海量模式高效匹配方法研究

张宏莉[1],徐东亮[1],梁 敏[2],刘宇峰[3]
(1.哈尔滨工业大学计算机科学与技术学院,黑龙江哈尔滨 150001;2.空军大连通信士官学校信息网络系,辽宁大连 116600;
3.中国人民解放军63851部队90分队网络管理,吉林白城 137001)

摘 要: 本文提出了一种基于随机指纹模型的Wu and Manber(WM)算法(Randomizing Fingerprint WM,RFP-WM),它通过为每一个模式串计算唯一指纹可以有效降低误报率.与WM算法相比,RFP-WM算法极大地降低了哈希冲突率,提高了命中率,在海量模式集上这一效果更为显著.实验结果表明,相对于传统WM算法,该算法的匹配效率更高,而且模式集的规模越大,性能越优越.

关键词: 指纹模型;模式匹配;指纹Wu and Manber(WM)算法
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2014)06-1220-028
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2014.06.028

Massive Strings Efficient Matching Method Research

ZHANG Hong-li[1], XU Dong-liang[1], LIANG Min[2], LIU Yu-feng[3]
(1. *School of Computer Sicience and Technology, Harbin Institute of Technology, Harbin, Heilongjiang* 150001, *China*;
2. *Dept. of Information Network, Air Force Communications Officer School in Dalian, Dalian, Liaoning* 116600, *China*;
3. *Network Management*, 63851 *PLA Troops* 90 *Unit, Baicheng, Jilin* 137001, *China*)

Abstract: This paper presents a randomizing fingerprint-based Wu and Manber(WM) algorithm(RFP-WM), which can effectively reduce false positives rate by calculating a unique fingerprint for each pattern. Compared with WM algorithm, RFP-WM algorithm greatly reduces the hash collision rate and increases the hit rate, especially in the massive patterns set. Experiment results show that the performance of the RFP-WM algorithm is more superior than traditional Wu and Manber(WM) algorithm on the larger pattern set.

Key words: fingerprint model; pattern matching; fingerprint-based Wu and Manber(WM) algorithm

1 引言

模式匹配(pattern matching, string matching),也称为特征匹配,就是给定一个模式集合 $P=\{P^1, P^2, \cdots, P^c\}$,对于给定的一个输入文本,找出模式集中的模式在文本中出现的所有位置.其中,P^c 是定义在有限字母表 Σ 上的字符串 $P^c=\{p_1^c, p_2^c, \cdots, p_l^c\}$.

模式匹配方法应用广泛,也有大量成熟的研究,文献[1]提出了一种基于单模式方法的优秀改进算法.文献[2]提出了一种基于智能有限自动机的正则表达式匹配算法.文献[3]提出了一个基于多窗口滑动的通用模式匹配方法,该方法可以应用在基于自动机和基于比较的高效模式匹配算法上.作者将该方法应用在Fast-Search[4]和TVSBS[5]算法上,变种算法在大字母表和自然语言文本上效果很好.

WM(Wu and Manber)算法[6]是最著名的多模式匹配算法之一,该算法采用不良字符块转移机制来实现匹配,并利用哈希函数选择匹配阶段可能匹配的模式串.文献[7]提出了一种共享位置的并行WM算法,利用多种操作几个扫描窗口同时对同一内存内容进行并行匹配.文献[8]使用OpenCL架构在GPU上实现并行WM算法,利用GPU的高性能并行计算能力加强了WM算法的处理能力.

本文从待处理的模式集和网络流量的特点出发,提出一种基于随机指纹模型的方法,并将其应用于WM算法,进一步对WM算法进行改进,提出并实现了一种实时的针对海量数据集的在线高效匹配方法.

