计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(4): 963−977, 2025

基于可重构架构的数据中心异构加速软硬件系统级平台

王彦伟[1] 李仁刚[2,3] 徐 冉[1] 刘钧锴[1]
[1](浪潮(北京)电子信息产业有限公司 北京 100095)
[2](清华大学计算机科学与技术系 北京 100084)
[3](浪潮电子信息产业股份有限公司 济南 250101)
(wangyanwei@ieisystem.com)

Data Center Heterogeneous Acceleration Software-Hardware System-Level Platform Based on Reconfigurable Architecture

Wang Yanwei[1], Li Rengang[2,3], Xu Ran[1], and Liu Junkai[1]
[1](*Inspur* (*Beijing*) *Electronic Information Industry Co., Ltd.*, *Beijing* 100095)
[2](*Department of Computer Science and Technology*, *Tsinghua University*, *Beijing* 100084)
[3](*Inspur Electronic Information Industry Co., Ltd.*, *Jinan* 250101)

Abstract Constructing a software and hardware system-level prototype platform for accelerating data center services requires the consideration of factors such as high computing power, scalability, flexibility, and low cost. To enhance data center capabilities, research from the perspective of software-hardware synergy has been conducted on the innovation of heterogeneous computing in cloud platform architecture, hardware implementation, high-speed interconnection, and applications. A reconfigurable and combinable software-hardware acceleration prototype system is designed and built to simplify existing processor-centric system-level computing platform construction methods, enabling rapid deployment and system-level prototype validation of target software-hardware designs. To achieve these objectives, methods such as decoupled reconfigurable architecture device virtualization and remote mapping are utilized to uncover the potential of independent computing units. An ISOF (independent system of FPGA) software-hardware computing platform system is constructed to surpass the capabilities of conventional server designs, enabling low-cost and efficient expansion of computing units while allowing clients to flexibly utilize peripheral resources. To address system-level communication challenges, a communication hardware platform and interaction mechanism between computing units are designed. Additionally, to enhance the agility of the software-hardware system-level platform, ISOF provides a flexible and unified invocation interface. Finally, through the analysis and evaluation of the system-level objectives of the platform, it has been verified that the platform meets the current computing and acceleration requirements, ensuring high-speed, low-latency communication, as well as good throughput and efficient elastic scalability. In addition, improvements have been made in congestion avoidance and packet recovery mechanisms based on high-speed communication, meeting the stability requirements of communication at data center scale.

Key words heterogeneous computing; acceleration platform; prototype system; high-speed interconnect; reconfigurable architecture; FPGA

收稿日期: 2024-01-18; **修回日期**: 2024-08-12

基金项目: 新一代人工智能国家科技重大专项(2021ZD0113004); 山东省自然科学基金项目(ZR2023LZH010)
This work was supported by the National Science and Technology Major Project (2021ZD0113004) and the Shandong Provincial Natural Science Foundation (ZR2023LZH010).

通信作者: 徐冉(xuran03@ieisystem.com)

摘 要 构建数据中心加速服务的软硬件系统级原型平台,需要考虑高计算能力、扩展性、灵活性和低成本等因素.为了提高数据中心的能力,从软硬件协同的角度研究数据中心异构计算在云平台架构、硬件实现、高速互连和应用等方面的创新,研究设计并构建了一个可重构组合的软硬件加速原型系统,简化了现有以处理器为中心的系统级计算平台构建方法,实现目标软硬件设计的快速部署与系统级原型验证.针对以上目标,通过解耦的可重构架构设备虚拟化和远程映射等方法,发掘独立计算单元的潜力,构建了一套 ISOF(independent system of FPGA(field programmable gate arrays))软硬件计算平台系统,可使其超越普通服务器设计所能提供的能力,实现计算单元低成本高效扩展,使客户端可灵活使用外设资源,并且为满足系统级通信挑战,设计了一套计算单元之间的通信硬件平台和交互机制.此外,为提升软硬件系统级平台的敏捷性,ISOF 提供了灵活统一的调用接口.最后,通过对平台目标系统级的分析评估,验证了该平台在满足了当下计算与加速需求下,保证了高速、低延时的通信,以及良好的吞吐率和弹性扩容效率,另外在高速通信的基础上改进的拥塞避免和丢包恢复机制,满足了数据中心规模通信的稳定性需求.

