专题：SDN/NFV 技术与应用

网络功能虚拟化场景下的并行加速

毕军，孙晨，于恒
（清华大学，北京 100084）

摘　要：在网络功能虚拟化场景下，每个网络功能都以软件的形式来实现。但传统的串行网络功能组链方式将带来极大的性能损耗。而如今针对网络功能进行加速的工作，主要集中在优化该串行链的每个组成成分上。提出了一个高性能的系统框架，通过让网络功能并行地对数据分组进行处理，提高网络功能虚拟化场景下服务链的整体性能。该系统由 3 个部分组成。首先，该系统为管理员提供了一个策略描述语言来直观地描述串行或并行的组链意图。然后，该系统的编排器智能地鉴别网络功能之间的依赖性，并基于所提供的策略，生成高性能的“服务图”。最后，该系统的底层实现通过执行轻量级的数据分组复制、分布式的并行分组转发和负载均衡后的数据分组合并来支持网络功能的并行处理。基于 DPDK 技术，在 Linux 容器中实现了该系统的原型机。通过实验验证可知，该系统能极大地减少真实世界中服务链的处理时延。

关键词：网络功能虚拟化；并行；服务链
中图分类号：TP393
文献标识码：A
doi: 10.11959/j.issn.1000−0801.2018216

NFP: enabling network function parallelism in NFV

BI Jun, SUN Chen, YU Heng
Tsinghua University, Beijing 100084, China

Abstract: Software-based sequential service chains in network function virtualization (NFV) could introduce significant performance overhead. Current acceleration efforts for NFV mainly target on optimizing each component of the sequential service chain. However, based on the statistics from real world enterprise networks, it was observed that 53.8% network function (NF) pairs could work in parallel. In particular, 41.5% NF pairs could be parallelized without causing extra resource overhead. NFP was presented, a high performance framework, that innovatively enabled network function parallelism to improve NFV performance. NFP consisted of three logical components. Firstly, NFP provided a policy specification scheme for operators to intuitively describe sequential or parallel NF chaining intents. Secondly, NFP orchestrator intelligently identified NF dependency and automatically compiled the policies into high performance service graphs. Thirdly, NFP infrastructure performed light-weight packet copying, distributed parallel packet delivery, and load-balanced merging of packet copies to support NF parallelism. An NFP prototype based on DPDK in Linux containers was implemented. The evaluation results show that NFP achieves significant latency reduction for real world service chains.

Key words: NFV, network function parallelism, service chain

收稿日期：2018−06−21；修回日期：2018−07−06

1 引言

为解决传统的专用 middlebox[1]所带来的一系列问题，网络功能虚拟化（network function virtualization，NFV）基于商用服务器硬件，利用虚拟化技术，实现各种各样的网络功能（network function，NF）。在运营商网络[2]、数据中心[3-4]、移动网[5]和企业网[6]中，网络管理员通常需要数据流以特定的顺序穿过多个 NF（例如依次通过防火墙、入侵检测系统和代理服务器）[7-9]，而以上需求通常被定义为服务组链需求。同时，软件定义网络（software defined networking，SDN）也常被用来在各个特定的 NF 之间进行导流，以执行该组链策略[3, 9-14]。所以 NFV 和 SDN 技术能共同提升在串行服务组链场景下的灵活性与动态性。

NFV 在带来如此多好处的同时，也伴随着一些问题出现[15]。基于软件实现的 NF 尤其容易带来较明显的性能损失。并且，服务链所带来的时延将随着组链长度的增加而线性增加。而对某些时延敏感型应用来说，这种情况将是不可接受的[16-17]。

对于在 NFV 场景下基于软件实现的 NF 所带来的性能缺陷问题，国际上已有多项研究工作来对其进行改善。通过图 1 所示的服务链来展示各自的优化点。

- 单个 NF 的加速：ClickNP[17]提出了将部分本应由软件实现的逻辑功能通过可编程硬件（例如 FPGA）实现的方式加速单个 NF 的处理速度。而 NetBricks[18]则遗弃了传统的基于虚拟机或容器技术的 NF 实现方式，通过将同一服务链中的所有 NF 运行在单个处理核上的方式来提高性能。
- 数据分组传递的加速：Intel DPDK[19]、ClickOS[20]和 NetVM[21-22]通过优化数据分组从网卡端口到虚拟机和虚拟机之间的传递过程来提高性能。
- NF 模块化：OpenBox[7]对 NF 进行了模块分解（在参考文献[6, 10, 23]中也有提到），然后通过在串行服务组链时，共享 NF 之间相同的组建模块，提高服务链的整体性能。