收稿日期:2013-08-20;修回日期:2014-02-19;责任编辑:马兰英
基金项目:国家重点基础研究发展计划"973"计划基金(No.2011CB302605);国家高技术研究发展计划("863"计划)基金(No.2011AA010705, No.2012AA012502, No.2012AA012506);"十一五"国家科技支撑计划(No.2012BAH37B01)

2 随机指纹模型

首先,我们给出本文中使用的模式串随机指纹的定义.

定义 1 $\varphi(P)$是模式串 P 的指纹,当且仅当 $\varphi(P)$ 满足以下两个条件:

(1)φ 是模式串 P 或者其子串的函数,如果两个模式串或者取其指纹的子串相同,那么它们的指纹是相同的.

(2)对于任意两个模式串 $P_1 \neq P_2$, $\rho_r(\varphi_{f,\alpha}(P_1) = \varphi_{f,\alpha}(P_2)) \ll 1$,即 P_1与 P_2指纹相同的概率远远小于 1.

定义 2 对于一个给定的模式串 $P = p_1, \cdots p_l, \alpha \in \theta(N^4)$, P 的一个多项式指纹为 $\varphi_{f_i,\alpha}(P) = (\sum_i^l p_i f_i) \bmod \alpha$,其中 $1 \leqslant i \leqslant l$, $f \in F_\alpha$,且 a 为一个质数, F_a 是整数与质数 a 模后的域.

一个多项式指纹有以下两个性质:

(1)可以根据 $p_1 \cdots p_l, \varphi(p_1, \cdots p_l)$和 p_{l+1}计算出指纹 $\varphi(p_1, \cdots p_l, p_{l+1})$.

(2)对于 $1 \leqslant i \leqslant l$,可以根据 $p_1 \cdots p_l, \varphi(p_1, \cdots p_l)$和 $p_1, \cdots p_i$ 的指纹 $\varphi(p_1, \cdots p_i)$计算出 $p_{1+i}, p_{2+i}, \cdots p_l$ 的指纹函数 $\varphi(p_{1+i}, p_{2+i}, \cdots p_l)$.

定理 1 对于两个长度为 l 的不同模式串 P_1 和 P_2, $l < n$, n 为待匹配文本 T 的大小. $\varphi_{f,\alpha}$为一个多项式指纹同, $\alpha \in \theta(N^4)$. 那么, $\rho_r(\varphi_{f,\alpha}(P_1) = \varphi_{f,\alpha}(P_2)) < \frac{1}{n^3}$,即模式串 u 与 v 指纹相同的概率小于$\frac{1}{n^3}$.

证明 如果模式串 P_1 和 P_2 的多项式指纹函数 $\varphi_{f,\alpha}$满足等式 $\varphi_{f,\alpha}(P_1) = \varphi_{f,\alpha}(P_2)$,则 $\varphi_{f,\alpha}(P_1) - \varphi_{f,\alpha}(P_2) = 0$,该等式可以看作 f 的函数. $f \in F_\alpha$, F_a 是整数与质数 a 模后的域,令域中 f 的可选个数最多为 γ 个,所以从 F_a 中任选一个 f 的概率为$\frac{\gamma}{n^4} \leqslant \frac{n}{n^4} = \frac{1}{n^3}$.

根据以上内容,我们给出了一个基于指纹技术的随机模式匹配算法.为便于理解,我们先给出一个单模式匹配的算法.设模式串为 $P = p_0 p_1 \cdots p_l$,文本为 $T = t_0 t_1 \cdots t_n$, $l \leqslant n$, $P(i) = p_i p_{i+1} \cdots p_{i+w-1}$, $T(j) = t_j t_{j+1} \cdots t_{j+w+1}$, $0 \leqslant i \leqslant l - w + 1$, $0 \leqslant j \leqslant n - (j + w + 1)$, i 为模式子窗口的指纹索引,即计算指纹函数的起始字符, n 为文本长度, w 为窗口长度, $\varphi_{f,a}(P(i))$为模式串 P 的指纹函数,matchnum 为命中的次数.算法伪代码如 Algorithm 1.

