基于可编程硬件的 NFV 数据包加速处理结构

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

摘 要: 针对网络功能虚拟化(Network Function Virtualization,NFV)在通用服务器中部署的处理性能受限问题,该文提出了一种基于硬件加速的虚拟网络功能(Virtual Network Function,VNF)处理结构:FARD(Function Adaptive and Resource Dividable hardware structure).通过可编程的包头解析器和动作处理器,FARD 可实现任意 L2/3/4 层功能实例的硬件加速处理;通过动态可分割的匹配表结构,FARD 支持不同功能实例间的资源动态分配和隔离.基于 NetFPGA-10G 的实验结果表明,对比基于纯软件实现的 VNF,FARD 加速结构提升了近 60 倍的包处理吞吐率.

关键词: 网络功能虚拟化;可编程硬件;通用加速结构;可分割匹配表;资源分割优化
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2017)12-3076-05
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.12.034

Enabling Fast Packet Processing for NFV via Programmable Hardware Acceleration

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

Abstract: To improve the forwarding performance of NFV (Network Function Virtualization) in general server,this paper comes up with a hardware based VNF (Virtual Network Function) packet processing acceleration structure which is called FARD (Function Adaptive and Resource Dividable hardware structure). FARD can implement the hardware acceleration of any function instance on the L2/3/4 layer with programmable parser and action processor. And FARD also supports the dynamic resource allocation and function isolation between different instances with dynamic division on match table. The results based on the NetFPGA-10G platform show that the FARD acceleration structure increases by nearly 60 times in packets processing throughput compared with VNF which is only implemented by software.

Key words: network function virtualization;programmable hardware;common acceleration structure;dividable match table;resource division optimizing

1 引言

网络功能虚拟化(Network Function Virtualization,NFV)技术[1]实现了网络功能与专用硬件的分离,应用前景广阔.NFV 在通用 x86 架构服务器中部署虚拟网络功能(Virtual Network Function,VNF).然而,由于基于软件的虚拟机的数据处理性能较差,难以满足以 L2/L3/L4 层处理为主的对流处理速率要求较高的网络功能处理需求[2],因此需要对 VNF 数据包处理进行加速.

目前,主要的 NFV 加速方案包括专用硬件加速和通用硬件加速结构.例如,文献[2]对 NFV 的硬件加速进行了抽象,并提出了"统一化处理"的加速方案.针对网络密集型(Network-Intensive,NI)功能,采用嵌入式系统和网络处理器(Network Processor,NP)进行加速;而对于计算密集型(Compute-Intensive,CI)功能,则采用通用架构的 CPU 或 GPU 实现数据加速处理.文献[3]在 SDN + NFV 架构中,将无状态功能卸载到数据平面中进行处理,减少软硬件之间的上下行流量,提高数据处理速率.文献[4]提出了带状态数据平面抽象(Stateful Data Plame Abstraction,SDPA)架构,该架构重新设计了数据平面,并在数据平面上提供了状态信息存储和管理功能,将带状态功能有效地转移到了数据平面中,提高

收稿日期:2016-09-12;修回日期:2017-02-06;责任编辑:覃怀银
基金项目:国家"973"计划资助项目(No.2012CB315901,No.2013CB329104);国家自然科学基金资助项目(No.61309019,61372121);国家"863"计划资助项目(No.2015AA016102)

带状态功能的处理效率. 文献[5]利用云存储技术,通过将状态信息与数据包分离,并存储在专用 RAMCloud 中,实现在数据平面上处理带状态功能,从而获得较高的数据处理性能. 然而,使用专用硬件虽然能够有效提高 VNF 的数据包处理性能,但也会增加网络中软硬件之间的耦合度,这与 NFV 的设计目标背道而驰.

综上,为了提升 NFV 数据处理速率并保证网络灵活性,该文提出了 FARD 加速处理结构. 它采用基于通用硬件的同构设计,将 VNF 中所有的无状态处理转移到硬件中,并保证了各功能之间相互隔离,实现对 NFV 数据包处理的通用硬件加速.

