文章编号:1671-4598(2022)11-0065-06 DOI:10.16526/j.cnki.11—4762/tp.2022.11.010 中图分类号:TP393.08;TP181 文献标识码:A

基于混合模式匹配算法的网络入侵检测

周 琰,马 强
(国家气象信息中心,北京 100081)

摘要:为了提升中央处理单元(CPU)和图形处理单元(GPU)协同检测网络入侵的性能,提出了一种具有数据包有效载荷长度约束的CPU/GPU混合模式匹配算法(LHPMA);在分析CPU/GPU混合模式匹配算法(HPMA)的基础上,设计了长度约束分离算法(LBSA)对传入数据包进行提前分类;当传入数据包加载到CPU之前,LBSA根据有效载荷长度约束减少有效载荷长度的多样性;长度超过约束的数据包直接分配给CPU的预过滤缓冲区进行快速预过滤,剩余数据包则直接发送至CPU主存储器中的全匹配缓冲区,并将较短数据包直接分配给GPU进行全模式匹配,提升了CPU/GPU协同检测网络入侵的性能;实验结果表明,LHPMA的性能优于HPMA以及CPU和GPU的单独处理方法;LHPMA增强了HPMA的处理性能,充分发挥了GPU并行处理较短数据包的优势,并且LHPMA提高了网络入侵检测的吞吐量。

关键词:网络安全;模式匹配算法;有效载荷;网络入侵检测系统;图形处理单元

Network Intrusion Detection based on Hybrid Pattern Matching Algorithm

ZHOU Yan, MA Qiang
(National Meteorological Information Centre, Beijing 100081, China)

Absrtact: In order to improve the performance of central processing unit (CPU) and graphics processing unit (GPU) in detecting network intrusion, a CPU/GPU hybrid pattern matching algorithm (LHPMA) with the packet payload length constraint is proposed. Based on the analysis of the CPU/GPU hybrid pattern matching algorithm (HPMA), a length constrained separation algorithm (LBSA) is designed to classify incoming packets in advance. Before the incoming packets are loaded into the CPU, the LBSA reduces the diversity of payload length according to the payload length constraint. The packets whose length exceeds the constraint are directly allocated to the pre filtering buffer of the CPU for fast pre filtering, and the remaining packets are directly sent to the full matching buffer in the main memory of the CPU. The shorter packets are directly allocated to the GPU for full pattern matching, which improves the performance of the CPU/GPU cooperative detection of network intrusion. The experimental results show that the performance of the LHPMA is better than that of the HPMA, CPU and GPU. The LHPMA enhances the processing performance of the HPMA, which gives full play to the advantage of the GPU parallel processing of shorter packets, and the LHPMA improves the throughput of network intrusion detection.

Keywords: network security; pattern matching algorithm; payload; network intrusion detection system; GPU

0 引言

随着计算机网络入侵技术的快速发展,防火墙、用户身份验证和数据加密等传统安全技术已无法充分保障网络安全[1]。网络入侵检测系统(NIDS)作为对网络传输进行即时监视的防御机制,可以有效提高网络系统的安全性[2]。NIDS可分为两类[3]:基于异常的检测和基于模式的检测。其中,基于异常的检测将入侵者不同于正常网络的异常行为识别为恶性攻击,从而可以检测到未知和已知攻击,但检测结果假阳性率高[4];基于模式的检测通过匹配预定义的攻击模式来检测异常,具有操作简单、误报率低的优点,但无法检测新型入侵攻击模式[5]。

通常,当数据包有效载荷中识别出任何异常模式,NIDS将发出警告并拦截该数据包,从而保护网络中的正常资源[6]。当采用基于模式的检测设计NIDS时,模式匹配过程占NIDS执行时间约70%[7],设计出高效的模式匹配算法对模式识别至关重要。模式匹配消耗系统执行时间的70%,是影响NIDDS系统整体性能的关键因素。模式匹配算法可以分为单模式匹配和多模式匹配。其中,单模式匹配是在一个字符串内找到单一模式,例如,字符串查找(KMP)算法[8]和字符串匹配(BM)算法[9];多模式匹配可以同时识别一个字符串中的多个模式,例如,Wu-Manber(WM)算法[10]和Aho-Corasick匹配(AC)算法[22]。

利用硬件处理的模式匹配算法主要通过可编程门阵列(FPGA)[11]、内容可寻址存储器(CAM)[12]和特定应用集

收稿日期:2022-05-29; **修回日期**:2022-08-18。
基金项目:国家重点研发计划《全球变化及应对》重点专项基金项目(2016YFA0602101);国家气象信息中心信息网络安全与"信创"技术研发创新团队基金项目(NMIC-202011-05)。
作者简介:周 琰(1978-),女,山东日照人,大学本科,高级工程师,主要从事气象信息技术方向的研究。
通讯作者:马 强(1972-),男,江苏吴江人,大学本科,高级工程师,主要从事气象信息技术方向的研究。
引用格式:周 琰,马 强.基于混合模式匹配算法的网络入侵检测[J].计算机测量与控制,2022,30(11):65-70.

