第38卷 第18期
Vol.38 No.18

计 算 机 工 程
Computer Engineering

2012年9月
September 2012

·工程应用技术与实现· 文章编号：1000—3428(2012)18—0231—03 文献标识码：A 中图分类号：TP303

基于FPGA的多模式匹配算法研究与实现

骆 潇[1]，郭 健[1]，邓 敏[1]，白 斌[2]

(1. 西南电子电信技术研究所，成都 610041；2. 中国电子科技集团公司第30研究所，成都 610000)

摘 要：针对模式匹配软件算法速度慢、正确率低等问题，提出一种基于FPGA的硬件多模式匹配算法，通过设计窗口折叠的布鲁姆过滤器，有效减小资源消耗。与多级确认机制结合可降低虚警率，并采用不同于传统方式的并行结构提高查询速度，实现高效率无虚警的精确模式匹配。实验结果表明，采用该算法的高速入侵检测系统吞吐率可达到10 Gb/s。

关键词：布鲁姆过滤器；入侵检测；模式匹配；现场可编程门阵列；流水线结构

Research and Realization of Multiple Patterns Matching Algorithm Based on FPGA

LUO Xiao[1], GUO Jian[1], DENG Min[1], BAI Bin[2]

(1. Southwest Electronics and Telecommunication Technology Research Institute, Chengdu 610041, China;
2. The 30th Institute of China Electronics Technology Group Corporation, Chengdu 610000, China)

【Abstract】In view of the low efficiency of software performance in multi-pattern matching algorithm, a new hardware-based solution based on Field Programmable Gate Array(FPGA) is proposed. By using Bloom Filter(BF) data window collapsing, the on-chip hardware resource consumption is reduced and efficiency is improved. To reduce the possibility of false positive, the multilevel-verification is applied. A pipeline parallel structure is used to improve the query efficiency, which makes the pattern matching work in the high performance with no false positive. The method is applied in a high-speed network intrusion detection system. Experimental result shows that the throughout of system can reach the line speed of 10 Gb/s.

【Key words】Bloom Filter(BF); intrusion detection; pattern matching; Field Programmable Gate Array(FPGA); pipeline structure

DOI: 10.3969/j.issn.1000-3428.2012.18.062

1 概述

多模式匹配是指对于一个给定数据串T，一次扫描判断一组含有n个数据串的模式串$P[1,2,\cdots,n]$在T中所有的出现。其相应技术在数据压缩、病毒检测和基因序列、搜索引擎、入侵检测等领域有着广泛应用。目前，主流软件实现的算法上常采用的方式是跳跃字符策略[1]、hash散列移位方法[2]、基于排除的字符串匹配[3]以及有限状态自动机扫描[4]等，如BM[1]、WM[2](Wu-Manber)、ExB[3]、AC(Aho-Corasick)[4]算法等。多数算法在软件实现时是顺序执行，且受CPU计算能力、内存容量、cache大小、存储延迟等方面的制约，时间复杂度较高。如常用的入侵检测软件snort，实现模式匹配算法最高只能达到接近300 Mb/s的吞吐率[5]。特别是在网络安全应用中，随着网络带宽迅速增长，入侵检测和包内容过滤等功能对模式匹配速度提出了更高要求，用软件实现难以达到G比特级的匹配扫描速度。因此，很多研究开始倾向于利用硬件的快速并行处理能力来实现模式匹配算法。

目前，适合于硬件实现的匹配算法主要有bloom[6-8]、WM、AC等，WM和AC算法的匹配速度快，但基于跳跃式查询和哈希(hash)散列的WM算法在大模式集情况下性能显著下降，基于有穷自动状态机的AC算法对内存量要求高，而布鲁姆过滤器(Bloom Filter, BF)[6-8]是一种基于多个hash函数映射来压缩参数空间的数据结构，采用一个位串表示并支持集合元素的hash查找，具有简单、空间利用率高的特点，自从1970年BF首次被提出，即在计算机系统中广泛应用于大数据集表达与查找。本文在基本型BF的基础上设计了一种新的适于FPGA实现的多模式匹配算法，通过窗口折叠的BF计算以及高效的并行流水线结构设计，从根本上使得模式匹配性能在原有算法基础上得到了较大提高，并以更少的硬件资源消耗得到更大的模式容量和更快的处理速度。