FARD 通过采用可编程的数据平面[6],允许 VNF 包处理在硬件中完成,从而提升数据传输性能,并降低软件硬件间的上/下行流量,提高资源利用率. 因此,与纯软件 VNF 处理方式相比,FARD 结构显然有更快的处理速度,与专用硬件加速和软件加速方案相比更具前瞻性,能够为 NFV 领域的创新提供一个更加开放的平台.

2　FARD 整体架构介绍

2.1　整体架构

FARD 结构提供了一个具有快速包处理能力的通用硬件加速平台,并保证了较高灵活性和独立性,其可在现场可编程门阵列(Field Programmable Gate Array, FPGA)和网络处理器(Network Processor, NP)等设备上进行实现. 如图 1 所示,FARD 主要由一个可编程解析器、可分割匹配表(Dividable Match Table, DMT)和一个灵活的动作处理器组成. 首先由解析器对数据包的协议类型进行识别并获得指定的匹配域,然后在匹配表中进行匹配查询,得到匹配结果后,由动作处理器对数据包执行相应的操作. 同时,在 FARD 中每个虚拟数据平面都分配有一对 DMA RX/TX 缓存和位于 TCAM 的独立路由表空间,这使得该结构的 I/O 性能和查找速度得到有效的提高.

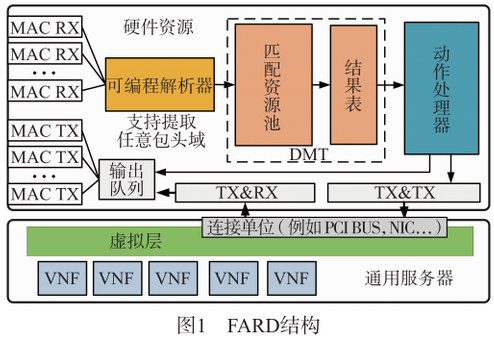

图1　FARD结构

另外,为了保证数据平面的灵活性,FARD 结构需使各功能间相互独立,不同 VNF 在分离的逻辑实例中并行运行,在性能方面互不影响. 为此,FARD 结构必须提供一种机制来确保每个 VNF 仅使用系统分配给它的硬件资源.

2.2　解析器和动作处理器

接收到数据包后,数据平面需要对其进行解析. 解析器须在提取指定域之前依次识别出包头格式. 然而,不同类型的网络、数据包之间,包头的长度和格式存在差异. 为了解决该问题,FARD 采用了可编程解析器结构[7,8].

可编程解析器由 TCAM 来存储各数据包头的类型值,由 RAM 来存储匹配域的偏移量和长度信息(〈offset, length〉). 在解析过程中,首先解析器根据包头信息识别数据包协议类型,并在 TCAM 中进行匹配,然后根据 RAM 中存储的〈offset, length〉值从包头中提取匹配域. 解析过程中,该操作会不断循环执行,直到提取出最后一个匹配域. 最后,结果输出模块将解析过程中提取的全部匹配域合并,生成匹配域组合并输出. 由于可重构解析器的 TCAM 和 RAM 能够通过 VNF 进行配置,因此用户能够提取任何所需的匹配域或向包头中添加新的域,从而实现数据平面包处理的功能自适应性[7,8].

为了在数据平面中实现协议无关的包处理,FARD 结构采用了具有协议无关性的可编程动作处理器[9]. 该动作处理器与网络协议解绑,能够根据用户需求对任何类型网络协议的数据进行处理[8,9]. 动作处理器能够根据 VNF 下发的策略对指定包头域执行修改、添加或删除操作. 另外,为了对数据包头执行指定的操作,需相应包头域,而这项工作已由解析器完成. 因此,可以利用解析器的 TCAM 和 RAM 中存储的包头域位置信息,对已由动作指令所指定的匹配域进行快速定位,然后处理器对指定域执行相应的操作.

