PARA-AC:一种基于 AC 自动机的高性能匹配算法

熊仁都[1],杨嘉佳[1],朱广宇[1],唐 球[1],隋 然[2]
(1.华北计算机系统工程研究所,北京 100083;2.中央军委后勤保障部 信息中心,北京 100842)

摘 要:原始 AC 自动机由于匹配性能低,无法满足当前大数据环境下大规模特征串实时匹配的应用需求。针对这一问题,提出一种基于多线程的多模式串匹配加速算法,称之为 PARA-AC(Parallel Aho-Corasick automaton)。该算法将待匹配字符串切割成若干字符子串以及若干切割点边界字符集,并将字符子串、切割点边界字符集输入至线程池中进行匹配,从而实现字符串的并行化加速处理。实验结果表明,与原始 AC 自动机匹配算法相比,PARA-AC 算法显著提高了匹配速度,约为原始 AC 的 13.91 倍。

关键词:多模式串匹配;AC 自动机;多线程;并行化
中图分类号:TP391.1 **文献标识码**:A DOI:10.16157/j.issn.0258-7998.200096

中文引用格式:熊仁都,杨嘉佳,朱广宇,等. PARA-AC:一种基于 AC 自动机的高性能匹配算法[J].电子技术应用,2020,46(11):87-90,95.
英文引用格式:Xiong Rendu,Yang Jiajia,Zhu Guangyu,et al. PARA-AC:a high performance matching algorithm based on Aho-Corasick automaton[J]. Application of Electronic Technique,2020,46(11):87-90,95.

PARA-AC:a high performance matching algorithm based on Aho-Corasick automaton

Xiong Rendu[1],Yang Jiajia[1],Zhu Guangyu[1],Tang Qiu[1],Sui Ran[2]
(1.North China Institute of Computer Systems Engineering,Beijing 100083,China;
2.Information Center,Logistics Support Department,CMC,Beijing 100842,China)

Abstract:Due to low matching performance,the original AC automaton cannot meet the application requirements of real-time large-scale feature string matching under the current big data environment. To solve this problem,a accelerated multi-mode string matching algorithm based on multi-threading is proposed,which is called PARA-AC. The algorithm cuts the string to be matched into several character substrings and a number of boundary character sets. Then these character substrings and boundary character sets to be input to the pool of threads for matching. The experimental results show that the performance of the PARA-AC algorithm is 13.91 times better than that of the original AC matching algorithm.

Key words:multi-mode string matching;Aho-Corasick automaton;multi-threading;parallelization

0 引言

模式串匹配的作用是给定一组特定的字符串集合 $S=\{s_1,s_2,\cdots,s_m\}$,对于任意一个字符串 $T=t_1t_2\cdots t_n$,找出 S 中所有字符串在 T 中出现的位置[1]。基于 Aho-Corasick(AC)自动机的模式串匹配算法在当前的串匹配算法中占据着重要地位,它以 Trie 树为基础,通过 fail 指针来实现状态匹配失效的过程跳转,保持了较为稳定的匹配性能。因此,基于 AC 自动机的串匹配算法在字符串搜索、生物特征识别、网络安全等领域有着广泛的应用。

截至目前,已经提出了各式各样的 AC 自动机优化算法,包括基于前缀识别的自动机算法 AC[2]、基于状态转移表加速的算法[3]、利用字符跳跃的加速匹配算法[4]。但是,这些算法的处理过程本质上为串行匹配,因而匹配性能较低,无法满足大数据环境下的高性能数据实时处理要求。此外,直接对 AC 自动机进行简单并行化易出现假阴性错误。

因此,针对原始 AC 自动机匹配速度较慢的问题,本文提出了一种基于多线程并行化的多模式串加速匹配算法。通过将文本分割成若干文本段进行多线程加速匹配,同时为保证算法功能的正确性,提取出切割点附近的边界字符形成切割点边界字符集进行处理。理论分析与实验结果表明,此算法与原始 AC 自动机的性能加速比达到 8.38,性能提高接近 1 个数量级,非常适合于大规模数据的实时处理。