成电路（ASIC）[13]，从而实现高效匹配速度。然而相比硬件实现算法相比，在通用处理器（GPP）[14]上通过软件实现算法具有成本低、可扩展性强的优点。目前，单独利用中央处理器（CPU）运算能力无法满足高速网络条件下模式识别速度，现有研究侧重于在软件平台部署具有更高并行处理能力的图形处理单元（GPU）来实现模式匹配算法[15]。尽管 GPU 比 CPU 具有更强的处理能力，但基于 GPU 的模式匹配算法在内核启动和数据传输方面具有延迟，导致整体性能受到限制[16]。此外，CPU 所支持的单指令多数据（SIMD）操作可用于加速模式匹配。因此，在 CPU 和 GPU 之间设计高效的协同模式匹配算法已成为研究热点。文献［17］提出了基于抛物线 Radon 变换的 CPU/GPU 协同模式匹配算法，通过 CPU 多线程与多个 GPU 协同并行提高网络流量检测的吞吐量。文献［18］利用基于队列的混合调度策略设计了 CPU/GPU 协同模式匹配算法，缩短了 CPU 与 GPU 完成各自计算任务后的等待时间，加快了网络流量检测的整体延迟。文献［19］和文献［20］根据 CPU 处理后的数据包加载到 GPU 平台的移植过程，分别设计了非确定性有穷自动机（NFA）和确定性有穷自动机（DFA），优化了网络接口与 CPU 和 GPU 的并行性，从而提高了僵尸网络检测处理性能。文献［21］设计了基于 CPU/GPU 混合模式匹配算法（HPMA）的 NIDS，实现 CPU 与 GPU 任务分配平衡，但具有不同长度的输入流量数据包会导致 GPU 并行处理多线程执行时间的增加，从而导致网络流量检测的吞吐量下降。

本文提出了一种具有长度约束的 CPU/GPU 混合模式匹配算法（LHPMA）来设计 NIDS。当传入数据包加载到 CPU 时，LHPMA 根据预先设定的有效载荷长度约束对数据包进行提前分类。长度超过约束的数据包直接分配给 CPU 进行快速预过滤；否则，数据包直接分配给 GPU 进行全模式匹配。通过减少有效载荷长度的多样性，提升了 CPU/GPU 协同检测网络入侵的性能。

1 CPU/GPU 混合模式匹配算法（HPMA）

文献［21］提出的 HPMA 将网络入侵检测任务分为两部分：数据包预过滤和全模式匹配。利用 CPU 执行预过滤对“正常”和“可疑”数据包进行分类，结合 GPU 对可疑数据包进行全模式匹配。HPMA 的整体架构，如图 1 所示。

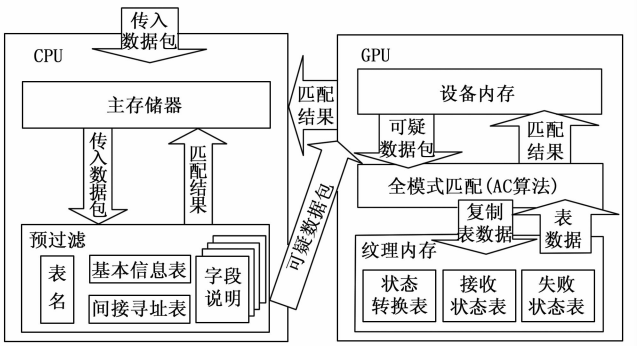

图 1 HPMA 的整体架构

首先，传入数据包发送至 CPU 的主存储器中，CPU 访问数据包并根据查找表（表名、基本信息表、间接寻址表和字段说明）对数据包进行预过滤。当表的大小足够小时，可以降低内存访问延迟并减少预过滤时间，从而提高过滤率并降低数据传输延迟。然后，将预过滤后的可疑数据包发送至 GPU 进行全模式匹配。由于 GPU 纹理内存的访问延迟比设备内存的访问延迟低得多，采用 AC 算法[22]对 GPU 纹理内存中存储的可疑数据包进行全模式匹配，并将查找表（状态转换表、接受状态表和失败状态表）复制到 GPU 纹理内存中，从而加快检测速度。最后，当全模式匹配完成后，将数据包匹配结果复制并返回 CPU 主存储器。