2.3　匹配表结构

解析器提取出指定匹配域后发送到数据平面的匹配单元,经过匹配后,将其发送到动作处理器并执行相应的操作. 一般情况下,不同功能的匹配域在类型和数量上存在差异,但对于同类 VNF,需要的匹配域和执行的动作是固定的. 因此,FARD 能够为每个 VNF 动态地分配匹配表. 本文提出的 DMT 如图 2 所示,DMT 由 VID 查找表、一个比较器和多个细粒度的匹配表组成.

在 FARD 结构中,各 VNF 用不同的虚拟 ID(Virtual ID, VID)标记,VID 查找模块根据解析器的输出结果对数据包进行分类,并能得到表 ID(Table ID, TID)和匹配组合类型. 在 DMT 中,采用支持模糊查找的 TCAM 来存储和查找包头域.

由于不同 VNF 所需的匹配宽度和深度不同,因此在 FARD 结构中引入了多个细粒度的匹配表. 首先,各匹配表对应的选择器从包头域中选取宽度为 m bit 的字

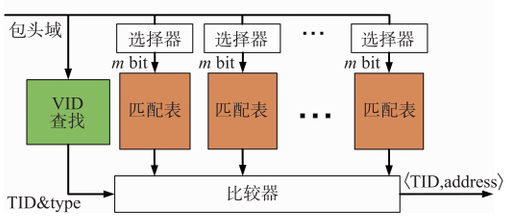

图2 可分割匹配表模型

段,然后在相应的匹配表中对其并行匹配,比较器根据表 ID 从匹配表的匹配地址中选出指定地址,结果会出现三种情况,如图 3 所示.

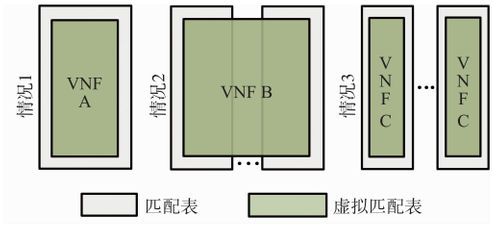

图3 不同情况下的资源分配

表 1 DMT 中不同匹配情况说明

情况	说明	TID 数量	备注
1	VNF 匹配深度和宽度均小于匹配表	单个	比较器输出〈TID, address〉,据此信息可得到结果表中的匹配结果
2	VNF 匹配宽度超过 m bit	多个	将多个匹配表合并实现匹配操作,比较器比较各匹配地址,若相等,则输出〈TID, match address〉
3	VNF 匹配深度超过匹配表深度	多个	由多个匹配表共同存储和匹配 VNF 规则,比较器找到最小地址后,作为最终的匹配地址

例如,假设当 NAT 功能用源 IP 来匹配,而防火墙功能使用源 IP、目的 IP、源端口和目的端口(〈IP-src,IP-dst,src-port,dst-port〉)来进行匹配,且每个匹配表的匹配宽度是 32bit,则会出现两种情况:情况 1 和情况 2. 对于防火墙(firewall)功能,它需要三个匹配表,那么就需要把三个匹配表合并起来,如同情况 2,该处理流程如图 4 中的流程①所示;对于 NAT 功能来说,它只需要 1 个匹配表,如同情况 1,该处理流程如图 4 中的流程②所示. 而对于硬件无法处理的数据包,它们会被发送到 VNF 中,然后对其做进一步的处理,如图 4 中的流程③.

