DOI:10.7544/issn1000-1239.20200629
59(1):1-21,2022

# 智能网卡综述

马潇潇[1,2] 杨 帆[1] 王 展[1] 元国军[1] 安学军[1,2]
[1](中国科学院计算技术研究所高性能计算机研究中心 北京 100190)
[2](中国科学院大学 北京 100190)
(maxiaoxiao@ncic.ac.cn)

# Survey on Smart Network Interface Card

Ma Xiaoxiao[1,2], Yang Fan[1], Wang Zhan[1], Yuan Guojun[1], and An Xuejun[1,2]
[1](*High Performance Computer Research Center*, *Institute of Computing Technology*, *Chinese Academy of Sciences*, *Beijing* 100190)
[2](*University of Chinese Academy of Sciences*, *Beijing* 100190)

**Abstract** In the era of the rapid increase in network speed, memory access bottleneck has been a prominent problem, and network processing overhead has been increasing significantly. Ordinary network cards have gradually exposed defects in network protocol processing, data migration, and programmable flexibility in modern applications. As a programmable intelligent network device, the SmartNIC (smart network interface card) has received extensive attention in the fields of data center and scientific computing cluster applications. SmartNIC has become a key technology to solve network bottlenecks. It can bring significant benefits in terms of protocol processing offload, network function virtualization, application-specific acceleration, and other usage scenarios. We survey the basic architecture, programming framework, application direction, and other hot research issues of SmartNIC. We summarize the typical products in the current industry and important achievements in academia. We also clarify the advantages and disadvantages of different design architectures. The application scenarios applicable to different programming frameworks are introduced, and the value of SmartNIC in typical data center applications and scientific computing applications is introduced. After that, we give some efficient suggestions about software and hardware collaborative design of SmartNIC in different application scenarios. Finally, we present the hot issues that still exist in the design and use of SmartNIC, and put forward the important future research points and a universal design solution for SmartNIC.

**Key words** smart network interface card (SmartNIC); programmable architecture; programming framework; network offloading; high performance network; in-network computing

**收稿日期**:2020-08-14;**修回日期**:2021-01-04

**基金项目**:国家科技创新2030重大项目(2020AAA0104402);国家自然科学基金青年科学基金项目(61702484);中国科学院战略性先导科技专项(B类)(XDB24050100);中国科学院青年创新促进会;国家自然科学基金面上项目(61972380)
This work was supported by the National Science and Technology Innovation Program 2030 (2020AAA0104402), the National Natural Science Foundation of China for Young Scientists (61702484), the Strategic Priority Research Program of Chinese Academy of Sciences (XDB24050100), the Youth Innovation Promotion Association of Chinese Academy of Sciences, and the General Program of the National Natural Science Foundation of China(61972380).

**通信作者**:王展(wangzhan@ncic.ac.cn)

**摘 要** 在网速飞速提升、内存瓶颈突出、网络处理开销愈发显著的时代,普通网卡在网络协议处理、数据搬移、使用灵活性等方面逐渐暴露出缺陷.智能网卡,作为可编程的智能网络设备,在数据中心、科学计算领域均得到广泛关注,成为解决网络瓶颈的关键技术.在网络协议处理卸载、网络功能虚拟化、特定应用加速等应用场景中发挥着重要作用.综述从智能网卡的基础架构、编程框架、应用方向和热点问题4个方面进行分析,总结了目前产业界中的典型产品、学术界中的重要成果,明确了不同设计架构的优势和不足,分析了不同编程框架适用的应用场景,介绍了智能网卡在典型数据中心应用、科学计算应用实例中的作用,对不同应用场景中智能网卡的软硬件协同设计提供了建议.最后,综述对当前智能网卡设计、使用中仍然存在的热点问题进行总结,总结了通用的智能网卡设计思路,指明未来有价值的重要研究点.

**关键词** 智能网卡;可编程架构;编程框架;网络卸载;高性能网络;在网计算

**中图法分类号** TP303

随着网络技术、存储技术、芯片设计制造技术的不平衡发展,目前,计算机网卡的设计面临新的问题.网络速度在2020年已经迈向400 Gb/s以太网大关,并正向着更快的800 Gb/s,甚至1.6 Tb/s发展[1],而后摩尔时代意味着CPU的频率已经趋于稳定,在这种不平衡的现状下,使用传统的CPU来进行网络处理已经显得不尽如人意.在速度上,现代CPU需要用10~15 ns来访问L3 Cache,而400 Gb/s的网络仅需1.2 ns便可传送64 B的消息[2];在计算能力上,CPU适合于处理串行的复杂指令操作,对大量并行的固定模式的计算并不适用;再者,在云环境多租户的情况下,开放虚拟交换(open virtual switch, OVS)等网络功能虚拟化将占用更多的CPU资源[3-4].此外,通过CPU访问内存进行数据搬移的开销在很多应用中占据了极大的比例,如在快速傅里叶变换(fast Fourier transform, FFT)计算中,数据搬移占据了40%的开销[5],旁路CPU已经成为一种重要的解决方式.因此,能够满足高速的网络处理需要、卸载CPU不适合的网络处理任务、提供一定编程灵活性的智能网络终端设备——智能网卡(smart network interface card, SmartNIC),应运而生,并且在协议处理[2,6-7]、网络功能[8-10]、数据中心云服务[11-14]、人工神经网络加速[6,15-17]、科学计算[5,18]等诸多场景中发挥了重要作用.

在学术界,智能网卡的雏形是微软亚洲研究院在2014年提出的基于现场可编程门阵列(field programmable gate array, FPGA)的Catapult设计,一种用于加速大规模数据中心服务的可重构网络[19],并在后续的一系列研究[3,20-22]中逐渐发展,且已经部署到Azure云中;在产业界,智能网卡的产品最初主要由有一定市场和技术储备的成熟网络设备生产商Mellanox[23],Netronome[24],Broadcom[25],Cavium[26]提供,Netronome公司于2016年9月在公司网站发文,对智能网卡的需求和定义进行了阐述,提出智能网卡必须具备实现复杂网络数据平面功能的能力,可以灵活地更改数据平面,并且与现有生态无缝衔接[27].而Mellanox公司则于2018年8月发文,借助PC Magazine对智能网卡的定义——能够卸载CPU通用处理任务的网卡[28],介绍了该公司推出的3种基于不同处理器架构的智能网卡产品[29].此外,近几年陆续出现了许多推出智能网卡的小公司,如BittWare[30],Ethernity[31]等,而Amazon[32]、华为[33]这样的大公司在近5年里也陆续收购小公司或者研发智能网卡用于自身部署,甚至推向市场.综合以上学术界和产业界的观点,本文认为智能网卡至少应当具有4个特性:

1) 满足现有数据平面网络处理需求;

2) 兼容现有网络协议生态;

3) 能够灵活卸载通用CPU不适合的处理任务;

4) 提供用户友好的可编程性.

本文将从智能网卡的基础架构设计、编程框架、重点应用、未来研究热点4个方面进行综述,系统介绍智能网卡在产业界和学术界的发展过程和现状,分类对比不同设计的优势和不足,列举典型的智能网卡应用场景和潜在问题,为今后智能网卡的软硬件协同设计提供参考和思路.

本文的主要贡献包括4个方面:

1) 对基于FPGA、多核处理器(multi-core processors, MP)、特定应用集成电路(application specific integrated circuit, ASIC)的不同处理器架构和基于On-Path,Off-Path的不同数据通路设计架构进行分类,明确不同设计思路的优势和不足;

2) 对可用于智能网卡的多种编程框架进行总结,权衡分析数据流驱动、控制流驱动的设计思路;

3) 对智能网卡在数据中心、科学计算中的典型应用场景进行总结,明确智能网卡的使用价值和设计目标;

4) 对当前智能网卡的设计缺陷和使用瓶颈进行总结,指出未来智能网卡软硬件协同设计的研究方向和值得思考的问题.

## 1 智能网卡基础架构

目前,产业界和学术界中智能网卡的硬件设计架构多种多样,在性能、成本、使用灵活性等方面亦表现得参差不齐.本节我们将从核心处理器选择和数据通路设计2个维度对智能网卡的基础架构设计进行分类综述.

### 1.1 按核心处理器分类

从核心处理器角度来分析,目前智能网卡的设计主要有三大类,分别为基于FPGA,MP,ASIC的设计.下文将依次对基于不同处理器智能网卡的研究和商业产品进行介绍.

#### 1.1.1 基于FPGA的设计

在学术界,以FPGA作为智能网卡核心可编程处理器的研究主要以微软研究院为代表[3,19-22].图1描述了微软一系列设计架构的演进.2014年,微软提出了基于高端FPGA——Altera Stratix V D5[34]的Shell(通用逻辑)+Role(可重构处理逻辑)的可重构数据中心云服务加速方案,用于解决商用服务器满足不了飞速增长的数据中心业务需求、定制化加速器成本开销大且灵活性不足的问题[19].如图1(a)所示,其中Shell为可重用的通信、管理、配置等通用逻辑,包含2个DRAM(dynamic random access memory)控制器(管理FPGA上的2块DRAM)、4个10 Gb/s轻量级FPGA间串行通信接口SerialLite3、管理DMA通信的PCIe核、路由逻辑(用于管理来自PCIe,Role,SerialLite3的数据)、重新配置逻辑(用于读、写、配置Flash)、事件翻转逻辑(用于阶段性的监督FPGA状态以减少错误);而Role则位于FPGA芯片的固定区域中,是跟用户加速应用紧密相关的逻辑,文中以加速Bing搜索为例,将排序逻辑映射到Role中进行加速.在Catapult设计中,考虑到FPGA的管理和使用,同机架下的所有FPGA以6×8的2维Torus网络拓扑的形式组成一套新的网络进行连接,可以将同机架下的所有FPGA作为加速资源使用.但是,使用第2套网络的设计方式:一方面,增加了网络的开销和容错管理;另一方面,对于网络流、存储流、分布式应用仅能提供有限的加速.此外,机架内的2维Torus直连使得用户对跨机架的FPGA资源无法得到有效的使用.

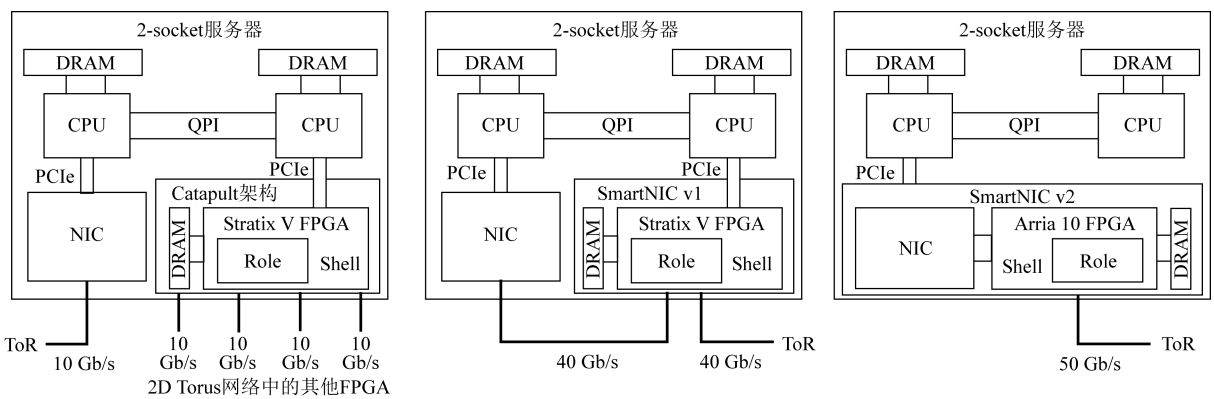



Fig. 1 The evolution process of Microsoft SmartNIC architecture[19-22]

图1 微软智能网卡设计架构的演化过程[19-22]

微软在2016年的研究工作中对Catapult进行了改进,将FPGA网络与数据中心网络融合,提出了新的云加速架构设计[20].如图1(b)所示,在Stratix V D5 FPGA板卡上设计了2个40 Gb/s的QSFP(quad small form-factor pluggable)端口,分别与主机端已有的普通网卡和架顶交换机(top-of-rack, ToR)相连接,对应地,在新的Shell设计中,原来Catapult的4端口SerialLite3被替换为轻量级传输层(lightweight transport layer,LTL)引擎用于处理2个40 Gb/s端口.这样,所有的网络数据

都要经过 FPGA 的以太网端口,FPGA 便可以更直接、高效地对网络数据流、存储数据流进行加速,在可扩展性上也有很大提升.文中除了加速网页搜索应用,还对传输中的网络数据加密进行了 FPGA 加速,以体现该设计的加速效果.此时,微软初代智能网卡已成雏形,只是 FPGA 和通用网卡还未集成在一块板卡上.同年,微软提出了针对该 FPGA 智能网卡的一套高级编程语言——ClickNP[21](类似 C 语言),实现 FPGA 和 CPU 之间的协同编程,使用模块化的实现方法,向用户提供友好的编程接口.文中以多种网络功能为例进行了实验评估验证,一定程度上解决了权衡 FPGA 高性能和编程复杂的问题.