通过在 HPMA 中使用预过滤方法，CPU 可以进行快速的预过滤并达到较高的过滤率，从而减少 GPU 的延迟，提升网络流量检测效率。然而，在以下条件下，HPMA 性能会受到限制：①所有传入数据包最终都传送至 GPU，造成多余操作[23]；②网络流量由不同长度的数据包组成，造成 GPU 并行处理中每个线程的执行时间不一致[24]。针对限制①，文献［25］提出了基于性能的 CPU/GPU 混合模式匹配算法（CHPMA），CHPMA 首先估计系统的 CPU 和 GPU 处理能力，在运行期间，CHPMA 根据 CPU 的历史过滤率和 GPU 处理能力自动进行处理选择。针对限制②，本文提出了 LHPMA 来提升 CPU/GPU 协同检测网络入侵的性能。

2 具有长度约束的 HPMA（LHPMA）

2.1 整体架构和程序

与文献［21］中的 HPMA 不同，本文提出的 LHPMA 在 CPU 对数据包进行预过滤期间，增加了长度约束分离算法（LBSA），即传入数据包发送至 CPU 主存储器之前，通过 LBSA 处理后进入预过滤缓冲区。LBSA 主要用于检查数据包有效载荷长度并对数据包进行分类。在经过 LBSA 处理后，部分数据包存储在 CPU 主存储器中的预过滤缓冲区中，从而直接交付给 CPU 作预过滤的准备；剩余数据包则直接发送至 CPU 主存储器中的全匹配缓冲区，用于存储要由 GPU 处理的全模式匹配数据包。LHPMA 的整体架构，如图 2 所示。

当 CPU 主存储器中的预过滤缓冲区变满时，则由 CPU 访问数据包并根据查找表（表名、基本信息表、间接寻址表和字段说明）对数据包进行预过滤，预过滤的数据包也发送至全匹配缓冲区；当全匹配缓冲区变满时，缓冲区内的数据包就会转移到 GPU 设备内存中，采用 AC 算法[22]对 GPU 纹理内存中存储的可疑数据包进行全模式匹配，并将查找表（状态转换表、接受状态表和失败状态表）复制到 GPU 纹理内存中，从而加快检测速度。最后，当全模式匹配完成后，将数据包匹配结果复制并返回 CPU 主存储器。LHPMA 的整体过程，如图 3 所示。

2.2 长度约束分离算法（LBSA）

LBSA 的伪代码，如算法 1 所示。LBSA 算法根据预先设定的有效载荷长度约束对传入数据包进行分类，从而减少 GPU 并行处理的有效载荷长度多样性。给定数据包有效

