基于可编程硬件的有状态网络功能硬件加速架构

兰天翼,郭云飞,范宏伟,兰巨龙
(国家数字交换系统工程技术研究中心,河南郑州 450002)

摘 要: 为了解决无状态加速器对有状态虚拟网络功能(Virtual Network Function,VNF)的加速效果较差的问题,该文提出了一种基于可编程硬件的有状态功能处理加速架构(Stateful Function Processing Acceleration Architecture,SFPA).SFPA通过为数据平面提供有状态处理单元(Staeful Processing Unit,SPU),将数据包处理任务卸载到数据平面上.此外,SFPA能够为多个VNF独立地分配加速资源,并采用资源分配优化算法降低硬件资源开销,提高了加速架构的灵活性.基于NetFPGA-10G平台的实验结果表明,SFPA架构下,VNF的吞吐量是采用DPDK加速时的2.9倍,是无状态硬件加速器的1.7倍;资源分配优化算法的优化率最高可达41.9%.

关键词: 网络功能虚拟化;可编程硬件;有状态处理;硬件加速;资源分配优化
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2018)07-1609-08
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.07.010

Programmable Hardware-Based Stateful Network Functions Hardware Acceleration Architecture

LAN Tian-yi,GUO Yun-fei,FAN Hong-wei,LAN Ju-long
(*National Digital Switching System Engineering & Technology Research Center,Zhengzhou,Henan* 450002,*China*)

Abstract: It's far less effective for the stateless accelerator to accelerate the stateful network function. In order to solve the problem,this paper presents a programmable hardware-based stateful network function acceleration architecture which is called Stateful Function Processing Acceleration (SFPA) architecture. Providing the Stateful Processing Unit (SPU) to the data plane,SFPA can offload the data processing task to the data plane. In addition,SFPA can allocate the acceleration resources to multiple VNFs independently,decrease hardware cost and improve the flexibility of the acceleration architecture with the resource allocation optimization algorithm. Results of the experiments which are based on the NetFPGA-10G platform show that the throughput of VNF is 2.9 times faster than that of DPDK,and 1.7 times faster than that of stateless hardware accelerator in the SFPA. The optimal rate of resource allocation optimization algorithm is up to 41.9%.

Key words: network function virtualization;programmable hardware;stateful processing;hardware acceleration;resource allocation optimization

1 引言

在互联网技术飞速发展的背景下,用户需求呈现出复杂化和多样化的趋势.因此,对于采用专用硬件的传统网络来说,为了提高网络服务质量(Quality of Service,QoS),需要不断地添加新的专用设备来支持相应的网络功能.除此之外,网络设备的增加提高了系统的功耗,从而使运营商面临着网络建设和运维成本迅速提高的困境.为了解决上述问题,业界提出了网络功能虚拟化(Network Function Virtualization,NFV)架构,它的核心思想是将软件和硬件解耦合,通过构建基于标准硬件的通用平台,使硬件资源能够更加灵活的共享和分配,有效降低了网络运营商的成本,实现了网络业务的灵活配置和新业务的快速开发与部署[1].在NFV架构

收稿日期:2017-03-14;修回日期:2017-07-16;责任编辑:孙瑶
基金项目:国家863高技术研究发展计划(No.2015AA016102);国家973重点基础研究发展计划(No.2013CB329104);国家自然科学基金创新研究群体科学基金(No.61521003)

中,由虚拟网络功能(Virtual Network Function,VNF)实现相应的网络业务.但与传统网络功能相比,VNF 在数据处理速率上大概有 30% ~40% 的损失[1],成为了 NFV 发展的主要瓶颈,因此需要对 VNF 数据包处理进行加速.而在这些需要加速的 VNF 中,有很大一部分是有状态功能.目前,大部分硬件加速器是无状态的,在对有状态功能进行加速时效果较差,因此需要专门为有状态 VNF 设计一种硬件加速架构.

文献[2]采用基于单根 I/O 虚拟化(Single Root-IO Virtualization,SR-IOV)技术和数据平面开发工具(Data Plane Development Kit,DPDK)对 VNF 进行加速.其利用 SR-IOV 技术将规模较大的网络功能分割为多个轻量级的虚拟功能,使多个 VNF 共享物理资源,提高了硬件资源的利用率和 I/O 性能,另外高性能数据包处理框架 DPDK 代替原 Linux 内核协议栈,提高了数据处理速率.文献[3]根据不同 VNF 加速需求之间的差别,将 VNF 抽象为两类:网络密集(Network-Intensive,NI)型和计算密集(Compute-Intensive,CI)型,并对两种 VNF 分别提出了硬件加速模型,在有效提升数据包处理性能的同时动态分配加速资源.文献[4]提出了一种 NFV 加速架构 OpenANFV,其对云环境下的 VNF 实现硬件加速.OpenANFV 架构采用基于 PCIe 的 FPGA 板卡作为硬件加速平台,该平台支持动态局部可重构,使加速器可以对多个 VNF 独立地加速.

