doi :10.3969/j.issn.1671-1122.2011.09.039

基于 CUDA 的多模式匹配技术

张光斌[1]，谢维盛[1]，吴鸿伟[2]

（1. 厦门市美亚柏科信息股份有限公司，福建厦门 361008；2. 厦门大学，福建厦门 361005）

摘 要：文章以经典的多模式匹配算法—AC 算法为例，通过对 CUDA 特性的分析，提出了基于 CUDA 的并行模型，设计了适合 CUDA 并行技术的 AC 匹配算法。实验结果表明，基于 CUDA 的 AC 匹配算法较 CPU 上获得了 22 倍的加速比，有效提高了入侵检测系统的性能。

关键词：CUDA；并行技术；多模式匹配；AC 算法

中图分类号：TP393.08 **文献标识码：**A **文章编号：**1671-1122（2011）09-0126-03

Multi-pattern Matching Technology Based on CUDA

ZHANG Guang-bin[1], XIE Wei-sheng[1], WU Hong-wei[2]

(1.*Xiamen Meiya Pico Information Co.,Ltd., Xiamen Fujian* 361008*, China;*
2. *Xiamen University, Xiamen Fujian* 361005*, China*)

Abstract: By analyzing the characteristic of CUDA, a parallel module of AC matching algorithm based on CUDA is proposed. Experiment shows that the AC multi-pattern matching algorithm based on GPU gets 22 times speedup ratio than it based on CPU, and it improves the performance of intrusion detection system effectively.

Key words: CUDA; parallel technology; multi-pattern matching; AC algorithm

0 引言

多模式匹配技术是网络入侵检测系统（Network Intrusion Detection System，NIDS）的计算核心，主要用于从网络包中搜索出已知网络攻击的特征数据，检测到利用合法的包头穿过防火墙的网络包攻击。对于现有的 NIDS，以 Snort 为例[1]，模式匹配过程在整个系统中占用的时间约为 40 %-70 %，属于计算密集型的应用。

随着网络规模与传输速度急速增长，如何在高速网络环境和海量的数据中检测出异常行为，成为目前网络入侵检测领域面临的一大难题。同时，随着网络攻击类型和攻击次数的急速增加，网络入侵检测系统的规则库不断膨胀，以此带来的是规则定义的复杂性也越来越高，这就直接导致了网络入侵检测系统检测复杂性也迅速提高，进而对其处理能力和执行效率提出了更高的要求，因此提高模式匹配的速度对改进 NIDS 的性能有重要意义。

加速模式匹配的早期研究致力于设计更高性能的软件算法，然而基于软件的方法已经很难实现匹配效率上的大幅提升[2]，不能满足现实需求。为了达到更快的匹配速度，一些研究者提出了基于专有硬件的解决方案：如基于 FPGA、TCAM 的方案，但是其价格昂贵且稳定性、扩展性不够好，应用范围有限。

随着图形处理器（Graphic Processing Unit，GPU）的高速发展，特别在近年 NVIDIA 公司推出了基于 G80 以上系列的 GPU 的统一计算设备架构（Compute Unified Device Architecture，CUDA），其并行线程执行模型和线程同步技术为大规模并行计算提供了一种崭新的研究视角与实现解决方案[3-5]。通过把 NIDS 的模式匹配计算任务移植到 GPU，可以有效提升模式匹配的速度，如 PixelSnort[6] 的解决方案。最好情况下，PixelSnort 的模式匹配部分的性能比 Snort 要提高 85 %。

本文分析了 CUDA 的特性，以 Aho-Corasick（AC）匹配算法为例分析了其在 CUDA 架构上实现存在的问题，提出了基于 CUDA 的解决方案，并对实验结果进行了对比分析。

1 CUDA 介绍

CUDA 是用于 GPU 计算的开发环境[7]，它是一个全新的软硬件架构，可以将 GPU 视为一个并行数据计算的设备，对所进行的计算进行分配和管理，其巧妙的线程体系设计、存储体系和线程同步技术是 CUDA 成功的重要因素。