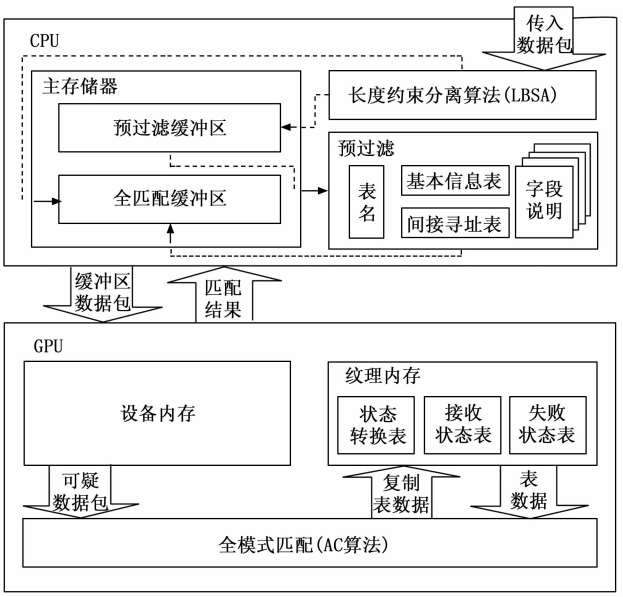

图 2　LHPMA 的整体架构

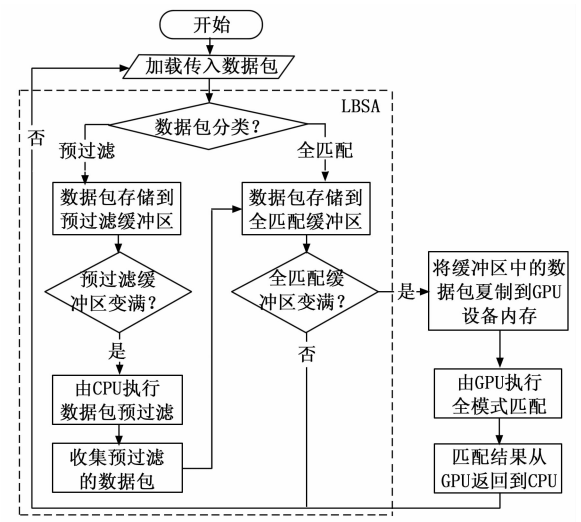

图 3　LHPMA 整体过程

载荷集 P 和长度约束 L_b，LBSA 算法可以用 L_b 确定 P 中的有效载荷。

算法 1：长度约束分离算法（LBSA）

输入：数据包有效载荷集 P 和长度约束 L_b

输出：数据包有效载荷匹配结果

1) T_{PB} ← 空；//预过滤缓冲区
2) T_{FB} ← 空；//全匹配缓冲区
3) P_f ← 空；//预过滤有效载荷
4) idx_{PB} ← 0；//预过滤缓冲区的索引
5) **for** 每个 P_i **do**
6) L_i←P_i 的有效载荷长度
7) **if** $L_i > L_b$ **then**
8) $T_{PB}[idx_{PB}]$ ←P_i；//将 P_i 存储在预过滤缓冲区中
9) idx_{PB}←idx_{PB}+1；
10) **else**
11) T_{FB}←$T_{FB} \cup P_i$；//将 P_i 存储在全匹配缓冲区中
12) **end**
13)
14) **if**T_{PB}变满时 **then**
15) P_f←预过滤的 T_{PB}；//执行预过滤算法
16) T_{FB} ←$T_{FB} \cup P_f$；//将 P_f 存储在全匹配缓冲区中
17) P_f←空；
18) T_{PB}←空；
19) idx_{PB}←0；
20) **end**
21)
22) **if** T_{FB}变满时 **then**
23) 结果←全匹配的 T_{FB}；//执行全匹配算法
24) T_{FB}←空；//匹配结果复制并返回
25) **end**
26) **end**

最初，预过滤缓冲区 T_{PB} 和全匹配缓冲区 T_{FB} 为空。同时，设置有效载荷集 P_f 用于临时存储预过滤结果，也是空（第 1～3 行）。预过滤缓冲区的索引 idx_{PB} 设置为 0（第 4 行）。对于 P 中的每个有效载荷，LBSA 测量数据包 P_i 的有效载荷长度 L_i，从而与长度约束 L_b 进行比较（第 5～6 行）。如果有效载荷长度 L_i 超过长度约束 L_b，则系统将数据包 P_i 存储在预过滤缓冲区 T_{PB} 中，并等待 CPU 执行预过滤过程（第 7～8 行）。在数据包 P_i 存储在 T_{PB} 后，预过滤缓冲区的索引 idx_{PB} 将增加（第 9 行）。否则，数据包 P_i 将存储在全匹配缓冲区 T_{FB} 中，并等待 GPU 执行全模式匹配（第 10～11 行）。

当预过滤缓冲区 T_{PB} 变满时，则缓冲区中的所有数据包由 CPU 执行预过滤得到有效载荷集 P_f，并将其存储在全匹配缓冲区 T_{FB} 中（第 14～16 行），系统清除存储在 P_f、T_{PB} 和 idx_{PB} 中的数据并继续处理下一批传入数据包（第 17～19 行）。同时，当全匹配缓冲区 T_{FB} 变满时，缓冲区中的所有数据包将复制到 GPU 设备内存中并由 GPU 执行全模式匹配过程。最后，将数据包匹配结果复制并返回 CPU 主存储器，系统会清除存储在 T_{FB} 的数据（第 22～24 行）。

在本文设计的 LBSA 算法中，较短的数据包有效载荷由 GPU 处理，从而提高整体效率。由于使用较短数据包是一种典型的攻击方式，大量的数据包可以在较短的时间内生成并发送，如果将此类数据包直接发送至 GPU，则可以减少由此类数据包引起的 CPU 预过滤的冗余操作。此外，GPU 可以对较短数据包进行较强的并行处理。同时，较长数据包由 CPU 进行预过滤而不发送至 GPU 进行全模式匹配检测，从而减轻了 GPU 处理负担并减少数据传输延迟。

3　实验分析

3.1　实验设置

本文设计的预过滤和全模式匹配分别基于 CPU 和 GPU

组成的通用并行计算架构（CUDA）平台实现多线程并行处理。实验的硬件规格，如表 1 所示。

表 1 实验的硬件规格

硬件	设备规格
CPU	英特尔酷睿 i7－3770(3.40GHz) 内核数量:4 主机内存:8GB DDR3
GPU	英伟达 GeForce GTX680(1058 MHz) 内核数量:1536 设备内存:2GB GDDR5

CUDA 平台中配置的 GPU 块数和每块的线程数分别为 128 和 64。同时，设置的预过滤缓冲区和全匹配缓冲区分别为 1 MB 和 256 MB。利用文献［21］中的 HPMA 算法的源代码创建 LHPMA 算法的源代码，结合算法 1 创建 LBSA 算法的源代码，所有源代码采用 GCC4.8.4 进行编译，操作系统利用 64 位的 Ubuntu14.04（内核版本 3.13）。