关键词 异构计算;加速平台;原型系统;高速互连;可重构架构;FPGA

中图法分类号 TP303

DOI: 10.7544/issn1000-1239.202440041 **CSTR:** 32373.14.issn1000-1239.202440041

海量数据涌现的时代,数据中心作为承载存储、计算和数据分析等方面的基础设施,对系统能力的需求不断增长.但是随着摩尔定律的消退,传统数据中心中仅由 CPU 组成的计算架构已经不能满足需求.领域专用体系架构得到越来越多的关注,以 GPU, FPGA(field programmable gate arrays), ASIC(application specific integrated circuit)为代表的领域专用处理器虽然专注于特定应用的执行,但因其执行效果极好,其重要性日益提高.现在已经形成了以 CPU 为中心, GPU, FPGA, ASIC 等为扩展加速设备所组成的异构计算平台,平台以更高的性能共同应对现有的算力需求[1].

目前数据中心服务器节点配备越来越多的与 CPU 紧耦合的高性能异构计算外围设备[2],设备使用的附加设备模式与配套服务器构成强依赖关系,每个节点的计算单元设备数量受到服务器限制.可重构架构已经成为计算和数据密集型领域的加速器,因为它们提供了接近 ASIC 的能量和面积效率,同时保留了足够的可编程性以降低开发成本[3].在可重构架构中 FPGA 具备特有的灵活性等特征,在性能、可编程及功耗等方面也具备较好的平衡性,除了可以作为硬件量产前的验证平台,FPGA 原型作为系统级验证也具备明显优势[4].考虑到数据中心和云服务器工作负载的多样性和它们的快速变化速度等,FPGA 可以在满足严格能耗需求的情况下,在逻辑层面对计算或者通信在内的各种应用进行加速,而且能够实现高吞吐和可预测的延迟,可在加速应用保持灵活性的同时,保持较高性能,提供有效的时间到价值的转换[5-8].

因此当今人工智能时代在数字经济潮流的推动下,基于 FPGA 的研究和应用越来越多,比如基于 FPGA 开发的开源 GPGPU(general purpose GPU)项目 Vortex[9]、加速 Bing 搜索的微软 Catapult 项目[10],以及亚马逊尝试实现基于 FPGA 云加速的 AWS[11]等.无论是基于 FPGA 提供底层开源硬件的探索,还是把 FPGA 应用于数据中心各种网络或者计算等功能的加速,或者将底层可重构硬件资源从计算平台中分离出来,用以实现 FPGA 云服务的异构计算资源池化等[12],都可以通过采用 FPGA 加速的形式,把任务卸载到以 FPGA 为主的扩展硬件资源上.以上方式的好处在于:一方面,可以释放宝贵的 CPU 内核资源减少系统负载和降低延时;另外一方面,通过将基于软件处理的方式卸载到硬件进行处理,可以大大提高系统性能的同时,还可以打破服务器的固定设计降低扩展成本.

在面向通信密集及计算密集型等多并发高性能计算场景中,往往需要运行多设备协作大规模应用程序,而以 CPU 为中心的架构无法有效实现对加速设备的协同调度,而且数据需要经 CPU 进行处理,也加大了平台整体延时,这就部分抵消了加速单元所带来的效率提升.比如在金融业务的加速中,部分计算结果并不需要返回给服务器,仅需要返回给请求者,但现有以服务器为中心的模式,远端数据流转路径首先必须将数据传输到运行在其本地 CPU 上的守护进程,该守护进程将数据传递给远程 CPU,然后远程 CPU 最终将数据写入目标设备,就会存在数据传

输路径过长，无法有效满足业务低延迟的需求[13]. 设备的操作需要频繁调用复杂的内核例程，尤其高性能外围设备的密集使用会使主机端的CPU非常繁忙，而且网络数据也需要先经网卡设备到达CPU，经CPU处理后到达各加速设备[13]. 因此服务器的整体性能受到了软件组件的延迟和CPU的负载等方面的限制.

