·移动互联与通信技术· 文章编号：1000-3428(2018)08-0112-08 文献标志码：A 中图分类号：TP393

# 基于 FPGA 的虚拟网络功能数据包处理加速架构

范宏伟，胡宇翔，兰巨龙

（国家数字交换系统工程技术研究中心，郑州 450002）

**摘 要：** 针对虚拟网络功能（VNF）的数据包处理性能较差的问题，提出一种基于现场可编程门阵列（FPGA）的通用硬件加速器（GHA）架构。GHA 架构在动态可重构 FPGA 上实现数据包处理流水线，提高 VNF 吞吐量并保证不同 VNF 加速器之间的独立性。采用基于离散粒子群优化算法的加速资源优化分配策略，实现加速收益的最大化。实验结果表明，GHA 架构能将 VNF 的吞吐量提升 50.7 倍，加速资源分配策略优化率达到 24.5% 。

**关键词：** 虚拟网络功能；通用硬件加速器；动态可重构；加速资源分配优化；离散粒子群优化

**中文引用格式：** 范宏伟，胡宇翔，兰巨龙. 基于 FPGA 的虚拟网络功能数据包处理加速架构[J]. 计算机工程，2018，44(8)：112-119，126.

**英文引用格式：** FAN Hongwei，HU Yuxiang，LAN Julong. Data packet processing acceleration architecture for virtual network function based on FPGA[J]. Computer Engineering，2018，44(8)：112-119，126.

# Data Packet Processing Acceleration Architecture for Virtual Network Function Based on FPGA

FAN Hongwei，HU Yuxiang，LAN Julong

（National Digital Switching System Engineering and Technology Research Center，Zhengzhou 450002，China）

**【Abstract】** In order to improve the packet processing performance of Virtual Network Function（VNF），this paper comes up with a Field Programmable Gate Array（FPGA）-based General Hardware Accelerator（GHA） architecture. The GHA architecture implements the packet processing pipeline on a dynamically reconfigurable FPGA，increasing VNF throughput and ensuring independence between different VNF accelerators. The Discrete Particle Swarm Optimization（DPSO） algorithm is utilized to optimize the allocation of acceleration resources. Experimental results indicate that GHA architecture can increase the throughput of VNF by 50. 7 times approximately，and the optimization rate of the acceleration resource allocation strategy can reach 24. 5% .

**【Key words】** Virtual Network Function（VNF）；General Hardware Accelerator（GHA）；dynamic reconfigurable；acceleration resource allocation optimization；Discrete Particle Swarm Optimization（DPSO）

**DOI：** 10. 19678/j. issn. 1000-3428. 0050266

## 0 概述

为了满足未来网络的发展需求，由运营商主导发起的新型网络技术——网络功能虚拟化（Network Function Virtualization，NFV）随之兴起。在 NFV 架构中，网络功能由软件实现，并以虚拟网络功能（Virtual Network Function，VNF）的形式部署在通用架构的服务器中，从而实现了网络功能与硬件设备的解耦合，加快网络功能的部署，促进了网络功能的创新。然而，基于软件实现的 VNF 在数据包处理方面性能较差，和专用设备相比，大概有 30% ～40% 的性能损失[1]，是 NFV 部署和发展的主要瓶颈之一。因此，有必要对 VNF 的数据包处理进行加速。

目前，针对 VNF 数据包处理性能较差的问题，主要有软件加速和硬件加速 2 类解决方案。其中，软件方案通过减少虚拟化和 I/O 开销[2-4]来加速数据包处理，但单纯的软件方案难以满足不断增长的数据包处理需求。硬件加速方案是指将 VNF 部分处理任务卸载到硬件中来，利用硬件的高性能来提升 VNF 的数据处理速率。与软件相比，当同样的数据包处理任务由硬件来完成时，可以大幅提高 NFV 系统吞吐率并减少处理时延。在 NFV 背景下，要求

**基金项目：** 国家高技术研究发展计划项目（2015AA016102）；国家自然科学基金创新研究群体项目（61521003）；国家网络空间安全专项课题（2017YFB0803204）。

**作者简介：** 范宏伟（1994—），男，硕士研究生，主研方向为新型网络体系结构；胡宇翔，副研究员；兰巨龙，教授。

**收稿日期：** 2018-01-24 **修回日期：** 2018-02-27 **E-mail：** fhwxd@ foxmail. com

承载 VNF 的加速硬件具有可编程性,由于现场可编程门阵列(Field Programmable Gate Array,FPGA)可以根据需求改变逻辑功能,成为了业内不少研究所采用的硬件加速平台。文献[5]提出了一种基于 FPGA 的 NFV 平台设计,利用 FPGA 实现完整的网络功能,在满足灵活性要求的同时,可获得比拟专用硬件设备的性能。文献[6]提出的 OpenANFV 架构在 OpenStack 的云环境下,实现特定 VNF 的加速性能最大化。文献[7]提出的 ClickNP 架构使用类 C 的高级语言模块化编程 FPGA,并和 CPU 细粒度分工合作,实现高性能和低时延的网络功能。文献[8]提出的 SLA-NFV 架构专注于实现用户的服务水平协议(Service Level Agreement,SLA),通过 FPGA 加速平台增强 NFV 性能。综上,现有的基于 FPGA 的硬件加速方案针对不同的 VNF 进行加速时,加速器的设计与 VNF 之间形成了“绑定”关系,与 NFV 分层解耦的思想相违背,而且 FPGA 的编程难度相对较高,增加了 VNF 开发者的负担,不利于 NFV 生态系统的构建。

基于以上分析,为实现 VNF 数据包处理加速,本文提出一种基于 FPGA 的 VNF 通用硬件加速器(General Hardware Accelerator,GHA)架构。通过设计能够承载多种 VNF 加速的 GHA 架构,并采用 FPGA 动态局部重构的方法在加速板卡上独立动态地部署加速器,保证整体架构的灵活性和功能之间的独立性。采用基于改进离散粒子群优化(Discrete Particle Swarm Optimization,DPSO)的分配策略,为数据中心内的 VNF 分配虚拟加速资源,提升有限加速资源的收益。最后,基于 NetFPGA-10G[9] 板卡实现 GHA 加速平台的原型,并对算法的有效性进行仿真测试。