微软在 2018 年的研究中将软件定义网络(software define network, SDN)栈卸载到其二代智能网卡[3],用以更好地支持单根 I/O 虚拟化(single root I/O virtualization, SR-IOV)[35].如图 1(c)所示,此时,二代智能网卡已将通用网卡和高端 Intel Arria 10[36] FPGA 集成到 1 个板卡上,对外的 ToR 端口已经达到 50 Gb/s,但从架构上而言并无实质的变化,仍然采用将 FPGA 放置在通用网卡和 ToR 数据通路之间的设计,用于高效地处理数据流,提供路径上的网络功能、特定应用加速.微软在后来的研究中指出,鉴于当前可编程网卡、可编程交换机的硬件条件支持,充分利用可编程网络设备组成高效的全网可编程云将成为一种趋势[22].

在产业界,近些年,涌现出众多基于 FPGA 设计智能网卡的小公司,如 BittWare,Ethernity 等.同时,部分老牌公司如 Mellanox,Intel,Xilinx 也相继推出基于 FPGA 的智能网卡类产品.

BittWare[30]公司推出了基于 FPGA 的 100 Gb/s 智能网卡 Shell.支持网络功能虚拟化(network function virtualization, NFV)、网络监控(network monitoring)、预防 DDoS(distributed denial of service)攻击等功能.此外,用户可以自定义对数据包的处理,其中自定义 IP 的设计支持 Xilinx SDNet 编译器,因此满足 P4[37]编程,可以通过 Match-Action[38]的方式对数据包进行用户自定义处理.Netcope[39]于 2017 年推出的基于 Xilinx Virtex UltraScale+[40] FPGA 的 NFB(Netcope FPGA board)系列智能网卡亦支持 P4 编程.Eynx[41]公司亦推出了与 BittWare 类似的基于 FPGA 的 1～40 Gb/s 智能网卡产品.Ethernity[31]推出 ACE-NIC 系列基于 Xilinx Ultrascale+FPGA 智能网卡,主要提供网络功能虚拟化的卸载、OVS 的卸载.Reflex CES[42]推出了基于 Intel FPGA 和 Xilinx FPGA 的两大类 PCIe 终端网络设备.其中 XpressGX S/A** 系列分别基于 Intel 的 4 款 FPGA 产品:1)带有 2 块 HBM(high bandwidth memory)的 Stratix 10[43];2)无 HBM 的 Stratix 10;3)Arria 10;4)Stratix V.XpressV*** 系列则分别基于 Xilinx 的 4 款 FPGA 产品:1)分别使用带有 HBM 的 Virtex UltraScale+;2)无 HBM 的 Virtex UltraScale+;3)Kintex UltraScale+[44];4)Virtex 7.其中 XpressGX S10-FH800G 板卡使用 Intel Stratix 10 FPGA,可以满足 800 Gb/s 以太网需求,可用于数据中心、云计算、安全、高性能计算、军队安防、视频广播等领域.Silicom[45]同样推出基于 Intel FPGA 和 Xilinx FPGA 的可编程网卡.带宽分为 1 Gb/s,10 Gb/s,40 Gb/s,100 Gb/s 这 4 个级别.基于 Xilinx FPGA 的设计中,分别有基于 Virtex 6,Virtex 7,Virtex UltraScale, Kintex UltraScale, Kintex UltraScale+的产品;基于 Intel FPGA 的设计中,分别有基于 Airra 10,Stratix 10 的产品.

Mellanox 推出了 Innova 系列[46]基于 Xilinx Kintex UltraScale 高端 FPGA 的智能网卡,包含 Innova 和 Innova-2 Flex 共 2 代产品.最新的 Innova-2 智能网卡中内嵌其 ConnextX-5[47]网卡控制器,提供 40 Gb/s,100 Gb/s 双端口以太网或者 InfiniBand 网络,ConnextX 系列 ASIC 已经满足基本的智能网卡卸载功能,如 RoCE(remote direct memory access over converged Ethernet)网络协议、vSwitch/vRouter、I/O 虚拟化的硬件卸载,而高端的可编程 FPGA 则可以为用户提供更高效的特定应用加速服务,例如安全、存储、机器学习等方面的应用加速.

Intel 则推出了基于两大类可编程 PCIe 加速卡,其中基于 Arria10/Arria10 GX FPGA 的可编程加速卡 Intel FPGA PAC(Intel FPGA programmable acceleration card) N3000[48],用于加速协议栈处理、NFV 等应用;此外,另有基于 Stratix 10 SX 的可编程加速卡 Intel FPGA PAC D5005[49],面向数据流分析、视频编码转换、金融、人工智能、基因分析等领域.

Xilinx 于 2019 年 4 月收购 Solarflare[50]公司,其实,自 2017 年 Xilinx 便与 Solarflare 合作,其推出的网卡含有 XtremeScale X2 和 8000 共 2 个系列以太网卡.其中 X2 系列产品是面向数据中心的设计,带宽达到 10 Gb/s,25 Gb/s,40 Gb/s,100 Gb/s,其 Cloud Onload[51]旁路内核技术、TCP-Direct 技术与 X2 的结合可以在负载均衡、数据库缓存、容器应用、

网页服务方面减轻操作系统的开销,提高性能;其中8000系列产品,带宽达到10 Gb/s,40 Gb/s,延时小于1 μs,提供用户自定义功能的软件接口,可用于特定应用加速,以及网络包抓取、监控、分析、过滤等.2020年3月,Xilinx将已有技术进行整合,将XtremeScale以太网控制器与高端Zynq UltraScale+XCU25 FPGA结合,推出其最新的Alveo U25[52]智能网卡一体化平台,应对业界的挑战性需求与工作负载,如SDN,OVS,NFV,NVMe-oF(non-volatile memory express over fabric)[53],以及电子交易、AI推理、视频转码和数据分析等,在编程框架方面,Alveo U25支持高级综合语言(high level synthesis,HLS)、P4高级编程抽象,同时支持Xilinx的Vitis[54]统一软件平台计算加速框架,方便Xilinx及第三方应用加速.

小结:总体而言,基于FPGA的智能网卡产品设计大多数与Catapult中的设计方案类似,即FPGA分为Shell+Role,再与网卡芯片集成到一个板卡上.在具体的设计细节中,部分设计将逐渐趋于成熟的卸载技术转移到ASIC网卡中,FPGA的使用也逐渐向高端产品迈进.基于FPGA的设计方式,在产业界得到了一定的认可,因为可以极大地利用FPGA丰富的逻辑单元实现对数据快速地并行处理,并且引入较小的能耗开销;但是,FPGA对应的硬件编程语言在编程复杂度上较繁琐,需要高效的编程框架(如ClickNP)支持,其次,FPGA的价格相对昂贵,在数据中心中大量部署需要具备雄厚的经济实力.

1.1.2 基于MP的设计

另一种得到业内认可的智能网卡的设计方式为采用片上多核的方式来进行网络数据的可编程加速处理,多数使用片上系统(system on chip, SoC)的实现方案,使用的处理器核可以是专用的网络处理器(network processor,NP),如Netronome NFP系列[55]、Cavium Octeon系列[56],也可以是通用处理器(general processor,GP),如ARM.下文将从通用处理器和网络处理器2个方面进行介绍.

1) 基于NP-SoC的智能网卡

Netronome早期在2016年推出了NFE-3240系列用于网络安全相关应用的智能网卡,对数据包可达到20 Gb/s的C语言可编程线速处理.在2018,2019年,Netronome陆续推出了三大系列Agilio[24]智能网卡:①面向计算节点的Agilio CX,基于NFP-4000或者NFP-5000网络处理器,可以完全卸载虚拟交换机对网络功能中数据平面的处理、卸载典型的计算密集型任务;②面向Bare-Metal服务器的Agilio FX,基于NFP-4000网络处理器和4核ARM v8 Cortex-A72 CPU(可运行Linux OS);③面向服务节点的Agilio LX,基于NFP-6000网络处理器,主要用于虚拟化、非虚拟化的X86服务节点和广域网网关.Agilio系列产品支持灵活的包解析和Match-Action处理,可以进行eBPF,C,P4编程.

Cavium推出基于cnMIPS III网络处理器的LiquidIO[26]系列智能网卡.其中cnMPIS III是Cavium公司实现的基于MIPS64指令集架构(instruction set architecture,ISA)的Octeon系列第3代产品,此外,Octeon系列产品中还有基于ARM的产品.cnMPIS III中的CN7***系列产品频率可达2.5 GHz,集成48个处理器核,cnMPIS III系列处理器面向智能网络相关应用(从Layer2到Layer7)的可编程需求,吞吐可满足100 Mb/s到200 Gb/s的网络,以此为核心处理器设计的LiquidIO,LiquidIO II智能网卡,可进行C语言编程,可以用于OVS、NFV、安全、存储、应用加速等智能网络服务.

华为[33]推出了IN系列三大类智能网卡:早期2012年的iNIC系列、2017年的SD100(基于ARM通用处理器,含16个2.1 GHz Cortex-A57处理核心)、2018年5月的IN500系列(IN200基于海思Hi1822芯片,IN300 FC HBA基于海思高性能Fibre Channel HBA芯片).此外,SolidRun[57]推出基于NXP LX2160A通信处理器的智能网卡,Silicom[58]推出基于NetLogic XLP316和RMI XLS416网络处理器的2类智能网卡,Kalray推出基于第3代MPPA[59](massively parallel processor array)架构Coolidge处理器的KONIC200[60]系列智能网卡,其中每个Coolidge处理器含有80个64 b超长指令字核、80个协处理器、其他加速部件及外围连接逻辑.

2) 基于GP-SoC智能网卡

Mellanox除了推出基于FPGA的Innova系列可编程智能网卡,还推出了基于BlueField IPU(I/O processing unit)系列可编程智能网卡[23],支持Ubuntu,Centos系统.其中BlueField初代产品集ConnectX-5控制器、ARM v8 A72处理器阵列(最多16核,0.8 GHz)、8 GB,16 GB DDR4内存控制器于一体,最大支持双端口25 Gb/s,50 Gb/s,100 Gb/s的以太网或者Infiniband网络连接.BlueField-2则集成了最新的ConnectX-6控制器,仍然使用ARM处理器阵列,可支持单口200 Gb/s以太网或者

Infiniband 网络连接,该系列智能网卡可用于加速数据中心或者超算中的安全、存储、网络协议及功能的卸载和加速.

Broadcom 推出 Stingray 系列智能网卡产品[25],其 PS410T, PS225, PS250 产品分别定位为 4×10 Gb/s, 2×25 Gb/s, 2×50 Gb/s 高性能数据中心智能网卡,支持数据平面加速和软件定义存储(software defined storage, SDS),如 NVMe-oF.以 PS250 为例,Stingray SoC 集成了 NetXtreme E 系列 100 Gb/s 以太网卡控制器、TruFlow 可配置流加速器、8 个 ARM v8 Cortex-A72 处理器核(3.0 GHz)及多种加速引擎,支持 RoCE v1/v2, SR-IOV,使用标准的 Linux 系统、GNU 工具库,定位用于 Bare Metal 和虚拟化服务器平台(OVS 卸载)、存储服务器场景.

Amazon 于 2015 年初收购以色列芯片制造商 Annapurna,次年,Annapurna 实验室发布 Alpine 系列[32]基于 ARM v7 或者 ARM v8 架构的芯片,可用于网络存储、虚拟化、云服务等场景,并在 AWS 中得到使用.在此基础上,Amazon 相继推出几代 Nitro 系统,并于 2018 年发布 Graviton 处理器,虽然并未公开 Nitro 系统的架构,但其中应当存在基于 ARM 的智能网卡的影子.

小结:基于 MP 的智能网卡设计框架如图 2 所示,均含有 5 个重点模块:①多种已经成熟的加速部件,如 Hash 计算、加解密(Crypto)等;②用于与主机通信的 PCIe 接口,多数支持 SR-IOV;③多种与外设通信的接口,如 $I^2C$, JTAG 等;④访问智能网卡板上内存的控制器;⑤ 片上 NP 或者 GP 多核,用于 OVS, RSS(receive side scaling)等网络功能,以及用户自定义功能.NP 或者 GP 多核的具体片上布局会有差异,多数设计采用 Mesh 方式,但也有例外,如 MPPA[59]则采用多个 Cluster 的方式,Cluster 内部共享内存.此外,有的 NP 内部含有多种处理器核,如 Netronome NFP 系列[55] NP 内部有包处理器核和流处理器核两大类,分别用于包的解析、分类和数据流的处理.

基于 NP 和 GP 的 SoC 具体设计上略有差异:①如图 2 中左斜阴影部分所示,基于 NP-SoC 的设计,会将网络协议如 TCP、远程数据直接访问(remote direct memory access, RDMA)放在 NP 核上处理,如华为智能网卡[33];②如图 2 中右斜阴影部分所示,基于 GP-SoC 的设计,其内部多数会集成专门的网络控制器,用于网络协议的处理,甚至部分典型的网络功能卸载,而将更复杂的任务放在 GP 核上处理,如 Mellanox BlueField 智能网卡内部集成了 ConnectX 控制器[23],而 Broadcom Stingray 智能网卡内部集成了 NetXtreme E 控制器[25,61]专用于网络协议处理;③如图 2 中网格阴影部分所示,部分基于 NP-SoC 设计,除了集成众多 NP 核用于实现可编程功能,还有可能集成几个 GP 核,可运行 Linux 系统,用于 Bare-Metal 服务器场景,如 Netronome Agilio FX 智能网卡[24]同时集成了 NFP-4000 和 ARM v8 Cortex-A72.