以上几种 VNF 加速架构是无状态的,当处理有状态功能时,需要等待同一条流中所有数据包到达后才能处理,降低了网络处理性能.因此需要有状态加速器对有状态 VNF 进行加速,文献[5]提出了一种有状态数据平面抽象(Stateful Data Plane Abstraction,SDPA)架构,SDPA 架构通过采用“匹配-状态-状态操作-指令”的新型数据平面抽象范式实现了在数据平面中处理有状态功能,在一定程度上提高了有状态 VNF 的数据包处理速率.但考虑到 SDPA 架构所需表项资源开销较高且增加了系统的复杂度,直接在资源有限的可编程硬件中部署会有一定的困难,因此需要提出一种优化方案来对 SDPA 进行简化.

文献[6]针对 SDN 中,数据平面对网络级状态信息(如 TCAM 开销、CPU 开销等)缺乏有效的管理机制的问题,提出了一种网络级状态管理架构(Network-Level State Management Architecture,NeSMA).该架构以较低的资源开销实现了对 SDN 中状态有感(state-aware)应用的支持.NeSMA 主要通过向数据平面添加网络功能处理器对网络状态进行监控,并保证数据平面有较高的转发性能.文献[7]提出了有状态功能的无状态化处理架构,该架构将待处理功能的数据包与其状态分离,通过 RAM 云来缓存状态信息,从而将带状态功能转移到数据平面中以无状态的方式进行处理,提高了网络中带状态功能的处理速率,降低了通信延迟.文献[8]提出了混合体系中有状态硬件加速器的优化算法,该算法针对有状态硬件加速器在进行硬件状态切换过程中的资源开销过高的问题,根据请求队列对不同的网络流请求进行动态重排序,有效降低加速器的平均响应时间,提高数据处理速率,但另一方面,该算法的不足是没有考虑到公平性因素.

为了提高有状态网络功能的数据包处理速率,本文提出了有状态功能处理加速架构(Stateful Function Processing Acceleration Architecture,SFPA).该架构采用可编程硬件作为加速平台,为数据平面提供了状态管理机制,从而将有状态 VNF 的数据转移到相互独立的硬件加速器中进行处理,实现了对有状态网络功能的硬件加速;还采用了资源分配优化算法,降低了加速资源开销.

2 SFPA 基本架构

2.1 整体架构

SFPA 整体架构如图 1 所示,其主要由一个可编程的解析器和有状态处理单元(Stateful Processing Unit,SPU)组成.为了提取状态信息,SFPA 采用了可编程解析器,实现了数据包头中指定匹配域的提取,并从中获得数据包状态信息,对其协议类型进行识别;然后解析器将数据包发送到 SPU 中,由状态选择器和匹配表(Match Table,MT)互相协作,完成有状态数据包的状态处理和匹配查询操作;当得到匹配结果后,数据包被发送到动作执行器中,再根据相关指令对数据包进行相应的操作.在 NFV 环境中,保证网络体系的灵活性至关重要.因此,SFPA 提供了灵活的配置机制,用户高度可定制.其中,可编程的解析器[9]可以为多种不同协议类型的包头进行解析,具有协议无关性,能够有效支持新型数据包头格式.该解析器可以根据用户的配置情况,提取对应功能的数据包头中指定匹配域,并缓存相关数据,然后经过数据平面流水线后级模块处理,完成对数据包的操作.对于 SPU,相比于传统的“匹配—动作”单元,其增加了状态选择器和状态转移信息表(State Transition Information Table,STIT),它主要根据解析器提取出的包头域和状态信息对数据包进行相关处理,并完成指定的状态操作,然后将其发送到匹配表以完成后续处理.动作执行器主要承担了对数据包的处理任务,它利用解析器共享的包头信息缓存,对匹配域快速定位,并对数据包执行相关指令.另外,在 SFPA 加速架构中,一些硬件加速器无法完成或不适合其完成的任务会上传到 VNF,由 CPU 对数据进行处理,从而降低了软硬件之间的上下行流量,提高处理速率,同时也能够避免不必要的资源开销,提高加速资源的利用率.