## 1　设计目标

本文的设计目标是设计一种基于 FPGA 的 VNF 通用加速架构 GHA,能够灵活承载不同的 VNF,同时保证较高的数据处理性能,并实现对硬件资源的高效使用。因此,GHA 应当遵循以下 3 个设计目标:

1)通用、灵活的加速架构。对于 GHA,要能够灵活地支持网络中多样的数据包协议,除了最基本的 IPv4 协议外,还能处理 STT 和 VxLAN 等新型数据包格式,实现对不同 VNF 数据处理的承载,保证架构的通用性。架构的通用性易于实现不同 VNF 对加速平台的调用。

2)支持较高的数据包处理性能。在网络中,通用性和高性能之间往往存在矛盾,为了实现较强的通用性,难免会降低硬件的处理性能。因此,在设计上需要实现通用性和高性能之间的平衡,保证在引入 GHA 加速架构后,能够有效提升 VNF 的数据处理速率,满足用户需求。

3)对硬件资源的高效使用。基于 FPGA 的 GHA 架构要充分利用 FPGA 的动态可重构技术,可独立配置加速器,为不同生命周期的 VNF 实现加速,彼此之间互不影响。被加速的 VNF 生命周期所占用的硬件资源能被重新使用,从而保证对硬件资源利用的高效性。

## 2　数据包处理加速架构

根据以上设计目标,基于 FPGA 硬件平台的 GHA 整体架构如图 1 所示。GHA 架构将整个 FPGA 板卡分为一个静态逻辑区和多个局部可重构区(Partially Reconfigurable Region,PRR)。其中,静态逻辑区在运行时不用重新配置,将不同 VNF 的共享模块部署在静态逻辑区,包括 GHA 架构的基本组件——解析单元,以及负责通信与数据存储的相关资源。每个 PRR 内包含相同数量的逻辑单元和存储单元,彼此间通过总线实现互连,从而构成处理单元,也就是 GHA 架构的另一个基本组件。GHA 架构在单块 FPGA 板卡中可以包含多个处理单元,用于不同 VNF 的数据处理,彼此间相互独立。不同 VNF 处理单元内包含的 PRR 个数由 VNF 的类型和配置信息条目的数量所决定。当旧的 VNF 生命周期结束,有新的 VNF 加速请求到达,相应处理单元内的 PRR 资源将被释放,用来构建新 VNF 的处理单元。解析单元与 VNF 相应的处理单元共同组成 GHA 架构中的加速器,负责卸载到硬件部分的数据包处理任务,加速器的配置信息生成和 VNF 的剩余功能部分则由软件完成。

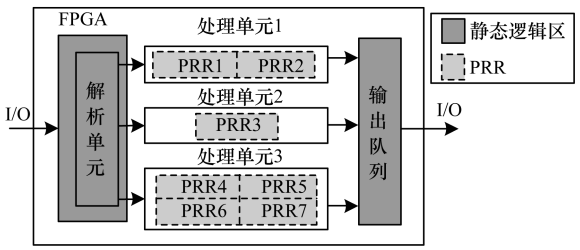


**图 1　GHA 整体架构**

### 2.1　GHA 基本组件

VNF 数据包处理流水线主要由“解析-匹配-动作”这 3 个阶段组成。GHA 基本组件分为解析单元和处理单元 2 个部分,实现对数据包流水线 3 个阶段的处理。为了支持多种类型的 VNF,解析单元采用可编程的解析器结构,根据用户进行配置,识别不

同的包头格式,对到达 GHA 架构的数据包进行解析[10-11]。处理单元对解析结果进行匹配、查找后,利用可编程动作处理器[12-13],根据 VNF 配置需求对数据包进行动作处理。GHA 相对固化简单的流水线架构,能够充分利用硬件的高性能特点,保证了加速器较高的数据包处理速率。GHA 结合 FPGA 的动态重构和可编程特性,通过对 2 个组件的功能设计,将“解析-匹配-动作”这一结构[12]应用到 VNF 加速场景中,能够承载诸多类型 VNF 的部分数据包处理任务,实现了一定的通用性,而且对于不同的 VNF,仅需配置处理单元内相应的表项,不需要重新设计加速器的内部结构,降低了 VNF 加速器设计的复杂性。

解析单元整体结构如图 2 所示。解析单元包含类型域提取模块、类型域查找模块、匹配域提取模块和匹配域组合模块 4 个模块。当数据包到达后,类型域提取模块读取 RAM1 中存放的初始化类型域偏移信息,将初始状态类型域信息提取出来,交给类型域查找模块。类型域查找模块在 TCAM1 中存放的类型域表项信息进行查找,根据查找结果提取出存放在 RAM2 中的相应匹配域和下一状态类型域的偏移量信息。其中,匹配域偏移量信息送到匹配域提取模块,该模块根据偏移量提取相应匹配域,而类型域提取模块则读取下一状态类型域的偏移量信息,进入下一状态类型域的提取。整个提取过程将重复执行,直到提取出所有的类型域和相应匹配域信息。最后将全部的匹配域信息送入匹配域组合模块,由该模块将其组合成包头域,作为该单元输出结果,送往处理单元完成剩余的数据包处理环节。

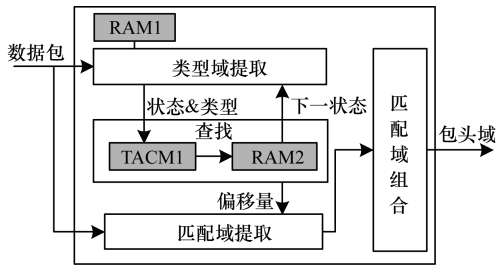


**图 2 解析单元整体结构**

处理单元整体结构如图 3 所示。处理单元包含匹配域选择模块、匹配模块和动作执行器。当数据包的包头域进入处理单元后,首先由匹配域选择模块根据 RAM3 中存放的 VNF 需要的匹配域信息,对包头域内的信息进行筛选,然后将筛选结果送入匹配模块。匹配模块包括匹配、查找 2 个部分。首先该模块读取 TACM2 中存放的 VNF 配置的匹配信息,根据查找结果读取 SRAM 中存放的相应动作字段。动作执行器根据动作字段完成相应报文的添加、删除和修改等动作。