1 相关工作

基于 AC 的串匹配算法是计算机科学领域的研究热点之一。近些年来,相关工作主要集中在解决 AC 的空间开销以及性能匹配方面。

AHO A V 等人[3]为使压缩后的状态转移维持在 $O(1)$ 时间复杂度,将状态转移表的行存储于数组中。同时为

保证自动机功能的正确性，存储的非空转移表须不重叠，并通过记录行位置和行偏移量以获得较好的匹配效果。但是，此方法在进行字符串的匹配过程中，要求系统具有较高的内存。

文献[4]提出一种改进的多模式匹配算法 ACTE。通过构建字符串跳跃表和哈希表，使在不发生漏检的情况下，最大移动距离为最短模式串长度加 3。实验结果表明，与 AC 算法相比，ACTE 算法消耗更少时间。

为解决自动机因存储空间引发的性能匹配瓶颈，DENCKER P 等[5]利用状态转移表存储下一跳状态、非空转移边、输入字符，达到减小存储空间的目的。但此方法只有在边相对稀疏的条件下才能取得显著效果。

陈永杰[6]利用 Trie 树转化成双数组形式的思想，提出快速多模式匹配算法。实验结果表明，提出的算法在性能上优于原始的 AC 自动机匹配算法。

文献[7]针对内存消耗大的问题，基于字符集规约的原理提出了新的算法——FilterFA。其主要思想为将原字符集压缩为多个像字符集，从而达到减小空间复杂度的目的。实验结果表明，算法消耗的存储空间仅为原始 AC 算法的 3%左右。

熊刚等人[8]研究发现自动机节点受数据流的访问影响，提出了基于数据访问特征的混合自动机构建算法 HybridFA。通过结合访问层次和访问频率，保证了算法的匹配速度。实验结果表明，此算法的存储空间为高级 AC 自动机的 5%，同时还具有较好的匹配速度。

尽管基于 AC 的串匹配算法研究得到了较大的进展，但这些算法在大数据环境下进行大规模模式串的匹配时，仍然无法满足单链路条件下的高性能匹配要求。因此，研究高性能的 AC 自动机匹配算法意义重大。

2 PARA-AC 匹配算法设计

2.1 原始 AC 自动机

AC 自动机匹配主要包括 3 个过程[4]：字典树构建、fail 指针的构建、字符串匹配。具体地，利用模式串中的字符进行树节点的创建；构建完字典树后，利用广度优先搜索算法计算 fail 指针，并将各父子节点按照预设的规定进行指向连接；在字符串匹配过程中，如果从当前节点沿着树边有路径可到达目的字符串，则表示匹配当前字符，否则需沿着 fail 指针所指向的路径继续进行匹配。

原始 AC 自动机匹配算法是串行处理。在多核 CPU 的环境下，无法充分利用计算资源进行匹配性能的加速。因此本文利用多核技术对原始 AC 自动机进行加速优化。

2.2 PARA-AC 算法原理及处理流程

RASOOL A[9]提出了一种针对 KMP 单模式匹配算法的并行加速方案，在多核的计算环境下取得了相对原 KMP 单模式匹配算法 4.3×的加速效果。但是该算法在实现过程中需要两次访问内存，导致访问时间增加，因而还有较大的匹配性能优化空间。

受该算法的影响，本文基于多线程技术对多模式串匹配 AC 自动机进行优化改进：通过字符串切割成若干子串，然后在各个子串上分别执行 AC 匹配，从而提高 AC 匹配算法的匹配效率。同时，为防止对字符串的切割导致假阴性匹配的出现，通过分析可得，假阴性匹配的出现只会存在于分割点附近的特定区域，且此区域与模式串的长度有直接关系。因此，对这部分区域单独处理，可防止假阴性匹配的出现，保证 AC 自动机匹配功能的正确性。通过改进 AC 自动机的匹配过程，减少访存次数，减少时间开销。