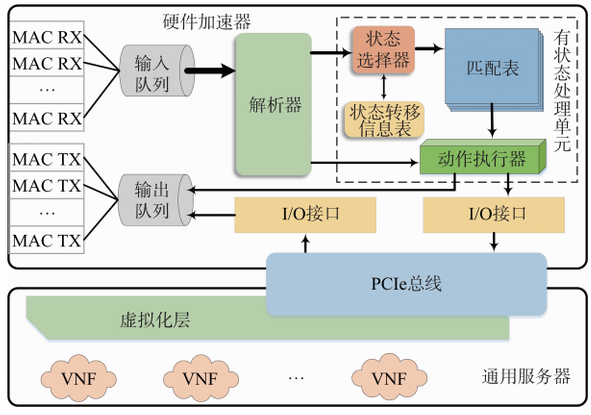


图1　SFPA整体示意图

2.2　有状态解析

数据包到达硬件加速器后,首先由解析器对其进行包头解析. 在 SFPA 中,解析器需要识别包头协议类型、对指定的匹配域进行提取,并获取数据包状态信息. 由于来自不同类型的网络或采用不同协议的数据包之间,在包头的格式和长度方面存在一定的差异,对它们的解析也会有所不同,因此必须提供一种能够支持多种协议的解析器,通过用户配置来实现协议无关的数据包解析. SFPA 采用可编程解析器[10]实现了协议无关的有状态解析,如图 2 所示,该包头解析器主要包括类型域识别模块、状态信息提取模块、匹配域提取模块和缓存模块. 在加速器中,数据包通过总线进行传输,为了提高传输效率,通常将多个数据包组成宽度与总线带宽相同的数据块. 数据块进入解析器后,RAM 1 被初始化,为类型域提取模块提供类型域起始偏移量,同时类型域提取模块也被初始化,将解析状态设为第一个类型域所在的数据块编号,在将到达的数据块编号与当前解析状态作对比,若一致则根据 RAM 1 中存储的类型域偏移量将类型域提取出来,再将当前的解析状态和提取出的类型域一并发送给缓存模块. 在缓存模块中,TCAM 存放类型域提取模块的解析状态和由用户配置的类型与信息. RAM 2 中存放当前处理的类型域所对应的状态域偏移量,在缓存模块中,TCAM 匹配类型域和解析状态后,RAM 2 根据匹配结果向类型域提取模块返回下一解析状态,并将匹配域偏移量发送到匹配域提取模块,同时 RAM 3 向状态域提取模块发送状态域偏移量. 匹配域提取模块和状态域提取模块分别根据相对应的偏移量信息将指定的匹配域和状态域提取出来,并分别发送给匹配域组合模块. 匹配域组合模块将收到的匹配域和状态域组合成包头域并发送到后级 SPU 中进行下一步处理.

2.3　有状态处理单元

解析器完成对数据包头的解析之后,会将包含有

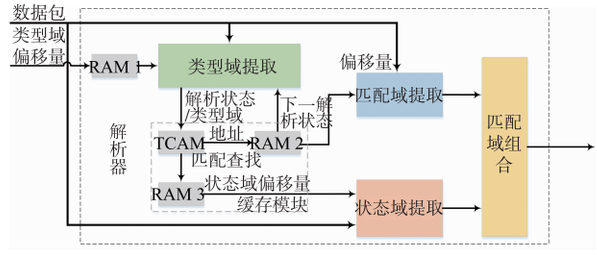


图2　解析器结构示意图

状态信息的匹配域组合连同数据包一起发送给有状态处理单元 SPU. SPU 实现了对数据包的状态信息进行管理,并完成数据包的转发等操作. 如图 3 所示,SPU 主要由状态选择器、状态转移信息表(State Transition Information Table,STIT)、匹配表(Match Table,MT)和动作执行器四部分组成. 当接收到相应的触发事件后,SPU 会执行相应的操作,这些事件通常是数据包到达、VNF 的生成和失效、网络拓扑变化、网络拥塞等等. SPU 能够通过 PCIe 接口与上层 VNF 进行通信,VNF 能够对状态选择器、STIT 和 MT 进行初始化、读取和修改. 一般情况下,实现不同功能的 VNF 对数据包的处理会有所不同,因此需要由 VNF 下发数据处理规则到数据平面,并对 SPU 进行配置.

状态转移信息表 STIT 主要用于指定 VNF 对应的硬件加速器的状态更新策略,并对状态选择器进行配置. 对于不同 VNF,STIT 表项一般不同,然而由于同一类网络功能的状态转移策略是相对固定的,因此可以将其缓存在加速平台中,当新的 VNF 加速需求到达后,硬件加速器对功能类型进行识别,然后读取相应功能的状态转移策略,并对 STIT 进行初始化. 一个 STIT 包含了 3 个域,依次是当前状态、下一状态和触发事件,而且正常情况下有状态 VNF 的状态数量是有限的,所以相应的 STIT 所需的资源开销较低. 以有状态防火墙为例,该 VNF 采用 TCP 协议,通过三次握手建立连接,其触发事件依次为 SYN,SYN + ACK 和 ACK,并且共有 LISTEN,REQUEST、ESTABLISH 和 TRANSFER 这 4 种状态,其 STIT 只需 3 条表项,如图 3 中 STIT 的前 3 条.

状态选择器主要由流控制表 FCT、状态表 ST 和结果处理器组成,根据数据包的状态信息和 VNF 的转发规则,选择对应的匹配表完成数据平面的处理过程. VNF 对状态选择器进行初始化,并配置 FCT 表项,VNF 发送到 FCT 的消息中包含了状态选择器处理相应数据包状态所需的匹配域,且匹配域可扩展,具有较强的灵活性. 在 FCT 表项中,由匹配域代表数据包处理过程中可能涉及到的连接,通常是源地址或目的地址. 以有状态防火墙功能为例,当只允许主动连线的数据包或具有指定匹配域信息的数据包通过防火墙时,加速器将

在 FCT 中进行匹配查找,对包头匹配域中的目的地址 Dip 进行过滤,并将匹配结果以指令“Instruction”的形式发送到结果处理器中,实现对数据包的相应操作. 状态表 ST 的主要任务是维护数据平面的状态信息. 加速器根据 VNF 类型,由 STIT 将状态操作规则下发到状态选择器中,根据各状态之间的转移关系生成状态表 ST. 当新的 VNF 加速请求到达后,对应的 STIT 会下发相应的状态处理规则,从而实现了 ST 的动态加载,降低了 VNF 与数据平面之间的流量. 包头中的状态域在 ST 中进行匹配查找,并将数据包当前状态和对应匹配表序号以向量〈当前状态,MT_id〉的形式发送到结果处理器中,以便下一步处理.