Algorithm 1　Single-Pattern_Match(P, T)

```
Input: P = p0p1⋯pl, T = t0t1⋯tn
Output: matchnum  //number of hit
Procedures:
01:  matchnum←0
02:  for(j = 1, i = 3; j + l − i < n; j + +)
03:    α←randomly chosen a big prime
04:    f←randomly chosen elements from Fa
05:    if(φf,α(P(i)) = φf,α(T(j)))
06:      if(P = T(j))//P = p0p1⋯pl,指纹相同,进行验证
07:        matchnum + +
08:      End if
09:    End if
10:  End for
11:  return matchnum
```

从 Algorithm 1 中可以看出,指纹函数从第三个字符开始计算模式串的指纹($i = 3$),设定指纹窗口的大小为 l-i,参数 f, a 可以在算法运行时随机选取.我们基于以上理论又提出了一个基于随机指纹的多模式匹配算法,Algorithm 2.其中,lmin 是最短模式串的长度,为了方便表述,窗口长度统一设为 lmin-i, matchnum 为大小为 c 的数组,记录每个模式串的命中次数.

Algorithm 2　Multi-Pattern_Match(P, T)

```
Input: P = {P1, P2, ⋯, Pc}, T = t0t1⋯tn
Output: matchnum  //count array
Procedures:
01: for(k = 0; k < c; k + +)
02:   matchnumk←0
03: End for
04: for(j = 0, i = 3; j + lmin-i < n; j + +)
05:   for(k = 0; k < c; k + +)
06:     α←randomly chosen a big prime
07:     f←randomly chosen elements from Fa
08:     if(φf,α(P(i)) = φf,α(T(j)))
09:       if(P^k = T(j))//P^k = p1^k p2^k ⋯ pl^k,验证
10:         matchnumk + +
11:       End if
12:     End if
13:   End for
14: End for
15: return matchnum
```

从 Algorithm 1 和 Algorithm 2 中,可以看出,指纹窗口的长度可以根据每个模式串的长度来设定,即 i 的值可变.以上两个算法是一种朴素的模式匹配算法.这两

个算法的平均时间复杂度分别为 $O(n)$和 $O(nk)$.我们将随机指纹函数的思想引入到 WM 算法中使之产生可以满足当前网络环境下在线流量匹配要求的算法.

3 随机指纹 WM 算法

WM 算法首先对模式串集合进行预处理.预处理阶段将建立 3 个表格:SHIFT 表,HASH 表,PREFIX 表.SHIFT 表根据读入字符串决定可以跳过的字符数.HASH 表用来存储尾块字符散列值相同的模式串.PREFIX 表用于存储尾块字符散列值相同的模式串的首块字符散列值.我们设定模式串窗口 w 的长度为模式集合 P 中最短的模式串的长度,$l\min$. l^k 为模式集中第 k 个模式串的长度.

随机指纹 WM 算法(Randomizing Fingerprint WM,RFP-WM)将随机指纹模型应用到了 WM 算法中时,对 WM 算法和指纹模型都做了一些改进:

(1)针对我们的模式集进行训练,选取较优参数 f 和 a.

(2)计算出模式集合中每一个模式串的所有长度为 $l\min$ 的窗口 w 的指纹,并且相互比较,找出其指纹冲突率最小的指纹窗口,然后在该窗口基础上计算 WM 算法中的 SHIFT 表和 HASH 表.

(3)将冲突率最小的指纹保存为 FPRINT 表,并将该窗口第一个字符的位置记录下来,作为该模式串的指纹索引(index),用该 FPRINT 表替换传统 WM 算法中的 PREFIX 表.改进算法如 Algorithm 3.