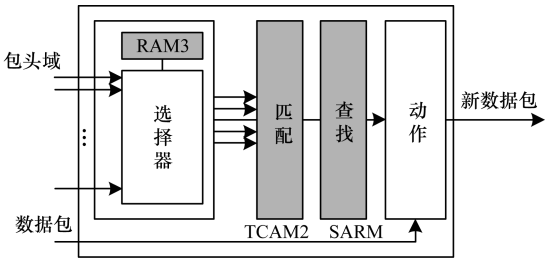


**图 3 处理单元整体结构**

## 2.2 GHA 的部署

作为 VNF 通用加速架构,GHA 需要部署在 NFV 系统中,根据需求为不同类型的 VNF 加速。将 GHA 架构的 FPGA 板卡作为加速硬件(Acceleration Hardware,AH)资源部署到 NFV 基础设施(NFV Infrastructure,NFVI)中,通过 PCIe 总线接入到通用服务器中。

为了使 VNF 能够灵活地调用硬件加速资源,如图 4 所示,利用虚拟化技术将 AH 资源转化为虚拟加速资源。根据 GHA 架构在 FPGA 板卡上的实现方式,将板卡中的 PRR 作为最小粒度的虚拟加速单元;VNF 的加速资源需求同样以 PRR 为单位进行量化,衡量加速资源与 VNF 之间的供求关系。因此,为了提高 AH 资源利用率,所划分的 PRR 内资源数量十分关键,要根据具体的 FPGA 板卡资源总量和不同场景下的 VNF 需求进行确定。

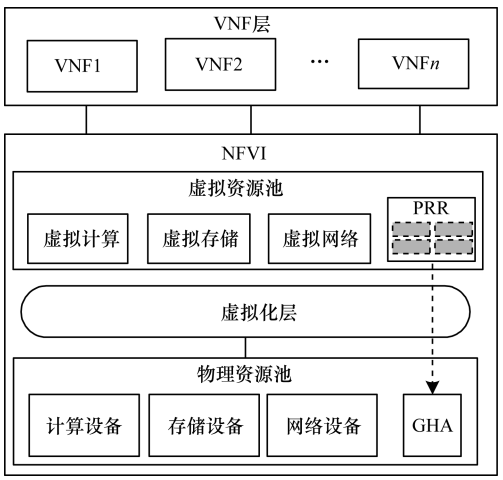


**图 4 GHA 在 NFV 中的部署**

将硬件加速资源虚拟化后,为了减少管理开销,并降低结构的复杂性,可以由虚拟基础设施管理器(Virtual Infrastructure Manager,VIM)负责加速资源的管理和分配[14]。VIM 通过南向接口与 Hypervisor 交互,实现对虚拟加速资源的管理,动态检测加速资源的数量和状态。当加速资源被分配或者释放时,

VIM 会记录资源的上升和下降,并保存到加速资源管理清单中。有新的加速资源加入时,VIM 对其资源数量、能力和链路连接进行初始化和记录。此外,VIM 要对加速资源进行全方面的维护,负责资源的安装和拆卸,升级和软件更新。VIM 要对加速资源的运行情况进行检测,当硬件发生故障时,具有启动恢复能力。

VNF 管理功能块(VNF Management,VNFM)负责 VNF 实例的生命周期管理,包括在部署过程中对所需的虚拟资源的支持。为了实现 VNF 对加速资源的调用,需要对当前 VNFM 规范进行扩展,将虚拟加速资源包含其中。当有新的 VNF 加速请求到达,VNFM 会与 VIM 之间进行通信,将 VNF 的加速资源请求与 VIM 加速资源管理清单中的可用加速资源相匹配。相比于存储和计算资源,硬件加速资源的数量有限,不能满足当前时刻所有 VNF 的加速请求,要根据 VNF 加速资源请求的优先级,尽可能满足优先级高的 VNF。此外,为了保证公平性和稳定性,加速资源应当留有一定的裕量,以满足下一时刻新的 VNF 加速请求。如果 VNF 加速资源请求被满足,根据相应的配置信息对 VIM 分配的 PRR 进行处理单元的重构配置。为了实现 VNF 之间功能的隔离,加速资源一旦被占用,直到 VNF 生命周期结束后才会被释放,并重新成为 VIM 加速资源管理清单中的可用资源。

VNF 与 GHA 之间的映射关系如图 5 所示。用户流量 $T$ 要按顺序依次经过网络中部署的 3 个 VNF。其中,VNF3 由 GHA 承载,VNF1 和 VNF2 仍基于软件进行数据处理。在数据传输、处理时,VNF1 和 VNF2 依旧映射到计算资源内,由 CPU 进行数据处理,而 VNF3 则映射到 GHA 架构的加速资源内,由 FPGA 内的加速器负责数据处理。因此,在部署 GHA 架构到 NFV 系统后,当有 VNF 被 GHA 加速时,原有的数据通路会发生改变,被加速的 VNF 的数据流会传输到 GHA,剩余的数据处理仍在基于软件的 VNF 中进行。

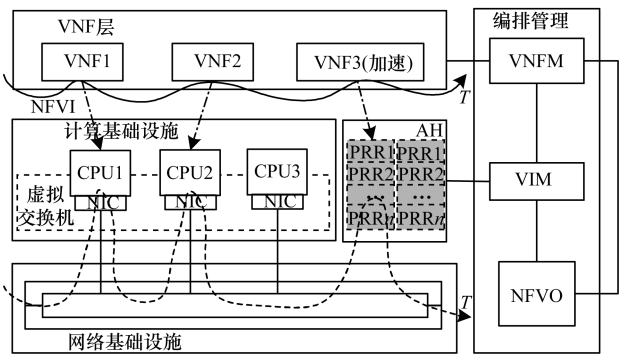


**图 5　VNF 映射到 GHA 的示意图**

## 3　加速资源优化分配策略

基于 FPGA 的 GHA 加速资源可以部署到数据中心,加速数据中心内 VNF 的数据处理,降低处理时延,提高网络吞吐量。数据中心内会存在大量有加速需求的 VNF,但部署的加速资源数量与之相比却是有限的,如果无差别地满足 VNF 的加速需求时,加速资源会马上被耗尽,无法满足加速优先级更高的 VNF 的加速需求。因此,需要一种基于 GHA 架构的资源优化分配策略,能够有效分配数据中心内有限的加速资源。