而以上研究都是在水平范围内对 NF 的性能问题进行改善。例如，对服务链中的每个 NF 水平地进行了加速，但仍然保持了所有 NF 之间串行的组链方式。

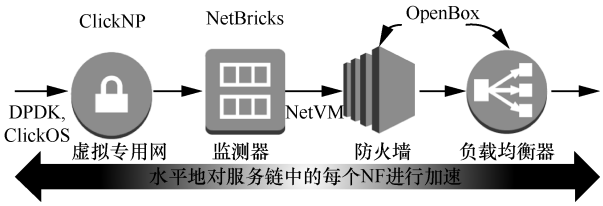


图 1 传统的串行服务组链方式[4]

但是，通过对服务链中 NF 的细致观察发现，一些 NF 之间并不存在任何依赖关系，可以并行地对数据分组进行处理。在如图 1 所示的服务链中，monitor（监测器）仅仅维护数据分组的统计数据而不需要对数据分组做任何修改。所以，如图 2 所示的服务框架，可以同时将数据流送入 monitor 和 firewall（防火墙）中，然后保留防火墙中的数据流，该方式将获得和串行服务组链情况下相同的处理结果。通过这种方式，该服务链的相对长度将减少为 3，从而在理论上减少 25%时延。

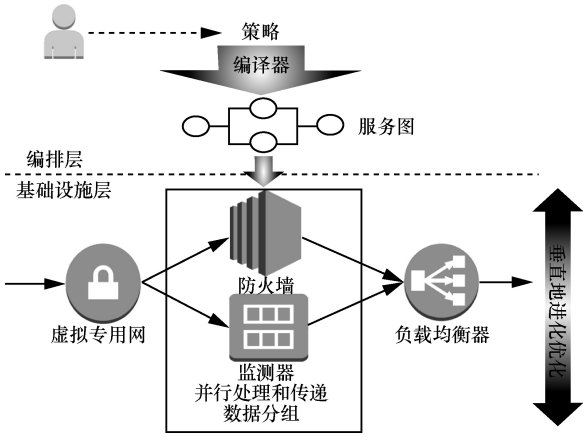


图 2 支持 NF 并行的系统框架

因此，不同于以上所提及的所有 NFV 加速思想，本文抓住了在垂直范围内对 NFV 性能进行提

升的机会。基于以上的观察，提出了一种高性能的系统框架，创新性地引入 NF 并行的思想以减少 NFV 场景下时延。如图 2 所示，该系统框架由三大逻辑部分组成，包括策略描述语言、系统编排器和基础设施层。

2　动机与挑战

首先讨论了在 NFV 场景下引入 NF 并行思想的动机，然后介绍了在设计过程中所面临的挑战。

2.1　动机

NF 并行将极大地降低时延：为了获得在引入并行思想后更直观的优化效果，收集了在企业网中常见的 NF，其所进行的操作和所占的比例见表 1[1,6]，其中，%列代表其在企业网中所占的百分比，R 代表读操作，W 代表写操作，T 代表执行操作。基于这些统计数据，计算出 53.8%的 NF 之间都能对数据分组并行进行处理，这将给 NF 并行带来极大的优化空间。

NF 并行能兼容其他加速技术：首先，对于单个 NF 性能的加速技术[17-18]，NF 并行思想能对每个加速后的 NF 进行并行从而获得更高的性能。然后，能利用快速数据分组传递技术[19, 21-22]在并行情况下加速数据分组的传递过程。最后，无论是一个整体的 NF，还是模块化的 NF，都能从并行中受益。