2 变长模式的布鲁姆过滤器

BF是一种特殊的数据结构，该数据结构通过对含n个关键词的模式集进行hash转换形成，可用于实现对集合

基金项目：教育部人文社科基金资助项目(10YJCZH169)；四川省金融智能与金融工程重点实验室基金资助项目(FIFE2010-P05)

作者简介：骆 潇(1980－)，女，工程师，主研方向：光纤网络通信，高速网络数据处理；郭 健、邓 敏、白 斌，工程师

收稿日期：2011-11-14 **修回日期：**2012-01-14 **E-mail：**ivy_xx@163.com

成员的高效查询。设一个位串长度为 i 的数据集合 $T=\{t_1,t_2,\cdots,t_i\}$，模式集合 $P=\{p_1,p_2,\cdots,p_n\}$，服从均匀分布的独立 hash 函数集合 $H=\{h_1,h_2,\cdots,h_k\}$，则 BF 数据结构为 m 维比特向量表 M，即通过 M 表示模式集 P。

对于插入操作，将模式集 P 中的每个元素分别与 hash 函数集 H 的 k 个函数计算得到 k 个 hash 值 $d_1 \sim d_k$ $(0 \leqslant d_i \quad m-1,1 \quad i \quad k)$。将向量表 M 的第 $d_i(1 \leqslant i \quad k)$ 位设置为 1。若存在 2 个不同的关键字通过 hash 计算后得到的向量地址相等，即发生冲突，此时该地址会被多次置 1。

对于查找操作，将数据集合 T 中每个元素分别与 hash 函数集 H 进行运算，得到 k 个 hash 值，若向量表 M 中这 k 个 hash 值对应位置值都为 1，则表示被查询数据可能存在于 P 中；反之任一个位置的值为 0，则数据不存在 P 中。

基本的 BF 算法只能处理定长模式，变长模式采用多个 BF 组成一个 LPM(Longest Prefix Matcher)来处理。当字串在数据窗口移动时，$L_{\min}$ =3~ $L_{\max}$ =w 长度的字串并行对相应模式长度的 BF 向量表进行查找匹配，形成一个 LPM 引擎(图 1)，可在 1 个时钟内搜索出从窗口第 1 个字节位置起的前任意长度数据在模式集中的全部匹配。而将 w 个这样的 LPM 引擎并行起来形成 PLPM，并且起始字位置依次偏移 1 Byte，则可以搜索整个数据窗口内数据的全部匹配模式。一个时钟内移动整个窗口字串的长度(w Byte)，可将处理效率提高 w 倍。

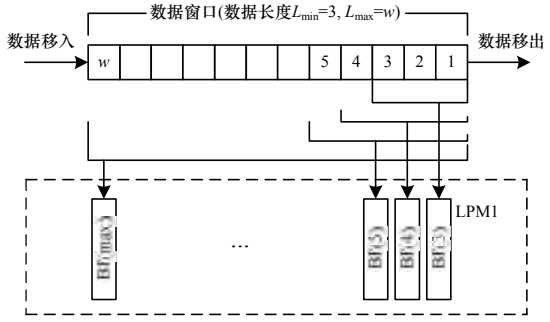

图 1 并行 BF 的 LPM 引擎

可见，搜索时间与模式集大小无关，而与每次移出的步长呈正比；BF 向量表所占的存储空间与每次移出的步长和数据窗口宽度呈正比，因此，在设计时需要在搜索效率和存储空间之间进行平衡。