### 3.1　加速资源分配问题建模

在数据中心中,与 GHA 加速资源分配相关的主要参数有:基于 FPGA 板卡的 GHA 加速资源总量,VNF 加速资源开销及相应的加速优先级。在 GHA 架构中,用户可以根据需求设定 VNF 的加速优先级为 1 ~3 共 3 个级别。VNF 的加速优先级为 1 时,需要优先分配相应的加速资源,满足该网络功能的加速需求,数量相对较少;当 VNF 的加速优先级为 3 时,需要根据分配后加速资源的剩余情况,决定是否满足该类 VNF 的加速需求;而加速优先级为 2 时,也是 VNF 的默认优先级,是用户在一般情况下的选择,在满足优先级为 1 的 VNF 加速需求后,再将剩余加速资源进行分配,尽可能满足其需求。

对于相同加速优先级的 VNF,尤其是 2 级加速优先级,在使用 GHA 架构加速后,对于加速资源开销不同的 VNF,其吞吐量提升幅度一般也会有差别。当加速资源无法满足全部相同优先级的 VNF 时,为使得运营商获得最大的收益,对不同 VNF 的加速收益 $e$ 作出如下定义:

$$e = \frac{tp_a - tp_n}{r} \tag{1}$$

其中,$tp_n$ 为 VNF 加速前的吞吐量,$tp_a$ 为 VNF 加速后的吞吐量,$r$ 为 VNF 的加速资源开销。显然,不同类型的 VNF 通过 GHA 架构所获得的加速收益不同,因此,存在一个有限加速资源收益最大化的问题。基于以上分析,可以得出加速资源分配的优化模型。

一个数据中心内某一时刻有 $m$ 块 FPGA 板卡内存在可分配的加速资源,第 $i$ 块板卡的空闲资源总量为 $C_i$;有 $n$ 个新生成的 VNF 需要加速,第 $j$ 个 VNF 的资源开销为 $r_j$,其加速收益为 $e_j$,硬件资源的总加速收益为 $E$。

$$\max: E = \sum_{i=1}^{m}\sum_{j=1}^{n} e_j x_{ij} \tag{2}$$

$$\text{s.t.} \sum_{j=1}^{n} r_j < \sum_{i=1}^{m} C_i, i = 1,2,\cdots,m, j = 1,2,\cdots,n \tag{3}$$

$$\sum_{j=1}^{n} r_j x_{ij} \leqslant C_i, i = 1,2,\cdots,m \tag{4}$$

$$\sum_{i=1}^{m} x_{ij} \leqslant 1, j = 1, 2, \cdots, n \tag{5}$$

$$x_{ij} \in \{0, 1\}, i = 1, 2, \cdots, m, j = 1, 2, \cdots, n \tag{6}$$

式(2)是模型的优化目标,即在分配有限硬件加速资源的情况中,保证获得最大的加速收益。式(3)~式(6)为约束条件。其中,式(3)是模型建立的前提,是对资源总量不足的描述。式(4)使得每块板卡可提供的加速资源不超过其空闲资源上限,这也是对单个 VNF 的加速资源只能由单块 FPGA 板卡提供的限制条件。位于不同板卡内的 PRR 以跨板卡的方式组成处理单元时,会对加速器的构建带来额外的开销和难度,同时增加整个架构管理的复杂性。因此,单个 VNF 的分配加速资源要限制在单块板卡内。式(4)保证了 VNF 的加速资源最多只能被分配一次。式(5)对 VNF 加速资源分配矩阵的元素进行了约束,保证元素的取值范围只能是 0 或 1。若 $x_{ij}$ 为 1,表示第 $j$ 个 VNF 的加速资源由第 $i$ 块板卡提供;若 $x_{ij}$ 为 0,表示第 $j$ 个 VNF 的加速资源不由第 $i$ 块板块提供。

### 3.2 加速资源分配算法

根据上述加速资源资源分配模型,同时考虑到数据中心内所能承载的有加速需求的 VNF 数量较多,可以将该优化问题归纳为一个大规模多背包问题(Multiple Knapsack Problem,MKP)[15]。MKP 是 0-1 背包问题的一个扩展形式,也是完全 NP 难问题,其计算复杂度为 $O(2^{mn})$。因此,精确算法难以在实际应用的可行时间内获得 MKP 的最优解,而目前用于求解 MKP 的启发式算法仅限于遗传算法(Genetic Algorithm, GA)[16]、人工鱼群算法(Artificial Fish Swarm,AFS)[17-18]和粒子群优化算法(Particle Swarm Optimization,PSO)[19]。其中,GA 算法对 MKP 的个体编码形式一般为 0-1 矩阵,式(5)的约束使得该矩阵为稀疏矩阵,降低了算法的求解效率,对于大规模 MKP,在实际应用时间很难得到令人满意的结果;AFS 算法对不可行解一般采用罚函数处理,而 MKP 问题本身约束的程度较高,当问题规模较大时,不可行解在初始化人工鱼群及执行行为算子的过程中不断增加,算法性能随之劣化[20]。本文采用的 DPSO 算法对速度和位置更新方式进行了改进,编码方式从 0-1 矩阵形式变为整数向量,而且对于不可行解的处理采用了随机修复策略,避免对无效解的搜索。基于以上改进,DPSO 算法在对大规模 MKP 问题求解时,在有限的时间内,能够得到满意解,具有较好的实用性。

在本文算法中,将式(2)中 $x_{ij}$ 的 0-1 编码方式转换为整数编码。如果 $x_{ij} = 1$,则 $x_j(t) = i$,表示第 $t$ 次迭代时,第 $j$ 个 VNF 的加速需求由第 $i$ 块 FPGA 板卡提供;如果 $x_j(t) = 0$,表示第 $t$ 次迭代中,第 $j$ 个VNF 的加速请求被拒绝。则粒子 $a$ 的位置编码为 $\boldsymbol{X} = [x_1^a(t), x_2^a(t), \cdots, x_n^a(t)]$,其中,$0 \leqslant x_j^a(t) \leqslant 1$,并且 $x_j^a(t)$ 为整数。$x_j^a(t)$ 表示第 $t$ 次迭代中,第 $j$ 个VNF 所分配板卡的编号,则每个粒子的位置编码便是一组 VNF 加速请求和 FPGA 板卡资源之间的分配关系。$\boldsymbol{V}^a(t) = [v_1^a(t), v_2^a(t), \cdots, v_n^a(t)]$ 为粒子 $a$ 的速度编码。随机修复策略是指对不满足约束条件式(4)的 FPGA 板卡,依次随机剔除一个所分配的 VNF,直到该板卡满足式(4)。该策略针对 MKP 中庞大的解空间,以最简单的方式有效扩大了 DPSO 算法的搜索能力。此外,将粒子的适应度定义为:

$$E(t) = \begin{cases} \sum_{j=1}^{n} e_j, x_j(t) \neq 0 \\ \sum_{j=1, j \neq j_0}^{n} e_j, x_{j_0}(t) = 0 \end{cases} \tag{7}$$