2.2　设计挑战

基于以上动机，提出了一个创新性的系统框架，将 NF 并行的思想引入 NFV 场景以提高其性能。总结了在该系统设计过程中的 4 个关键挑战，具体如下。

（1）服务图策略描述语言的设计

为了支持 NF 的并行，需要一种新的更直观的方式来描述 NF 串行和并行的组链意图，以此来获得更优化的并行服务图。本文提出了一种拥有更丰富语义的策略描述语言来解决该问题。

（2）构建并行服务图的系统编排器设计

支持并行对编排器分析 NF 之间的依赖性和自动将策略描述语言编译为高性能服务图的能力提出了挑战。但是，人工对所有 NF 对之间的依赖关系进行分析，不仅耗费时间，而且缺乏可扩展性。为了解决该问题，提出了 NF 依赖性分析原则，通过在编排器中运行特定的算法，自动检测 NF 之间的依赖性。

（3）减少资源损耗的系统编排器设计

引入 NF 并行的思想，可能需要对每个数据分组进行一次甚至多次的复制。因此，系统编排器在构建高性能服务图时，还面临着最小化资源损耗的挑战。一种解决方法是采用 consolidation 的思想[6, 21]，将服务图中所有的 NF 放在同一台硬件服务器上运行，以便只在内存中存储复制的数据分组。为了获得更好的优化效果，该系统编排器还需智能地进行分析，以争取在无需进行数据分组复制的情况下实现 NF 之间的并行。

表 1　常用 NF 及其对数据分组的操作 [1, 6, 24]

虚拟网络功能	产品	%	源 IP 地址	目的 IP 地址	源端口	目的端口	负载	增添/移除	丢弃
防火墙	iptables	26%	R	R	R	R			T
入侵检测系统	NIDS	20%	R	R	R	R	R		
网关	Cisco MGX	19%	R	R					
负载均衡器	F5, A10	10%	R/W	R/W	R	R			
缓存器	Nginx	10%	R		R		R		
虚拟专用网	OpenVPN	7%	R	R			R/W	T	
地址转换器	iptables		R/W	R/W	R/W	R/W			
流量代理	Squid		R/W	R/W					
流量压缩器	Cisco IOS						R/W		
流量整形器	Linux tc								
监测器	NetFlow		R	R	R	R			

（4）支持 NF 并行的基础设施层设计

首先，该基础设施层需要引入一个轻量级的数据分组复制机制来最小化复制时所带来的性能损耗。然后，该基础设施层还需要一个合并模块来对来自多个 NF 处理后的多个数据分组进行合并。最后，目前对 NF 之间的数据分组进行转发的方案是基于一个集中的虚拟交换机[19-21]。但是，在该集中式交换机中的排队时延将降低性能。

3 策略描述语言定义

对于传统的串行服务组链方式，网络管理员只需定义一种语言，为串行服务链中的每个 NF 分配具体的位置即可，见表 2 中的第 1 行。但是，本系统希望构建支持并行的高性能服务图，所以需要一种更直观的方式来描述串行和并行的 NF 组链意图。因此，本系统定义了 3 种策略描述规则，网络管理员可以通过将几种描述规则组合起来表达自己的组链意图。对示例的串行组链和策略语言组链的不同表达方式见表 2，其中，FW 代表防火墙，LB 代表 load balance（负载均衡），VPN 代表 virtual private network（虚拟专用网络）。

表 2 对示例的串行组链和策略语言组链的不同表达方式

描述方法	表达方式
图 1 的传统串行服务组链描述方法	assign(VPN, 1) assign(monitor, 2) assign(FW, 3) assign(LB, 4)
图 1 的策略语言服务组链描述方法	order(VPN, before, monitor) order(monitor, before, FW) order(FW, before, LB)
图 2 的并行服务图组链描述方法	position(VPN, first) order(FW, before, LB) order(monitor, before, LB)

order（NF1，before，NF2）：这条规则描述了两个 NF 之间需要的执行顺序。如图 1 所示的服务链，网络管理员可以通过指定 order（VPN，before，monitor）让数据流首先通过 VPN，再通过 monitor。然后，系统编排器会在 NF 之间的 order 规则中寻求并行执行的机会来提高性能。