Algorithm 3 是整个算法的预处理部分.首先,针对我们的模式集训练出较优的参数 f 和 a,将其用于对后面窗口指纹的计算;其次,计算每个模式串中所有长度为 $l\min$ 的窗口的指纹;再次,统计每个模式串的所有指纹在整个模式集中出现的频率;最后,对每个模式串,选其中指纹出现频率最小的那个窗口的指纹作为该模式串的指纹,并记录下该窗口的起始位置.块 B、SHIFT 和 HASH 表都是在每个模式串的最小指纹频率的窗口上计算的.搜索阶段与传统 WM 算法相似,将当前位置 pos 初始化为窗口长度 w,对于每个当前位置 pos,从后向前读入 B 个字符的块.如果 SHIFT$[k]>0$,那么将窗口移动到位置 pos + SHIFT$[k]$,并继续搜索;如果 SHIFT$[k]=0$,则用指纹函数找出模式集中所有与该块文本指纹相同的模式串,并逐个与文本比较.将 Algorithm 3 和 Algorithm 4 相结合,就够成了我们的改进算法随机指纹 WM 算法(RFP-WM).

Algorithm 3 RFP-WM_Preprocess(P)

Input: $P=\{P^1,P^2,\cdots,P^c\}$

Output: HASH, SHIFT, FPRINT

Procedures:

01: $\alpha\leftarrow$training a big prime

02: $f\leftarrow$training an element from F_a

03: for($k=1;k<c+1;k++$)

04: for($i=0;i<l^k-l\min+1;i++$)

05: fingerprint$[k]$.win$[i]\leftarrow\varphi_{f,\alpha}(P^k(i))$

06: End for

07: End for

08: for($k=1;k<c+1;k++$)

09: for($i=0;i<l^k-l\min+1;i++$)

10: fingerprint$[k]$.win$[i]$.frequency$\leftarrow$FrequencyCount(fingerprint$[k]$)

11: End for

12: End for

13: for($k=1;k<c+1;k++$)

14: for($i=0;i<l^k-l\min+1;i++$)

15: if(fingerprint$[k]$.win$[i]$.frequency is the smallest)

16: index$\leftarrow i$

17: PRINT$[k]$.fingerprint$_{\text{index}}\leftarrow\varphi_{f,\alpha}(P^k(i))$

18: End if

19: End for

20: End for

21: Computation of B at win$[i]$

22: Construction of the tables SHIFT and HASH at win$[i]$

Algorithm 4 RFP-WM_Searching(P, T)

Input: $P=\{P^1,P^2,\cdots,P^c\}$, $T=t_0t_1\cdots t_n$

Output: matchnum //count array

Procedures:

01: pos$\leftarrow w$

02: while(pos$<m-w$)

03: $k\leftarrow$hash($t_{\text{pos-}B+1}\cdots t_{\text{pos}}$)

04: if(SHIFT$[k]=0$)

05: index$\leftarrow$pos-w

06: list$\leftarrow\varphi_{f,\alpha}(P^k(\text{index}))$

07: verify all the patterns in list one by one

08: if(there is a pattern in the text)

09: matchnum$_k$++

10: End if

11: pos$\leftarrow$pos + 1

12: else

13: pos$\leftarrow$pos + SHIFT$[k]$

14:　End if
15:End while

我们用一个实例来说明我们的算法.给定一个模式集 $P = \{fried, filed, hundred, thing, something, algorithm\}$,因为模式集中最短模式串的长度是 5 个字符,所以我们的窗口长度设为 5,首先,我们要计算出每个模式串的唯一指纹,并确定相应的窗口位置.假设 P 中的每个模式串的指纹是在以下窗口(用斜体标出)上取得:*fried*, *filed*, h*undre*d, *thing*, s*omeths*ing, alg*orith*m,那么我们的 HASH 表、SHIFT 表和 FPTINT 表分别如图 1、表 1 和图 2 所示.

