ， 此 时 $f_{\text{conflict\_min}} \approx (1/2)^{35} \approx 3\times10^{-11}$ 。

由于 TCAM 不会出现误判或漏判，因此，则对于输入宽度为 R 的关键字，算法正确识别的概率为

$$F = \prod_{i=1}^{R}(1-f_{i\_\text{conflict}}) \tag{4}$$

3.3 硬件化的 BF-TCAM 算法分裂性能分析

在 BF 中，每个哈希函数的映射都相当于在比特向量存储空间上的随机读取，因此对于 35 个哈希函数，相当于算法需要在一个周期内要实现 35 次的读取操作，如图 5(a)所示。而嵌入式的存储资源无法支持如此多的读取操作，但可以通过并行使用多个查找能力较低的存储资源来实现。例如文献[14]提出的存储核具有 5 个读写口，利用这种存储核可以在一个时钟周期内实现 5 次的随机读取，7 个并行的存储核可以实现 35 次的哈希查找，如图 5(b)所示。

设 h 为 RAM 在单周期内的最大查找次数，对于 k 次的哈希查找，需要 k/h 个相同的向量，每个向量的宽度为 $m/(k/h)$。这样便将向量宽为 m，哈希函数个数为 k 的 BF 分裂成了 k/h 个向量宽为

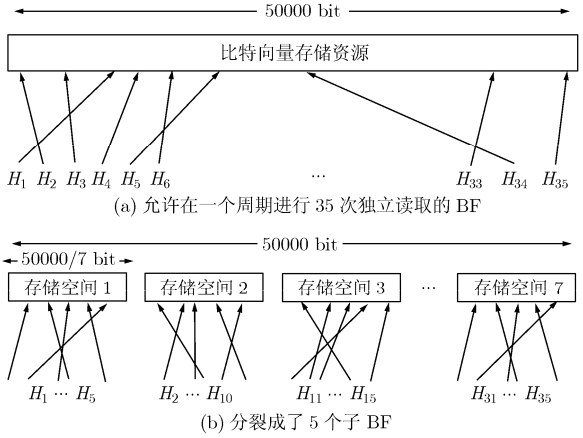

图 5 BF 算法中哈希映射的硬件实现

$m/(k/h)$，哈希函数个数为 h 的子 Bloom 滤波器。由式(1)，式(2)可知这种分裂的实现方法所产生的冲突率为

$$f' = \left[1-\left(1-\frac{1}{m/(k/h)}\right)^{hn}\right]^{h(k/h)} \approx [1-e^{-nk/m}]^k \quad (5)$$

比较式(1)和式(5)可以看出，可以通过并行的使用多个向量来满足多哈希函数的需求，且冲突概率近似相同。

BF-TCAM 算法为每个字节设置了一个 BF 引擎，为了降低冲突概率，当单字节上出现的模式集个数 n 一定时，一般通过加大 m 的宽度来降低冲突概率。由式(2)，式(3)，式(5)的分析可知，冲突概率 f_{conflict} 在极值点 $k=(m/n)\ln 2$ 的左边是递减的曲线，并在 $k=(m/n)\ln 2$ 时出现最小值。当 $m/n=50$ 时，哈希函数的最优个数 $k\approx 50\times 0.7=35$，而硬件实现所支持的哈希函数要远远小于 35，因此在 BF 的硬件化过程中需要实现更多的哈希函数来降低冲突概率。由于存储核实现比较困难，文献[11]利用片内的双端口的RAM存储资源实现了多哈希的映射，但这种特殊的 RAM 在一个周期内仅支持两次读取操作，所以该实现方式只能完成两个哈希函数的映射。

传统的 BF 算法在模式匹配时是通过多次判断哈希函数映射的位置上是否为 1 来判定该字符串是否属于模式集，这种特点决定了该算法在硬件化实现时需要在一个周期内进行多次读取来实现多个哈希函数的映射。而 BF-TCAM 算法在 BF 寻码时是通过读取向量存储空间中存储的编码来将字符串进行分类，因此本算法可以通过一次读取的方式来实现多个哈希函数的映射，并可以动态调整 BF 中的哈希函数的个数。如图 6 所示，关键词经过 3 个哈希函数映射后得到 3 个地址，算法将其合并成一个地址 A，并读出 RAM 中地址 A 上的编码。