priority（NF1 > NF2）：在该系统中，网络管理员需要能描述两个 NF 之间并行执行意图的规则。但是，这两个 NF 之间的动作可能会产生冲突。例如，firewall 和 IPS 会在是否丢弃某个数据分组的意见上产生分歧。因此，网络管理员可以通过指定 priority（IPS > firewall）规则来指示系统，当两个 NF 并行时，如果发生冲突，则采用 IPS 的处理结果。

position（NF，first/last）：图 1 所示的数据中心服务链[4]需要所有的数据分组首先被 VPN 处理。这提出了将某个 NF 放于服务图中特定位置的需求。但是，由于不能预先知道经过优化的服务图的最终结构，只能为某个 NF 在服务图中指定最前或最后的位置。设计了 position（NF1，first/last）规则来描述这种需求。例如，可通过 position（VPN，first）规则来保证所有的数据分组首先穿过 VPN。

4 编排器设计

本系统编排器以策略描述规则作为输入，并自动分析 NF 之间的依赖关系，以尽可能并行的方式将这些规则编译为高性能的服务图。

（1）NF 并行可能性的分析

对于策略描述语言中以 priority 规则定义的两个 NF，可直接视为可并行的两个 NF。对于以 position 规则定义的两个 NF，可直接以串行的方式将其放置于服务图的头部或尾部。但是，对于以 order 规则定义的两个 NF，需要进一步分析其并行的可能性。为了分析这 NF 之间是否可并行，提出了结果一致性原则：当两个 NF 并行时对相同数据分组的处理结果和每个 NF 各自的内部状态与串行服务组链时的处理结果和内部状态都相同时，本文认为这两个 NF 是能并行执行的。例如，假设 NF1 对分组头进行读操作，而 NF2 接着对相同的分组头域进行修改。为了保证 NF1 读到的是分组头的原始信息，而不是经过 NF2 修改后的信息，可以对数据分组进行复制，然后将这两份数据分组并行地发送

到这两个 NF 中。若 NF1 首先对分组头进行写操作，而 NF2 再接着对分组头进行读操作，该管理员希望传递给 NF2 的数据是经过 NF1 修改后的结果。则此时，这两个 NF 应该以串行的方式来组链。

（2）减少额外资源损耗

在 priority 规则或可并行的 order 规则下，如果两个 NF 之间能通过复制数据分组的方式来并行执行，则会招致额外的资源损耗。为了解决这个问题，该系统引入了许多优化技巧来减少数据分组复制时的额外资源损耗。本文根据结果一致性原则，总结出了不需要数据分组复制就能支持并行的情况。例如，假设 NF1 和 NF2 都对数据分组进行读操作，因为读操作并不对数据分组进行任何修改，所以这两个 NF 能同时对数据分组进行读操作。为了进一步减少数据分组复制所带来的额外资源损耗，提出了以下资源优化的技巧。

- 内存重利用：对于读—写和写—写操作的情况，需要分析这两个操作是否在同一个操作域中执行来判断是否对其进行数据分组复制。如果这两个 NF 读或写该数据分组中的不同域，则它们能同时对该数据分组安全地进行操作。
- 只复制数据分组头部：表 1 中，观察到只有 7%的 NF 需要对数据分组的有效载荷进行修改。并且，在数据中心中数据分组的平均长度为 724 byte[25]。据此可知，分组头仅占整个分组长度的 8.8%。因此，对于一些 NF 并行的情况，本文提出了只复制数据分组头部的技巧。

（3）NF 并行可能性分析算法

基于以上提出的 NF 并行可能性分析和资源优化技巧，本文提出了 NF 并行可能性分析算法来对以 order 规则定义的两个 NF 进行分析。该系统编排器以 order 规则定义的两个 NF 作为输入。这两个 NF 可能出现不需复制即可并行、需复制才能并行和不可并行 3 种结果。首先，该算法从 AT 表中获取这两个 NF 要执行的所有操作。然后该算法穷尽地对这两个 NF 中的所有操作对进行分析以确定其是否有并行的可能性。对于读—写和写—写的情况，本文需要进一步确认这两个操作是否针对同一域。如果这两个 NF 在复制数据分组的情况下可以并行，还需对冲突的动作进行记录。最后，该算法将产生这两个 NF 是否能并行的结果以及是否存在冲突的操作。