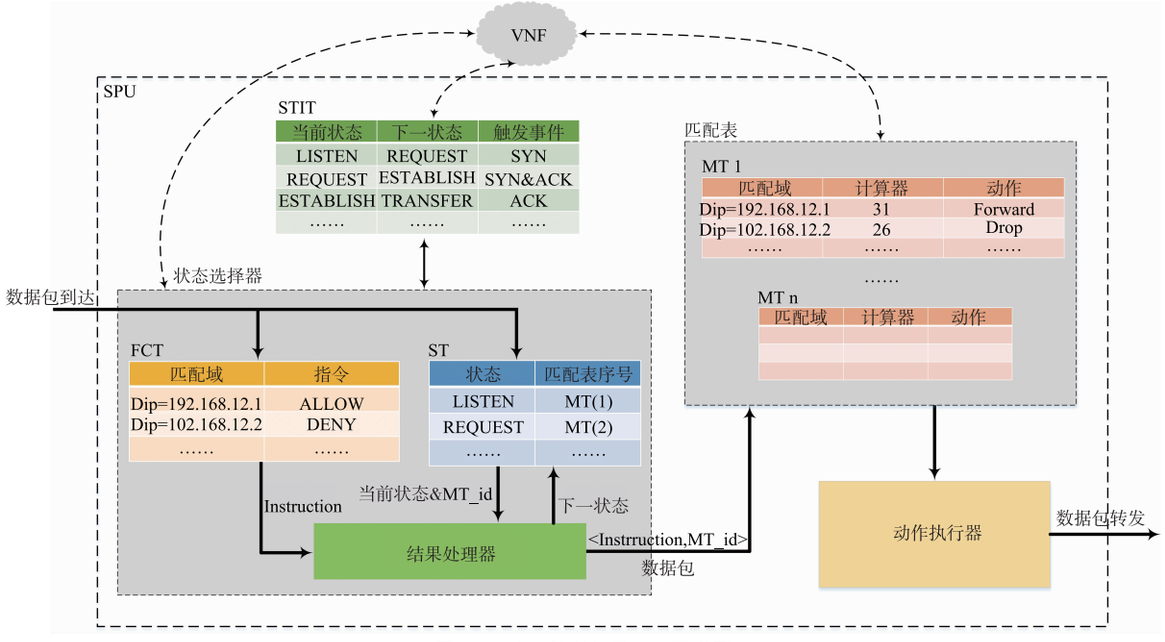


图3 有状态处理单元结构示意图

结果处理器可由用户进行配置,在其接收到 FCT 和 ST 的处理结果后,对相应数据包进行操作. 根据 FCT 的指令,若结果为“ALLOW”,则将处理结果〈Instruction,MT_id〉和相应的数据包发送到匹配表模块,并向 ST 发送消息,告知其当前状态数据包的相关操作已完成,进入下一状态;若 FCT 指令为“DENY”或未匹配到,则需要将数据包上传到 VNF,由 VNF 对其进行处理.

状态选择器根据〈Instruction,MT_id〉将数据包发送到相应的匹配表,如 MT(n)等,然后对数据包进行匹配查询. MT 的结构与 OpenFlow 协议中的流表类似,每条表项包含有 3 个域,分别是匹配域、计数器和动作指令. SFPA 架构中,MT 可以存储在 TCAM 中,由于 TCAM 支持模糊查找,因此能够对包头进行快速匹配. 当得到匹配结果后,将数据包发送到动作执行器中,完成对数据包的处理.

在 SPU 中,可编程动作执行器支持 VNF 对数据包头进行修改、添加或删除操作. 一般情况下,数据平面处理指定包头域时,需要将其从包头中提取出来,然而为了避免重复工作以及处理时延的增加,动作执行器能够读取解析器中 TCAM 和 RAM2 内存储的包头域位置信息,从而实现对相关包头域进行快速定位.

3 资源分配优化

硬件加速器需要为每个 VNF 独立地分配资源,加速器在工作状态下,对该部分资源独占,实现了 VNF 之间的相互隔离. 而 SFPA 采用了基于可编程硬件平台的同构化设计,如 FPGA 等. 一般来说,与 VNF 的加速需求相比,硬件平台上的资源十分有限,因此需要采用优化算法,从而降低加速器的硬件资源开销.

3.1 资源分配模型

在 SFPA 中,包头解析器的资源开销要远小于 SPU,且变化相对较小,因此,其资源开销可忽略不计,主要考虑 SPU 所占用的硬件资源. SPU 主要有四部分组成:状态转移信息表、状态选择器、匹配表和动作执行器. 其中,动作执行器的功能和结构均较为简单,资源占用少,为简化问题,其资源开销忽略不计. 对于状态转移信息表,其开销主要与相关功能的状态总数 k 和状态信息长度 l_{SI} 有关,对于不同类型的网络功能,k 值和 l_{SI} 值一般不同;对于状态选择器,其开销和其处理的功能状态总数 k 和选择项长度 l_S 有关;对于匹配表,其资源开销与匹配宽度 m 和深度 d 有关. 为了更加高效地利用硬件平台的资源,SFPA 将硬件资源划分为细粒

度的加速单元(Acceleration Unit,AU),设单个 AU 的宽度为 ω,深度为 D,由于 AU 的宽度和深度二者均可配置,因此可以选择恰当的 AU 宽度和深度参数,降低硬件资源开销.