PARA-AC 匹配算法分成 2 个阶段：(1)AC 自动机构建与加载，(2)字符串切割与匹配。

2.2.1 AC 自动机构建与加载

根据 AC 匹配算法原理，首先利用模式串构建字典树，然后使用 BFS 算法搜索每个字符的 fail 指针，当 fail 指针搜索构造完成后便形成了模式串对应的 AC 自动机。由于在相同的模式串所构造的 AC 自动机一致，因此将其作为公共资源，供各线程调用加载，从而减少构建的时间和空间需求。设模式串为 lea、ea，模式串的最大长度为 3，其对应的 AC 自动机如图 1 所示。

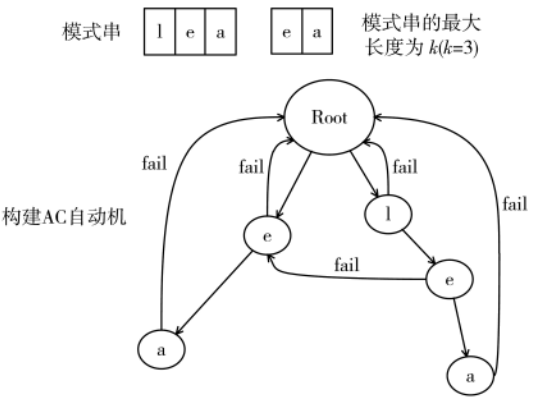

图 1 构建 AC 自动机

2.2.2 字符串切割与匹配

为方便算法描述，在后文约定以下记号：S 表示模式串集合，T 表示文本，$m=\max|p|$ 表示最长模式串长度，$n=|T|$ 表示待匹配文本长度，p 表示将文本切分成的段数。

上一小节中，假设已创建的线程数为 $p+1$ 个。在 AC 自动机的基础上，增加并行化处理以加速文本的匹配，具体流程为：(1)将长文本切割成 p 个文本段，每个文本段包含为长 n/p 的字符串。(2)确定边界字符串集合。通过分析可得，边界字符串所处边界的半长应该为 $m-1$，因此从分割点往左右各延伸 $m-1$ 个字符，形成长为 $(m-1)\times2$ 的切割点边界字符串。对于 p 段文本，总共有 $p-1$ 个边界字符串。通过提取这些边界字符串形成边界字符集，并将边界字符集分配给 1 个独立的线程进行处理。

(3)利用 $p+1$ 个线程对文本段以及边界字符集进行匹配,得出最终的匹配结果。

以 $p=4$ 为例,算法匹配过程示意图如图 2 所示。将待匹配字符串等分为 4 段,每段分配给特定的线程进行匹配。每个分割点向两边延伸 2 个字符形成 3 个长度为 4 的边界字符串,交由线程 5 处理。对于文本 T=anappleadayaanappleadaya,其被切割成 4 段,分别为 anappl、eadaya、anappl、eadaya。边界字符集为{plea,yaan,plea}。线程 1 处理 anappl,线程 2 处理 eadaya,线程 3 处理 anappl,线程 4 处理 eadaya,线程 5 处理边界字符集{plea,yaan,plea}。