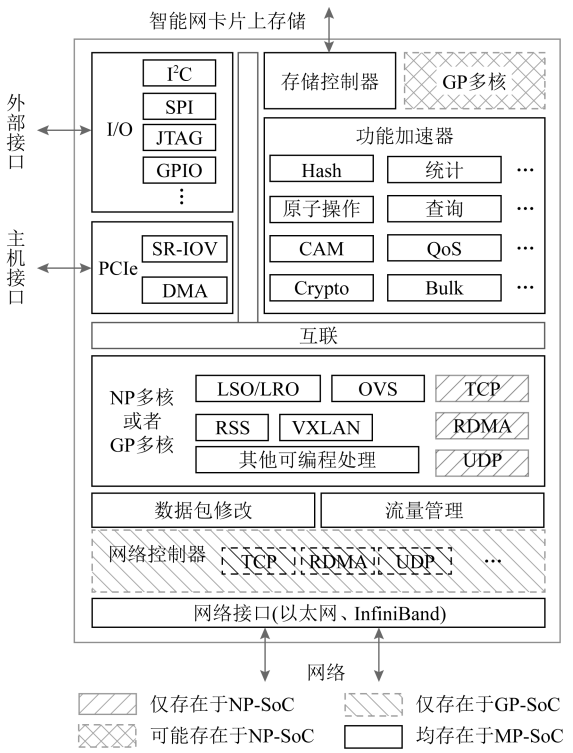



Fig. 2 MP-based SmartNIC block diagram

图 2 基于 MP 智能网卡模块架构图

在性能方面,基于 NP-SoC 的智能网卡会略胜于基于 GP-SoC 的智能网卡,因为在并行性上,NP 相比 GP 更占优势,但基于 MP 的性能均不及基于 FPGA 的性能;在成本方面,基于 NP-SoC 的智能网卡会低于基于 GP-SoC, FPGA 的智能网卡;在编程方面,基于 NP-SoC 的编程复杂度居于 FPGA 和 GP-SoC 之间,基于 GP-SoC 的智能网卡在编程方面最友好.

### 1.1.3 基于 ASIC 的设计

目前,基于 ASIC 的智能网卡并不多,ASIC 芯片主要以网络控制器的角色出现在智能网卡中,如 Mellanox 的 ConnectX 系列[47]、Broadcom 的 NetXtreme 系列[61]、Cavium 的 FastLinQ 系列[62].此类 ASIC

网络芯片除了能够满足传统的网络协议(如 TCP,RoCE)处理需求,又具备一定的卸载 CPU 处理能力和可编程性.以 Mellanox 最新的 ConnectX-6 产品为例,其在一定程度上提供对数据平面的可编程处理和硬件加速,提供虚拟化、SDN 的支持,可硬件卸载网络虚拟化中的 VxLAN(virtual extensible local area networks),NVGRE(network virtualization use generic routing encapsulation)等协议,卸载网络安全中的部分加解密运算,支持 NVMe-oF 等用于存储场景的存储协议处理,支持 GPU-Direct 等机器学习应用场景中数据零拷贝的低延时通信.

小结:1)在性价比方面,基于 ASIC 的智能网卡,基本上可以满足多数通用网络处理的应用场景,可以在预定义的范围内对数据平面进行可编程处理,并提供有限范围内的硬件加速支持,如果是批量使用,在性价比上会有较大的优势;2)在编程复杂度方面,基于 ASIC 的智能网卡虽不及基于 MP 的智能网卡那么简单,却也远易于基于 FPGA 的智能网卡;3)在使用灵活性方面,基于 ASIC 的智能网卡相比于其他的智能网卡灵活性最差,对于更复杂的应用场景则显得力不从心,更明确地来说,单纯基于 ASIC 的智能网卡应该称之为卸载网卡,因为其可编程性并不完全.从长远的角度分析,其定制化的逻辑,对于已经成熟的应用场景虽然能够提供显著的性能提升,但是随着时间的推移,新的应用场景对智能网卡将会提出新的功能要求.目前,很多厂家采用 ASIC+GP 的设计方式来解决这一问题,类似前文 Mellanox 的 BlueField 产品(集成了 ConnectX-5 和 ARM).同时,商家不断地更新 ASIC 产品,将成熟的技术定制化到网卡中,如 ConnectX 系列已更新到第 6 代.可见,体系结构中灵活性和性能之间的斗争依然在继续着.

## 1.2 按数据通路分类

从核心处理器与数据通路的关系来分析,目前智能网卡的设计主要有 On-Path,Off-Path 两大类设计[13].其中 On-Path 架构如图 3(a)所示,核心处理器在数据发送、接收的路径上,直接对数据包进行处理;Off-Path 架构如图 3(b)所示,核心处理器并不在数据发送、接收的路径上,而是通过网卡上的交换部件(图 3(b)中的 SmartNIC Switch)决定将数据直接送往主机端还是由核心处理器做处理.下文将依次对 On-Path,Off-Path 智能网卡的研究和商业产品进行综述.

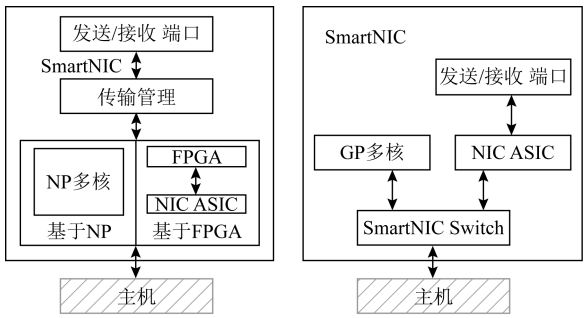



Fig. 3 On-Path,Off-Path SmartNIC architecture

图 3 On-Path,Off-Path 智能网卡架构图

### 1.2.1 On-Path 设计

如图 3(a)所示,基于 FPGA,NP-SoC,ASIC 的智能网卡一般采用 On-Path 的设计架构,NP 处理器核、FPGA、ASIC 位于数据通路上,便于处理器直接对数据进行预定的处理.基于 NP-SoC 的智能网卡,如华为的 IN 系列、Cavium 的 LiquidIO 系列、Netronome 的 Agilio 系列,基于 FPGA 的智能网卡,如微软 Catapult 系列、Mellanox Innova 系列,基于 ASIC 的智能网卡(卸载网卡),如 Mellanox 的 ConnectX 系列、Cavium 的 FastLinQ 系列,均属于 On-Path 架构.当主机端服务器需要发送数据时,主机端处理器向网卡下达发送请求,一般包括具体的处理指令(如发送、原子操作等普通指令,或者用户自定义的其他操作)和包地址,网卡中的 DMA 引擎从主机端内存中取出数据到网卡中的缓冲区,然后由核心处理单元(FPGA/ASIC/NP)进行对应的数据处理,处理完的数据再由发送端口发出.反之,当主机端服务器接收数据时,接收的数据经网卡缓冲区,直接由处理器核进行对应的数据处理,处理完成后通过 DMA 引擎将数据存入主机端对应的内存中.

On-Path 架构的设计方式优点在于处理器可以直接对数据通路上的网络数据进行处理,而使用该架构的处理器(FPGA/ASIC/NP)一般具有较高的并行度,可以提供低时延、高带宽的服务;缺点则是,在编程灵活性和易用性方面不及基于 GP-SoC 的智能网卡.因此,On-Path 架构在大多数普通需求的应用场景下能够提供较好的性能,而在较为复杂的情况下,如 Bare-Metal、以智能网卡为中心搭建加速平台等情况[4,6]下,该架构的智能网卡则略显逊色.

### 1.2.2 Off-Path 设计

如图 3(b)所示,基于 GP-SoC 的智能网卡一般

采用 Off-Path 的设计架构,GP 处理器核与数据通路松耦合,位于网卡上的可编程交换部件根据预先设定的规则决定数据流的转发对象是否为 GP 处理器核.基于 GP-SoC 的智能网卡,如 Broadcom 的 Stingray 系列、Mellanox 的 BlueField 系列,均属于 Off-Path 架构.当主机端服务器需要发送数据时,主机端处理器向网卡下达发送请求,一般包括具体的处理指令(如发送、原子操作等普通指令,或者用户自定义的其他操作,包含网卡中交换机用于转发的标志)和包地址,网卡中的 DMA 引擎从主机端内存中取出的数据经过网卡中交换机的转发,或者流向通用网卡 ASIC 逻辑直接由发送端口发出,或者流向 GP 处理器核,由 GP 处理器核进行对应的处理,最终再由发送端口发出.反之,当主机端服务器接收数据时,数据先通过网卡 ASIC,然后经网卡中交换机转发,或者直接通过 DMA 引擎存入主机端对应的内存中,或者进入 GP 处理器核处理并根据处理结果进行后续操作.

Off-Path 架构的设计方式优点有:1)数据通路与网卡处理器松耦合,对于无需特定处理的普通数据流可以直接旁路 GP 处理器;2)使用该架构的处理器一般具有较高的编程灵活性和易使用性,可以运行独立的 Linux 操作系统,能够应对复杂的应用场景.缺点则有:1)网卡上用于数据转发的交换部件需要具备灵活的可编程性和高效的转发能力,如果设计不合理将很有可能成为性能瓶颈,因而使用 Off-Path 架构的 Mellanox BlueField 和 Broadcom Stingray 分别研发了 ASAP2(accelerated switching and packet processing)技术[63]和 TruFlow 技术[64]用于提高网卡交换部件的可编程性及转发能力;2)使用该架构的 GP 处理器在处理并行性方面较差,相比于其他架构,对于固定模式的数据流处理性能略差.因此,Off-Path 架构更适用于较为复杂的应用场景下,如 Bare-Metal 服务器、以智能网卡为中心的加速平台等[4,6],可以充分利用 GP 的易用性和灵活性,而对于较为成熟的应用场景,如 NFV 卸载、网络协议卸载等,则 Off-Path 架构并非最优选择.

### 1.3 基础架构小结

表 1 从智能网卡的核心处理器和数据通路架构 2 个维度,对智能网卡基础架构部分内容进行总结,对比了不同基础架构设计的特性.图 4 以相关公司官网站发布的产品简介和公开的论文为时间节点,从核心处理器和数据通路架构 2 个维度,列举了近 10 年典型智能网卡相关设计成果.

**Table 1 The Comparison of Different SmartNIC Designs**

**表 1 不同智能网卡设计的对比**

| 对比方面 | On-Path 架构 | On-Path 架构 | On-Path 架构 | Off-Path 架构 |
|---|---|---|---|---|
| 核心处理器 | FPGA | NP-SoC | ASIC | GP-SoC |
| 性能 | 较高 | 较高 | 高(通用功能) | 低 |
| 开销 | 高 | 较低 | 低 | 中等 |
| 可编程性 | 较高 | 较高 | 受限 | 高 |
| 易用性 | 困难 | 较难 | 较难 | 简单 |
| 灵活性 | 较高 | 较高 | 低 | 高 |
| 操作系统支持 | 多数不支持 | 多数不支持 | 多数不支持 | 支持 |
| 典型产品 | Microsoft Catapult, XilinxAlveo U25 | Netronome Agilio CX, CaviumLiquidIO | MellanoxConnectX, Broadcom NetXtreme | MellanoxBlueField, Broadcom Stingray |

从核心处理器分析,总体而言,目前,产业界对核心处理器的选择仍众说纷纭,基于 FPGA 的设计除了被 Microsoft 和 Xilinx 采用外,因 FPGA 产品逐渐成熟,并有 Catapult 作为设计框架参考而受到很多小公司青睐;而基于 MP,ASIC 的设计则需要较成熟的网络技术积累,因而只有少数典型的网络设备供应商推出对应的产品.从技术发展的角度分析,已经成熟的网卡卸载技术(如协议处理、特定加解密计算等)将被逐渐定制化到网卡 ASIC 芯片中;趋于成熟且需要一定编程灵活性的功能(如 Match-Action 操作)则使用 NP 处理可以获得更高的性价比和易用性;对于涌现出来更为复杂的新需求(如 Bare-Metal 服务器)则需要 GP 来做更通用的处理.智能网卡的设计将根据应用的需求走向多元化、异构化,如 Netronome Agilio 系列网卡分为 CX,FX,LX 三大类产品,分别对应计算节点、Bare-Metal、服务器节点 3 种应用场景,其 Agilio FX 智能网卡中已集成了 NetXtreme 网卡控制器 ASIC、NFP 网络

处理器、ARM 通用处理器 3 种处理器,是异构化的典型代表.