（4）服务图构建

该系统编译器首先将策略描述规则转换为预先定义的数据结构，然后将这些数据结构转化为独立的子图，最后将这些子图合并，产生最后的服务图。

（5）将策略转化为数据结构

本文设计了两种类型的数据结构来存储策略，如图 3 所示。对于 position 规则，通过图中左边所示的数据结构来保存该 NF 的类型和其在服务图中的位置。对于 order 规则，本文通过算法 1 来确定其是否能并行处理和可能的冲突操作。order 和 priority 规则最后都被转换成了如图 3 右边所示的数据结构。

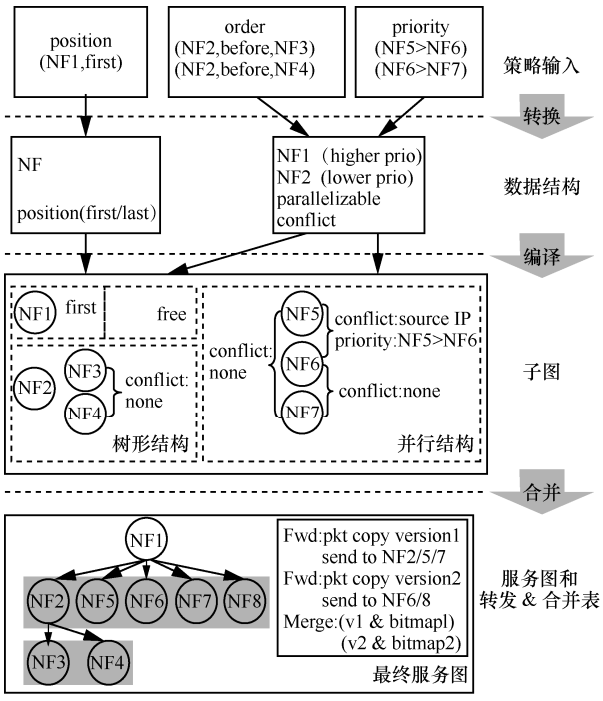


图 3　并行服务图构建过程

（6）将数据结构编译为子图

首先将不能并行的 NF 以串行的方式进行组链（如图 3 中的 NF2、NF3 和 NF4），然后将具有同一 NF 的数据结构组合成子图。该系统拥有 3 种类型的子图结构，包括单个 NF（例如 NF1、NF8）、树形结构（例如 NF2、NF3 和 NF4）和并行结构（例如 NF5、NF6 和 NF7）。

（7）将子图合并成服务图

将子图进行合并，以产生最后的服务图。首先，以 position 规则定义的 NF 被放置到服务图的头部或尾部（NF1）。然后，将每一个生成的子图（包括单个的 NF）视为一个 NF，穷尽地对每一个子图对进行分析，以确定其之间是否存在依赖性。最后，将所有的独立子图并行放置（如图 3 所示）。基于对 NF 的依赖性分析，还生成了数据分组转发表和合并表。

5 基础设施层的设计

图 4 展示了该基础设施层的设计全貌。正如前面所提到的，该系统为了避免占用额外的带宽资源，采用了 consolidation 的设计思想。为了能在 NF 之间进行数据分组传递时获得高性能，采用了零数据分组复制的导流方法[20-21]。