假设有 N 个 VNF 需要采用 SFPA 进行加速,其中第 i 个 VNF 的 AU 宽度参数为 ω_i,深度参数为 D_i,并且 $1\leqslant i\leqslant N$,则其加速单元的开销为 $C_{\mathrm{AU}}(\omega_i,D_i)$,以下简记作 C_{AU}. 因此,该资源分配问题可以描述为:

$$\min: C_{\mathrm{sum}}=\sum_{i=1}^{N} C_i \tag{1}$$

s. t.

$$C_i=C_{STI\_i}(k,l_{SI})+C_{SS\_i}(k,l_S)+C_{M\_i}(m_i,D_i) \tag{2}$$

$$C_{z\_i}(x,y)=n_{z\_i}\times C_{\mathrm{AU}} \tag{3}$$

$$n_{z\_i}=\langle x/\omega_i\rangle\times\langle y/D_i\rangle \tag{4}$$

$$\forall i\in(0,N]:0<\omega_i\leqslant\omega_{\max},0<D_i\leqslant D_{\max} \tag{5}$$

在式(1)~(5)中:

C_{sum}为硬件平台上的加速器总资源开销;

C_i 为第 i 个 VNF 所在加速器的硬件资源开销;

$C_{STI\_i}(k,l_{SI})$ 为状态转移信息表资源开销,其与相应 VNF 的状态总数 k 和状态转移信息长度 l_{SI}有关;

$C_{SS_i}(k,l_s)$为状态选择器资源开销,其与相应 VNF 的状态总数 k 和选择项长度 l_s 有关;

$C_{M_i}(m_i,d_i)$为匹配表资源开销,其与相应 VNF 的匹配宽度 m 和深度 d 有关;

$C_{z_i}(x,y)$为 SPU 中各模块的资源开销,其中 $z\in\{STI,SS,M\}$,$x\in\{k,m_i\}$,$y\in\{l_{SI},l_S,D_i\}$;

n_{z_i}为各模块所需 AU 的数量,其中$\langle x/y\rangle$代表 x/y 的上舍入值;

$\omega_{\max}$和 $D_{\max}$为硬件加速平台资源宽度深度的上限.

3.2　资源分配优化算法

针对加速器资源分配模型,可将该优化问题归纳为一个整数线性规划(Integer Linear Programming,ILP)问题[11]. 在有状态功能的加速过程中,SFPA 上的硬件加速器由一个或多个 AU 组成,而 AU 的尺寸(即 ω 和 D 值)可以根据实际情况来配置,同时加速器的资源开销情况也将随着 ω 和 D 值的变化而变化. 为了保证求解算法较快收敛,本文采用分支定界法求解该优化问题.

算法 1 利用 ILP 描述了硬件资源分配进行优化的过程. 首先求解 ILP 所对应的 LP 问题,找到规划问题的最优解(行 7),然后将该解作为初始解提供给 ILP 求解器(行 10).

算法 1　资源分配优化算法

输入:硬件加速平台的资源上限 $\omega_{\max}$和 $D_{\max}$,状态信息总数 k,状态转移信息长度 l_{SI}和选择项长度 l_s

输出:加速器资源总开销 C_{sum}

```
0.  for ω = 1:ω_max
1.      for D = 1:D_max
2.          C_AU = f(ω,D)                    /* 加速单元的资源开销 */
3.          n_STI = ⟨k/ω⟩ × ⟨l_STI/D⟩, n_SS = ⟨k/ω⟩ × ⟨l_S/D⟩,
4.          n_M = ⟨m/ω⟩ × ⟨d/D⟩
5.          C_AU = f(ω,D), C_STI = n_STI × C_AU, C_SS = n_SS × C_AU,
6.          C_M = n_M × C_AU
7.          minFind:(ω,D) → C_sum0 = C_STI + C_SS + C_M   /* 不考虑
            整数可行域时的初始解 */
8.      end for
9.  end for
10. min:C_sum = ILP_Solver(ω,D,C_sum0)
```

算法 2 给出了基于分支定界法的 ILP 求解器“ILP_Solver”的实现过程. 首先对初始解进行判断,如果其满足整数可行域的约束条件,则作为结果输出(行 1);如果参数 ω 不满足整数约束条件,而参数 D 满足,则对 ω 进行分支,在各分支中分别求取资源总开销,取较小者返回(行 2~4);如果参数 D 不满足整数约束条件,而参数 ω 满足,则对 D 进行分支,在各分支中分别求取资源总开销,取较小者返回(行 5~7);如果参数 ω 和 D 都不满足整数约束条件,则分别对两个参数分支,共有 4 种情况,分别算出个情况下的资源开销,取最小者为所求结果(行 8~10).