3 FARD 资源分割建模及算法

3.1 匹配表资源分析

FARD 匹配所需的资源开销要远高于解析和动作执行,且相对固定. 因此,匹配表资源开销占资源总开销的主要部分. 匹配表资源开销可以与匹配宽度和深度

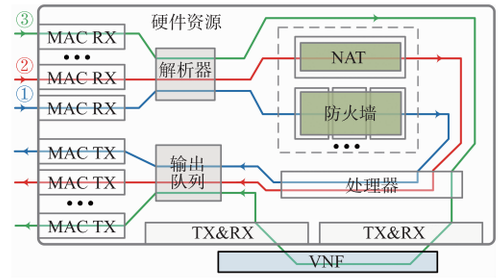

图4 FARD包处理流程示例

有关,因此存在选择恰当的匹配参数使总资源开销最小化的问题. 为了正式描述这个问题,假设有 K 个 VNF 需要映射到硬件中,各 VNF 所需的匹配深度和宽度记作:d_i 和 w_i,其中 $1 \leqslant i \leqslant K$. 为了使用尽可能少的匹配表资源,所以这个问题可以表述为:

$$\min: C(m,D) = n \times (C_s + C_M) \tag{1}$$

$$n = \sum_{i=1}^{K} (\langle w_i/m \rangle \times \langle d_i/D \rangle) \tag{2}$$

在式(1)和(2)中:C 为匹配表总代价,包括匹配选择器和匹配表代价;m 为各匹配表的匹配宽度;n 为所有匹配表的数量,且式(2)中的$\langle x/y \rangle$代表 x/y 的上舍入值;D 为各匹配表的深度;C_s 为匹配选择器代价,它与匹配宽度 m、包头域长度 L(为常数)有关,该关系可以表示为 $C_s = f(m)$;C_m 为匹配表代价,它与匹配宽度 m 和匹配深度 D 有关,$C_m = g(m,D)$.

对于不同的 VNF 设置,匹配宽度 m 和匹配深度 D 的取值会有所不同. 一般来说,在设计硬件之前,应该考虑最典型和最通用的 VNF,并仔细地选择适当的匹配表参数.

由于 C_S 和 C_m 函数非线性且与具体硬件设计相关,因此对于式(1)想找到最优解较困难,在此,本文仅给出公式化分析,具体的匹配表宽度与深度的设置可根据具体硬件资源进行确定.

3.2 资源分割优化算法

DMT 需要对存储资源进行分割,并动态地分配给各 VNF. 而为了提高资源利用率,可对资源分割进行优化. 在 FARD 结构中,不同 VNF 的虚拟匹配表宽度和深度一般不同. 因此,应当根据 VNF 的需求和硬件存储资源的实际情况,动态分配硬件资源.

该优化问题可以表述为:

$$\min: M_{\mu,h} = \sum_{i=1} M_{i,\mu,h} \tag{3}$$

约束条件为:

$$\forall i > 0: \min m \leqslant m_k \leqslant \max m \tag{4}$$

$$\forall i > 0: \min D \leqslant D_k \leqslant \max D \tag{5}$$

$$\max m \leqslant \mu \leqslant \max \mu \tag{6}$$

$$\min h \leqslant h \leqslant \max h \tag{7}$$

在式(3)~(7)中:$M_{\mu,h}$为匹配表存储资源总开销;m_k 为第 k 个 VNF 匹配宽度;max$m$,min$m$ 为 VNF 匹配表宽度值的上下限;D_k 为第 k 个 VNF 匹配深度;max$D$,min$D$ 为 VNF 匹配表深度值上下限;μ 为优化过程中所使用匹配表宽度;maxμ 为匹配表存储资源宽度上限;h 为匹配表深度;min$h$,max$h$ 为匹配表存储资源深度的下限和上限.

上述资源分割优化问题,可以归纳为一个整数线性规划(Integer Linear Programming,ILP)问题. 该问题的经典的求解方法有采用割平面法和分支定界法进行求解. 但是考虑到采用割平面法经常会发生收敛速度较慢的情况,因此该文采用“分支定界法”求解资源分割优化过程中所涉及的 ILP 问题.