算法给出了用 DPSO 优化分配的求解过程。首先确定粒子种群规模 *Size*、最大迭代次数 *iter*、位置更新公式中的惯性权重 $w$。接着初始化每个粒子的位置编码和速度编码。其次对超出约束条件的粒子进行随机修复,然后评价每个粒子的适应度,并更新个体历史最优解和全局最优解,计算下一代每个粒子的速度和位置。最后将算法循环 $q$ 次后,判断粒子是否达到预设的资源收益或是否达到最大迭代次数,根据输出结果进行加速资源的分配。

DPSO 优化分配算法如下所示:

**算法** DPSO 资源分配算法

**输入** $\{Size, w, (r_1, r_2, \cdots, r_n), (e_1, e_2, \cdots, e_n), (C_1, C_2, \cdots, C_m)\}$

**输出** 最佳位置 $P_g$

```
1. For each particle
2. initialization,generating X(0) and V(0) randomly
3. End
4. t←0
5. Do
6. For each partilce
7. For j = 1:n
8. If x_j(t) ≠0
9. R_{x_j}(t)←R_{x_j}(t) + r_j
10. End
11. End
12. If ∃R_i(t) > C_i
13. H = rand(n) andj←1
14. Do
15. If x_{H(j)}(t) ≠0
16. x_{H(j)}(t)←0 and R_i(t)←R_i(t) - r_{H(j)}
```

17. End
18. While ($R_i(t) > C_i$)
19. End
20. For j = 1:n
21. If $x_j(t) \neq 0$
22. $E(t) \leftarrow e_j(t) + E(t)$
23. End
24. End
25. If $E(t) > P_b(t)$
26. $P_b(t) \leftarrow E(t)$
27. End
28. End
29. For k = 1:Size
30. If $E_k(t) > P_g(t)$
31. $P_g(t) \leftarrow E_k(t)$
32. End
33. End
34. For each particle
35. $V(t+1) = w \otimes s(V(t)) \oplus \text{rand}() \otimes [P_b(t) \Theta X(t)] \oplus \text{rand}() \otimes [P_g(t) \Theta X(t)]$
36. $X(t+1) = X(t) \oplus V(t+1)$
37. End
38. $t \leftarrow t+1$
39. While(t ≥ iter or $P_g$ achieved the scheduled gain)

定义如下公式:

$$s(v_j) = \frac{1}{1+\exp(-v_j)} \tag{8}$$

$$a \otimes b = round(ab) \tag{9}$$

$$a \oplus b = \begin{cases} a, & s(a) \geq s(b) \\ b, & \text{其他} \end{cases} \tag{10}$$

$$a \Theta b = \begin{cases} a, & a = b \\ |a-b|, & \text{其他} \end{cases} \tag{11}$$

## 4　性能测试

对基于 NetFPGA-10G 平台的 GHA 架构原型系统进行性能测试,而且对算法的加速收益优化率进行仿真验证。

### 4.1　原型系统测试

为了测试 GHA 架构的数据处理性能,本文基于 NetFPGA-10G 平台实现了 GHA 原型,进行了数据吞吐量的测试。NetFPGA-10G 板卡上的芯片系列为 Xilinx Virtex 5(XC5VTX240T),具有 4 个 10 Gb/s 物理端口和 4 个支持直接内存访问并连接了 PCIe 总线的虚拟端口。加速平台的开发软件环境为 Xilinx ISE 14.7,利用 Xilinx 提出的动态局部可重构设计流程划分出 4 个 PRR,每个 PRR 中资源为 9246 Slice,75 BRAM。硬件加速平台通过 PCIe 总线与部署有 VNF 的通用服务器(Intel Xeon E5-2420 CPU 1.9 GHz,16 GB DDR3 RAM,Ubuntu14.04)相连,OpenVZ 作为虚拟层,由 Spirent Tester 产生测试数据流,与 FPGA 的 4 个物理端口相连接,整个性能测试的搭建平台如图 6 所示。

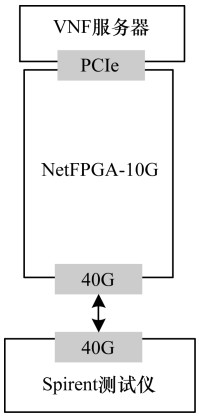


**图 6　数据平面的服务器和 FPGA 架构**

为了验证 GHA 加速架构对不同 VNF 数据处理的性能,分别采用虚拟路由器(vRouter)和防火墙作为 VNF 实例,而且与纯软件和采用 DPDK 软件加速方案的包处理速率进行对比。为了比较实验结果与纯软件 VNF 和 DPDK 软件加速方案,在 NetFPGA-10 Gb/s 板卡上分别构建了一个 vRouter 和一个防火墙,并对不同长度的数据包处理速率进行测试,其中一个 vRouter 的资源为 14 588 Slice,84 BRAM,需要 2 个PRR 构建处理单元;一个防火墙的资源为 14 588 Slice,65 BRAM,需要一个 PRR 构建处理单元。采用软件方案测试时,将服务器的 4 个 10 Gb/s 端口直接与测试仪相连,其中服务器运行 vRouter 的软件版本为 Click[21],防火墙则为网络层防火墙[22],负责检测包的五元组信息。使用 DPDK 作为软件加速方案时,VNF 服务器分配 2 个核心用作数据包转发,并且将内存大页设置为 4 G。Spirent 测试仪在每次实验中模拟生成 20 000 条流,而且分别设定数据包长度为 64 Byte、512 Byte 和 1 512 Byte。