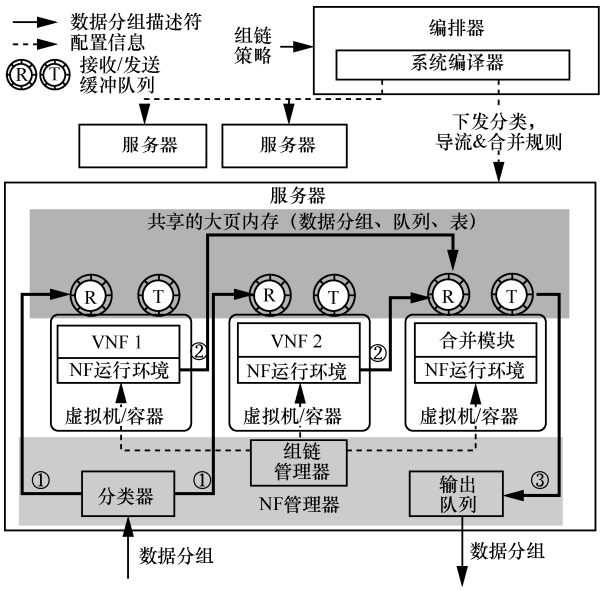


图 4 基础设施层设计架构

图 4 中的加粗实线代表该系统中的数据分组处理流程。当数据分组到达服务器时，通过一个分类器将数据分组描述符发送到相应的服务图上。对于数据分组的传递，本文设计了一个分布式的 NF 运行环境，高效地在各个 NF 之间并行地传递数据分组。最后，同一数据分组的多个版本被送入合并模块中，以产生最后的输出，并且该模块能被编排器动态地进行配置。

数据分组分类器：该分类器模块从网卡处取得到达的数据分组，并找出与之相应的服务图信息，包括合并模块中需要多少份复制的数据分组、如何对这些复制的数据分组进行合并以及该服务图的第一跳是哪个 NF 等信息。

NF 之间的数据分组传递：在 NF 对数据分组进行处理后，该系统需将该数据分组传递到服务图中相应的下一个 NF 中，并在需要时对其进行复制。如前所述，用集中的虚拟交换机进行数据分组转发可能会导致性能下降。为了解决该问题，该系统将数据分组的转发任务进行分解，并利用每个 NF 来独立并行地将数据分组转发至下一个 NF 中。图 4 中，对数据分组处理完毕后，由该 NF 的运行环境将该数据分组的描述符复制到下一个 NF 的接收队列中，以实现数据分组的传递。

数据分组合并模块：在所有的 NF 都对数据分组处理完毕后，该数据分组的多个版本被送入合并模块中来产生最后的输出。为处理每个数据分组的所有版本，该合并模块需承受巨大的负载，并有可能因此成为整个系统的性能瓶颈。为了解决这个挑战，本文提出了在同一台服务器中部署多个合并模块的方法，并设计了合并代理以在多个合并模块中进行负载均衡。

6 系统实现与实验评估

本文基于 16.11 版本的 DPDK 实现了该系统原型和 NF 操作收集工具。基于两个配备有两块 Intel(R) Xeon(R) E5-2690 v2 CPU（3.00 GHz, 10 个

物理核）、256 GB RAM 和两块 10 Gbit/s 网卡的服务器组成的试验床对其性能进行评估。该服务器为运行版本号为 4.4.0-31 的 Linux 内核，并利用 Docker 容器技术[25]来运行 NF。

对于测试所用的流量，通过在另一台服务器上运行基于 DPDK 的数据分组发送器来生成数据分组，并通过网线与测试服务器直连。该分组发送器通过发送并接收数据分组来测试时延和没有分组丢失情况下的最大吞吐率。为了评估该系统，参考以往研究，实现了 L3 forwarder（转发器）、load balancer（负载均衡器）、firewall（防火墙）、IDS、VPN、monitor（监视器）等 NF。

6.1　性能提升

串行服务链的性能：在 OpenNetVM 和本系统中以相同的方式实现了一个 L3 forwarder，并将多个相同的 L3 forwarder 组成一条串行的服务链。使链的长度从 1 到 5 发生改变。相比 OpenNetVM，本系统会带来较小的时延损耗，但在任意数据分组大小时，都能达到线速的转发性能。

NF 复杂度的影响：对在本系统中实现的 6 个 NF 的性能进行测量，将并行度控制为 2。本文还比较了在 64 byte 小分组的情况下，需要进行数据分组复制和不需要进行数据分组复制时的优化程度。可以发现，随着 NF 复杂度的增加，NF 并行化所带来的时延优化程度也增加。