表 1　SHIFT 表

fr	ri	ie	ed	fi	il	le	un	nd	dr	re	th	hi	in	ng	om	me	et	or	ri	it
3	2	1	0	3	2	1	3	2	1	0	0	2	1	0	3	2	1	3	2	1

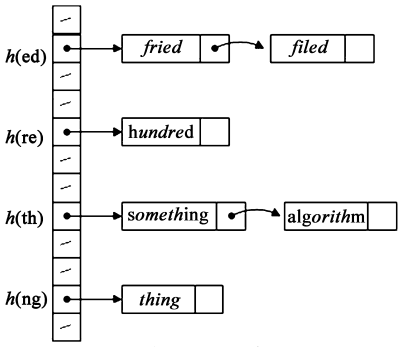


图1　HASH表

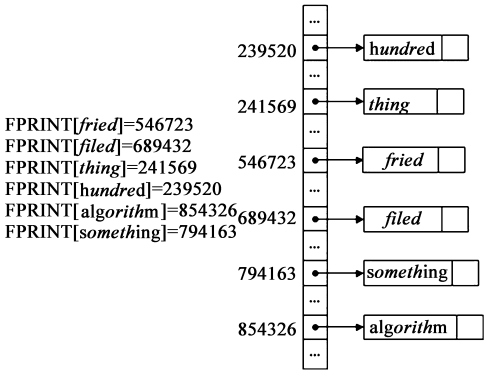


图2　FPRINT表

HASH 表和 SHIFT 表与传统 WM 算法相似.我们主要是添加了 FPRINT 表.图 2 左侧是每个模式串的指纹,右侧是相应的指纹表,当跳跃值为 0 时,就进入指纹表进行查找.

4　实验评价

本节中,我们通过实验来评价我们方法的性能.我们的实验环境为:CPU,Interl i5-243002.4GHz,内存 1GB,Fedora13 系统.我们将 RFP-WM 算法在上述环境下实现,并将其与传统的 WM 算法进行比较.实验数据来自国家骨干网国际出口抓取的 50 万条 URL,具体如表 2.

图 3 是用表 2 中不同的模式集在相同的 50 万条待匹配 URL 上匹配得出的结果.由图中可以看出,RFP-WM 算法的性能始终好于 WM 算法,而且,模式集越大,RFP-WM 算法的性能越优越.在模式为 8 万条的模式集上,WM 算法的执行时间接近于 RFP-WM 算法的两倍;在 10 万条的模式集上,WM 算法的执行时间为 7364ms,已经是 RFP-WM 算法 3543ms 的两倍了.

表 2　实验数据集

数据集(Data Set)	模式串数量	模式串长度
Data Set1	20,000	8 – 50
Data Set 2	40,000	8 – 50
Data Set 3	60,000	8 – 50
Data Set 4	80,000	8 – 50
Data Set 5	100,000	8 – 50

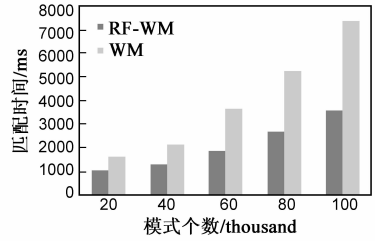


图3　RFP-WM与WM算法在不同模式集的性能对比

图 4 是 RFP-WM 算法与 WM 算法用相同的模式集(2 万条模式串)分别在 10,20,30,40,50 万条 URL 文本上匹配得出的结果.由图中同样可以看出,文本越大,RFP-WM 算法的性能越优越.在文本大小为 50 万条 URL 时,WM 算法的执行时间为 RFP-WM 算法的 1.5 倍.