实验结果对比如图 7 所示。采用纯软件数据平面处理时,受到虚拟化开销和多层 I/O 处理限制,2 个VNF 最多分别能达到 677 Mb/s 和 670 Mb/s。利用 DPDK 的软件加速方案,通过旁路内核协议栈和零复制等技术,极大减轻了数据包在服务器中的 I/O 开销,但受到 VNF 本身的软件处理速率的限制,吞吐量最多达到 19 Gb/s 和 17.2 Gb/s。GHA 架构下的 vRouter 和防火墙的数据包处理完全在 FPGA 内部完成,没有经过 CPU,吞吐量达到 34 Gb/s 和 33.3 Gb/s。可见,GHA 的最大吞吐量为纯软件的 50.7 倍、DPDK 加速方案的 1.8 倍,通过 GHA 架构能够提升 VNF 的数据处理速率。与其他类型 VNF 相比,vRouter 和防火墙 2 类网络功能的全部数据包处理任务由硬件实现,不经过 CPU,加速效果显著;但其他类型 VNF 的部分数据处理可能在 CPU 中实现,另一部分卸载到 GHA 硬件中,加速效果无法比拟实验中的 2 类 VNF。GHA 架构的相对通用性降低了网络功能加速的复杂性,但也限制了所能承载的数据处理任务,对不同类型的 VNF 加速效果不

同。总地来说,相比纯软件和软件加速方案,GHA 架构以一种相对通用的加速方式,能够对不同类型的 VNF 取得较好的加速效果。

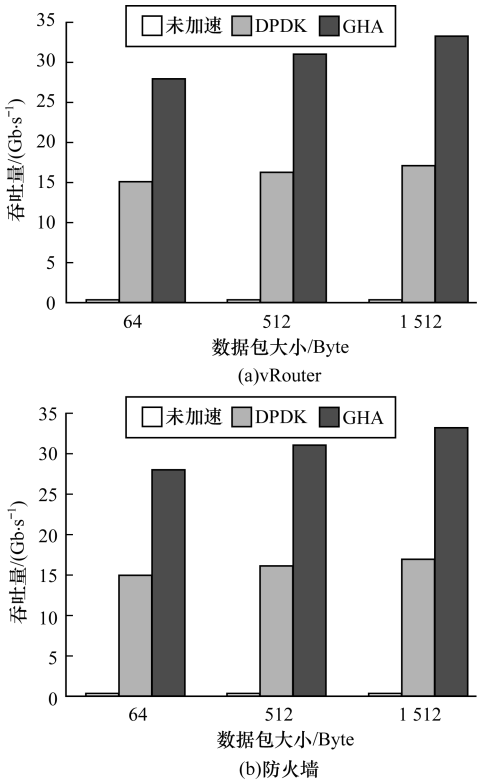


**图 7 3 种类型加速方案的处理性能对比**

### 4.2 算法性能仿真

对于分配算法的仿真,用 Matlab R2015a 编程,在 PC 机上(Intel Core CPU i5-3450 3.10 GHz,4 GB RAM)运行。为了对资源分配算法进行性能分析,将上述 2 类 VNF 的测试数据作为参考,设置 8 种不同参数的 VNF 作为初始数据,如表 1 所示。

**表 1 VNF 类型和属性**

| VNF 类型 | 未加速吞吐量/(Gb·s⁻¹) | GHA 吞吐量/(Gb·s⁻¹) | PRR 数量 |
|---|---|---|---|
| 1 | 0.781 | 16.0 | 3 |
| 2 | 6.000 | 40.0 | 3 |
| 3 | 0.312 | 6.2 | 4 |
| 4 | 16.000 | 40.0 | 4 |
| 5 | 1.562 | 12.8 | 2 |
| 6 | 6.410 | 14.8 | 2 |
| 7 | 7.830 | 40.0 | 1 |
| 8 | 0.568 | 40.0 | 4 |

在仿真实验中,设定单块 FPGA 板卡所能提供的加速资源均为 8 个 PRR。为了尽可能保证资源分配的公平性,给后到达的 VNF 留有可分配的加速资源,通过限制每次分配的硬件资源总量,实现系统的稳定性,根据每次加速资源的总开销设定每次的资源分配总量。在此次实验中,引入分配系数 $\gamma$,表示可分配加速资源与总开销之间的比值,分别设定为 1/2、3/5 和 2/3,用于模拟真实环境,显示不同情况下可分配的加速资源的总量。

为了测试算法对不同 VNF 数量的优化性能,一共做了 5 组实验,每组实验生成不同数目的服务功能链(Service Function Chain,SFC),每条 SFC 由8 类设定的 VNF 中随机挑选4 个 ~6 个来组成。在每组实验中,对于同一数目的 SFC,测试 5 次,然后取平均值。另外,设定 DPSO 算法中种群规模 *Size* 为 100,最大迭代次数 *iter* 为 200,惯性权重 $w$ 为 $0.65+0.25\times\cos(t\times\pi/iter)$,其中,$t$ 为迭代次数,$w$ 随着算法迭代的进行而逐渐减小,可降低算法陷入局部最优的频次。仿真不同分配系数下 DPSO 算法的加速收益,并与传统的贪心算法(Greedy Algorithm)和遗传算法(SGA)[23] 的加速收益作对比。贪心算法按照 VNF 的加速收益大小依次选取剩余资源最大的 FPGA 板卡进行分配;SGA 算法通过在适应性函数中引入容量约束惩罚和物体单一归属惩罚,并采用一致交叉的交叉算子[23],设定与 DPSO 算法相同的迭代次数,求解出资源分配方案。