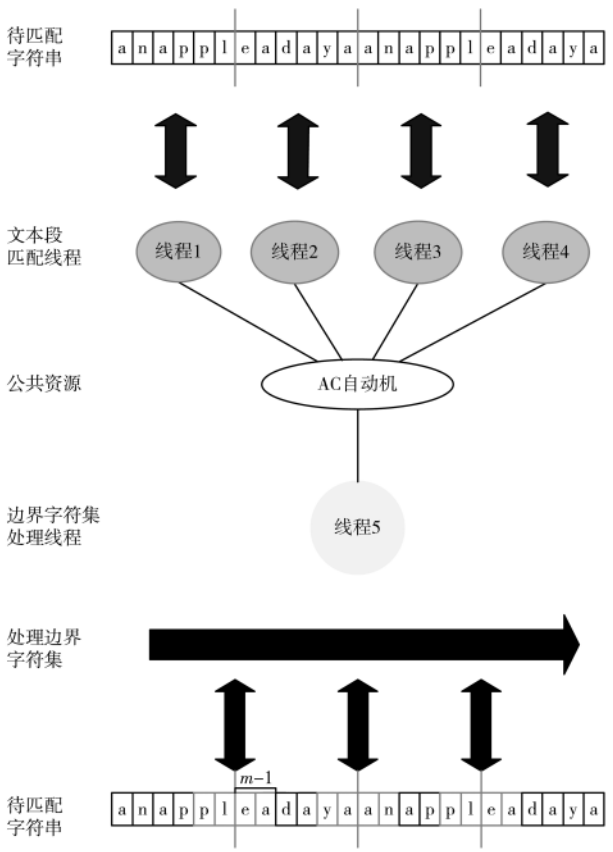

图 2 PARA-AC 算法匹配过程

2.2.3 算法伪代码

PARA-AC 算法的伪代码包括初始化函数 Init()、并行处理函数 Parallel_Search 两部分。PARA-AC 预处理如算法 1 所示,其中 keywords 为构建的模式串集合、p 为切割的文本段数。文本的匹配过程如算法 2 所述,其中 T 为待匹配的文本,Boundary_Set 为待处理的边界字符集,进程 P^1、P^2,…,P^p 用于处理切割的文本段,进程 P^{p+1} 用于处理边界字符集,Result 为命中模式串的次数。

(1)算法 1:初始化加载 AC 自动机

算法 1:Init (keywords,p)

输入:keywords,p;

输出:无。

① BuildACAutomation(keywords)
//根据 keywords 构建 AC 自动机

② Process(P^1,P^2,…,P^p,P^{p+1}) //初始化 $p+1$ 个线程

③ LoadACToProcess() //加载 AC 至各线程

(2)算法 2:文本匹配过程

算法 2:ParallelACSearch(T)

输入:T //待匹配的文本

输出:Result //命中模式串的次数

① For all processors Pi where $1\leqslant i\leqslant p$

② do {

③ AC_Search($(i-1)*n/p$,$(i)*n/p$);
//匹配切割的文本段

④ } end for

⑤ For processor Pi where $i=p+1$

⑥ do{

⑦ AC_Search(Boundary_Set);
//匹配边界字符集 Boundary_Set

⑧ }end for

⑨ Result=CombineAllResult()
//合并所有线程的匹配结果

2.2.4 算法时间复杂度分析

原始 AC 自动机在处理长度为 n 的文本所消耗的时间[1]为 $O(n)$。基于多线程的 PARA-AC 算法在真实环境下,需要考虑线程调度时间 cp,因为其占据了较大的时间比,而在实际的文本处理中,边界字符集的字符数远小于文本段的字符数,即:

$$(p-1)\times 2\times(m-1)<<n/p \tag{1}$$

所以在理想情况下($p+1$ 个线程都能并行执行),PARA-AC 时间复杂度为:$O(\frac{n}{p}+cp)$。因此,理论上 PARA-AC 与原始的 AC 自动机的加速比约为 $n/(\frac{n}{p}+cp)$。

3 实验

本文主要从模式串的长度、待测试的文本大小、线程数等方面对 PARA-AC 算法的匹配性能进行实验测算。

3.1 实验环境和数据

实验环境采用的服务器 CPU:Intel® Xeon® CPU E5-2640 v3@2.60 GHz(8 核 16 线程),采用的操作系统为 Ubuntu 14.04.5 LTS,算法实现的语言及版本为 Python 3.5。