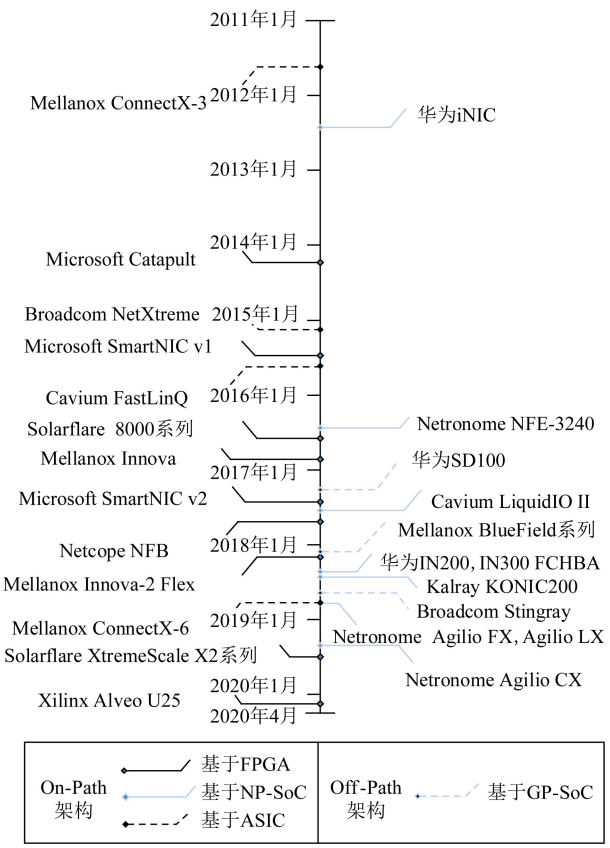



Fig. 4 SmartNIC milestone list

图 4 智能网卡里程碑

从数据路径分析,On-Path 架构更适合数据密集型应用场景,尤其适合传输路径上的流式处理,这也与此架构对应的 FPGA,ASIC,NP 处理器特性相符合;Off-Path 架构则更适合控制密集型应用场景,这也与此架构对应的 GP 处理器特性相符合.后文将对数据密集型和控制密集型的编程框架进行综述.

## 2 智能网卡编程框架

智能网卡的处理器架构是设计智能网卡的硬件基础,智能网卡的编程框架则是设计和使用智能网卡的软件基础,软硬协同是设计智能网卡的重要方法.在此,将智能网卡的编程框架设计分为面向数据密集型和面向控制密集型两大类设计,分别偏向于对数据流和控制流提供更友好的支持.

### 2.1 数据密集型

网络带宽发展迅速,正向着 Tbps 时代迈进,尤其是在云环境、虚拟化的情况下,出现了大量的应用需要对数据流进行模式固定、运算简单的可编程操作,如防火墙、网关、深度包检测等网络功能,以及 Hash 计算等,我们称这种计算相对简单、模式固定、控制相对较少的应用为数据密集型应用.智能网卡作为一种可编程网络终端设备,具备高效的 I/O 处理能力、内存读写能力、可观的软件处理开销,是智能网卡设计的必然需求[2,7],而智能网卡编程框架是其中的重要一环.我们把智能网卡编程框架中,对数据密集型应用提供良好的编程支持和性能加速的编程框架,称为数据密集型编程框架.

数据密集型智能网卡编程框架至少要具备 3 个特点:1)尽可能减少主机服务器在数据通路上对数据搬移的开销;2)向用户提供具备一定编程能力的接口,满足可编程应用的需求;3)将软件编程和硬件架构之间进行合理的映射,以实现对数据流进行高效的流水处理.

RDMA 作为一种在高性能计算中常用的通信方式,近些年逐渐被广泛应用到数据中心网络中[65-70],使用 RDMA 通信可以旁路主机端操作系统、减少通信中数据拷贝的开销,提供低延时、高带宽的通信性能,是进行数据通路优化的一种重要方式.Mellanox 的 OFED[71] 提供了标准的 RDMA 支持,Portals 4[72] 也提供了类似 RDMA 的通信接口,FlexNIC[7] 和 sPIN[2] 则在支持 RDMA 的基础上对网卡进行了可编程功能的强化和流水处理的设计.此外,基于 FPGA 的 ClickNP[21] 和基于 MP 的 Floem[73] 也是典型的数据密集型智能网卡编程框架,后文将对以上提到的典型究进行介绍.

如图 5 所示,FlexNIC[7] 的设计继承了 RMT (reconfigurable match table)可编程交换机架构[38],对数据包的处理分为入口流水线和出口流水线 2 部分,每一部分都包含包解析、Match-Action(匹配包头字段并执行对应的操作)、数据包整合 3 个处理阶段,可编程的功能以 Match-Action 的形式映射到多核处理器中,对数据包中自定义的域进行流水处理,由于智能网卡需要与主机端进行数据交互,如图 5 中左斜阴影部分所示,FlexNIC 的设计中增加了 DMA 操作流水线和 DMA 引擎.FlexNIC 使用 P4[37] 语言进行编程,并且增加了部分原语用于简化编程.这种设计方法,对于简单的操作,经过一次流水便可以完成处理;但是,对于复杂的操作,需要的 Match-Action 操作数量大于流水线中 Match-Action 单元总数 $n$ 时,未处理完的数据包需要重新进入流水线进行新一轮流水处理,这样将大大增加处理延时.

sPIN[2]的设计则是在 Portals 4[72]的基础上进行了数据通路的进一步优化和可编程功能的强化，充分利用了智能网卡片上内存和处理单元，在数据到达接收端网卡后，网卡直接向发送端返回响应，然后由网卡进行数据处理并搬移到对应的应用存储空间，减少了接收端对数据包的响应时间，将 RDMA 的性能进一步提高.如图 6 所示，sPIN 将每个消息的多个数据包处理划分为 3 部分，即包头 handler、负载 handler、完成 handler，由智能网卡上的 1 个或者多个逻辑处理单元(handler processing unit，HPU)进行流水处理，HPU 可以映射到智能网卡的多核处理器上.在编程方面，为了方便对智能网卡进行管理和对 3 种 handler 进行编程处理，sPIN 对 Portals 4 编程接口进行了扩充，提出了 P4sPIN 编程接口，提供了智能网卡片上内存管理、HPU 管理、handler 处理的多种原语.

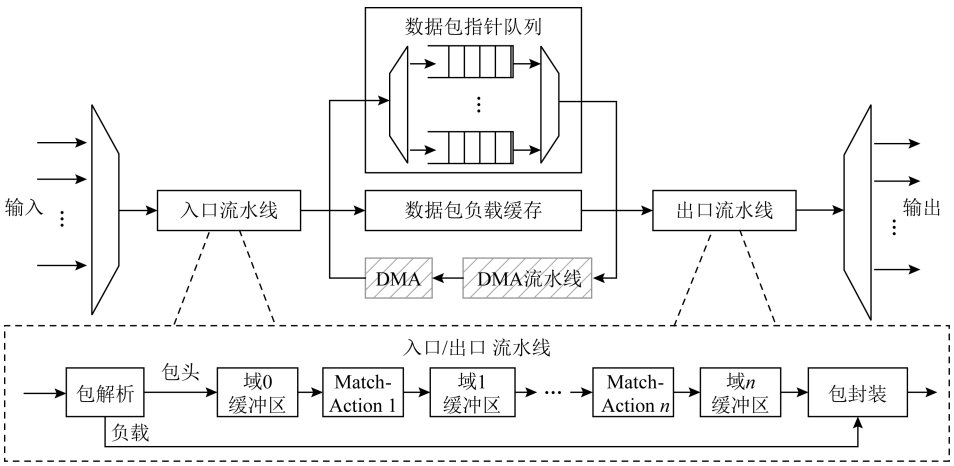



Fig. 5 FlexNIC (RMT-enhanced DMA architecture)

图 5 FlexNIC(RMT+DMA 架构)

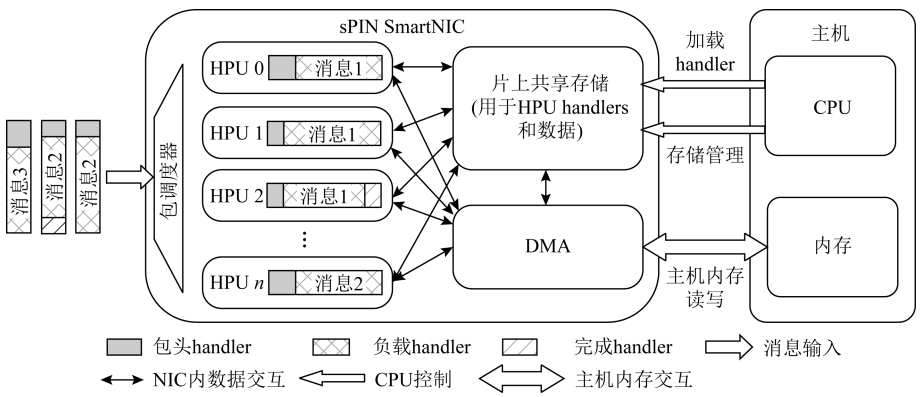



Fig. 6 sPIN SmartNIC network interface design

图 6 sPIN 智能网卡网络接口设计

微软针对基于 FPGA 的智能网卡——Catapult 架构提出了 ClickNP[21]编程框架，用于商用服务器中，以提供高性能的网络功能，如防火墙、网关、负载均衡器等.ClickNP 向用户提供了类似 C 语言语法、面向对象的编程语言，并且提供了近 100 个处理单元(elements)库，解决 FPGA 编程困难的问题，用户将各个功能模块以 elements 为对象进行编写，如图 7 所示，编写的程序经过编译器的预编译后得到中间 C 文件，然后由 FPGA 厂商提供的后端编译器和 C 编译器分别对运行在 FPGA 和主机 CPU 的程序进行编译处理，将不同的任务分配到 FPGA 和 CPU 中，达到 FPGA 与主机端 CPU 协同工作的效果.FPGA 内部采用了模块化设计，映射到 FPGA 的多个 elements 可以异步并行处理数据，类似于多核处理器，elements 之间通过缓冲通道连接，而非共享内存.ClickNP 的设计主要面向商用服务器中网络功能加速的应用场景，在后续工作 AccelNet[3]中，微软使用智能网卡卸载了 SDN 协议栈，向虚拟机提供

高效的 SR-IOV 功能,均是典型的数据密集型设计,对控制复杂的应用逻辑则并非最佳选择.

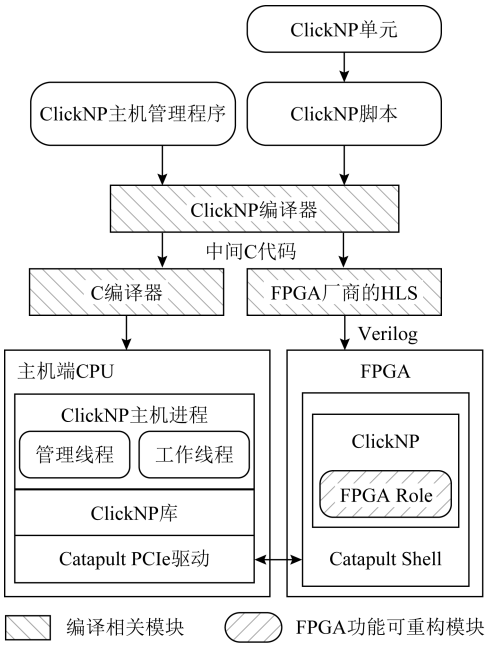



Fig. 7 ClickNP architecture(used in Catapult)

图 7 ClickNP 架构(用于 Catapult)

华盛顿大学研究团队提出基于数据流的 CPU-SmartNIC 协同编程框架——Floem[73],运用类似于 ClickNP 模块化设计的思想,将数据包模块化处理逻辑组件——elements(C 语言实现的 Python 类),映射到智能网卡的硬件资源上,并设计了通用的 elements 库便于用户编程,向用户提供了编程语言、编译器、运行时管理一整套技术支持.通过编程抽象用户可以完成硬件资源访问、逻辑映射、数据包元数据访问、计算卸载、应用加速等操作,编译器负责维护 CPU 和网卡之间的数据传输和缓存机制,运行时则负责 DMA 数据通路的优化.文中使用基于多核网络处理器的智能网卡——Cavium LiquidIO 作为硬件平台,实现了键值存储(key-value store,KVS)、分布式实时数据分析(real-time analytics,RTA)系统的智能网卡应用加速.

### 2.2 控制密集型

数据密集型智能网卡编程框架设计注重对数据流提供高效的数据通路和流水处理,与网络功能、协议处理应用场景更加匹配;而控制密集型的编程框架则更注重对数据进行控制相对复杂的处理,卸载到智能网卡的计算模式一般相对简单,通信模式并不固定,有一定的延时敏感要求,因此,控制密集型设计在调度策略、性能隔离、操作系统支持等方面更有优势,与分布式应用、Bare-Metal 场景更加匹配.控制密集型编程框架的典型设计有 iPipe[13],NICA[74-75],INCA[76],λ-NIC[4].

鉴于 actor[77-79]编程模型支持异构硬件、并行处理、独立内存、动态迁移的特点,华盛顿大学研究团队在 actor 编程模型的基础上进行设计,提出面向多核 SoC 硬件平台的 iPipe[13]编程框架,iPipe 的贡献主要包括调度器、分布式内存抽象和安全隔离 3 个方面.iPipe 的核心是将 FCFS(first come first serve)和 DRR(deficit round robin)调度策略进行结合的混合调度方案,用来协调调度 CPU 和智能网卡之间开销不断变化的执行任务,最大化智能网卡资源的利用率;分布式内存抽象是指各个 actor 之间不共享内存,每个 actor 拥有独立的 ID,actor 可以进行灵活的迁移;安全隔离是指维护多个 actor 在智能网卡上可以并发执行、互不影响,并对意外状况作出反应.由于灵活的调度策略和编程抽象,与 Floem 相比,iPipe 更适合复杂逻辑的分布式应用加速,如 RTA,KVS 等,对于描述状态简单甚至 stateless 的网络功能卸载,基于 FPGA 的设计更合适,文中实现了防火墙、网关、深度包检测等网络功能,其性能均不及 ClickNP.