为满足数据中心动态业务处理需求，在构建基础平台架构时，对于可扩展性和灵活性等方面有至关重要的考虑. 但现有平台大多采用把FPGA作为扩展外设集成，通过PCIe(peripheral component interconnect express)将FPGA连接到CPU的单机单设备和单机多设备的绑定模式[2,13]. 这种CPU与计算加速设备强绑定的模式，无法分配独立于CPU数量运行的计算单元负载[12]，存在设备扩展和独立性差、远端设备无法灵活使用、业务平台整合移植代价高、系统通信带宽链接不足等一系列弊端[12-13]. 基于当前固定的计算架构，难以满足当前计算平台按需弹性部署及业务快速响应的市场需求，限制了数据中心的灵活扩展能力. 无法满足数据中心工作负载需要高计算能力、扩展性、灵活性、能效和低成本等方面的需求.

为了充分利用FPGA提供的各类外设接口及计算等资源，研究数据中心异构计算在平台架构、硬件实现、高速互连，以及应用和加速的可行性等方面的创新，实现了灵活独立的算力扩展系统目标，在此研究中以FPGA为可重构载体，实现了一套名为ISOF(independent system of FPGA)的异构计算平台. 该平台突破传统架构限制，通过CPU解耦的架构设计实现机器与设备的分离，使得平台可以超越普通服务器设计所能提供的能力. 并设计新型传输协议实现计算单元间的高速直连通信，在解耦与互连的基础上，构建可组合的、可重构的结构，实现计算平台对远端设备本地映射和使用. 为用户提供网络通信和计算加速服务的全栈解决方案. 其主要创新点及贡献包括5个方面：

1) 提出一种CPU解耦的体系结构，探索数据中心以数据为中心的去中心化计算架构，并验证了架构的可行性.

2) 以FPGA为载体，设计了名为ISOF的原型平台，对计算资源进行解耦和重构，突破了设备绑定的模式，可以脱离主机实体进行硬件资源扩展组合，实现了新型异构计算加速平台.

3) 通过设计高可靠的传输协议，以FPGA为实现载体，构建了可重构的高速网络传输硬件平台，实现了计算单元间稳定可靠的高速连接.

4) 通过将加速设备抽象为网络节点，使得以FPGA为原型的计算单元可以作为独立实体存在，可实现主机端与FPGA间、FPGA与FPGA间的控制指令传输，以及近、远程数据搬移，应用可通过网络共享可重构的硬件资源，实现计算单元的远程映射、支持计算资源调度等.

5) 基于所提数据中心异构加速可重构架构实现了ISOF软硬件系统平台，并通过理论分析与实验验证，证明了ISOF的可扩展性和多点应用有效性，以及其在网络和计算加速等方面的优势.

1 相关工作

数据中心计算平台在人工智能计算中发挥着重要的基础支撑作用，需要具备高效计算力、灵活性、高能效和低成本等特性，但是同时改善所有这些因素是具有挑战性的. 为了提高数据中心的能力，使其超越普通服务器设计所能提供的能力，设计新型可组合的、可重构的结构并提供加速服务，对现实需求有着重要的研究意义.

考虑数据中心服务变化多样性，数据中心硬件加速设计必须高度可管理，这意味着单一的硬件加速设计必须跨服务器使用. 关于数据中心跨硬件协同加速研究有很多，典型的就是以FPGA为基础开发了FPGA加速平台[14]，它基于传统FPGA架构利用统一的框架进行数据中心的通用卷积神经网络推理加速，实现了最佳的FPGA峰值性能和吞吐量. 文献[15]旨在通过对多样的FPGA以及CPU异构加速平台检测，确定哪些微架构结构特征会对特定的应用性能产生影响. 文献[12]提出了一种在异构云数据中心中创建网络FPGA集群的框架，FPGA集群是通过使用逻辑内核描述和FPGA映射文件创建的，该描述介绍了如何连接一组FPGA内核，但与这些内核位于哪个FPGA无关. 文献[5]是在现有云基础设施中FPGA硬件资源上独辟蹊径，以FPGA资源虚拟化的形式支持细粒度的资源共享，并以提供软件离线编译和硬件资源动态分配的方式充分利用底层资源. 文献[16]需使用延迟不敏感通信接口来启用多FPGA设备加速. 文献[17]采用将多个应用封装为单个应用，然后将其编译到单个物理FPGA上实现高吞吐. 文献[18]关注于高性能并行计算机，FPGA与CPU通过特定的扩展模式耦合，并提供机架级直接FPGA到FPGA的通信.