3 窗口折叠的 BF

本文在基本算法基础上提出窗口折叠的 BF 算法，有效地减小表数量和 hash 计算量，可在有限的硬件资源中容纳更多的模式集，并实现更高的处理效率。

3.1 窗口折叠的 BF 设计

由于 hash 计算位宽与 BF 表深度和模式集大小正相关，因此大规模模式集需要更多硬件逻辑资源。因此，本文选用特定的 hash 函数，实现窗口折叠的 BF，以时间延迟换取资源空间并获得更高的计算效率。

hash 函数的选取方法采用文献[9]中的 H_3，若窗口字串长度为 w，则一个 LPM 的 hash 函数为 $k \times w$ 的矩阵。设窗口数据的第 i 个字节 $byte_i=(b_i^1,b_i^2,\cdots,b_i^8)$。其中，$byte_i$ 的第 r 个 hash 值 h_{ir} 的计算为 $h_{ir}=d_{ir}^1 \cdot b_i^1 \oplus d_{ir}^2 \cdot b_i^2 \oplus \cdots \oplus d_{ir}^8 \cdot b_i^8$。其中，$d_{ir}^j$ 是服从均匀分布的随机数序列，对于 m 维的 BF 映射表，d_{ir}^j 取值范围为 $0 \sim 2^m-1$。w 个字节的第 r 个 hash 值为 $h_{w,r}=h_{w-1,r} \oplus h_{w,r}$。

由以上 hash 函数选取方法可知，前 i+1 个字节的 hash 可以通过前 i 个 hash 和第 i+1 个字节的 hash 值异或计算得到，下面证明定理，前 2i 个字节的 hash 可以通过前 i 个字节 hash 和第 i+1~2i 的 i 个字节 hash 结果异或得到。

定理 设字节集合∑、$x_i \in \sum$、n 为偶数，有运算关系：

$$y_n=x_1 \oplus x_2 \oplus \cdots \oplus x_n, \quad y_n=y_{n-1} \oplus x_n \tag{1}$$

成立，若：

$$y_{n/2}= \quad x_1 \oplus x_2 \oplus \cdots \oplus x_{n/2}, \quad z_{n/2}=x_{n/2+1} \oplus x_{n/2+2} \oplus \cdots \oplus x_n \tag{2}$$

则有 $y_n=y_{n/2} \oplus z_{n/2}$ 成立。

证明：根据式(1)，有：

$$y_{n-1}=x_1 \oplus x_2 \oplus \cdots \oplus x_{n/2} \oplus x_{n/2+1} \oplus \cdots \oplus x_{n-1} \tag{3}$$

根据异或运算自反性，由式(2)推出：

$$z_{n/2} \oplus x_n=x_{n/2+1} \oplus x_{n/2+2} \oplus \cdots \oplus x_{n-1} \tag{4}$$

$$y_{n-1}= \quad y_{n/2} \oplus \quad z_{n/2} \oplus x_n \tag{5}$$

再根据式(1)和式(5)推出：

$$y_n=y_{n-1} \oplus x_n=y_{n/2} \oplus z_{n/2} \tag{6}$$

利用定理的特性本文设计了窗口折叠的 BF。数据窗口内数据按上述方法计算出 hash 值后，结果保存在 D 触发器；下一个时钟第 2 组数据移入窗口按同样方式计算，第 2 组 hash 结果与前一组的 D 触发器保存的结果异或，可得到 2 倍窗口长度的数据的 hash 值。流水线计算方法如图 2 所示。