实验数据集为从真实网络环境中爬取的 1 GB 网页内容。利用随机算法随机抽取出大小为 128 KB、256 KB、512 KB、1 MB、4 MB、20 MB、80 MB 的测试文件。

本文主要根据不同大小的测试文件(128 KB、256 KB、512 KB、1 MB、4 MB、20 MB、80 MB)、不同长度的模式串(4、16、64、256、1024)以及不同数量的线程上对算法进行测试。

3.2 实验结果与分析

为了探究 PARA-AC 算法在不同条件下的匹配性

能，选取文件大小、模式串和线程数作为实验因素进行测量评估。通过实验结果可以知道，当处理的文本大小为 80 MB、线程数为 40 时(略超过 16 核 32 线程 CPU 的线程数)，算法的匹配吞吐率达到 31.9 MB/s。相较原始 AC 自动机取得了 13.91×的加速比。

首先为方便起见，选取了有代表性的 4 组数据进行绘图显示。图 3 描述了在处理较大的文件时算法的性能与线程数的关系，图 4 则描述了处理小文件时算法的耗时变化。

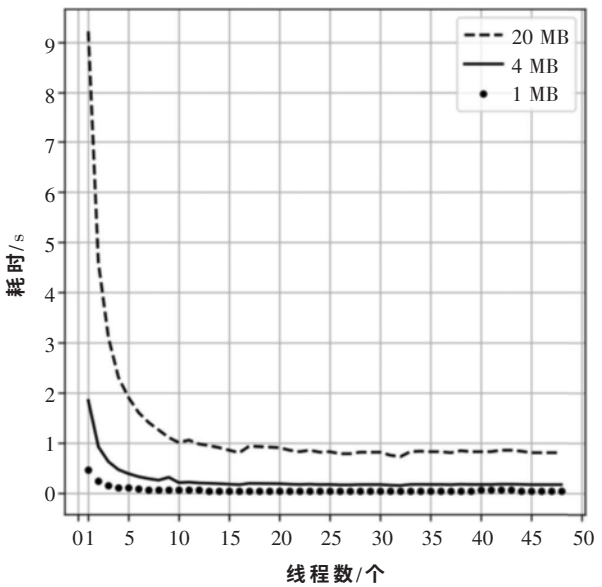

图 3 处理大文件时算法耗时变化

在处理大文本时，随着线程数的增加，匹配耗时逐渐下降，并在 CPU 核数附近逐渐趋于饱和。如图 3 所示，当处理较小文本时，随着线程数开始增加，匹配处理耗时开始会下降，而当线程数达到一定阈值时耗时反而上升。原因如下：

(1)随着线程数增加，因切割产生的连接字符串增多，增加了系统的处理压力；

(2)过多的线程会增加系统调度时间。

假设处理连接部字符串的耗时为 x，线程调度的耗时为 y，文本匹配耗时为 z，那么总的处理时间约为 x、y 和 z 之和。当线程数开始增加时，文本匹配耗时 z 的减少量大于 x、y 耗时的增加量，因此总的处理时间呈下降