4 实验环境的建立

本文提出的算法已应用于高可信网络业务管控系统。该系统板采用 Netlogic 公司 33100 系列的 TCAM 芯片，存储空间为 36 Mbit，总线宽度为 72 bit，选用 CYPRESS 公司的 CY1470V25 的 SRAM 芯片，存储空间 18 Mbit，总线宽度为 36 bit，系统的主体程序在 Xilinx 公司生产的型号为 V5 系列 240 T 的 FPGA 上实现。240 T 芯片上含有 325 个嵌入的块存储器，将每个存储器配置成 1024 bit 宽，4 bit 深的 RAM。采用这类 RAM 并通过流水线的方式可以实现 m=1024, k=10 的 BF，当 n=32 时，$f_{\text{conflict}}=1.94\times 10^{-6}$。将 3 个这样的存储器合成一个 BF 引擎，则 $f_{\text{conflict}}=7.29\times 10^{-18}$，设查找关键字的宽度为 R 字节，则算法总的误判概率为

$$F_{\text{conflict}} = 1-(1-f_{\text{conflict}})^R \approx Rf_{\text{conflict}} \quad (6)$$

当 R=50 时，$F_{\text{conflict}}=50\times 7.29\times 10^{-18}=3.64\times 10^{-16}$，该算法通过一次读取来取代传统算法对于向量存储空间的多次读取，使得应用 3 个存储器产生的假阳性概率便达到了文献[11]提出的算法中 8 个存储器的效果。

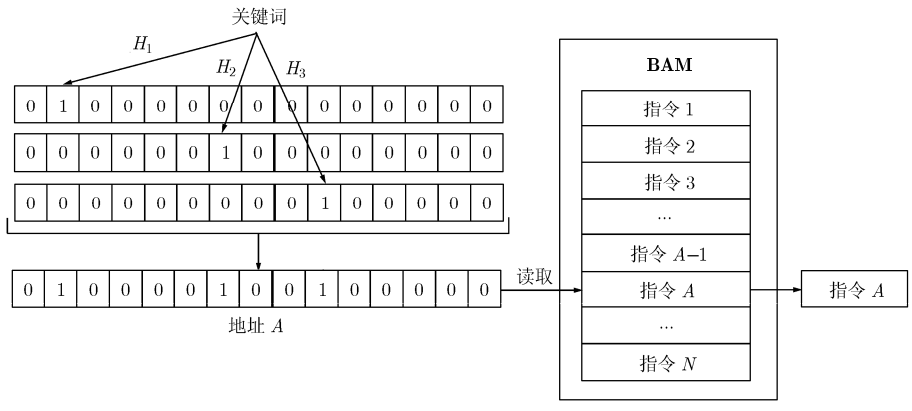

图 6 BF-TCAM 算法中哈希函数映射的硬件实现

5　实验验证

假设每个字节中可能出现的模式集的个数均小于15，每个字节的BF引擎可以将编码配置为4 bit，通过这样的处理并配合流水线操作，算法可以实现的最大吞吐率为 $T = 200\ \text{MHz} \times 72\ \text{bit} \times 2 = 28.8$ Gbps。

由于业务识别的关键字通常为五元组和部分报文负载，由于五元组中的源IP，目的IP，源端口和目的端口需要进行范围匹配，而应用本算法对其进行处理后往往需要将一条模式转换为多条模式进行存储，反而增大了TCAM的存储空间，因此，本算法对于五元组不进行压缩，而是直接将其输入TCAM中利用X(无关项)来实现范围匹配和最长前缀匹配的查找。

表2–表4分别统计了当模式集的个数为$32k$时，文献[11]中提出的并行化硬件实现的BF算法、传统的TCAM查找方式以及BF-TCAM算法在查找内容大小、吞吐率、包处理能力、系统吞吐率以及TCAM存储空间方面的对比，其中算法的吞吐率是指算法每秒钟处理的比特数；包处理能力是指只对固定关键字查找的系统每秒中处理的包的个数；系统支持的吞吐率是指该系统的包处理能力与平均包长的乘积。

由表2和表3可以看出，在TCAM查找周期相同时，BF-TCAM算法和传统的TCAM查找方式在包处理能力以及存储空间的占用上是相同的，但BF-TCAM算法的查找内容大小以及吞吐率要优于传统的TCAM方式。由表4可以看出，当查找内容大小相同时，BF-TCAM算法比TCAM查找算法多出了一倍的吞吐率和包处理能力，同时存储空间下降到了原来的50%，取得了很好的效果。

6　结论

本文针对骨干网上高速的长模式串匹配算法，提出了基于BF与TCAM并行查找相结合的算法——BF-TCAM算法，该算法解决了传统的基于TCAM的查找方式在性能上受限于TCAM芯片本身的输入宽度和存储容量的问题。通过应用改进型的BF完成了关键字的压缩，降低了冲突概率并提高了存储空间利用率；通过将压缩后的关键字送往TCAM进行查找，提高了算法的吞吐率。假设每个字节中可能出现的模式集的个数均小于15，与传统TCAM查找方式相比，BF-TCAM算法在编码长度为4 bit时可将吞吐率加倍，同时存储空间缩短为原来的二分之一。本文的下一步工作是根据每个字节上模式集的数目，动态的调整编码长度，从而更好地优化资源，增强算法的可适用范围。