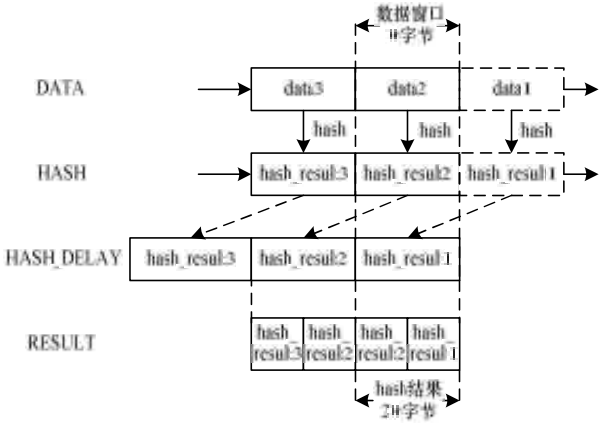

图 2 数据窗口折叠的 hash 计算

在 altera EP2S130 中用 VHDL 实现一个 BF 的 hash 计算，取 k=5，分别按一般方法和按折叠方式计算 20 Byte 窗口数据 hash 的逻辑资源消耗情况如表 1 所示。

表 1 逻辑资源消耗情况比较

方式	register	LUT	LUT-FF	D-type
正常方式	2 044	4 648	1 928	5 496
折叠方式	1 456	3 664	1 348	4 084

可见，采用折叠方法节省了约 30%的计算资源，尤其在多个 LPM 并行起来时更加明显。

3.2　多级确认

由于数据经过压缩，BF 不可避免地存在冲突，存在某一个元素不在模式集中，经 hash 映射到 BF 中的 k 个位置却均返回 1 的可能性，称为虚警。设模式数量 n，映射空间地址向量深度 m，hash 函数个数 k，由表示算法可知，在 n 个模式进行 $k \times n$ 次置位之后，向量表中某一位仍为 0 的概率是 $(1-\frac{1}{m})^{kn}$，虚警率 f 表示为 $\forall_i \in \{1,2,\cdots,k\}$ 均有 h_i 对应向量表的位置置 1 的概率。根据文献[7]，有：

$$f=(1-(1-\frac{1}{m})^{kn})^k \approx (1-\mathrm{e}^{-\frac{nk}{m}})^k \tag{7}$$

设 w 个 BF 构成一个 LPM，则 w 个 BF 都没有虚警的概率是 $(1-f)^w$，因此一个 LPM 的虚警为 $f_{\mathrm{lpm}}=1-(1-f)^w \approx Wf$[10]，$x$ 个 LPM 构成一个 PLPM，且共用 BF 向量表，则虚警为：

$$f_{\mathrm{plpm}}=1-(1-f_{\mathrm{lpm}})^x \approx xf_{\mathrm{lpm}} \tag{8}$$

由于虚警的存在，因此 PLPM 的匹配的结果必须经过二次确认。传统方式通常是建立两级引擎，第 1 级过滤掉不可能匹配，再直接进入第 2 级作精确匹配。这样 BF 虚警的影响会直接带给第 2 级引擎，由于精确匹配耗时多效率低，虚警过高会使花在精确匹配上的时间大大增加。本文设计了一种有效的多级确认机制，在一二级引擎间再增加一级确认过滤器对疑似匹配数据进行确认，确认通过的数据再送入精确匹配模块处理。实验证明，采用该方法能过滤掉 90%以上的虚警。