收稿时间：2011-07-15

作者简介：张光斌（1982-），男，福建，硕士，主要研究方向：网络安全；谢维盛（1982-），男，福建，硕士，主要研究方向：网络安全；吴鸿伟（1976-），男，福建，博士研究生，主要研究方向：网络安全。

CUDA 只对 ANSI C 进行了最小的必要扩展，以实现其关键特性——线程按照两个层次进行组织、共享存储器和栅栏同步。这些关键特性使得 CUDA 拥有了两个层次的并行：线程级并行实现的细粒度数据并行，和任务级并行实现的粗粒度并行。

CUDA 采用了 SIMT（Single Instruction Multiple Thread，单指令多线程）执行模型，也就是说在一个多处理器上的每个线程在任何时间执行的指令都是相同的。在 CUDA 中由多个线程组成线程块，多个线程块进一步组成线程格。SIMT 中的每个线程的寄存器都是私有的，线程之间只能通过共享存储器和同步机制进行通信。在 SIMT 执行模型中如果需要控制单个线程的行为，必须使用分支，这会大大的降低效率。

CUDA 设备拥有多个独立的存储空间，其中包括：全局存储器、本地存储器、共享存储器、常量存储器、纹理存储器和寄存器。对于同一个应用程序启动的内核而言，全局、固定和纹理存储器空间都是持久的。而且除共享存储器外，常量存储器，纹理存储器和全局存储器都可与主机内存通信。由于共享存储器是片上内存，所以存取速度很快，纹理存储器和常量存储器采用高速缓存技术加快相应存储器空间的读写速度。

表1 存储器属性

存储器类型	高速缓存	存取方式	访问速度
寄存器	N/A	R/W	快
全局存储器	No	R/W	慢
本地存储器	No	R/W	慢
共享存储器	N/A	R/W	快
常量存储器	Yes	R	快
纹理存储器	Yes	R	快

2 基于 CUDA 的 AC 算法实现

2.1 AC算法

AC 算法是一种经典的基于有穷状态自动机 FSA（Finite State Automata）的多模式匹配算法。AC 匹配过程主要分为两个阶段：在进行匹配之前，先对模式串集合进行预处理，构建树型有穷状态自动机；在匹配过程中，依据 FSA 对文本串 T 扫描找出所有匹配模式串。无论是在最好情况还是最坏情况下，AC 算法模式匹配的时间复杂度都是 O（n），其性能几乎不受模式集合大小、模式长度分布等的影响，鲁棒性和适应性较好。

预处理过程生成三个转移函数：转移函数，失效函数和输出函数。转移函数表明在当前状态下读入下一个待比较文本的字符后到达的下一个状态。失效函数用来表明在当前状态下，当读入的字符不匹配时应该转移到的下一个状态。输出函数的作用是输出匹配到的模式。

AC 算法的匹配过程是：从初始状态 0 出发，每次取出文本串中的一个字符，由当前的状态和取出的字符，根据转移函数或失效函数进入下一状态。当某个状态的输出函数不为空时，表明出现了匹配的模式。

2.2 AC算法的并行化设计

根据上一节的分析，在一次匹配任务中，由于模式串不会改变，FSA 只需要构造一次就可以在匹配过程中重复使用，构造 FSA 并不是 AC 匹配算法的性能瓶颈，因此本文主要研究 AC 算法中匹配阶段的并行设计。

根据 CUDA 的 SIMT 执行模型特点，本文为匹配过程设计了以下两种并行模型：

1）模式并形：将多模式分解为多个小的模式，每个匹配线程完成其中一部分模式的匹配。由于模式串的规模很难达到 GPU 满载时的线程规模，在通常情况下无法充分发挥 GPU 的计算能力。并且，分解后的模式之间难以保持相等的规模，将导致 GPU 各线程的负载不均衡，也将导致性能的下降。因此，在多模式规模较小、以及无法均分的情况下，不适合采用该模型。