4　性能测试与分析

为了解 FARD 结构的工作流程并测试其数据处理性能和资源开销,该文基于 NetFPGA-10G 平台实现了 FARD 原型,该平台与部署了 VNF 的通用服务器(2.5GHz 64bit CPU,16GB DDR2 RAM)连接. 该方案实现 4 个 VNF 的并行处理,使用的资源包括 4 个 10Gbps 的物理端口和 4 个连接了 PCI-E 总线并支持直接内存访问(Direct Memory Access,DMA)的 CPU 虚拟端口,并用 TCAM 和 hash 表来存储和匹配相关的功能规则,并将匹配表分为匹配宽度为 32bit 的 8 个小匹配表.

4.1　转发性能对比

首先通过实验测试不同包长度下 FARD 的处理速率. 实验中,采用 OpenVZ 作为虚拟层,4 个虚拟路由器(vRouters)作为 VNF 实例,由 Spirent TestCenter 来测量数据平面的数据吞吐率.

图 5 显示当 VNF 数量增加时,FARD 吞吐量的变化情况. 从图中可知,FARD 的吞吐量随着 VNF 的增加而增长,在 VNF 达到 3 个以上时,达到最高处理速度 40Gbps. 同时,数据包长度对处理速率有一定应影响,在 VNF 较少时,数据包长度越长,吞吐量越高.

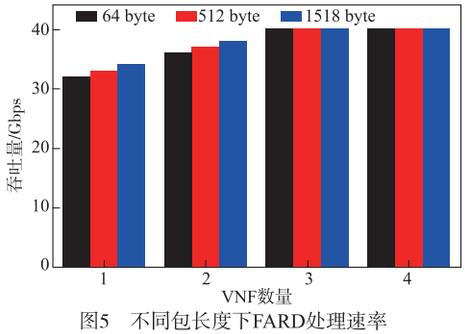

图5　不同包长度下FARD处理速率

为了测试未加速、DPDK 和 FARD 的吞吐率,本实验采用了两块 10G Intel 网卡进行测试,测试主机 CPU 为 Intel Core i5-3450,主频 3.10GHz,内存 8GB,操作系统为 64 位 Ubuntu14.04,FARD 性能测试基于 NetFPGA-10G 平台,转发包长度为 1024B. 图 6 显示了三种不同方案的包处理速率. 实验结果表明,并行处理的 VNF 数量越多,吞吐率越高,当 VNF 数量达到 3 个以上时,不同方案的吞吐量稳定在各自的性能上限. 从实验结果中可以看出,当 VNF 数量达到 4 个时,FARD 中的数据包吞吐量将达到 40Gbps,实现 NetFPGA-10G 的线速处理. 而在同样情况下,未加速时 VNF 最大吞吐量约为 680Mbps,DPDK 最大吞吐量约为 19Gbps. 可见,FARD 的最大吞吐量约为 DPDK 加速方案的 2 倍、纯软件 VNF 的 60 倍,表明通过 FARD 加速结构能使 VNF 获得更好的包处理性能.

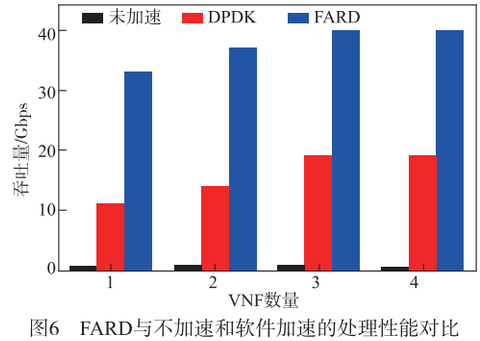

图6　FARD与不加速和软件加速的处理性能对比

4.2　资源开销分析

为了测试 FARD 的资源开销,首先对基于 NetFPGA-10G 平台的 OpenFlow1.0 和 FARD 结构的综合性能进行比较. 实验中,OpenFlow 和 FARD 的数据平面采用相同尺寸的匹配表(总匹配宽度为 256bit)和相同的包头域长度(256bit),所用到的 Slice 资源数目如表 2 所示.