本文从 Snort 入侵检测系统提取并生成实验所需的模式集。Snort 构成的模式集包含的模式数量最多，但字符数最少。模式集中的数据包长度从 1 到 208 bytes 不等，平均为 13.8 bytes。如果输入数据流中的字符串是模式集中任何模式的重要前缀字符串，则视为恶意攻击。如果前缀字符串覆盖了整个字符串的 80%以上，则视为正常流量。对于每个输入数据流量，根据匹配率，从模式集中随机选择适当的前缀，并嵌入正常流量中。对于 Snort 构成的模式集采用 HTML 格式，并在 Linux 服务器中的/usr/bin 下的所有文件连接起来作为正常数据流。同时，利用模式集生成预过滤和全模式匹配的查询表，并加载到检测系统中。从谷歌、亚马逊和雅虎的门户网站提取的网络流量中选取真实的数据包检测作为传入数据包的输入。输入流量中的数据包有效载荷长度信息，如表 2 所示。

表 2 输入流量的数据包有效载荷长度信息

有效载荷长度/bytes	数量	比率
＜100	3 334	0.009
100～300	6 664	0.018
300～500	32 919	0.090
500～700	14 670	0.040
700～900	4 326	0.012
900～1 100	3 946	0.011
1 100～1 300	4 441	0.012
＞1 300	297 419	0.809

3.2 CPU 与 GPU 之间的数据传输性能

在分析 LHPMA 的结果之前，还需分析影响 GPU 处理性能的因素。不同全匹配缓冲区中 CPU 和 GPU 之间的数据平均传输速率，如图 4 所示。其中，x 轴表示全匹配缓冲区的大小，y 轴表示以 Gbps 为单位的数据平均传输速率，AC-GPU 表示利用 GPU 通过 AC 算法检测数据包的匹配方法。

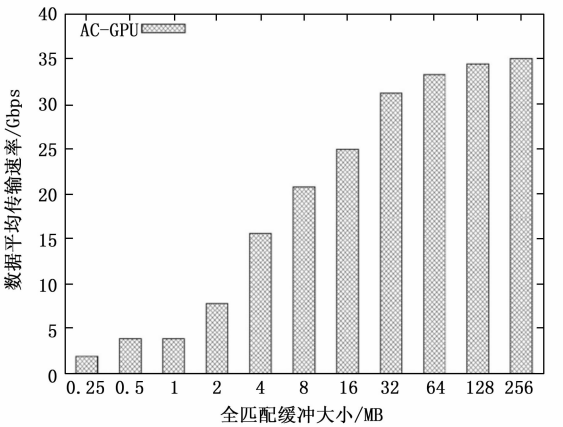

图 4 全匹配缓冲区中 CPU 和 GPU 之间的数据平均传输速率

由图 4 可以看出，当全匹配缓冲区较小时，数据传输率非常低，即传输的数据包有效载荷很少。例如，当全匹配缓冲区为 0.25 MB 时，数据传输速率为 1.95 Gbps。当全匹配缓冲区较大时，更多的数据包有效载荷在同一时间被传输，GPU 执行的数据传输速率会更高。例如，当全匹配缓冲区为 256 MB 时，数据传输速率为 35.1 Gbps。最佳方案的并不是将全匹配的缓冲区大小设置的越大越好，这是由于过大的缓冲区会导致数据包有效载荷等待时间过长，从而影响整体性能。因此，选择位于数据传输速率收敛的转折点作为全匹配缓冲区的临界值，本文选取全匹配缓冲区为 256 MB。

3.3 GPU 处理性能

不同全匹配缓冲区中 GPU 性能，如图 5 所示。其中，x 轴表示全匹配缓冲区的大小，y 轴表示以 Gbps 为单位的平均吞吐量，AC-GPU-100 和 AC-GPU-1460 分别表示较短（100 bytes）和较长（1 460 bytes）数据包有效载荷集。

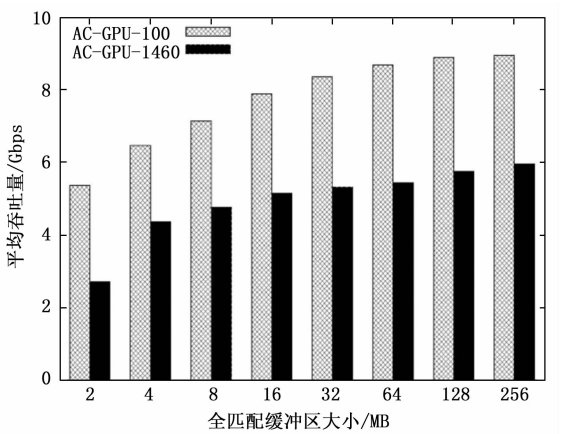

图 5 全匹配缓冲区中 GPU 性能

由图 5 可以看出，在全匹配缓冲区中，AC-GPU-1460 比 AC-GPU-100 的吞吐量低，较长的数据包有效载荷集在全匹配缓冲区的吞吐量较小，这是由于 LBSA 算法将较长的数据包过滤后直接发送至全匹配缓冲区，等待 GPU 的处理，造成了 GPU 性能降低。当全匹配缓冲区较小时，GPU