2）任务数据并行：将需要匹配的任务数据分解为多个小的任务数据，每个匹配线程完成其中一部分任务数据的匹配。该模型适用于任务数据量大，模式串较少的应用类型。通常情况，任务数据规模都能达到甚至超过 GPU 满载时的线程规模，尤其是对于 NIDS 的海量数据，能够保证 GPU 在匹配过程中的满载工作。同时，由 AC 算法可知，任务数据易于实现均分，可以保证每个 GPU 线程的负载均衡。因此，入侵检测系统的匹配过程更适合于使用该模型。

通过对两种并行模型的对比分析，入侵检测系统的匹配过程更适合于采用任务数据并行的模型。但采用该模型，需要注意数据段交界处漏检和重复检测匹配信息的问题。本文采用冗余数据的方法，将两线程交界处的数据同时分配给两个线程，冗余数据的长度不得小于最长的模式串的长度。冗余数据成功匹配的，只输出一次。任务数据并行原理如图 1 所示。

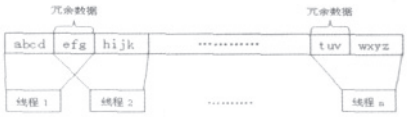

图1 任务数据并行原理

由于冗余数据在两个线程均要匹配计算，势必影响匹配的效率。因此应该尽可能提高每个线程的任务数据量，以降低冗余度。

根据任务数据并行模型，AC 匹配算法在 CUDA 上的匹配流程如图 2 所示。

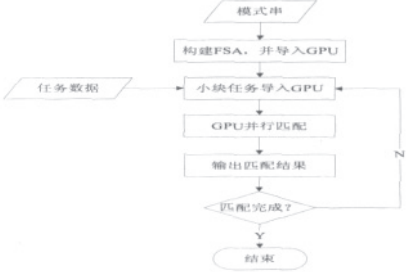

图2 基于CUDA的AC匹配算法流程

2.3 CUDA存储器选择

AC 匹配算法主要包含了基于 FSA 的查找表和任务数据两个数据源，整个匹配过程存在大量的数据访问。由于硬件特性，CUDA 不同存储器的存储周期相差上百倍，因此，选择合适的存储器对匹配效率至关重要。

根据表 1 存储器的属性，优先选择存储速度快的共享存储器、常量存储器和纹理存储器。由于共享存储器和常量存储器的存储空间有限，即使是最新的 Fermi 架构 GPU，也只有 64KB，无法满足查找表对空间的需求。而对于全局存储器，其存储空间很大，能够满足表的空间需求，但 AC 算法对查找表的访问位置是随机的，会造成严重的效率问题。纹理存储器的空间也能够满足表的空间需求，而且可以通过缓存加速，访问速度快。因此，选择纹理存储器来保存查找表。

任务数据所需要的空间很大，但并行设计后，可以通过控制每个线程的任务量，分批完成匹配的方法，以此来减少对任务数据空间的需求。共享存储器对于同一个 Block 的所有线程，共享 64KB 存储空间，可以通过控制每个 Block 中的线程数量以及每个线程的任务量，充分利用 64KB 的共享存储器，来达到最快的访问速度。

2.4 CUDA存储映射

许多研究者提出的基于 GPU 的模式匹配解决方案，在匹配过程比 CPU 的方案获得了 10 倍以上的性能提升，但在设备之间的数据传输上产生了新的性能瓶颈，如果加上 GPU 显存与系统内存之间的数据拷贝时间，基于 GPU 的匹配效率有时反而比基于 CPU 的匹配效率还低[4, 5]。

针对计算能力 1.0 以上的 GPU，CUDA 提供了一种存储映射方法。它可以将系统内存映射到设备地址空间，设备和宿主共用同一块内存空间。通过这种方法，不需要在设备中分配单独的显存空间以及系统内存和 GPU 显存间数据的拷贝，数据在需要时被内核隐式传输。使用存储映射时，由于内存数据是共享的，访问时必须同步。

3 实验结果分析

硬件环境：使用的 CPU 为 AMD Phenom(tm) II X4 945 Processor 3.01GHz，4G 内存，GPU 为 NVIDIA GTX 480。