算法 2　分支定界法

输入:加速单元宽度 ω,深度 D 和初始最优解 C_{sum0}

输出:硬件资源分配最优解 $\min C_{\mathrm{sum}}$

```
0.  def ILP_Solver(ω,D,C_sum0)
1.  if(ω ∈ Z, D ∈ Z) minC_sum = C_sum0
2.  else if(ω ∉ Z, D ∈ Z)
3.      ω_low = int(ω), ω_high = ceil(ω)
4.      minC_sum = min(C_sum(ω_low,D), C_sum(ω_high,D))
5.  else if(ω ∈ Z, D ∉ Z)
6.      D_low = int(D), D_high = ceil(D)
7.      minC_sum = min(C_sum(ω,D_low), C_sum(ω,D_high))
8.  else
9.      ω_low = int(ω), ω_high = ceil(ω), D_low = int(D),
        D_high = ceil(D)
10.         minC_sum = min{C_sum(ω_low,D_low), C_sum(ω_high,D_low),
                          C_sum(ω_low,D_high), C_sum(ω_high,D_high)}
11.     end if
12.     return minC_sum
13. end def
```

ILP 问题是一个 NP 难问题,假设硬件资源上限参数分别为 $\omega_{\max}=m$,$D_{\max}=n$,则采用分支定界法进行求解的时间复杂度为 $O(mn)$,虽然与贪婪算法相比,其在时间复杂度上不占优势,但能够覆盖整个可行域,避免陷入局部最优. 另一方面,分支定界法效率较高,通常

情况下并不需要遍历所有可行域内的点,因此与枚举法相比,其收敛速度大大提升.

4 性能分析

为了测试 SFPA 加速架构针对有状态 VNF 的实际加速效果,本文基于 NetFPGA-10G 平台实现了 SFPA 的原型,并在相关环境中对其性能进行了测试与分析. NetFPGA-10G 板卡采用了 Xilinx Virtex 5 可编程芯片,具有 4 个 10Gbps 的物理端口,以及 4 个连接了 PCIe 总线并支持直接内存访问(Direct Memory Access,DMA)的 CPU 虚拟端口[12,13]. 硬件加速平台开发的集成软件环境为 Xilinx ISE,并用其测量 SFPA 的硬件开销. VNF 部署在通用服务器中(Intel Xeon E5-2407 CPU 2.2GHz,16GB DDR3 RAM),服务器与硬件加速平台连接,采用 OpenVZ 作为虚拟层.

4.1 加速器转发性能分析

为了验证 SFPA 对有状态功能的加速性能,首先需要对 SFPA 架构针对不同长度数据包的处理速率进行测试. 实验中,我们采用 4 个有状态防火墙作为 VNF 实例,在硬件平台上能够被独立并行地加速,并由 Spirent TestCenter 测试仪来测量数据平面的吞吐量,实验结果如图 4 所示.

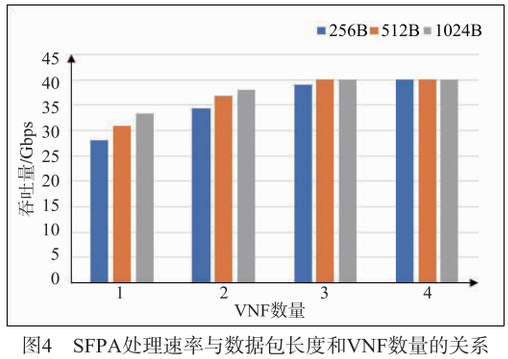
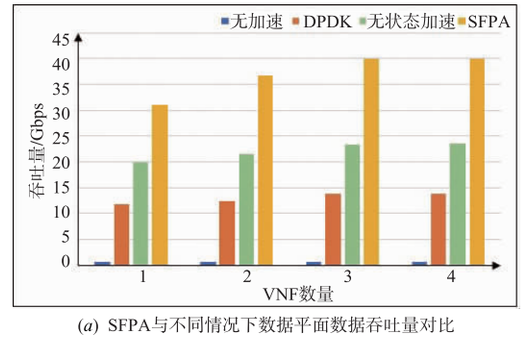


图4 SFPA处理速率与数据包长度和VNF数量的关系

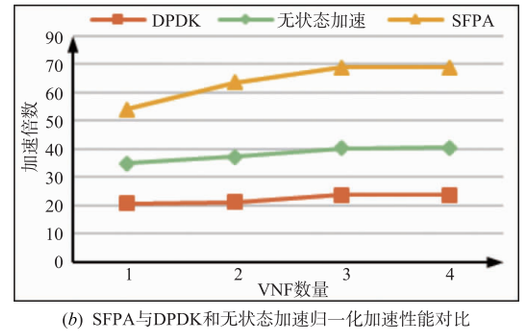
从图 4 的实验结果中可以看到,随着并行处理的 VNF 数量逐渐增多,SFPA 架构的吞吐量有所提高,同时在并行加速的 VNF 数量较少时,加速器的吞吐量与数据包长度呈正相关. 当 NetFPGA-10G 板卡为 4 个 VNF 提供并行加速处理时,硬件加速平台处于满负荷工作状态,I/O 资源被完全利用,因此吞吐量也达到了加速平台的极限,实现了 40Gbps 的线速处理. 对于数据包长度,其在 VNF 数量较小时影响较为明显,因为此时硬件加速平台上仍有较多的空闲资源,数据包越长,资源利用效率越高,吞吐量也就越高;而当 VNF 数量较多时,加速平台上大部分资源已被占用,因此再提高数据包长度也无法有效提升数据处理速率.