性能较低，尤其是在全匹配缓冲区为 2 MB 的 AC-GPU-1460 情况下。数据包有效载荷的数量很少，导致 GPU 线程利用率低，从而降低了整体性能。即输入有效载荷集的长度也会影响 GPU 效率。较短的数据包可收集成更多的数据包，有利于 GPU 的并行处理能力。因此，AC-GPU-100 的效率表现为 5.37 Gbps 到 9.03 Gbps，高于 AC-GPU-1460 从 2.71 Gbps 到 6.10 Gbps 的效率。

不同有效载荷长度中 GPU 性能，如图 6 所示。

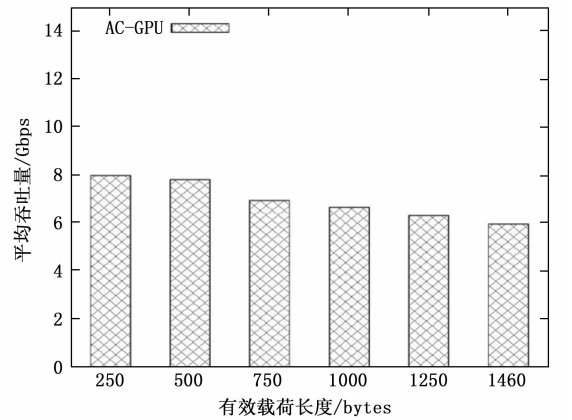

图 6　有效载荷长度中 GPU 性能

由图 6 可以看出，较短的数据包有利于 GPU 效率。这是由于 GPU 具有较强的并行处理能力，可将较短的数据包中的有效载荷集在 GPU 纹理内存中集中收集状态转换表并做统一识别处理，加速了并行处理较短的数据包的能力。当有效载荷集长度为 250 bytes 时，GPU 最多可以完成高达 8.0 Gbps 的吞吐量。随着有效载荷长度的增加，GPU 的吞吐量逐渐下降。这是由于 GPU 设备内存中较短的有效载荷数量多于较长的有效载荷数量，同时，GPU 内核的数量远大于 CPU 内核的数量，从而 GPU 利用多线程并行处理提高检测流量的吞吐量。

3.4　LHPMA 性能

不同有效载荷长度约束中 LHPMA 性能，如图 7 所示。其中，x 轴表示有效载荷长度约束 L_b，y 轴表示以 Gbps 为单位的平均吞吐量，LHPMA-AC 表示 LHPMA 通过 AC 算法检测数据包的匹配方法。

由图 7 可以看出，当 $L_b = 0$ 时，所有传入数据包都以 12.7 Gbps 的吞吐量通过预过滤缓冲区，即数据包通过 LBSA 算法后所有数据包都没有通过预过滤缓冲区就被发送至 CPU。随着 L_b 的增加，吞吐量提升至 14.0 Gbps。当 $L_b = 750$ 时，这与图 6 的结果相对应。即通过 LBSA 算法将较长的数据包存储在 CPU 主存储器中的预过滤缓冲区中，CPU 访问较长的数据包并根据查找表（表名、基本信息表、间接寻址表和字段说明）对数据包进行预过滤，降低内存访问延迟并减少预过滤时间，同时，较短的数据包存储在 CPU 主存储器中的全匹配缓冲区中，当全匹配缓冲区变满时，全匹配缓冲区内的数据包就会转移到 GPU 设备内存中，采用 AC 算法对 GPU 纹理内存中存储的可疑数据包进