Mellanox 研究团队提出了基于 FPGA 智能网卡加速数据平面应用的软硬件协同框架——NICA[74-75],并用于自身的 Innova 系列智能网卡,与微软的 ClickNP 框架不同,NICA 框架突破了基于 FPGA 智能网卡在支持操作系统、虚拟化方面的障碍,适用于 Bare-Metal、多租户的应用场景.NICA 通过新的 ikernel(inline kernel)抽象,动态管理智能网卡上的一个或多个专用的硬件加速部件——AFUs(accelerator functional units)[80],同时,NICA 集成了 VMA 内核旁路协议栈[81],实现了 KVM hypervisor 中 AFU 的虚拟化,满足云环境下对网络数据流进行灵活的自定义处理需求.其中,一个 ikernel 抽象即是一个 OS 对象,代表用户程序中的一个 AFU,对进程来说是私有的,可以保护 AFU 的应用和网络状态.AFU 可以由用户自定义,亦可以由云产商提供,根据需求部署,AFU 支持 I/O 通道的虚拟化和细粒度的时分复用来实现 NICA 对虚拟化的支持.文中实现了 KVS 和 IoT 身份认证 2 种应用的加速.

在网计算的抢占式处理编程框架 INCA[76]则在 Portals 4 的基础上,实现了优于 sPIN 的在网计算处理模式,解决了 sPIN 流式处理中智能网卡的

处理能力受计算复杂度(指令数量)、数据包速率、计算单元数量限制的问题.INCA 以抢占式的触发机制对智能网卡的计算资源进行更高效的调度,实现 deadline-free 的处理效果,即对每一个到达的包都能进行及时的处理,将已处理且未能完成处理的包的处理状态保存并发往下一节点做后续处理,支持更加复杂的计算.此外,网络空闲时,智能网卡中的包处理引擎(packet processing engine,PPE)可以用于非网络数据的处理.

基于 NP 多核智能网卡的编程框架 λ-NIC[4] 则面向云计算模式中的 serverless 负载(诸多细粒度的定制化小程序,如 Lambdas),在 P4 语言 Match-Action 编程抽象[37] 的基础上,设计了基于事件的 Match-Lambda 编程抽象,支持更加复杂的操作,数据包通过 Match 之后被发往主机端 CPU 或者智能网卡上对应的 Lambda 处理单元(NP 核)进行处理.同时,该工作使用远程过程调用(remote procedure calls,RPC)技术和 RDMA 技术加速通信,并对智能网卡存储空间访问和 Lambda 任务分配进行了优化,实现了数据隔离和性能隔离.

### 2.3 编程框架小结

根据数据密集型和控制密集型的分类,我们对编程框架面进行总结,得到表 2.数据密集型和控制密集型的设计分别强调对数据流和控制流处理,数据密集型的设计在流水处理和并行度上表现更好,更适合模式较为简单的应用,如网络功能;控制密集型的设计则在灵活性、资源调度、性能隔离,以及面向 Bare-Metal,Serverless 的 OS 支持方面更好,更适合云计算、控制略显复杂的分布式计算应用.2 种编程框架的设计均有基于 Portal 4,P4 Match-Action 的研究,在硬件方面对多核和 FPGA 方式都有实现,但从性能上看,面向数据密集型的设计更适合 FPGA,面向控制密集型的设计更适合多核架构.

根据以上工作的分析,我们发现编程框架的设计过程中需要注重 5 点:1)模块化设计,如 ClickNP,NICA 均采用了模块化设计思想;2)智能网卡与主机端 CPU 协同设计,这一点在基于 Click 包处理编程抽象[82] 的许多编程框架中皆有体现,如 Snap[83-84],NBA[85],ClickNP[21],UNO[86],在 λ-NIC,INCA 的设计中也有实现;3)良好性能隔离和虚拟化的支持,这一点成为了近些年研究中的一个热点问题,如性能隔离在 iPipe[13],NICA[74-75],λ-NIC[4],FairNIC[70] 中皆有体现,对网络虚拟化的研究也受到业内的重视,如 AccelNet[3],Freeflow[68],MasQ[87] 等;4)优化调度策略,提高智能网卡的资源利用效率,如 PIEO[88],Loom[89],在包调度方面提供更加灵活高效的硬件支持,iPipe[13],λ-NIC[4],FairNIC[70] 则在网卡资源方面进行更加合理的调度管理;5)简洁的编程接口,兼顾易用性和灵活性,这一点在 ClickNP,NICA,λ-NIC 等工作中皆有体现.

**Table 2 Comparison of Different SmartNIC Programming Framework**

**表 2 不同智能网卡编程框架的对比**

| 对比方面 | 数据密集型 | 控制密集型 |
|---|---|---|
| 侧重对象 | 数据流 | 控制流 |
| 计算复杂度 | 简单 | 复杂 |
| 灵活性 | 低 | 高 |
| 流水处理能力 | 高 | 低 |
| 并行度 | 高 | 低 |
| 可调度性 | 低 | 高 |
| 隔离性 | 低 | 高 |
| OS 支持 | 弱 | 强 |
| 框架基础 | P4,Portals 4,Click | P4,Portals 4,actor |
| 硬件平台 | MP,FPGA | MP,FPGA |
| 典型框架 | FlexNIC,sPIN,ClickNP,Floem | λ-NIC,INCA,iPipe,NICA |

## 3 应用方向

更快的网络速率、更加复杂的处理场景、更高昂的网络处理开销催生了智能网卡,智能网卡作为一种应用驱动的产物,在众多场景中得到应用.本节将从网络协议处理、网络功能、数据中心应用、科学计算应用 4 个方面介绍智能网卡的典型应用场景.

### 3.1 网络协议处理

智能网卡作为一种具备一定编程能力的网卡,其最基础的功能就是快速处理网络协议,提供高效的网络 I/O.

在网络协议种类方面,传统的网卡多是仅支持一种网络,标准的以太网,或者标准的 Infiniband,或者自定义的网络协议.部分智能网卡(如 Mellanox 网卡)则可以根据用户设置,兼容以太网和 Infiniband,除了传统的 TCP/IP 协议,智能网卡大多支持 RDMA 协议或者其他加速数据通路的协议,如 Portals 4,RoCE v1/v2,iWARP,在数据中心或者高性能计算机集群中提供低延时、高带宽的网络服务;智能网卡甚至支持存储方面的协议,如 NVMe-oF[53].

此外,如 Mellanox 产品还支持 MPLS(multi-protocol label switching)协议、GPU-Direct,迎合当下虚拟化、人工智能的应用场景.

在网络协议处理方面,以微软为例,早期的工作中已经将 TCP/IP 协议中模式固定、计算简单的处理卸载到网卡,如 TCP 校验(chueksum)、IPsec(Internet protocol security)的卸载[90];之后,逐渐有更多的网络协议处理卸载到网卡上,如 RSS 卸载[91]、VMQ(virtual machine queues)卸载[92].近些年,为加速虚拟化的云场景的网络处理,如 NVGRE,VxLAN 的处理也在网卡中得到卸载处理[93],进一步释放主机端 CPU.此外,智能网卡对多种加解密方法提供加速,如 Stringray 智能网卡,可以卸载 Hash 计算、SHA、MD5、PKA(public key accelerator)等.在学术界,不断有智能网卡卸载网络协议的新方式提出,如 TriEC[94]对现有智能网卡卸载纠删码(erasure coding, EC)的方式进行了改善,提出 3 分图式纠删码卸载模式.1RMA[95]则对现有 RDMA 网络可靠连接高开销的通信模式进行改善,提出 Connection-free 的网络连接模式,同时通过软件保证连接的可靠性,并且对安全方面提供了网卡卸载支持.

### 3.2 网络功能卸载

网络功能的作用是通过一系列方式对数据包进行检测和修改,典型的网络功能有防火墙(firewall)、网关(gateway)、入侵检测(instruction detection system,IDS)、负载均衡器(load balancer)、域名服务(domain name service,DNS)等.微软基于 Catapult[19]硬件架构的 ClickNP[21]编程模型一文中主要对卸载网络功能进行了实现和评估,iPipe[13]中也对卸载网络功能进行了实现和评估,文献[14]则对 DNS 进行了卸载,从网络功能的性能上看基于 FPGA 的 ClickNP 优于基于多核处理器的 iPipe,可见,相比于数据中心中的其他复杂应用,逻辑相对简单的网络功能更适合流式处理的实现方法.

此外,智能网卡在卸载 SDN 协议栈、加速 SR-IOV[3]、卸载 NFV、卸载 OVS[96]方面也有很好的应用场景,在微软的 AccelNet[3]工作中得到了充分的体现,NetBricks[97]也对 NFV 进行了卸载.在产业界,Mellanox 则将加速 OVS 的 ASAP2[63](accelerated switching and packet processing)技术应用到新一代的智能网卡产品中.

### 3.3 数据中心应用

智能网卡在数据中心应用的十分广泛,在此总结为 5 类:

1) 卸载一致性协议.如对 Paxos[98-99]一致性协议进行卸载,其中有在交换机端的卸载工作,如文献[100]中使用 P4 交换机完成了 Paxos 一致性协议的卸载,也有在网卡端完成一致性协议的卸载,如文献[14]中的 P4xos、文献[101]均在网卡上完成了一致性协议的卸载工作.

2) 卸载 KVS 相关的应用.如 KV-Direct[11],Lake[12],加速分布式共享内存(distributed shared memory,DSM)[102-103],如 FaRM[104],Grappa[105].在智能网卡软硬件设计的相关研究中,如 iPipe,Floem,FlexNIC,sPIN,NICA 等,皆以 KVS 作为性能评测的重要指标,在商业智能网卡产品中,KV 加速部件也已成为重要的组件.

3) 加速搜索引擎.在文献[19-20]中,微软加速了 Bing 搜索引擎业务,将吞吐提高了 95%.

4) 加速人工智能应用.如 Lynx[6],搭建了以智能网卡为中心来调度管理异构 AI 加速器的神经网络训练、推理加速平台,将 CPU 从任务中释放出来做其他事务的处理;而文献[15-17]则把网络设备作为一种神经网路加速器来使用,卸载神经网络模型中的某些层甚至整个模型,数据在网络传输中被计算,降低延时的同时减轻终端加速器的负载.

5) 提供虚拟化、云环境支持.如微软在 AccelNet[3]中加速 SR-IOV、卸载 OVS,Freeflow[68], MasQ[87]则对 RDMA 网络进行了虚拟化,向多租户提供接近物理网卡性能的虚拟 RDMA 接口.而 FairNIC[70],PIEO[88],Loom[89],1RMA[95]则对云环境下的包调度、性能隔离和数据加密方面进行了研究.Pythia[106]则对 RDMA 数据安全性方面进行了侧信道攻击的尝试,对网卡安全提出了更高的要求.

### 3.4 科学计算应用

智能网卡在科学计算中的应用首先表现在通信加速上,如利用 RDMA 的特性进行非连续数据通信的加速[5,18,107]、集合通信加速[108-110]、MPI 加速[5,107,111-112].其次,智能网卡在科学计算的应用中也可以起到卸载计算的作用,如集合通信加速时 Allreduce 操作中的计算、MPI Tag-Matching[112]均可卸载到网卡处理.再者,针对科学计算应用中存在大量的矩阵计算的特点,在 INCA[76]中,作者使用网卡进行了矩阵转置、卷积、矩阵乘等与应用紧耦合的计算任务的卸载,可将高性能计算应用加速 11%.

### 3.5 应用方向小结

智能网卡在分布式应用中几乎无处不在.在传统的网络通信方面,智能网卡可以满足 RDMA,

TCP/IP 等协议下,基本的网络数据传输甚至部分网络协议的硬件卸载,在如今数据中心网络虚拟化的大趋势下,智能网卡可以提供网络功能的卸载、SR-IOV 的支持、虚拟交换机的卸载等,将部分 CPU 的资源从网络处理中释放出来.E3[113]研究表明,高效利用智能网卡中的低功耗处理器处理合适的数据中心任务可以将能效比提高 3 倍;在用户应用方面,智能网卡在加速一致性协议、KVS 相关应用、搜索引擎、分布式共享存储、神经网络等数据中心应用方面皆有优秀表现,在科学计算领域的集合通信加速、MPI 加速、矩阵计算中也表现出重要价值.

相信在未来,依然会有各色的新应用驱动智能网卡向更强大的方向发展,如何适应新的应用需求将是智能网卡设计中的一个重要问题.在此,我们从应用场景的专用性和通用性 2 个方面做简单的总结:1)专用性.智能网卡是应用驱动的产物,在应用相对固定的情况下,如网络功能卸载、机器学习模型训练、KV 处理、特定算法的加解密,可以针对特定的一类或者几类应用,做针对性的加速设计,如采用 FPGA 架构、ASIC 的加速部件.2)通用性.在应用场景相对复杂的情况下,如云环境、多租户场景,需要通用性强的智能网卡,可以采用基于网络处理器、甚至通用处理器的多核架构,提供灵活性更友好的编程接口,满足新应用场景下的可用性.