软件环境：使用 Windows XP 操作系统，CUDA 3.2 版本的开发工具，Visual Studio 2005 开发环境。

本文主要测试 CPU 与 GPU 在不同情况下的匹配速度，以及使用 CUDA 存储映射方法对整体性能的影响。其中，4/16M 表示模式串长度为 4 字节，匹配数据长度为 16M，其余类推。实验结果如表 2、表 3 所示。

由表 2 的对比数据可以看出，由于 16M 的任务量还未使 GPU 达到满载工作状态，而 64M 时已经可以使 GPU 满载工作，因此任务量为 16M 时，GPU 与 CPU 在匹配阶段的加速比要低于 64M 时任务量的加速比。在未使用存储映射时，匹配阶段的加速比最大达到 32 倍，使用存储映射时，匹配阶段的加速比也达到 22 倍。

表2 CPU和GPU匹配时间比较（单位：ms）

测试平台	4/16M	8/16M	8/64M
CPU	7.1529	7.1973	28.3532
GPU（未使用存储映射）	0.4864	0.4931	0.8947
GPU（使用存储映射）	0.6852	0.6977	1.2836

表3 设备间数据拷贝时间（单位：ms）

时间	16M	64M
数据上传	6.9625	28.8163
数据下载	12.4681	49.6453

匹配阶段的加速比并未考虑设备间数据拷贝对性能的影响，由表 3 的测试结果可以看出，当未使用存储映射时，数据传输的时间甚至达到 CPU 匹配时间的 3 倍。虽然在匹配阶段获得一个很好的加速比，但整体性能反而下降。而使用存储映射的方案，它不需要在设备间拷贝数据，不存在这方面的时间消耗，整体性能最优。

由以上测试结果可以看出，与基于 CPU 的方法相比，GPU 的实现方案使 AC 匹配算法的匹配速度提升了 22 倍。基于 GPU 的应用中，关键是设计合理的并行模型、优化算法流程，同时考虑对存储器访问的优化、存储映射、GPU 满载以及线程的负载均衡等。

4 结束语

本文通过分析 CUDA 架构的特点以及 AC 算法原理，设计了适合于 CUDA 的 AC 算法并行模型和匹配流程，同时优化了数据的存储访问，使 AC 匹配算法在 GPU 下获得了 22 倍的加速比。为了获得更高的加速比，基于多个 GPU 的 AC 算法设计将是下一步的研究重点。（责编 张岩）

参考文献：

[1] Martin Roesh Sonort. Snort-Light Weight Intrusion Detection for Networks[C]. Proceeding of LISA'99:13th Systems Adminstration corference. Washington, 1999. 299-238.

[2] 李伟男，鄂跃鹏，葛敬国等．多模式匹配算法及硬件实现 [J]. 软件学报，2006，17（12）：2403-2415.

[3] HUANGNEN-FU，HUNG HSIEN-WEI，LAI SHENG-HUANG，et. A GPU-based multiple-pattern matching algorithm for network intrusion detection systems[C]. Proceedings of the 22nd International Conference on Advanced Information Networking and Applications Workshops. Washington, DC:IEEE Computer Society, 2008. 62-67.

[4] 张庆丹，戴正华，冯圣中等．基于 GPU 的串匹配算法研究 [J]. 计算机应用，2006，26（07）：1735-1737.

[5] 唐定车，刘任任，谭建龙．基于统一计算设备架构的并行串匹配算法 [J]. 计算机应用，2009，29（06）：399-401.

[6] JACOB N, BRODLEY C. Offloading IDS computation to the GPU[C]. 22dn Annual Computer Security Applications Conference: ACSAC. Washington, DC:IEEE Press, 2006. 371-380.

[7] NVidia Corporation. NVidia CUDA Compute Unified Device Architecture programming guide, V3.2[EB/OL]. http://developer.download.nvidia.com/compute/cuda/3.0/toolkit/docs/NVIDIA_CUDA_Programming Guide.pdf, 2010-02-20/2011-06-14.