作为对比,本实验中无状态加速方案采用文献[10]中基于 NetFPGA-10G 平台的 NFV 数据包加速处理架构. 该方案将数据平面部署在 NetFPGA-10G 板卡上,然后直接将 VNF 的数据处理任务卸载到加速平台中进行,实现 VNF 数据处理加速. 软件加速方案采用高性能的数据平面优化工具 Intel DPDK. DPDK 是以 IA(Intel Architecture)多核处理器为目标平台而设计的高速包处理数据平面,可从多个方面对网络数据包的处理进行优化,提高数据平面上数据包的处理速率[14]. 实验中,为 DPDK 分配转发核 2 个,CPU Xeon E5-2407,内存大页设置为 4G,2 个 10Gbps 物理端口,操作系统 Ubuntu16.04. 用测试仪模拟数据流,向 VNF 发送数据包. 模拟 20000 条流,构造 20GB 数据流,从 100 个源地址向 50 个目的地址发送业务流,数据包长度为 1024B.

下面实验将 SFPA 的数据处理性能与 DPDK 和无状态加速器进行对比,验证 SFPA 对有状态功能的加速效果. 实验结果如图 5 所示.



(*a*) SFPA与不同情况下数据平面数据吞吐量对比



(*b*) SFPA与DPDK和无状态加速归一化加速性能对比

图5 SFPA与其他数据平面数据处理性能对比

从图 5(*a*)中可见 SFPA 与无状态加速器、DPDK 以及无加速相比,吞吐量具有明显提升. 在满负荷工作状态下,采用传统数据平面(即无加速时),对有状态功能的数据吞吐量为 590Mbps, DPDK 处理速率为 13.7Gbps,无状态加速器为 24Gbps,而 SFPA 能够达到 40Gbps. 从图 5(*b*)中可以看到,SFPA 与 DPDK 和无状态加速器相比,显然有着更好的加速效果,在满负荷工作时,与无加速情况相比,SFPA 能够达到 68 倍的加速效果,是 DPDK 加速的 2.9 倍,无状态加速器的 1.7 倍. 但是,有必要说明的是,如果直接采用无状态加速器对

有状态功能进行加速,会因为频繁的状态切换而造成较大的处理时延;除此之外,由于功能类型的不同,在实际情况中还将存在较大的延迟抖动,严重降低网络 QoS[8]. 综上所述,SFPA 对有状态功能加速效果显著.

4.2　资源分配优化算法性能分析

由于硬件资源有限,因此需要在 SFPA 中使用资源分配优化算法,提高硬件的利用率. 该文中,用优化率 R 来表示算法对资源分配的优化效果,优化率可以通过式(6)得到.

$$R = \frac{C_{\text{sum\_0}} - C_{\text{sum\_1}}}{C_{\text{sum\_0}}} \times 100\% \tag{6}$$

其中,$C_{\text{sum_0}}$代表优化前的资源总开销,$C_{\text{sum_1}}$代表优化后的资源总开销.

通过 Xilinx ISE 软件,测得不同数量 VNF 在加速平台上的资源开销情况,结果如表 1 所示.

表 1　SFPA 资源开销情况

VNF 数量	Slice 资源开销	资源使用率/%
1	7145	19.1
2	14281	38.1
3	21437	57.2
4	28568	76.3

在 SFPA 架构中采用了资源分配优化算法后,测试了不同 VNF 数量下算法的优化率,实验结果如图 6 所示.

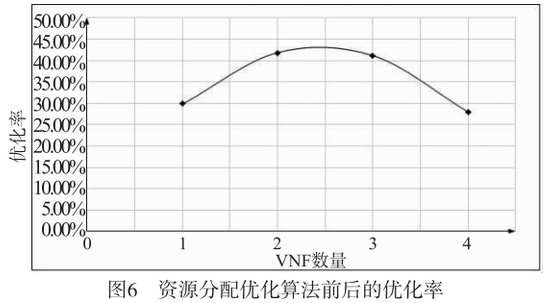


图6　资源分配优化算法前后的优化率

从图 6 中可以看到,在采用了资源分配优化算法后,SFPA 的总 Slice 资源开销有所下降,VNF 数量不同的情况下,算法的优化率也有所不同,当两个 VNF 并行加速时,优化率达到最高的 41.9%. 当 VNF 数量较少时,硬件平台上部署的加速器数量少,资源相对较多,因此当加速器数量增加时,优化算法能够更充分地发挥优化作用,从而明显提升优化率. 但当 VNF 超过 2 个之后,资源开销接近饱和,当 VNF 增加时,优化效果不再增加,优化率在 4 个 VNF 并行加速的情况时出现了明显的下降. 总的来说,基于 ILP 的资源分配优化算法是有效的,且存在最优工作区间,但本文只给出了有限情况下的分析,在 SFPA 部署时需要根据实际情况来确定.