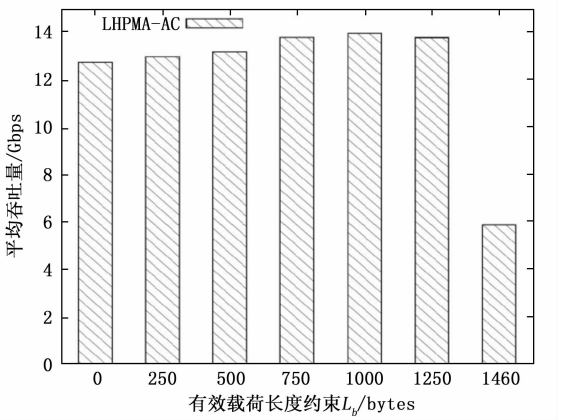

图 7　有效载荷长度约束中 LHPMA 性能

行全模式匹配，AC 算法通过插入许多模式，从前往后遍历数据包的整个字符串，要插入的字符其节点再先前已经建成，直接去考虑下一个字符即可，当发现当前要插入的字符没有再其前一个字符所形成的树下没有节点，就要创建一个新节点来表示这个字符，接下往下遍历其他的字符。并将查找表（状态转换表、接受状态表和失败状态表）复制到 GPU 纹理内存中，使得有效载荷长度的多样性减少，从而提高过滤率并降低数据传输延迟。LHPMA 提高了检测流量的吞吐量。当 $L_b = 1\ 460$ 时，吞吐量下降至 5.9 Gbps，这是由于 AC 算法从数据包的根节点开始，每次根据读入的字符沿着自动机向下移动。当读入的字符，在分支中不存在时，递归走失败路径。如果走失败路径走到了根节点，则跳过该字符，处理下一个字符。因为 AC 自动机是沿着输入文本的最长后缀移动的，所以在读取完所有输入文本后，最后递归走失败路径，直到到达根节点，这样可以检测出所有的模式。所有传入数据包都没有通过预过滤缓冲区就被发送至 GPU，从而数据传输延迟和有效载荷长度多样性导致 LHPMA 性能的下降。

将本文提出的 LHPMA 与 HPMA[21]、仅使用 CPU[26]、仅使用 GPU[27]的性能进行对比，如图 8 所示。其中，HPMA-AC 和 AC-CPU 分别表示 HPMA 和 CPU 通过 AC 算法检测数据包的匹配方法，同时，LHPMA-AC 选择的长度约束 $L_b = 750$。

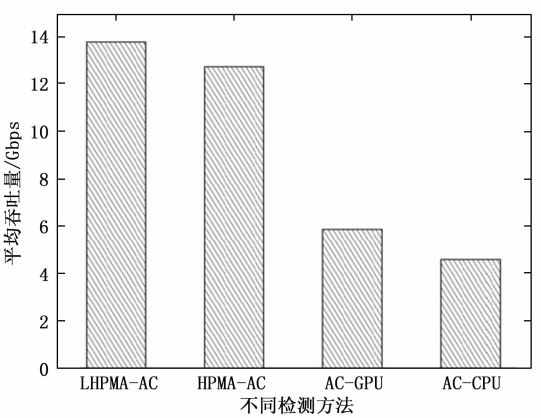

图 8　LHPMA 与其他方法的性能对比

由图 8 可以看出，LHPMA-AC、HPMA-AC、AC-GPU 和 AC-CPU 的吞吐量分别为 13.8、12.7、5.9 和 4.6 Gbps，表明 LHPMA-AC 比 HPMA-AC 增强了性能。这是由于 LHPMA-AC 比 HPMA-AC 增加了 LBSA 算法，根据预先设定的有效载荷长度约束对传入数据包进行分类，较长的数据包分配至 CPU 主存储器中的预过滤缓冲区中，凭借 CPU 更强的处理能力可以对较长的数据包进行检测，而较短的数据包分配至 CPU 主存储器中的全匹配缓冲区中，凭借 GPU 更强的多线程并行处理能力可以对较短的数据包进行检测，从而减少 GPU 并行处理的有效载荷长度多样性。同时，LHPMA-AC 分别比仅适用 GPU-AC 和仅适用 CPU-AC 方法高出了 2.3 和 3.0 倍。这是由于 LHPMA-AC 综合了 CPU 较强的处理能力和 GPU 并行处理能力，提升了 CPU/GPU 协同检测网络入侵的性能。

4 结束语

本文提出了一种具有数据包有效载荷长度约束的 CPU/GPU 混合模式匹配算法（LHPMA）来设计 NIDS，提高了输入流量中各种有效载荷长度情况下的入侵检测效率。通过预先设定长度约束分离算法（LBSA）对传入数据包进行分类，减少有效载荷长度的多样性。将较长的数据包分配给 CPU 主存储器的预过滤缓冲区中，直接由 CPU 访问数据包并根据查找表对数据包进行预过滤，并将预过滤的数据包也发送至全匹配缓冲区。同时，将较短的数据包分配给 CPU 主存储器的全匹配缓冲区中，并发送至 GPU 设备内存中，结合 AC 算法对 GPU 纹理内存中存储的可疑数据包进行全模式匹配，提升了 CPU/GPU 协同检测网络入侵的性能。实验结果表明，在 LBSA 算法中选择适当的长度约束，可以有效提升 LHPMA 的性能，并且 LHPMA 的性能优于 HPMA 以及 CPU 和 GPU 的单独处理方法。未来的研究中，将致力于分析输入流量中不同有效载荷长度分布情况，改进 LHPMA 方法对 CPU/GPU 协同处理网络入侵的性能。

参考文献：

[1] 叶开珍. 基于 COME 模块的网络入侵检测系统设计 [J]. 微型电脑应用，2022，38 (4)：160－162.