## 4 热点问题

目前,在产业界和学术界中,涌现出各种智能网卡相关的热点问题,在本节将分为架构及编程框架、应用、协议三大类进行介绍.

### 4.1 架构及编程框架探索

目前智能网卡的硬件架构主要分为三大类,分别是基于 FPGA,MP,ASIC 的设计,如 1.3 节中表 1 中所示各种架构在性能、成本、功耗上各有千秋.此外,近几年,粗粒度可重构架构[114](coarse-grained reconfigurable architecture, CGRA)作为一种使用多个可重构单元解决领域专用的处理器设计方案,以优于 FPGA 1~2 倍的能效比、更接近 ASIC 的性能、优于 FPGA 的编程灵活性,得到业内的认可和关注.在智能网卡的设计中,ClickNP 虽是基于 FPGA 的设计,但是其模块化的设计与 CGRA 有异曲同工之妙;GP-SoC 的多核设计思路成为部分厂家的选择,但是 GP 的通用性和易用性在另一方面则限制了专用性和能效比;NP-SoC 的设计更像 CGRA,但 CGRA 具备更短的功能重构时间,支持配置流和数据流同时驱动.目前,CGRA 的技术还不够成熟,如何设计智能网卡中的可重构单元、如何建立可重构单元之间的拓扑关系(Mesh, Torus 等)、是否增加其他处理器进行功能辅助、有异构处理器存在的情况下是否采用共享内存的设计、可重构单元之间使用类似于 RMT[38]架构的流水处理还是类似于 dRMT[115]的独立处理,各种问题需要进一步的探索.

近 5 年,涌现出诸如 ClickNP, Floem, FlexNIC, sPIN, NICA 等多种出色的智能网卡编程框架[2,7,21,73-76],定制与硬件结构协同优化的编程框架、功能调度机制[13]、任务切分机制[21,82-86,116]十分重要,新的硬件架构需要配套的编程框架作为支撑方可最大化发挥智能网卡的通信、计算能力.

### 4.2 应用探索

如本文第 3 节所介绍,智能网卡在网络协议处理、网络功能卸载、数据中心应用、科学计算应用中均表现出强大的加速能力.可见,智能网卡正在逐渐将成熟的加速部件模块化集成,同时,卸载的任务与用户应用的关系越来越紧密,如今已有使用智能网卡卸载神经网络模型的运算[15-17]、矩阵计算[76]的探索.在 Bare-Metal 和云环境中,智能网卡在提供虚拟化支持、性能保障、性能隔离等方向也将继续发挥重要作用[3,70,74-75,87,89,117].而使用智能网卡为中心搭建加速器互联平台——Lynx[6]的探索更是让人耳目一新,可见,智能网卡正在尽可能卸载更多力所能及的 CPU 处理任务,将 CPU 资源释放出来用于处理控制逻辑更加复杂的任务.

### 4.3 协议接口探索

智能网卡的可编程特性使其在通信协议的处理上具备一定的灵活性,为满足不同应用的需求,部分研究工作对已有的协议或者接口进行了拓展.sPIN[2]对 Portals 4[72]进行拓展,增加了智能网卡对加速通信和计算的支持;P4[37]则完全定义了以 Match-Action 为基础的协议无关的可编程包处理模式;StRoM[118]则对 RDMA 语义进行了拓展,增加了网卡进行可编程处理的的支持,如设计了 RDMA RPC verb 用于支持对 RPC 的加速处理;RIMA[69]针对 Infiniband 协议中的共享接收队列(shared receive queue, SRQ)造成的内存浪费问题,设计新的 append queue verb 及处理架构,在保证吞吐和时延的情况下实现了对 SRQ 的缓冲区更高效的管理;1RMA[95]则设计了 Connection-free 的连接模式,解决 RDMA 在数据中心中扩展性差的

问题.在应用需求更加灵活的情况下,拓展更丰富的智能网卡协议接口,甚至是制定标准化的面向智能处理的通信协议也将成为可能.

## 5 智能网卡设计与测试

### 5.1 设计方法

基于第 1~3 节对智能网卡相关工作的认识和理解,我们总结出一种系统的智能网卡设计方法,主要由图 8 所示的设计步骤组成:

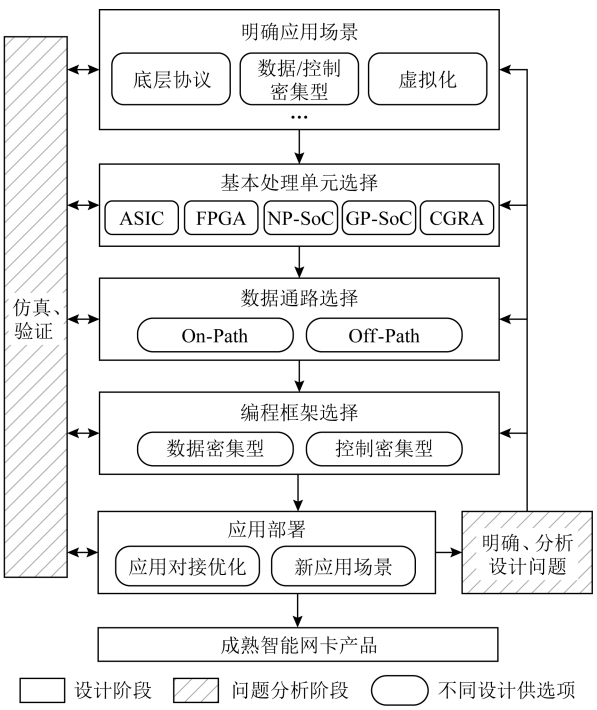



Fig. 8 SmartNIC design method

图 8 智能网卡设计方法

1) 明确应用场景.智能网卡是一种典型的应用驱动产物,因此,确定欲设计的智能网卡使用的底层协议(以太网、Infiniband,或者其他自定义协议)、卸载应用的主要类型(数据密集型、控制密集型)、虚拟化的支持情况等,是进行架构设计之前必要的工作.

2) 明确智能网卡设计的基础处理单元.根据明确的应用场景和成本预算,确定使用 FPGA,ASIC,MP,还是 CGRA 作为基础处理单元.在应用场景相对固定的情况下,需要采用专用性强的设计,可以针对特定的一类或者几类应用,做针对性的加速设计,可以优先考虑 FPGA,ASIC,CGRA 作为基础处理单元;在应用场景相对复杂的情况下,如云环境、多租户场景,需要采用通用性强的设计,可以优先考虑基于网络处理器、甚至通用处理器的 MP 架构,提供灵活性更友好的编程接口,满足复杂的应用场景.此外,可以考虑使用模块化开发的思想,对智能网卡进行完全异构化的设计,比如通用处理器搭配卸载成熟应用的专用加速部件和处理器网络协议的专用网络控制器,这一步可以参考 1.3 节、3.5 节、4.1 节.

3) 明确数据通路架构.根据设定的的应用场景和选择的基本处理单元选择 On-Path 或者 Off-Path 的数据通路设计.一情况下,On-Path 架构对应的基础架构一般为 FPGA,ASIC,NP,更适合数据密集型应用场景,可以对数据直接进行传输路径上的流式处理;Off-Path 架构对应的基础架构一般为 GP 处理器,更适合控制密集型应用场景,可以对数据进行更为复杂的通用处理.这一步可以参考 1.3 节及 2.3 节.

4) 设计软硬件协同编程框架.根据明确的应用场景、处理单元、数据通路设计编程框架,同时考虑编程框架的调度策略、资源使用效率、灵活性、易用性、可扩展性、虚拟化下的性能隔离以及智能网卡与 CPU 的协同工作问题,这一步可以参考 2.3 节表 2.

5) 仿真验证.对智能网卡进行软硬件协同设计开发周期漫长,尤其是硬件开发更为繁琐,确定合适的设计架构和设计参数是提高网卡设计效率的重要环节,因此,在设计的不同阶段,通过适当的软件模拟、硬件仿真验证是解决部分设计问题的重要方式,然而目前通用模拟器 GEM5,NS-3 等在网卡仿真方面依然存在适用性、准确性等问题,需要开发者自行开发对应的模拟器来进行系统级的仿真实验.

6) 对接应用.将智能网卡系统平台与真实应用对接,优化应用、充分发挥智能网卡的性能特性,由于智能网卡的设计周期较长,面对新的应用场景,需要及时发现、总结、解决智能网卡的设计问题,迭代新的设计版本.

### 5.2 测试方法

该部分对智能网卡的测试分为 3 个阶段进行介绍,涉及到开发阶段的软件仿真测试分析、硬件验证测试分析和商业产品测试分析.

1) 模拟器模拟测试分析.很多软硬件协同设计在具体设计方案以及相关参数确定之前需要进行细致的模拟分析,因此模拟器分析在芯片设计,尤其是复杂应用场景下的网络设计中十分重要.在智能网卡的模拟器测试分析中,需要重点关注 4 个方面:① PCIe 读写效率;②网卡 SRAM,DRAM 的开销和使用效率;③调度情况是否达到预期;④在吞吐、延时、使用规模方面是否达到预期.目前,虽然有

GEM5,NS-3 等系统仿真工具或者网络仿真工具,但是在网卡仿真,尤其是智能网卡仿真方面,没有较成熟的网络芯片微体系结构模拟器,进行细致的仿真需要很大的工程量,但是成熟的模拟测试分析会大大提高智能网卡的设计效率.

2) FPGA 验证测试分析.经过模拟分析、修正获得相对成熟的设计框架和设计参数之后,进而可以使用 FPGA 进行 RTL(register transfer level)验证,以获取更贴近真实设计的测试效果,FPGA 开发、芯片后端验证是一个复杂的过程,需要花费大量的人力和物力,并且需要相当的集成电路设计经验.在此,不做详细的介绍.

3) 智能网卡产品测评.对于商业智能网卡产品,可以从 5 个方面进行测评分析:①吞吐量、时延、线速处理能力,这 3 个方面是一个网络产品的基础性能指标;②支持的规模,如支持 RDMA 的智能网卡可以支持到多少 QP 连接;③卸载性能,对于相同应用的卸载场景,测试网卡资源使用效率情况、主机端 CPU 开销占比情况;④编程接口的灵活性、编译器的编译效率,比如测试从用户编写的特定的智能网卡应用程序到网卡发挥卸载作用的配置时间(如 Loom[89]中,测试了 Mellanox ConnectX-4 网卡配置 QoS 的时间)、对比使用纯硬件语言编写应用加速与使用编程接口经过编译器编译获得的硬件综合效果以及应用加速效果等;⑤核心处理器的处理能力,比如测试基于 NP 智能网卡的单核处理能力[13]、基于 FPGA 智能网卡的模块间并行处理吞吐等.

## 6 总结展望

本文总结了自智能网卡兴起之后相关的重要学术研究和产业界典型产品.对基础架构设计进行了基本处理单元、数据通路 2 维分类,分析对比了不同基础设计架构的优缺点;对编程框架进行了数据密集型和控制密集型分类,并结合基础架构进行了对比分析;对智能网卡的重点应用方向进行了归类总结;此外,本文指出了目前智能网卡相关的热点研究问题.最终本文根据总结和分析,提出了一种系统的智能网卡设计思路和测试方法.

目前,基于不同基础架构的智能网卡几乎都有产业界的商业产品,其内部微架构的具体实现方法对外均是不透明的,至今,业内没有较为一致的评价标准.因此,无论从学术角度还是产业发展的角度,对基础架构的分析依然需要做很多细致的工作,通过不断的架构探索、更加细粒度的测试来比较分析不同微架构实现智能网卡的详细差异,对性能、能效比、应用场景展开细致的讨论,提出可靠的数据分析,这对智能网卡的设计将产生重要的指导作用.此外,高效的软硬件协同编程框架和不断涌现出来的应用需求对智能网卡的系统化设计也提出了更高的要求,甚至,面向智能网卡的网络协议、面向在网计算的智能化网络处理模式也需要进行标准化的尝试.

**作者贡献声明**:马潇潇负责搜集、整理产业界和学术界的智能网卡研究工作,以及文章整体架构设计、撰写和修改;杨帆负责智能网卡基础架构部分的分类和特性分析,以及智能网卡软硬件协同设计部分的指导;王展负责相关研究现状的补充和未来研究热点的指导;元国军负责智能网卡应用场景相关内容的补充,尤其是智能网卡在机器学习中的应用;安学军负责文章分类逻辑的调整、整体思路的指导,以及文章最后的总结.