5　结论

本文针对有状态 VNF 的数据包处理速率较低的问题,提出了一种基于可编程硬件的有状态加速架构 SFPA. 通过改进数据平面流水线,为硬件加速器提供了状态管理机制,能够有效地将有状态 VNF 的数据处理任务卸载到硬件中. 同时,还为不同加速器独立地分配硬件资源,提高了网络设备的灵活性和扩展性;最后,基于 ILP 的资源分配优化算法为 SFPA 提供了更为高效的资源分配方案,降低了 Slice 资源开销. 本文基于 NetFPGA-10G 平台实现了 SFPA 的原型,并对其性能进行了测试. 实验结果表明,在相同网络环境中,SFPA 能有效提升原 VNF 的吞吐量且加速效果优于无状态加速器和 DPDK. 另外,资源分配优化算法降低了加速资源开销. 因此,SFPA 能够在保持资源开销较低和较好灵活性的情况下,为有状态 VNF 提供更加快速的数据包处理,为 NFV 未来的创新和进步提供帮助.

参考文献

[1] 徐雷. 网络功能虚拟化技术与应用[M]. 北京:人民邮电出版社,2016.

[2] Kourtis M A, Xilouris G, Riccobene V, et al. Enhancing VNF performance by exploiting SR-IOV and DPDK packet processing acceleration[A]. Proceedings of IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)[C]. San Francisco: IEEE, 2015. 74 – 78.

[3] Bronstein Z, Roch E, Xia J, et al. Uniform handling and abstraction of NFV hardware accelerators[J]. IEEE Network, 2015, 29(3): 22 – 29.

[4] Ge X, Liu Y, Du D H C, et al. OpenANFV: Accelerating network function virtualization with a consolidated framework in openstack[J]. ACM SIGCOMM Computer Communication Review, 2015, 44(4): 353 – 354.

[5] Bi J, Zhu S, Sun C, et al. Supporting virtualized network functions with stateful data plane abstraction[J]. IEEE Network, 2016, 30(3): 40 – 45.

[6] Sun C, Bi J, Hu H, et al. NeSMA: Enabling network-level state-aware applications in SDN[A]. Proceedings of IEEE 24th International Conference on Network Protocols (ICNP)[C]. Singapore: IEEE, 2016. 1 – 7.

[7] Kablan M, Caldwell B, Han R, et al. Stateless network functions[A]. Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization[C]. London: ACM, 2015. 49 – 54.

[8] 马宜科,常晓涛,范东睿,等. 混合体系结构中有状态硬件加速器的优化[J]. 计算机学报,2011,34(7):1314 – 1322.
MA Yi-ke, CHANG Xiao-tao, FAN Dong-rui, et al. Optimization of stateful acceleration in hybrid architectures[J]. Chinese Journal of Computers, 2011, 34(7): 1314 – 1322.

(in Chinese)

[9] Gibb G, Varghese G, Horowitz M, et al. Design principles for packet parsers[A]. Proceedings of the Ninth ACM/IEEE Symposium on Architectures for Networking and Communications Systems[C]. San Jose: IEEE Press, 2013. 13 - 24.

[10] 段通,兰巨龙,胡宇翔,等. 一种支持网络功能演进的可重构数据平面[J]. 电子学报, 2016, 44(7): 1721 - 1727.
DUAN Tong, LAN Ju-long, HU Yu-xiang, et al. A reconfigurable dataplane enabling network function evolution[J]. Acta Electronica Sinica, 2016, 44(7): 1721 - 1727. (in Chinese)

[11] Jose L, Yan L, Varghese G, et al. Compiling packet programs to reconfigurable switches[A]. Proceedings of Usenix Conference on Networked Systems Design and Implementation[C]. Oakland: USENIX Association, 2015. 103 - 115.

[12] Ruiz M, Ramos J, Sutter G, et al. Accurate and affordable packet-train testing systems for multi-gigabit-per-second networks[J]. IEEE Communications Magazine, 2016, 54(3): 80 - 87.

[13] Duggisetty P. Design and Implementation of a High Performance Network Processor with Dynamic Workload Management[D]. Amherst Massachusetts: University of Massachusettes, 2015.

[14] Bi H, Wang Z H. DPDK-based Improvement of Packet Forwarding[OL]. https://www.itm-conferences.org/articles/itmconf/pdf/2016/02/itmconf_ita2016_01009.pdf, 2016-11-21/2017-03-06.

作者简介

兰天翼 男,1993 年 1 月生于甘肃兰州. 现为国家数字交换系统工程技术研究中心硕士研究生. 主要研究方向为新型网络体系结构.
E-mail: tianyilan@foxmail.com

郭云飞 男,1963 年 10 月生于河南郑州. 现为国家数字交换系统工程技术研究中心副主任、教授、博士生导师. 主要研究方向为宽带信息网络.
E-mail: gyf@ndsc.com.cn

范宏伟 男,1994 年 3 月生于河南许昌. 现为国家数字交换系统工程技术研究中心硕士研究生. 主要研究方向为新型网络体系结构.
E-mail: fanhongwei151@qq.com

兰巨龙 男,1962 年生于河北张北. 现为国家数字交换系统工程技术研究中心总工程师、教授、博士生导师. 主要研究方向为新一代信息网络关键理论与技术.
E-mail: ndscljl@163.com