图 8 为 3 种加速资源分配算法的收益对比。由仿真结果可知,在相同的服务链数目下,3 种算法的加速收益随着分配系数的增大而增加。这是由于可分配的加速资源越多,VNF 的加速需求得到满足的可能性也就越大。加速资源分配数量与总开销的比值 $\gamma$、VNF 个数和算法优化率之间的关系如图 8(d)和图 8(e)所示。综合 5 次实验结果,在限定不同的可分配加速资源总量时,与贪心算法相比,DPSO 算法的收益优化效率平均能提升 24.5%、13.5% 和 8.0%;与 ASG 算法相比,在相同的迭代次数内,DPSO 算法得到的分配方案更优,收益优化率平均提升11.1%、7.8% 和 5.4%。当 $\gamma$ 确定时,随着 VNF 个数的增长,解空间不断增大,在设定的搜索次数内,算法所能找到的较优解质量随之下降,所提升的加速资源收益有限,可以通过增加种群规模和迭代次数扩大搜索范围,直到找到令人满意的结果;当 VNF 个数确定,加速资源分配数量与总开销的比值 $\gamma$ 越小,约束条件下的解空间越小,在相同的迭代次数内,算法能够搜索到更优的资源分配方案,算法优化率明显提高。可见,基于 DPSO 的资源分配优化算法是有效的,在实际情况中,需要根据具体 VNF 个数和加速资源总量以及所限定的加速资源分配时间,来确定种群规模和迭代次数,以获得预期的优化率。

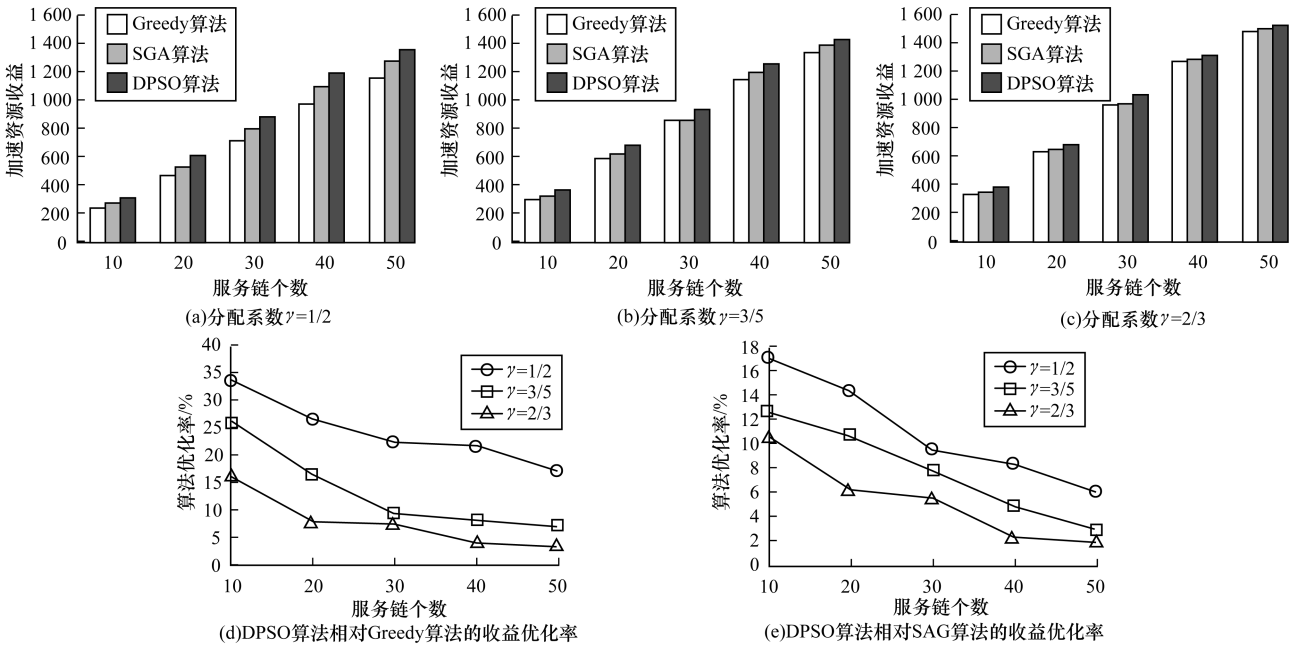


**图 8　DPSO 算法性能仿真结果**

## 5　结束语

本文针对 VNF 在通用服务器中数据包处理性能受限的问题,提出了基于 FPGA 的相对通用的硬件加速架构 GHA,将“解析-匹配-动作”的处理环节卸载到硬件中,由硬件中的解析单元、处理单元实现数据处理加速。GHA 能够利用 FPGA 的动态可重构技术,根据 VNF 需求部署 GHA 加速器。另外,采用 DPSO 算法对加速资源分配进行优化,提高了数据中心内有限硬件资源的加速收益。仿真测试证明,本文提出的 VNF 数据包硬件加速架构,在促进 NFV 性能提升的同时,降低了 VNF 加速的实现难度。

### 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] RUSSELL R. Virtio: towards a de-facto standard for virtual I/O devices[J]. ACM Sigops Operating Systems Review,2008,42(5):95-103.

[3] DONG Yaozu, YANG Xiaowei, LI Xiaoyong, et al. High performance network virtualization with SR-IOV[C]// Proceedings of International Symposium on High Performance Computer Architecture. Washington D. C., USA: IEEE Press,2010:1471-1480.

[4] CERRATO I, ANNARUMMA M, RISSO F. Supporting Fine-grained network functions through Intel DPDK[C]// Proceedings of the 3rd European Workshop on Software Defined Networks. Washington D. C., USA: IEEE Press, 2014:1-6.

[5] KACHRIS C, SIRAKOULIS G, SOUDRIS D. Network function virtualization based on FPGAs: a framework for all-programmable network devices [J]. Computer Science,2014,96(5):3227.

[6] LIU Yi, HU Xinyu, GE Xiongzi, et al. OpenANFV: accelerating network function virtualization with a consolidated framework in openstack[C]//Proceedings of ACM Conference on SIGCOMM. New York, USA: ACM Press,2014:353-354.

[7] GE Xiongzi, LI Bojie, TAN Kun, et al. ClickNP: highly flexible and high performance network processing with reconfigurable hardware [C]//Proceedings of ACM SIGCOMM Conference. New York, USA: ACM Press, 2016:1-14.

[8] SUN Chen, BI Jun, ZHENG Zhilong, et al. SLA-NFV: an SLA-aware high performance framework for network function virtualization [C]//Proceedings of ACM SIGCOMM Conference. New York, USA: ACM Press, 2016:581-582.

[9] NetFPGA-10 G Project [EB/OL]. [2018-01-24]. http://github. com//NetFPGA/NetFPGA-ublic/wiki.