## 参 考 文 献

[1] Ethernet Alliance. 2020 roadmap [EB/OL]. [2020-08-09]. https://ethernetalliance.org/technology/2020-roadmap/

[2] Hoefler T, Girolamo S, Taranov K, et al. sPIN: High-performance streaming processing in the network [C/OL] // Proc of the Int Conf for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2017 [2021-12-09]. https://doi.org/10.1145/3126908.3126970

[3] Daniel F, Andrew P, Sambhrama M, et al. Azure accelerated networking: SmartNICs in the public cloud [C] // Proc of the 15th Symp on Networked Systems Design and Implementa-tion. Berkeley, CA: USENIX Association, 2018: 51-66

[4] Sean C, Muhammad S, Balaji P, et al. λ-NIC: Interactive serverless compute on programmable SmartNICs [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2019: 151-152

[5] Ma Xiaoxiao, Lu Gang, Fu Binzhang, et al. Implementation method and performance analysis of non-contiguous data communication in network [J]. Chinese Journal of Computers, 2020, 43(6): 1123-1138 (in Chinese)
(马潇潇, 陆钢, 付斌章, 等. 非连续数据网络通信实现方法和性能分析[J]. 计算机学报, 2020, 43(6): 1123-1138)

[6] Maroun T, Lina M, Mark S. Lynx: A SmartNIC-driven accelerator-centric architecture for network servers [C] // Proc of the 25th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2020: 117-131

[7] Antoine K, SImon P, Naveen K, et al. High performance packet processing with FlexNIC [C] //Proc of the 21st Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2016: 67-81

[8] Kaushik K R, Jayaram M, Alan L, et al. SNICh: Efficient last hop networking in the data center [C] //Proc of the 6th ACM/IEEE Symp on Architectures for Networking and Communications Systems. New York: ACM, 2010: No.26

[9] Li Xiaoyao, Wang Xiuxiu, Liu Fangming, et al. DHL: Enabling flexible software network functions with FPGA acceleration [C] //Proc of the 38th Int Conf on Distributed Computing Systems. Piscataway, NJ: IEEE, 2018. doi: 10.1109/ICDCS.2018.00011

[10] Alexander R, Muhammad S, Tushar S, et al. Elastic RSS: Co-Scheduling packets and cores using programmable NICs [C] //Proc of the 3rd Asia-Pacific Workshop on Networking. New York: ACM, 2019: 71-77

[11] Li Bojie, Ruan Zhenyuan, Xiao Wencong, et al. KV-Direct: High-performance in-memory key-value store with programmable NIC [C] //Proc of the 26th Symp on Operating Systems Principles. New York: ACM, 2017: 137-152

[12] Tokusashi Y, Matsutani H, Zilberman N. LaKe: The power of in-network computing [C] //Proc of the Int Conf on ReConFigurable Computing and FPGAs. Piscataway, NJ: IEEE, 2018: 1-8. doi: 10.1109/RECONFIG.2018.8641696

[13] Liu Ming, Cui Tianyi, Henry S, et al. Offloading distributed applications onto smartNICs using iPipe [C] //Proc of the Special Interest Group on Data Communication. New York: ACM, 2019: 318-333

[14] Tokusashi Y, Dang T, Pedone F, et al. The case for in-network computing on demand [C/OL] //Proc of the 14th EuroSys Conf. New York: ACM, 2019: 1-16 [2021-12-09]. https://doi.org/10.1145/3302424.3303979

[15] Siracusano G, Bifulco R. In-network neural networks [J]. arXiv preprint, arXiv:1801.05731, 2018

[16] Sanvito D, Siracusano G, Bifulco R. Can the network be the AI accelerator [C] //Proc of the Morning Workshop on In-Network Computing. New York: ACM, 2018: 20-25

[17] Swamy T, Rucker A, Shahbaz M, et al. Taurus: An intelligent data plane [J].arXiv preprint, arXiv:2002.08987, 2020

[18] Salvatore D G, Konstantin T, Andreas K, et al. Network-accelerated non-contiguous memory transfers [C/OL] //Proc of the Int Conf for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2019: 1-14 [2021-12-09]. https://doi.org/10.1145/3295500.3356189

[19] Putnam A, Caulfield A, Chung E, et al. A reconfigurable fabric for accelerating large-scale datacenter services [C] //Proc of the 41st ACM/IEEE Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2014: 13-24

[20] Caulfield A, Chung E, Putnam A, et al. A cloud-scale acceleration architecture [C] //Proc of the 49th Annual IEEE/ACM Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2016: 1-13. doi: 10.1109/MICRO.2016.7783710

[21] Li Bojie, Tan Kun, Luo Layong, et al. ClickNP: Highly flexible and high performance network processing with reconfigurable hardware [C/OL] // /Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2016: 1-14 [2021-12-09]. https://doi.org/10.1145/2934872.2934897

[22] Caulfield A, Costa P, Ghobadi M. Beyond SmartNICs: Towards a fully programmable cloud [C/OL] //Proc of the 19th Int Conf on High Performance Switching and Routing. Piscataway, NJ: IEEE, 2018: 1-6 [2021-12-09]. https://ieeexplore.ieee.org/document/8850757

[23] Mellanox. Mellanox BlueField series SmartNIC white paper [EB/OL]. [2020-08-10]. https://www.mellanox.com/files/doc-2020/pb-bluefield-2-smartnic-vpi.pdf

[24] Netronome. Agilio series SmartNIC [EB/OL]. [2020-08-10]. https://www.netronome.com/products/smartnic/overview/

[25] Broadcom. Stingray series SmartNIC [EB/OL]. [2020-08-10]. https://www.broadcom.com/products/ethernet-connectivity/smartnic

[26] Cavium. LiquidIO SmartNIC family [EB/OL]. [2020-08-10]. https://www.marvell.com/products/ethernet-adapters-and-controllers/liquidio-smart-nics.html

[27] Netronome. What makes a NIC a SmartNIC, and why is it needed?[EB/OL]. 2016 [2020-08-10]. https://www.netronome.com/blog/what-makes-a-nic-a-smartnic-and-why-is-it-needed/

[28] PC Magazine. SmartNIC [EB/OL]. [2020-08-10]. https://www.pcmag.com/encyclopedia/term/smartnic

[29] Kevin D. Whatis a SmartNIC [EB/OL]. Mellanox, 2018 [2020-08-10]. https://blog.mellanox.com/2018/08/defining-smartnic/

[30] BittWare. SmartNIC shell: Jumpstart your 100G NIC project [EB/OL]. [2020-08-10]. https://www.bittware.com/fpga/smartnic/

[31] Ethernity Networks. ACE-NIC SmartNICs [EB/OL]. [2020-08-10]. https://ethernitynet.com/products/ace-nic-smartnics/

[32] Annapurna Labs. Announces availability of home network and storage platform-on-chip and subsystem solution [EB/OL]. 2016 [2020-08-10]. https://www.annapurnalabs.com/annapurna-labs-an-amazon-company-announces-availability-of-home-network-and-storage-platform-on-chip-and-subsystem-solutions.html

[33] Huawei. IN series SmartNIC [EB/OL]. [2020-08-10]. https://support.huawei.com/enterprise/zh/category/intelligent-accelerator-components-pid-1548148324389?submodel=doc

[34] Intel. Stratix V Device Handbook [EB/OL]. 2011[2020-08-10]. https://www.intel.com/content/www/us/en/programmable/documentation/nik1409774008946.html

[35] Dong Yaozu, Yang Xiaowei, Li Jianhui, et al. High performance network virtualization with SR-IOV.[J] Journal of Parallel and Distributed Computing, 2012, 72(11): 1471-1480

[36] Intel. Arria 10 Device DataSheet [EB/OL]. 2013 [2020-08-10]. https://www.intel.com/content/www/us/en/programmable/documentation/mcn1413182292568.html

[37] Bosshart P, Daly D, Gibb G, et al. P4: Pogramming protocol-independent packet processors [C] // /Proc of the Special Interest Group on Data Communication. New York: ACM, 2014: 87-95

[38] Bosshart P, Gibb G, Kim H. S, et al. Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN [C] //Proc of the Special Interest Group on Data Communication. New York: ACM, 2013: 99-110

[39] Netcope. Netcope P4 [EB/OL]. [2020-08-10]. https://www.netcope.com/en/products/netcopep4

[40] Xilinx. Virtex UltraScale FPGA data sheet [EB/OL]. 2019 [2020-08-10]. https://www.xilinx.com/support/documentation/data_sheets/ds893-virtex-ultrascale-data-sheet.pdf

[41] Eynx. nxFramework [EB/OL]. [2020-08-10]. https://www.enyx.com/nxframework/

[42] Reflex. Reflex CES [EB/OL]. [2020-08-10]. https://www.reflexces.com/intel-fpga/stratix-10-intel-fpga

[43] Intel. Stratix 10 FPGA [EB/OL]. [2020-08-10]. https://www.intel.com/content/www/us/en/products/programmable/fpga/stratix-10.html

[44] Xilinx. Xilinx ultrascale overview [EB/OL]. [2020-08-10]. https://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf

[45] Silicom. Silicom programmable FPGA server adapter [EB/OL]. [2020-08-10]. https://www.silicom-usa.com/cats/server-adapters/programmable-fpga-server-adapter/

[46] Mellanox. Innova series SmartNIC white paper [EB/OL]. [2020-08-10]. https://www.mellanox.com/sites/default/files/doc-2020/pb-innova-2-flex.pdf

[47] Mellanox. ConnextX family intelligent data-center network adapters [EB/OL]. [2020-08-11]. https://cn.mellanox.com/products/ethernet/connectx-smartnic

[48] Intel. The FPGA SmartNIC for network acceleration PAC N3000 [EB/OL]. 2020 [2020-08-11]. https://www.intel.com/content/www/us/en/programmable/products/boards_and_kits/dev-kits/altera/intel-fpga-pac-n3000/overview.html

[49] Intel. Intel FPGA programmable acceleration card D5005 data sheet [EB/OL]. 2019 [2020-08-10]. https://www.intel.com/content/www/us/en/programmable/documentation/cvl1520030638800.html

[50] Xilinx. X2 series Ethernet adapters [EB/OL]. [2020-08-11]. https://www.xilinx.com/products/boards-and-kits/x2-series.html#specifications

[51] Xilinx. Onload application acceleration software [EB/OL]. [2020-08-11]. https://www.xilinx.com/products/boards-and-kits/8000-series.html#onload

[52] Xilinx. Alveo U25 SmartNIC accelerator card [EB/OL]. [2020-08-11]. https://www.xilinx.com/publications/product-briefs/alveo-u25-product-brief.pdf

[53] NVMe-oF. NVMe_Over_Fabrics [EB/OL]. [2020-08-11]. https://www.nvmexpress.org/wp-content/uploads/NVMe_Over_Fabrics.pdf

[54] Xilinx. Vitis unified software platform [EB/OL]. [2020-08-11]. https://www.xilinx.com/products/design-tools/vitis/vitis-platform.html#documentation

[55] Netronome. NFP-4000 flow processer [EB/OL]. [2020-05-11]. https://www.netronome.com/m/documents/PB_NFP-4000.pdf

[56] Cavium. OCTEON multi-core MIPS64 processors [EB/OL]. [2020-08-11]. https://www.marvell.com/products/infrastructure-processors/multi-core-processors/octeon-multi-core-mips64-processors.html

[57] SolidRun. NXP LX2160A family [EB/OL]. [2020-08-11]. https://www.solid-run.com/nxp-lx2160a-family/#top

[58] Silicom. programmable NPU server adapters [EB/OL]. [2020-08-11]. https://www.silicom-usa.com/cats/server-adapters/programmable-npu-server-adapters/

[59] Kalray. MPPA manycore, a massively parallel processor array architecture [EB/OL]. [2021-12-07]. https://www.kalrayinc.com/products/mppa-technology

[60] Kalray. Acceleration from cloud to the edge and embedded systems [EB/OL]. [2021-12-07]. https://www.kalrayinc.com/products/cards

[61] Broadcom. BCM58800 NetXtreme S-series named linley group's best embedded processor [EB/OL]. 2018 [2020-08-11]. https://www.broadcom.com/blog/bcm58800-netxtreme-s-series-named-linley-group-s-best-embedded-processor

[62] Cavium. Marvell FastLinQ Ethernet NICs [EB/OL]. [2020-08-11]. https://www.marvell.com/products/ethernet-adapters-and-controllers/fastlinq-performance-nics/documents.html

[63] Mellanox. ASAP-accelerated switch and packet processing [EB/OL]. 2019 [2020-08-11]. http://www.mellanox.com/page/asap2?mtag=asap2

[64] Broadcom. The TruFlow flow processing engine [EB/OL]. 2019 [2020-08-11]. https://www.broadcom.com/applications/data-center/cloud-scale-networking

[65] Chen Guo, Lu Yuanwei, Li Bojie, et al. MP-RDMA: Enabling RDMA with multi-path transport in datacenters [J]. IEEE/ACM Transactions on Networking. 2019, 27(6): 2308-2323

[66] Ma Teng, Ma Tao, Song Zhuo, et al. X-RDMA: Effective RDMA middleware in large-scale production environments [C] //Proc of the IEEE Int Conf on Cluster Computing. Piscataway, NJ: IEEE, 2019: 1-12. doi: 10.1109/CLUSTER.2019.8891004

[67] Xue Jilong, Miao Youshan, Chen Cheng, et al. Fast distributed deep learning over RDMA [C/OL] //Proc of the 14th EuroSys Conf. New York: ACM, 2019, 44 [2021-12-09]. https://doi.org/10.1145/3302424.3303975