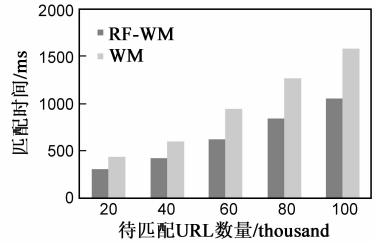


图4　RFP-WM与WM算法处理不同文本的性能对比

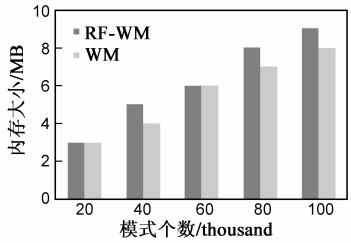


图5 RFP-WM与WM算法在不同大小模式集上占用内存对比

图5是RFP-WM算法与WM算法分别在2,4,6,8,10万条模式的模式集上所占用的内存大小对比.由图可以看出,RFP-WM算法在2,6万条模式的模式集上占用的内存与WM算法相同,分别是3MB和6MB;其它三个模式集,RFP-WM算法都比WM算法多用1MB的内存.

5 结论

本文中,我们提出了一个唯一指纹模型,该模型是一种过滤方法,可以大量地将大部分非关键字数据过滤掉,有效减少误报率.将其与WM算法结合,进一步提出了基于指纹的WM算法(RFP-WM).这种方法有效的减少了哈希的冲突率,极大地提高了算法的效率.从不同数据集上的实验结果可以看出,RFP-WM算法在匹配速度上都要快于原始WM算法,而且模式集越大,性能越优越;在内存的使用上,RFP-WM算法要略多于WM算法,但相对于模式集的规模,这几乎可以忽略不计.

参考文献

[1] 牟永敏,李美贵,梁琦.入侵检测系统中模式匹配算法的研究[J].电子学报,2006,34(S1):2488 – 2490.
MU Yong-min, LI Mei-gui, LIANG Qi. The survey of the pattern matching algorithm in intrusion detection system[J]. Acta Electronica Sinica, 2006, 34(S1): 2488 – 2490. (in Chinese)

[2] 张大方,张洁坤,黄昆.一种基于智能有限自动机的正则表达式匹配算法[J].电子学报,2012,40(8):1617 – 1624.
ZHANG Da-fang, ZHANG Jie-kun, HUANG Kun. A regular expression matching algorithm with smart finite automaton[J]. Acta Electronica Sinica, 2012, 40(8): 1617 – 1624. (in Chinese)

[3] Faro S, Lecroq T. A multiple sliding windows approach to speed up string matching algorithms[A]. Experimental Algorithms[C]. Berlin Heidelberg: Springer, 2012. 172 – 183.

[4] CAntone D M, FARo S M. Fast-search algorithms: New efficient variants of the Boyer-Moore pattern-matching algorithm [J]. Journal of Automata, Languages and Combinatorics, 2005, 10(5/6): 589 – 608.

[5] Huang Y, Ping L, Pan X, et al. A fast exact pattern matching algorithm for biological sequences[A]. IEEE International Conference on Biomedical Engineering and Informatics Proceedings [C]. New York: Institute of Electrical and Electronics Engineers, 2008. 8 – 12.

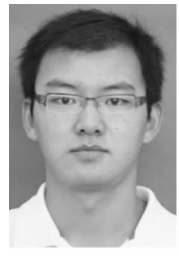
[6] Wu S, Manber U. A Fast Algorithm for Multi-pattern Searching [R]. Tucson: University of Arizona, 1994. 1 – 10.

[7] Kharbutli M, Aldwairi M, Mughrabi A. Function and data parallelization of Wu-Manber pattern matching for intrusion detection systems[J]. Network Protocols and Algorithms, 2012, 4(3): 46 – 61.

[8] Pyrgiotis T K, Kouzinopoulos C S, Margaritis K G. Parallel implementation of the Wu-Manber algorithm using the openCL framework[A]. Artificial Intelligence Applications and Innovations[C]. Berlin Heidelberg: Springer, 2012. 576 – 583.

作者简介

张宏莉 女,1973年出生,吉林榆树人,哈尔滨工业大学教授、博士生导师,主要研究方向为计算机网络信息安全、并行处理.



徐东亮(通信作者) 男,1984年出生,山东省威海人,博士研究生,主要研究方向:模式匹配,入侵检测和信息安全等.
E-mail: Ray198421@gmail.com