表2 3种算法在TCAM查找周期为2 CLK时的对比

算法	BF算法	TCAM方式	BF-TCAM算法
查找内容大小(bit)	144	144	192
算法的吞吐率(Gbps)	2.46	14.4	19.2
包处理能力(Mps)	17	100	100
系统支持吞吐率	17 MHz × 平均包长	100 MHz × 平均包长	100 MHz × 平均包长
TCAM存储空间(Mbit)		4.5	4.5

表3 3种算法在TCAM查找周期为4 CLK时的对比

算法	BF算法	TCAM方式	BF-TCAM算法
查找内容大小(bit)	288	288	480
算法的吞吐率(Gbps)	2.46	14.4	24
包处理能力(Mps)	8.5	50	50
系统支持吞吐率	8.5 MHz × 平均包长	50 MHz × 平均包长	50 MHz × 平均包长
TCAM存储空间(Mbit)		9	9

表4 3种算法在查找内容为24 byte时的对比

算法	BF算法	TCAM方式	BF-TCAM算法
查找内容大小(bit)	192	192	192
算法的吞吐率(Gbps)	2.46	9.6	19.2
包处理能力(Mps)	12.8	50	100
系统支持吞吐率	12.8 MHz × 平均包长	50 MHz × 平均包长	100 MHz × 平均包长
TCAM存储空间(Mbit)		18	9

参 考 文 献

[1] Kim Junghak and Choi Song-in. High speed pattern matching for deep packet inspection[C]. Proceedings of Communications and Information Technology, Icheon, Sep. 28–30 2009: 1310–1315.

[2] Yu Song, Zheng Jun, and Hu Wen-xin. Improved KMP

algorithm [J]. *Journal of East China Normal University* (*Natural Science*), 2009, (4): 92–97.

[3] Boyer R S and Moore J S. A fast string searching algorithm[J]. *Communications of the ACM*, 1975, 20(10): 762–772.

[4] Aho A and Corasick M. Efficient string matching: an aid to bibliographic search[J]. *Communications of the ACM*, 1975, 18(6): 333–340.

[5] Beate Commentz-Walter. A string matching algorithm fast on the average[J]. *Information Systems*, 1980, 5(3): 245–246.

[6] Coit J, Staniford S, and McAlerney J. Towards faster string matching for intrusion detection or exceeding the speed of snort[C]. Proceedings of DISCEX II, Anaheim, CA, USA, June 2001, Vol.1: 367–373.

[7] Smith P D. On tuning the boyer-moore-horspool string searching algorithm[J]. *Software*: *Practice and Experience*, 1994, 24(4): 435–436.

[8] Liu A X and Gouda M G. Complete redundancy removal for packet classifiers in TCAMs[J]. *Parallel and Distributed Systems*, 2010, 21(4): 424–437.

[9] Liu A X, Meiners R, and Torng. TCAM Razor: a systematic approach towards minimizing packet classifiers in TCAMs[J]. *Communication of the ACM*, 2010, 18(2): 490–500.

[10] Bloom B. Space/time trade-offs in hash coding with allowable errors[J]. *Communication of the ACM*, 1970, 13(7): 422–426.

[11] Dharmapurikar, Krishnamurthy, Sproull T S, *et al.*. Deep packet inspection using parallel Bloomfilters[C]. IEEE Symposium on High Performance Interconnects. Stanford, CA, 2003: 162–170.

[12] Che Hao, Wang Zhi-jun, Zheng Kai, *et al.*. DRES: dynamic range encoding scheme for TCAM coprocessors[J]. *IEEE Transactions on Computers*, 2008, 57(7): 902–915.

[13] Kim Young-deok, Ahn Hyun-seok, Kim Suhwan, *et al.*. A high-speed range-matching TCAM for storage-efficient packet classification[J]. *IEEE Transactions on Circuits and Systems*, 2009, 56(6): 1221–1230.

[14] Dipert B. Special purpose SRAMs smooth the ride[J]. *Electrical Design News*, 1999, 44(13): 93.

陈正虎： 男，1986年生，硕士生，研究方向为宽带信息网络和高速业务管理控制.

兰巨龙： 男，1962年生，博士生导师，教授，研究方向为网络路由理论与技术、并行交换结构和IPv6技术等.

黄万伟： 男，1979年生，博士生，研究方向为可重构计算、宽带信息网络.

李玉峰： 男，1976年生，博士生，讲师，研究方向为宽带信息网络、高速路由器核心技术.