NF 并行度的影响：为了获得并行度与优化效果的关系，将并行的 firewall 数量从 2 一直增加到 5，分析复制或不复制数据分组时的性能。可以观察到随着 NF 并行度的增加，当不需要复制数据分组时，时延减少比例从 33%升至 52%；而当需要复制数据分组时，时延减少比例也能达到 32%。因此能得出结论，当 NF 并行度增加时，时延也减少得更多，而吞吐率却不怎么受影响。

6.2　额外损耗

额外的资源损耗：为了评估该系统在复制过程中所带来的资源损耗，本文计算了额外资源占用率与 TCP 数据分组大小（64 ~ 1 500 byte）和 NF 并行度的函数关系。对于以太网中任何长度的 TCP 数据分组，其复制仅需额外占用 64 byte 的内存。由资源损耗（ro）、数据分组长（s）和并行度（d）构造了一个等式：$ro = 64 \times (d-1) / s$。本文根据数据中心的分组长度分布可计算出，该系统的额外资源损耗为 $ro = 0.088 \times (d-1)$。当并行度为 2 时，额外的资源占用率为 8.8%，但却能获得 30%的时延减少。

复制与合并时的性能损耗：对于数据分组复制的实现，本文利用由 DPDK 提供的经过优化后的快速内存复制接口来减少复制时的性能损耗。数据分组复制与合并仅带来了平均 9 μs 的时延损耗和极小的吞吐损耗。但相比串行的组链方式，仍能获得 20%的时延减少。

7　结束语

本文提出了一个高性能的框架，通过创新性地在 NFV 场景下引入 NF 并行的思想来提高 NFV 的性能。该系统通过定义一种新的策略描述语言，让网络管理员更直观地描述其 NF 之间串行或并行的服务组链意图。然后该系统编排器将这些策略描述规则以较小的资源损耗为代价编译为可并行的高性能服务图。最后，该系统的基础设施层执行数据分组复制，导流和合并的操作来支持 NF 之间的并行处理。本文基于 Linux 的容器技术实现了该系统的原型机，并验证了其性能的优化效果和所带来的额外资源损耗。

参考文献：

[1] SHERRY J, HASAN S, SCOTT C, et al. Making middleboxes someone else's problem: network processing as a cloud service[J]. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 13–24.

[2] QUINN P, NADEAU T. Service function chaining problem statement: draft-ietf-sfc-problem-statement-10 [S]. 2014.

[3] JOSEPH D A, TAVAKOLI A, STOICA I. A policy-aware switching layer for data centers[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(4): 51-62

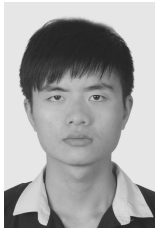
[4] IETF SFC WG. Service function chaining use cases in data centers[S]. 2015.