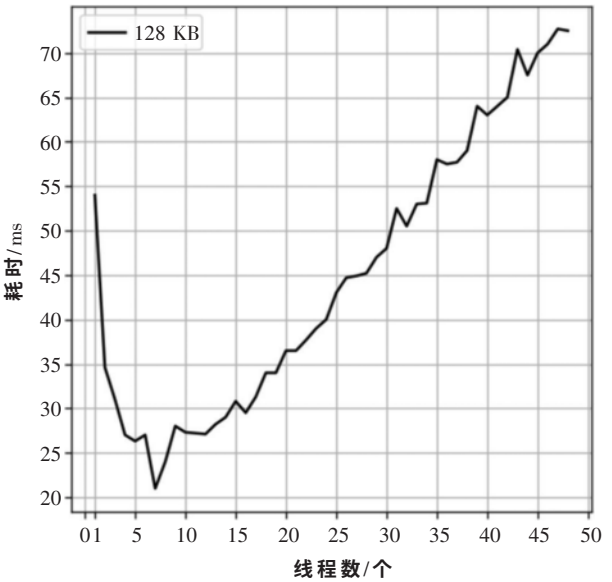

图 4 处理小文件时算法耗时变化

趋势。而当线程数量达到一定阈值时，x、y 耗时的增加量大于文本匹配耗时 z 的减少量，因此总的处理时间呈上升趋势。

另外当文本越小，x、y 耗时相对于 z 占总时间的比例越大，使得曲线极值点左移。在处理小文本时，几乎没有下降的阶段，主要因为 z 的减少量太小，对总时间的变化影响较小。

通过以上分析可知，文本量较大时，可以最完全发挥并行 AC 算法的潜力，选取进程数为 40 时，模式串的长度为(4、16、64、256)时的算法在 80 MB、20 MB、4 MB、1 MB 以及 128 KB 文件上的性能测试数据如表 1 所示。从表 1 可以看出，在大样本测试数据上(80 MB)，PARA-AC 算法的平均加速比 (计算不同模式串长度的加速比取平均)能达到原始匹配算法的 13.91×。而在小文本(128 KB)上，采用多段匹配的 PARA-AC 算法的匹配速度低于原始的 AC 匹配算法，主要是因为：在 PARA-AC 算法处理小文本的过程中，线程的调度以及连接部分处理耗时带来的性能下降量远大于因并行匹配而带来的性能增加量，导致其匹配性能反而不如原始 AC 自动机。虽然在图 4 中，线程数为 8 时，并行性能高于串行性能，但当文本进一步变小时，该极值点会进一步左移直到整个曲线为单调递增。

表 1 PARA-AC(并行)与原始 AC 匹配算法(串行)的耗时对比

(单位：s)

模式串的最大长度	80 MB		20 MB		4 MB		1 MB		128 KB	
	串行	并行	串行	并行	串行	并行	串行	并行	串行	并行
4	36.832	2.611	9.421	0.774	1.832	0.209	0.412	0.123	0.053	0.065
16	32.939	2.400	8.944	0.743	1.711	0.197	0.397	0.132	0.051	0.064
64	36.677	2.543	9.305	0.774	1.789	0.204	0.411	0.121	0.054	0.064
256	34.129	2.462	9.214	0.762	1.751	0.201	0.408	0.106	0.053	0.063

(下转第 95 页)

术应用,2019,45(6):28-36.
[4] LI L,WANG X,SONG J.Fuel consumption optimization for smart hybrid electric vehicle during a car-following process[J].Mechanical Systems and Signal Processing,2017,87:17-29.
[5] 龙克俊.增程式电动汽车动力系统参数匹配与效率优化控制研究[D].重庆:重庆理工大学,2018.
[6] WANG Y,LU E,LU H,et al.Comprehensive design and optimization of an electric vehicle powertrain equipped with a two-speed dual-clutch transmission[J].Advances in Mechanical Engineering,2017,9(1):1-13.
[7] WANG X,LI L,HE K,et al.Dual-loop self-learning fuzzy control for AMT gear engagement:design and experiment[J].IEEE Transactions on Fuzzy Systems,2018,26(4):1813-1822.
[8] WANG X,LI L,HE K,et al.Position and force switching control for gear engagement of automated manual transmission gear-shift process[J].Journal of Dynamic Systems Measurement Control,2018,140(8):081010.
[9] YUE H,ZHU C,GAO B.Fork-less two-speed I-AMT with overrunning clutch for light electric vehicle[J].Mechanism and Machine Theory,2018,130:157-169.
[10] TIAN Y,RUAN J,ZHANG N,et al.Modelling and control of a novel two-speed transmission for electric vehicles[J].Mechanism and Machine Theory,2018,127:13-32.
[11] 叶文,李翔晟,单外平.基于改进布谷鸟搜索优化神经网络的AMT换挡电机控制[J].计算机应用,2018,38(S2):67-71.
[12] 胡宇辉,乐新宇,吴洪振,等.混合动力汽车的转速转矩双同步换挡控制[J].汽车工程,2018,40(9):1076-1082,1095.
[13] 程靖.湿式DCT换挡过程同步器控制策略研究[D].长春:吉林大学,2018.
[14] MO W,WALKER P D,FANG Y,et al.A novel shift control concept for multi-speed electric vehicles[J].Mechanical Systems and Signal Processing,2018,112:171-193.
[15] LIANG J,YANG H,WU J,et al.Power-on shifting in dual input clutchless power-shifting transmission for electric vehicles[J].Mechanism and Machine Theory,2018,121:487-501.
[16] LI L,HE K,WANG X,et al.Sensor fault-tolerant control for gear-shifting engaging process of automated manual transmission[J].Mechanical Systems and Signal Processing,2018,99:790-804.