3.3　并行处理结构

如图 3 所示为两级流水线结构。

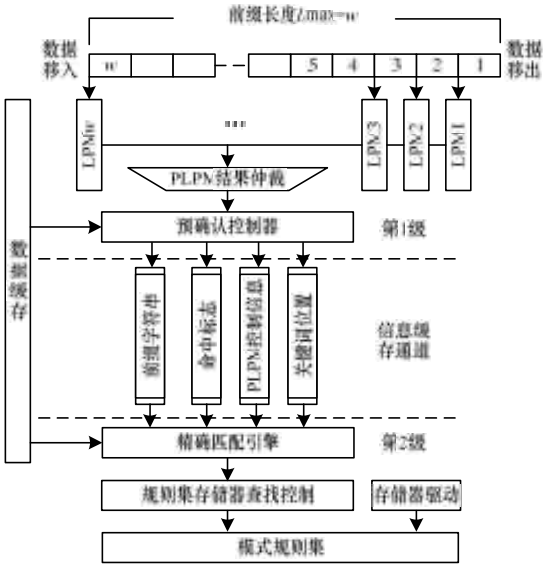

图 3　并行处理结构

在前缀匹配中，只要一个 LPM 中任意一个 BF 返回命中，则视为该 LPM 命中，在 PLPM 中可能一次产生多个 LPM 命中，因此，精确匹配无法与算法计算完全同步，其处理效率和方式制约着整个多模式匹配器的效率。因此，本文将算法计算和精确查找独立起来，在前缀匹配和精确匹配之间设置了数据和标志信息缓存通道，采用流水线技术使得两部分处理并行工作，对模式在真实数据流出现的不均衡性进行了平滑，避免了传统的查询方式中确认等待时间过多造成的效率浪费。

模式集存储于外部存储器中，查询提交和结果读取均采用 fifo 缓存的方式，同样实现了流水线设计。

4　系统性能分析与实验结果

本文将基于折叠窗口 BF 的模式匹配器设计应用于高速网络入侵检测系统中，包处理、协议分析和模式匹配设计均在一片 FPGA 上完成。主要分析 3 个方面：算法的时间复杂度和空间复杂度，并行结构系统吞吐率。

由算法可知，每次查找需要作 k 个 hash，软件的算法时间复杂度为 $O(k)$，硬件并行处理可将 k 次 hash 并行执行，此时计算时间复杂度仅为 $O(1)$，空间复杂度为 $O(k)$。

本文系统中由于前缀算法计算与精确匹配采用流水线方式，则系统吞吐率由两级流水处理能力的最小值决定。前缀部分 BF 算法计算以流水线方式实现完全线速处理，设系统时钟 F_1 MHz，处理窗口字节宽度 M，前缀部分计算效率为 $V_1=F_1\times 8\times M$。

精确匹配部分查询时钟为 F_2、工作时钟 F_1、且单次提交查找及其他控制操作需 N 个时钟周期，则理论上单次查找所需时间为 $(1/F_2+N/F_1)$ s，由于精确匹配部分的查询和结果读取同样采用流水线方式，因此本文设计的实际查询时间为 $\max(1/F_2, N/F_1)$。设 f 为前缀匹配的虚警，p 表示模式在数据流中出现的概率，由于虚警而造成本不该命中却需要进行二次确认的概率为 $(1-p)\times f$，则精确匹配处理流量为：

$$V_2=\frac{8}{(p+(1-p)\times f)\times \max(1/F_2, N/F_1)} \tag{9}$$

当系统时钟为 155 MHz、精确匹配处理时钟为 200 MHz、前缀匹配窗口数据 10 Byte、每次确认平均 6 个时钟、虚警率为 3%、线路数据模式出现率小于 1%时，V_1=12.4 Gb/s，V_2=10.4 Gb/s。因此，系统吞吐率可至少达到 10 Gb/s。如图 4 为吞吐率随虚警和字符出现率变化的情况。

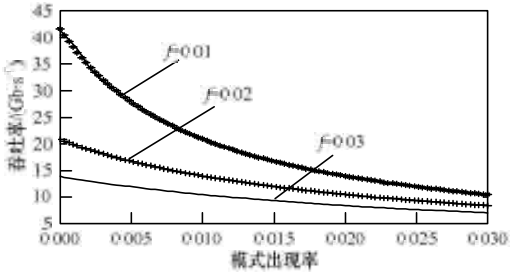

图 4　匹配处理吞吐率变化情况

通过采用测试仪 AX4000 作为测试信号源生成网络数据，对本文系统性能进行详细测试。在表 2 中数据包按一定比例加入符合规则的关键词，配置 3 万条字符串模式。结果表明，系统吞吐率与包长度和字符串出现率有关，短包情况下系统负担较重，但系统在字符串出现率低于 2.5%时，短包也能达到 10 Gb/s 的吞吐率。　(下转第 237 页)