[5] HAEFFNER W, NAPPER J, STIEMERLING M, et al. Service function chaining use cases in mobile networks: draft-ietf-sfc-use-case-mobility-01[S]. 2014.
[6] SEKAR V, EGI N, RATNASAMY S, et al. Design and implementation of a consolidated middlebox architecture[C]//The 9th USENIX Conference on Networked Systems Design and Implementation(NSDI'12), April 25-27, 2012, San Jose, CA, USA. New York: ACM Press, 2012: 24.
[7] BREMLER-BARR A, HARCHOL Y, HAY D. OpenBox: a software-defined framework for developing, deploying, and managing network functions[C]//The Workshop on Hot Topics in Middleboxes and Network Function Virtualization, August 22-26, 2016, Florianópolis, Brazil. New York: ACM Press, 2016: 511-524.
[8] HALPERN J, PIGNATARO C. Service function chaining (SFC) architecture: draft-ietf-sfc-architecture-07 [S]. 2015.
[9] QAZI Z A, TU C C, CHIANG L, et al. SIMPLE-fying middlebox policy enforcement using SDN[J]. ACM SIGCOMM Computer Communication Review, 2013, 43(4):27-38.
[10] ANWER B, BENSON T, FEAMSTER N, et al. A slick control plane for network middleboxes[C]// ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, August 16, 2013, Hongkong, China. New York: ACM Press, 2013: 147-148.
[11] FAYAZBAKHSH S K, CHIANG L, SEKAR V, et al. Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags[C]//Usenix Conference on Networked Systems Design and Implementation, USENIX Association (NSDI'14), April 2-4, 2014, Seattle, WA, USA. New York: ACM Press, 2014: 533-546.
[12] GEMBER-JACOBSON A, VISWANATHAN R, PRAKASH C, et al. OpenNF: Enabling innovation in network function control[C]//The 2014 ACM Conference on SIGCOMM, August 17-22, 2014, Chicago, Illinois, USA. New York: ACM Press, 2014: 163-174.
[13] 王歆平, 王茜, 刘恩慧, 等. 基于 SDN 的按需智能路由系统研究与验证[J]. 电信科学, 2014, 30(4): 8-14.
WANG X P, WANG Q, LIU E H, et al. Research and verification on SDN-based on-demand smart routing system[J]. Telecommunications Science, 2014, 30(4): 8-14.
[14] ETSI NFV ISG. Network functions virtualisation: an introduction, benefits, enablers, challenges and call for action [R]. 2012.
[15] GANDHI R, LIU H H, HU Y C, et al. Duet: cloud scale load balancing with hardware and software[C]//The 2014 ACM Conference on SIGCOMM, August 17-22, 2014, Chicago, Illinois, USA. New York: ACM Press, 2014: 27-38.
[16] LI B, TAN K, LUO L L, et al. ClickNP: highly flexible and high-performance network processing with reconfigurable hardware[C]//The 2016 ACM Conference on SIGCOMM, August 22-26, 2016, Florianópolis, Brazil. New York: ACM Press, 2016: 1-14.
[17] PANDA A, HAN S, JANG K, et al. NetBricks: taking the V out of NFV[C]//The 12th USENIX Conference on Operating Systems Design and Implementation (OSDI'16), November 2-4, 2016, Savannah, GA, USA. New York: ACM Press, 2016: 203-216.
[18] Intel. Data plane development kit[R]. 2018.
[19] MARTINS J, AHMED M, RAICIU C, et al. ClickOS and the art of network function virtualization[C]// USENIX Conference on Networked Systems Design and Implementation, April 2-4, 2014, Seattle, WA, USA. New York: ACM Press, 2014:459-473.
[20] HWANG J, RAMAKRISHNAN K, WOOD T. NetVM: high performance and flexible networking using virtualization on commodity platforms[J]. IEEE Transactions on Network and Service Management, 2015, 12(1): 34-47.
[21] ZHANG W, LIU G, ZHANG W, et al. OpenNetVM: a platform for high performance network service chains[C]//The Workshop on Hot Topics in Middleboxes and Network Function Virtualization, August 22-26, 2016, Florianópolis, Brazil. New York: ACM Press, 2016: 26-31.
[22] PALKAR S, LAN C, HAN S, et al. E2: a framework for NFV applications[C]// Symposium on Operating Systems Principles, October 4-7, 2015, Monterey, California, USA. New York: ACM Press, 2015: 121-136.
[23] BREMLER-BARR A, HARCHOL Y, HAY D, et al. Deep packet inspection as a service[C]// ACM International on Conference on Emerging Networking Experiments and Technologies, December 2-5, 2014, Sydney, Australia. New York: ACM Press, 2014: 271-282.
[24] BENSON T, AKELLA A, MALTZ D A. Network traffic characteristics of data centers in the wild[C]// 2010 Internet Measurement Conference, November 1-3, 2010, Melbourne, Australia. New York: ACM Press, 2010: 267-280.
[25] MERKEL D. Docker: lightweight linux containers for consistent development and deployment[J]. Linux Journal, 2014(239): 2.

[作者简介]

毕军（1972−），男，博士，清华大学网络科学与网络空间研究院副院长、计算机科学与技术学科和网络空间安全学科博士生导师，长江学者特聘教授，北京信息科学与技术国家研究中心未来网络理论与应用研究部主任，主要研究方向为互联网体系结构、SDN/NFV、IPv6 安全体系结构等。

孙晨（1992−），男，清华大学计算机系博士生，主要研究方向为 SDN/NFV、网络测量。



于恒（1995−），男，清华大学计算机系博士生，主要研究方向为 SDN/NFV。