[68] Daehyeok K, Yu Tianlong, Liu Hongqiang, et al. FreeFlow: Software-based virtual RDMA networking for containerized clouds [C] //Proc of the 16th Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2019: 113-126

[69] Xue Jiachen, Vijaykumar T N, Thottethodi M. Network interface architecture for remote indirect memory access (RIMA) in datacenters. [J/OL]. ACM Transactions on Architecture and Code Optimization, 2020, 17(2). [2020-10-07]. https://doi.org/10.1145/3374215

[70] Grant S, Yelam A, Bland M, et al. SmartNIC performance isolation with FairNIC: Programmable networking for the cloud [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2020: 681-693

[71] Hansen S, Sujal D. Fabric-agnostic RDMA with OpenFabrics enterprise distribution: Promises, challenges, and future direction [C/OL] //Proc of the 2006 ACM/IEEE Conf on Supercomputing. New York: ACM, 2006 [2021-12-09]. https://doi.org/10.1145/1188455.1188479

[72] Barrett B, Brightwell R, Grant R, et al. Portals 4 network programming interface [C/OL] //Proc of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis. Piscataway, NJ: IEEE, 2012 [2020-08-11]. https://doi.org/10.1109/SC.Companion.2012.264

[73] Phothilimthana P, Liu Ming, Kaufmann A, et al. Floem: A programming system for NIC-accelerated network applications [C] //Proc of the 13th USENIX Association Symp on Operating Systems Design and Implementatcion. Berkeley, CA: USENIX Association, 2018: 663-679

[74] Eran H, Zeno L, Malka G, et al. NICA: OS support for near-data network application accelerators [C/OL] //Proc of the Int Workshop on Multi-core and Rack-scale Systems. 2017 [2020-08-11]. https://haggaie. github. io/publications/2017-mars-nica

[75] Eran H, Zeno L, Tork M, et al. NICA: An infrastructure for inline acceleration of network applications [C] //Proc of the USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2019: 345-362

[76] Schonbein W, Grant R, Matthew G F, et al. INCA: In-network compute assistance [C/OL] //Proc of the Int Conf for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2019: No. 54 [2021-12-09]. https://doi.org/10.1145/3295500.3356153

[77] Agha G. Actors: A Model of Concurrent Computation in Distributed Systems [M/OL]. Cambridge, MA: Massachusetts Institute of Technology Press, 1987 [2020-10-07]. https://doi.org/10.1016/0167-6423(88)90028-7

[78] Hewitt C, Bishop P, Steiger R. A universal modular ACTOR formalism for artificial intelligence [C] //Proc of the 3rd Int Joint Conf on Artificial Intelligence. San Francisco, CA: Morgan Kaufmann, 1973: 235-245

[79] Srinivasan S, Mycroft A. Kilim: Isolation-typed actors for Java [C] //Proc of the 22nd European Conf on Object-Oriented Programming. Berlin: Springer, 2008: 104-129

[80] Intel. Accelerator functional unit (AFU) developer's guide [EB/OL]. 2018 [2020-08-11]. https://www. intel. com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-afu-dev-v1-1.pdf

[81] Mellanox. libvma: Linux user-space library for network socket acceleration based on RDMA compatible network adaptors [EB/OL]. 2018 [2020-08-11]. https://github. com/mellanox/libvma

[82] Morris R, Kohler E, Jannotti J, et al. The click modular router [C] //Proc of the 17th ACM Symp on Operating Systems Principles. New York: ACM, 1999: 217-231

[83] Sun Weibin, Ricci R. Fast and flexible: Parallel packet processing with GPUs and Click [C] //Proc of the 9th ACM/IEEE Symp on Architectures for Networking and Communications Systems. New York: ACM, 2013: 25-36

[84] Marty M, Kruijf M, Adriaens J, et al. Snap: A microkernel approach to host networking [C] //Proc of the 27th ACM Symp on Operating Systems Principles. New York: ACM, 2019: 399-413

[85] Joongi K, Keon J, Keunhong L, et al. NBA (network balancing act): A high-performance packet processing framework for heterogeneous processors [C] //Proc of the 10th European Conf on Computer Systems. New York: ACM, 2015: No.22

[86] Le Yanfang, Chang Hyunseok, Mukherjee S, et al. UNO: Unifying host and smart NIC offload for flexible packet processing [C] //Proc of the 2017 Symp on Cloud Computing. New York: ACM, 2017: 506-519

[87] He Zhiqiang, Wang Dongyang, Fu Binzhang, et al. MasQ: RDMA for virtual private cloud [C/OL] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2020: 1-14 [2021-12-09]. https://doi. org/10. 1145/3387514.3405849

[88] Shrivastav V. Fast, scalable, and programmable packet scheduler in hardware [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2019: 367-379

[89] Stephens B, Akella A, Swift M. Loom: Flexible and efficient NIC packet scheduling [C] //Proc of the 16th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2019: 33-46

[90] Microsoft. TCP/IP offload overview [EB/OL]. 2019 [2020-08-11]. https://docs. microsoft. com/en-us/windows-hardware/drivers/network/tcp-ip-offload

[91] Microsoft. Introduction to receive side scaling [EB/OL]. 2017 [2020-08-11]. https://docs. microsoft. com/en-us/windows-hardware/drivers/network/introduction-to-receive-side-scaling

[92] Microsoft. Virtual machine queue (VMQ) overview [EB/OL]. 2020 [2020-08-11]. https://docs. microsoft. com/en-us/windows-hardware/drivers/network/virtual-machine-queue--vmq-

[93] Microsoft. Network virtualization using generic routing encapsulation (NVGRE) task offload [EB/OL]. 2017 [2020-08-11]. https://docs.microsoft.com/en-us/windows-hardware/drivers/network/network-virtualization-using-generic-routing-encapsulation--nvgre--task-offload

[94] Shi Haiyang, Lu Xiaoyi. TriEC: Tripartite graph based erasure coding NIC offload [C] //Proc of the Int Conf for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2019: No.44, 1-34

[95] Singhvi A, Akella A, Gibson D, et al. 1RMA: Re-envisioning remote memory access for multi-tenant datacenters [C] // Proc of the Annual Conf of the ACM Special Interest Group on Data Communication. New York: ACM, 2020: 708-721

[96] Pfaff B, Pettit J, Koponen T, et al. The design and implementation of Open vSwitch [C] //Proc of the 12th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2015: 117-130

[97] Panda A, Han Sangjin, Jang Keon, et al. NetBricks: Taking the V out of NFV [C] // Proc of the 12th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2016: 203-216

[98] Lamport L. The part-time parliament [J]. ACM Transactions on Computer Systems. 1998,16(2): 133-169

[99] Kirsch J, Amir Y. Paxos for system builders: An overview [C] //Proc of the 2nd Workshop on Large-Scale Distributed Systems and Middleware. New York: ACM, 2008: No.3

[100] Dang H, Canini M, Pedone F, et al. Paxos made switch-y [J]. ACM SIGCOMM Computer Communication Review, 2016, 46(2): 18-24

[101] Yang Fan, Zhang Peng, Wang Zhan, et al. Accelerating Byzantine fault tolerance with in-network computing [J]. Journal of Computer Research and Development, 2021, 58(1): 164-177 (in Chinese)
(杨帆, 张鹏, 王展, 等. 基于在网计算加速的拜占庭容错算法[J]. 计算机研究与发展, 2021, 58(1): 164-177)

[102] An Zhongqi, Zhang Yunyao, Xing Jing, et al. Optimization of the key-value storage system based on fused user-level I/O[J]. Journal of Computer Research and Development, 2020, 57(3): 649-659 (in Chinese)
(安仲奇, 张云尧, 邢晶, 等. 基于用户级融合 I/O 的 Key-Value 存储系统优化技术研究[J]. 计算机研究与发展, 2020, 57(3): 649-659)

[103] Chen Youmin, Lu Youyou, Luo Shengmei, et al. Survey on RDMA-based distributed storage systems [J]. Journal of Computer Research and Development, 2019, 56(2): 227-239 (in Chinese)
(陈游旻, 陆游游, 罗圣美, 等. 基于 RDMA 的分布式存储系统研究综述[J]. 计算机研究与发展, 2019, 56(2): 227-239)

[104] Dragojević A, Narayanan D, Hodson O, et al. FaRM: Fast remote memory [C] //Proc of the 11th USENIX Conf on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2014: 401-414

[105] Nelson J, Holt B, Myers B, et al. Latency-tolerant software distributed shared memory [C] //Proc of the USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2015: 291-305

[106] Tsai S, Payer M, Zhang Yiying. Pythia: Remote oracles for the masses [C] //Proc of the 28th USENIX Security Symp. Berkeley, CA: USENIX Association, 2019: 693-710

[107] Li Mingzhe, Subramoni H, Hamidouche K, et al. High performance MPI datatype support with user-mode memory registration: Challenges, designs, and benefits [C] //Proc of the 2015 Int Conf on Cluster Computing. Piscataway, NJ: IEEE, 2015: 226-235

[108] Rabinovitz I, Bloch G, Bloch N. Overlapping computation and communication barrier algorithms and ConnectX-2 CORE-Direct capabilities [C] //Proc of the Int Symp on Parallel and Distributed Processing. Piscataway, NJ: IEEE, 2010: 1-8

[109] Venkata M, Graham R, Ladd J, et al. ConnectX-2 CORE-Direct enabled asynchronous broadcast collective communications [C] //Proc of the IEEE Int Symp on Parallel and Distributed Processing. Piscataway, NJ: IEEE, 2011: 781-787

[110] Schneider T, Hoefler T, Grant R, et al. Protocols for fully offloaded collective operations on accelerated network adapters [C] //Proc of the 42nd Int Conf on Parallel Processing. Piscataway, NJ: IEEE, 2013: 593-602

[111] Gainaru A, Graham R, Polyakov A, et al. Using InfiniBand hardware gather-scatter capabilities to optimize MPI all-to-all [C] //Proc of the 23rd European MPI Users' Group Meeting. New York: ACM, 2016: 167-179

[112] Xiong Qingqing, Skjellum A, Herbordt M. Accelerating MPI message matching through FPGA offload [C] //Proc of the 28th Int Conf on Field Programmable Logic and Applications. Piscataway, NJ: IEEE, 2018: 191-194

[113] Liu Ming, Peter S, Krishnamurthy A, et al. E3: Energy-efficient microservices on SmartNIC-accelerated servers [C] //Proc of the USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2019: 363-378

[114] Liu Leibo, Zhu Jianfeng, Li Zhaoshi, et al. A survey of coarse-grained reconfigurable architecture and design: Taxonomy, challenges, and applications [J/OL]. ACM Computing Surveys, 2019, 52(6). [2020-10-07]. https://doi.org/10.1145/3357375

[115] Chole S, Fingerhut A, Ma Sha, et al. dRMT: Disaggregated programmable switching [C/OL] //Proc the Conf of the ACM Special Interest Group on Data Communication. New York: ACM, 2017: 1-14. [2021-12-09]. https://doi.org/10.1145/3098822.3098823

[116] Wang Shuhe, Meng Zili, Sun Chen, et al. SmartChain: Enabling high-performance service chain partition between SmartNIC and CPU [C] //Proc of the IEEE Int Conf on Communications. Piscataway, NJ: IEEE, 2020: 1-7. doi: 10.1109/ICC40277.2020.9149136

[117] Kumar P, Dukkipati N, Lewis N, et al. PicNIC: Predictable virtualized NIC [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2019: 351-366

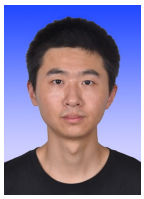
[118] Sidler D, Wang Zeke, Chiosa M, et al. StRoM: Smart remote memory [C] //Proc of the 15th European Conf on Computer Systems. New York: ACM, 2020: No.29

**Ma Xiaoxiao**, born in 1994. PhD candidate. Student member of CCF. His main research interests include high performance interconnection network and in-network computing.

**马潇潇**,1994 年生.博士研究生,CCF 学生会员.主要研究方向为高性能互联网络、在网计算.

**Yang Fan**, born in 1992. PhD, engineer. Member of CCF. His main research interests include high performance interconnection network, in-network computing and high-performance storage system.

**杨 帆**,1992 年生.博士,工程师,CCF 会员.主要研究方向为高性能互联网络、在网计算、高性能存储系统.(yangfan@ncic.ac.cn)



**Wang Zhan**, born in 1986. PhD, associate professor, master supervisor. Member of CCF. His main research interests include high performance interconnection network, distributed system architecture.

**王 展**,1986 年生.博士,副研究员,硕士生导师,CCF 会员.主要研究方向为高性能互联网络、分布式系统架构.

**Yuan Guojun**, born in 1983. PhD, associate professor, master supervisor. Member of CCF. His main research interests include optical networks and system interconnection.

**元国军**,1983 年生.博士,副研究员,硕士生导师,CCF 会员.主要研究方向为光网络、系统互联.(yuanguojun@ncic.ac.cn)

**An Xuejun**, born in 1966. PhD, senior engineer, PhD supervisor. Member of CCF. His main research interests include computer system architecture and high performance interconnection network.

**安学军**,1966 年生.博士,高级工程师,博士生导师,CCF 会员.主要研究方向为计算机系统结构、高性能互联网络.(axj@ncic.ac.cn)