表 2　FARD 和 OpenFlow 资源开销对比

可编程硬件结构	Slice 数目	利用率
OpenFlow1.0	27816	74%
FARD	28454	76%

从表 2 中可以看到,同样作为可编程硬件结构,OpenFlow 1.0 的 Slice 资源开销为 27816,占总体片内资源的 74%,FARD 的 Slice 资源开销为 28454,占片内资源的 76%. 相比于 OpenFlow1.0,虽然 FARD 需要使用额外的 Slice 资源来支持一些功能定义的解析状态机,但是它与 OpenFlow1.0 数据平面的资源开销仍然非常接近. 对于 OpenFlow 交换机,它需要解析和处理 12 元组匹配域,这个过程需要一个较为复杂的状态机来识别各层包头类型. 而 FRAD 用功能配置代替状态机,节省很多 Slice 资源. 但由于 FARD 结构中需要支持 DMT,因此开销会略高于 OpenFlow. 实验结果显示,FARD 结构的资源开销与 OpenFlow1.0 接近,在可接受的范围之内.

下面测试资源分割优化的效果. 实验中,VNF 的虚拟匹配表的约束条件取为:64bit≤w_i≤128bit,32≤d_i≤64. 对于匹配表,宽度和深度上限分别为 72bit,48. 实验结果如图 7 所示,随着 VNF 需求数量的增加,所需要的匹配表存储资源也随之增加,当 VNF 一定时,采用 ILP 算法优化后,所需的匹配表数量较优化前有所下降,能够有效地降低匹配所需的硬件资源开销.

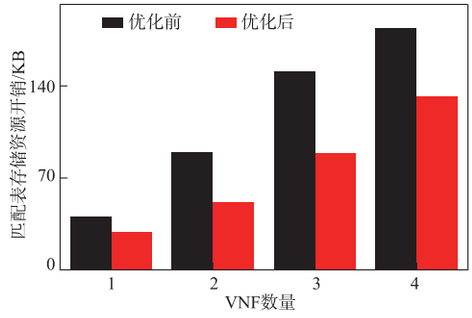

图7 ILP算法优化前后所需匹配表存储资源对比

5 结论

该文针对 NFV 架构处理性能受限问题,提出了一种基于硬件加速的 VNF 处理结构:FARD. 通过使用可编程的解析器和动作处理器,FARD 能够实现硬件加速结构的功能自适应性. 并通过使用 DMT,实现数据平面的资源可分割. FARD 结构为 NFV 包处理加速进一步创新提供可能.

参考文献

[1] M. Chiosi, et al. Network Functions Virtualization [EB/OL]. http://portal. etsi. org/NFV/NFV White Paper. pdf,2012-10-24.

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

[3] Matias J, Garay J, Toledo N, et al. Toward an SDN-enabled NFV architecture [J]. IEEE Communications Magazine, 2015,53(4):187 – 193.

[4] 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.

[5] Kablan M, Caldwell B, Han R, et al. Stateless network functions[A]. Workshop on Hot Topics in Middleboxes and Network Function Virtualization [C]. New York, NY, USA:ACM,2015. 221 – 234.

[6] 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.
Dun 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)

[7] 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]. Piscataway, NJ, USA: IEEE Press,2013. 13 – 24.

[8] Bosshart P, Gibb G, Kim H S, et al. Forwarding metamorphosis: fast programmable match-action processing in hardware for SDN[J]. ACM SIGCOMM Computer Communication Review,2013,43(4):99 – 110.

[9] Bosshart P, Daly D, Gibb G, et al. P4: programming protocol-independent packet processors [J]. ACM SIGCOMM Computer Communication Review,2014,44(3):87 – 95.

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

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

段 通 男,1992 年生于河南驻马店. 现为国家数字交换系统工程技术研究中心博士研究生. 主要研究方向为可编程数据平面.
E-mail:duantong21@ 126. com