[2] 赵 卫，方 诚. 基于大数据的复杂网络入侵数据智能化检测系统设计 [J]. 自动化技术与应用，2021，40 (11)：164－167.

[3] 谢 凯，代 康. 基于负载预测的通信网络入侵检测系统设计 [J]. 计算机测量与控制，2021，29 (8)：62－66.

[4] 李麒鑫. 网络入侵检测技术研究 [J]. 电子技术与软件工程，2021 (14)：234－235.

[5] 陈昌娜，李昭桦. 基于增量集成学习的动态自适应 SDN 入侵检测 [J]. 计算技术与自动化，2021，40 (3)：177－183.

[6] GIUSEPPINA A，ANNALISA A，DONATO M. Autoencoder-based deep metric learning for network intrusion detection [J]. Information Sciences，2021，569 (8)：706－727.

[7] 陶家栋，金 华. 基于多类攻击的 Modbus Tcp 网络入侵检测方法 [J]. 信息技术，2020，44 (8)：23－27.

[8] 陈天一，郑闻悦，邹 健，等. 基于 KMP 算法的字符串查找匹配研究 [J]. 科技创新导报，2019，16 (23)：242－243.

[9] 王元甲，吴 陈. 基于 BM 算法病毒实时扫描的设计与实现 [J]. 计算机与数字工程，2019，47 (6)：1437－1440，1458.

[10] 周延森，张维刚. 一种 WM 多模匹配算法的研究与改进 [J]. 计算机应用与软件，2021，38 (7)：251－257，309.

[11] 寇远博，邱泽宇，王 亮，等. 基于 CPU－FPGA 异构系统的排序算法加速 [J]. 电子技术应用，2022，48 (1)：18－23，30.

[12] 王 越，冯 振. 基于 CAM 与双线性网络的鸟类图像识别方法 [J]. 重庆理工大学学报（自然科学），2021，35 (11)：136－141，239.

[13] 马晓莹，卢文杰，陈 肯. 浅谈精确字符匹配技术的发展 [J]. 电子世界，2019 (14)：77－78.

[14] 黄劲安，蔡子华. 基于 3GPP 信道模型的快速 5G 网络预测方法研究 [J]. 广东通信技术，2021，41 (4)：36－41.

[15] 刘楚鸿，汪培萍. 基于自动编码器集合的入侵检测系统的研究与实现 [J]. 中国新通信，2019，21 (24)：71－74.

[16] 黎雷生，杨文浩，马文静，等. 复杂异构计算系统 HPL 的优化 [J]. 软件学报，2021，32 (8)：2307－2318.

[17] 张 全，林柏栎，杨 勃，等. CPU-GPU 异构平台的抛物线 Radon 变换并行算法 [J]. 石油地球物理勘探，2020，55 (6)：1263－1270，1163.

[18] 孙兆鹏，周宽久. 基于 PCIe 的高性能 FPGA-GPU-CPU 异构编程架构 [J]. 计算机工程与科学，2021，43 (4)：641－651.

[19] 林宏刚，张运理，郭楠馨，等. 基于图神经网络的 P2P 僵尸网络检测方法 [J]. 工程科学与技术，2022，54 (2)：65－72.

[20] 王靖雯，马梓尧. 基于人工神经网络的 DGA 僵尸网络检测 [J]. 信息技术与信息化，2022，15 (1)：205－208.

[21] YAMINI B，SWATI B，BALARAM G. Current trends in the development of HPMA-based block copolymeric nanoparticles for their application in drug delivery [J]. European Polymer Journal，2020，139 (5)：110－118.

[22] 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.

[23] 何梦乙，覃仁超，刘建兰，等. 基于 Adam-BNDNN 的网络入侵检测模型 [J]. 计算机测量与控制，2020，28 (2)：58－62，81.

[24] 章 缙，李洪赭，李赛飞. 针对基于随机森林的网络入侵检测模型的优化研究 [J]. 计算机与数字工程，2022，50 (1)：106－110，179.

[25] LIN Y S，LEE C L，CHEN Y C. A capability-based hybrid CPU/GPU pattern matching algorithm for deep packet inspection [J]. Control of Robotics and Informatics，2021，15 (6)：94－102.

[26] 周 颖. PCA 和 KNN 特征选取的网络入侵检测 [J]. 计算机与网络，2021，47 (22)：48－49.

[27] 洪 波，曹子建. 基于 Hadoop 的分布式入侵检测系统设计与实现 [J]. 西安工业大学学报，2018，38 (4)：390－395，407.