[10] GIBB G, VARGHESE G, HOROWITZ M, et al. Design principles for packet parsers[C]//Proceedings of IEEE Symposium on Architectures for Networking and Communications Systems. Washington D. C., USA: IEEE Press,2013:13-24.

[11] 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.

[12] 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.

[13] BOSSHART P, IZZARD M, IZZARD M, et al. P4: programming protocol-independent packet processors[J]. ACM SIGCOMM Computer Communication Review, 2014,44(3):87-95.

[14] BRONSTEIN Z, ROCH E, XIA J, et al. Uniform handling and abstraction of NFV hardware accelerators[J]. IEEE Network, 2015,29(3):22-29.


(下转第 126 页)

传输,基于小区分割方案以及分割因子 $R$,将 MU 划分为 CCU 和 CEU。同时,利用 RSA 策略将总可用频带依据 $p_m$ 划分为 2 个部分,其中,CCU 与 FU 共享 CCU 频带,CEU 与 D2D 共享 CEU 频带。在此异构网络模型下,分别研究了网络参数对 D2D、FU、CCU 和 CEU 链路覆盖性能的影响,得到了各自的 SINR 覆盖概率。研究表明,有效的小区分割与合理的频带分配能改善异构网络的覆盖性能,所得结果对于异构网络的最优化设计具有重要意义。基于上述研究结果,下一步将分析异构网络的物理层安全性能,并在被动窃听环境下研究 CCU 和 CEU 的安全概率。

## 参考文献


[1] Cisco System. Cisco Visual Networking Index: Global mobile data traffic forecast[EB/OL]. [2017-07-26]. http://www. cisco. com/c/en/us/solutions/collateral/serviceprovider/visual-networking-index-vni/mobile-white-paper-c11-520862. pdf.

[2] 贾向东,颉满刚,周 猛. 基于带内回程的全双工大规模 MIMO 异构网覆盖分析[J]. 计算机工程,2017,43(7):124-128,135.

[3] 尤肖虎,潘志文,高西奇,等. 5G 移动通信发展趋势与若干关键技术[J]. 中国科学:信息科学,2014(5):551-563.

[4] ANDREWS J G. Seven ways that HetNets are a cellular paradigm shift[J]. IEEE Communications Magazine, 2013,51(3):136-144.

[5] ZHENG Y,SUN S,RONG B,et al. Traffic aware power allocation and frequency reuse for green LTE-A heterogeneous networks[C]//Proceedings of IEEE International Conference on Communications. Washington D. C. ,USA:IEEE Press,2015:3167-3172.

[6] MAHMUD A,HAMDI KA. A unified framework for the analysis of fractional frequency reuse techniques[J]. IEEE Transactions on Communications,2014,62(10):3692-3705.

[7] ZAHID R, RAHMAN A U, HASSAN S A. On the performance of multiple region reverse frequency allocation scheme in a single cell downlink heterogeneous networks[C]//Proceedings of Wireless Communications and Mobile Computing Conference. Washington D. C., USA:IEEE Press,2014:387-392.

[8] NOVLAN T D, GANTI R K, GHOSH A, et al. Analytical evaluation of fractional frequency reuse for heterogeneous cellular networks[J]. IEEE Transactions on Communications,2012,60(7):2029-2039.

[9] IJAZ A,HASSAN S A,ZAIDIS A R,et al. Coverage and rate analysis for downlink HetNets using modified reverse frequency allocation scheme[J]. IEEE Access, 2017,5:2489-2502.

[10] XIE Bei,ZHANG Zekun,HU Rose,et al. Joint spectral efficiency and energy efficiency in FFR based wireless heterogeneous networks[J]. IEEE Transactions on Vehicular Technology,2017(5):1.

[11] ZHUANG He,OHTSUKI T. A model based on poisson point process for analyzing MIMO heterogeneous networks utilizing fractional frequency reuse[J]. IEEE Transactions on Wireless Communications, 2014, 13(12):6839-6850.

[12] LIU Jiajia, KATO Nei, UJIKAWA H, et al. Device-to-device communication for mobile multimedia in emerging 5G networks[J]. ACM Transactions on Multimedia Computing, Communications, and Applications, 2016, 12(5):75.

[13] LIN Y D,HSU Y C. Multihop cellular:a new architecture for wireless communications[C]//Proceedings of the 19th Annual Joint Conference of IEEE Computer and Communications Societies. Washington D. C. , USA: IEEE Press,2000:1273-1282.

[14] MOLTCHANOV D. Distance distributions in random networks[J]. Ad Hoc Networks, 2012, 10(6):1146-1166.

[15] ELSAWY H, HOSSAIN E. Two-tier HetNets with cognitive femtocells: downlink performance modeling and analysis in a multichannel environment[J]. IEEE Transactions on Mobile Computing, 2014, 13(3):649-663.


编辑 顾逸斐

(上接第 119 页)


[15] PISINGER D. An exact algorithm for large multiple knapsack problems[J]. European Journal of Operational Research,1999,114(3):528-541.

[16] FUKUNAGA A S. A new grouping genetic algorithm for the multiple knapsack problem[C]//Proceedings of IEEE World Congress on Computational Intelligence. Washington D. C. ,USA:IEEE Press,2008:2225-2232.

[17] LIU Qing, ODAKA T, KUROIWA J, et al. A new artificial fish swarm algorithm for the multiple knapsack problem[J]. IEICE Transactions on Information and Systems,2014,97(3):455-468.

[18] 覃 磊,周 康,易校尉. 一种求解多背包问题的改进的人工鱼群算法[J]. 科技通报,2016,32(6):166-171.

[19] REN Zihui,WANG Jian. A discrete particle swarm optimization for solving multiple knapsack problems[C]//Proceedings of the 5th International Conference on Natural Computation. Washington D. C. ,USA:IEEE Press,2009:166-170.

[20] 马 炫,刘 庆. 求解多背包问题的人工鱼群算法[J]. 计算机应用,2010,30(2):469-471.

[21] KOHLER E. The click modular router[M]. Cambridge, USA:Massachusetts Institute of Technology,2001.

[22] WACK J,CUTLER K,POLE J. Guidelines on firewalls and firewall policy[M]. [S. l. ]:Nist January Special Publication,2002.

[23] 虞安波,杨家本. 多背包问题的遗传算法求解[J]. 计算技术与自动化,2002,21(2):59-63.


编辑 顾逸斐