(收稿日期:2020-02-21)

作者简介:
汪秋婷(1982-),女,博士,讲师,主要研究方向:数字信号处理、锂离子电池能量管理。
戚伟(1984-),男,博士,讲师,主要研究方向:电子电路、合同能量管理。
沃奇中(1974-),男,硕士,研究员,主要研究方向:电子检测系统。

(上接第90页)

4 结论

本文基于并行加速的思想,对原始AC自动机的串行匹配过程进行改进。利用多线程技术对切割后的文本段进行处理以获得较高的性能加速比。实验结果表明,在实验环境下,改进后的算法在匹配大文本(取80 MB为例)时,可获得相比于原始AC自动机13.91×的加速比,当待匹配文本较小时,使用过多线程,该算法性能提升不明显。下一步将高级AC自动机的字典树压缩技术和并行加速思路相结合,进一步提升算法的时间与空间性能,并研究如何将算法应用到基于深度包检测的入侵检测系统中。

参考文献

[1] AHO A V,CORASICK M J.Efficient string matching:an aid to bibliographic search[J].Communications of ACM,1975,18(6):333-340.
[2] CURINO C,JONES E,Zhang Yang,et al.Schism:a workload-driven approach to database replication and partitioning[C].VLDB,2010:48-57.
[3] AHO A V,SETHI R,ULLMAN J D.Compilers:principles,techniques,and tools[M].Addison-Wesley Publishing,2006.
[4] 刘春晖,黄宇,宋琦.一种改进的AC多模式匹配算法[J].计算机工程,2015,41(10):280-285.
[5] DENCKER P,DORRE K,HEUFT J.Optimization of parser tables for portable compilers[J].ACM Transactions on Programming Languages and Systems,1984,6(4):546-572.
[6] 陈永杰,吾守尔·斯拉木,于清.一种基于Aho-Corasick算法改进的多模式匹配算法[J].现代电子技术,2019,42(4):89-93.
[7] 张萍,何慧敏,张春燕,等.FilterFA:一种基于字符集规约的模式串匹配算法[J].通信学报,2016,37(12):103-114.
[8] 熊刚,何慧敏,于静,等.HybridFA:一种基于统计的AC自动机空间优化技术[J].通信学报,2015,36(7):31-39.
[9] RASOOL A,KHARE N.Parallelization of KMP string matching algorithm on different SIMD architectures:multi-Core and GPGPU's[J].International Journal of Computer Applications,2012,49(11):26-28.

(收稿日期:2020-02-14)

作者简介:
熊仁都(1994-),男,硕士研究生,主要研究方向:软件工程。
杨嘉佳(1988-),男,博士,工程师,主要研究方向:社交网络、机器学习。
唐球(1985-),男,博士,工程师,主要研究方向:社交舆情、大数据存储、数据挖掘。

版权声明