深度神经网络等应用对FPGA之间的通信有很

高的要求[6,19]. 百度公司利用 FPGA 加速在深度神经网络上取得了相同的性能[20]. 亚马逊公司以 FPGA 与虚拟机 CPU 紧密耦合的方式把 FPGA 作为资源, 期望 FPGA 像 GPU 一样为市场提供公有的云加速服务[11]. 一些研究架构[21-23]采用点对点(P2P)连接和外围组件互连 PCIe 链路等, 跨多个节点传输数据. 微软公司的 Catapult 提出了数据中心中用于 Web 搜索服务加速和网络加速的一种新的云架构, 利用 FPGA 快速特性和主机及其他 FPGA 交换数据, 实现 Web 搜索应用快速排名, 它同样使用了 FPGA 可重构逻辑来加速网络平面功能和应用, 并且同样在 FPGA 上添加了网络通信功能[10]. 文献 [13] 提供了全局名称和用于跨各种资源通信的公共接口, 具备底层网络自动路由流量和资源管理多路复用等特性.

与之不同于的是在 ISOF 研究中, 提供了用于设备独立性的软硬件体系, 构建了支持灵活可重构的计算服务加速与网络服务加速的软硬件平台系统, 借助独立于网络单元存在的可重构硬件单元, 消除了计算单元数量和服务器数量之间的依赖关系, 并使应用端通过本地虚拟化和远程映射方式, 在数据中心规模上获取本地服务器和远程未使用的计算单元. 另外在设备间直接通信的基础之上, 进一步添加了用于通信链路的高效与可靠的通信协议.

2 系统概述

本文研究提出了一种按需可独立扩展的计算平台架构 ISOF, 并在此基础上设计了用于高速互连的可重构硬件及通信协议, 在保证高速稳定可扩展的同时提供了统一接口支持用户的灵活操作. 本节从平台的设计原则及总体结构设计介绍系统的整体设计思路.

2.1 设计原则

构建基于异构硬件的计算平台是前所未有的开发领域, 由于现有架构和软硬件的限制及灵活可重构的独特要求, 给本项研究带来新的挑战. 为充分考虑当前需求, 研究设计应遵循原则如下:

1)**原则 1.** 平台架构应该不局限于现有硬件体系结构

为了满足在数据中心中部署和扩展各种类型的计算单元, 直接在数据中心高密度服务器上部署, 或者重新设计整个数据中心服务器集群架构都是不符合实际的.

2)**原则 2.** 需要满足低延时和高吞吐

设想未来的平台有一个新的体系层次结构, 其中本地总线扩展和远端设备扩展比本地设备具备更强的扩展能力, 比如 RDMA(remote direct memory access)和 CXL(compute express link)[24]. 由于 ISOF 是基于总线和网络的, 它的合理性能目标首先是匹配先进的总线和网络速度, 然后能在新的体系层次结构中进一步降低所需的通信延迟.

3)**原则 3.** 需要满足低成本

构建异构平台的主要目标和好处是降低成本. 一个好的 ISOF 平台应该有较低的总体资本支出和运营成本. 因此, 这样的平台不应该使用昂贵的硬件来构建, 增加成本超过节省的成本, 更好的方案是依赖于现有的硬件体系构建.

4)**原则 4.** 需要高效利用

为了保持集群内设备数量和总成本较低, 每个客户应该托管多个设备, 这些设备将被充分利用. 为了最有效地使用设备, 应该允许应用程序在 ISOF 上创建和访问任意大小的计算单元区域.

5)**原则 5.** 需要满足灵活可重构

随着数据中心应用、硬件和网络的快速发展, 一个可持续的 ISOF 解决方案应该是灵活的和可扩展的, 例如支持高级 API, 以减轻一些到设备的应用逻辑.

2.2 系统结构

ISOF 新型异构计算加速平台的主要组成结构如图 1 所示, 主要包括了代表主机端的服务器、代表外围加速计算单元集中服务设备的 FPGA box 和远端请求应用等. FPGA box 可以通过网络接口接入到数据中心网络或者其他应用网络中, 在网络接入的模式下, PCIe 接口除应用于计算单元间的直接通信, 也提供应用于设备健康状态管理的硬件通路. 整个平台中代表计算单元的 FPGA 等异构计算资源可通过高速网

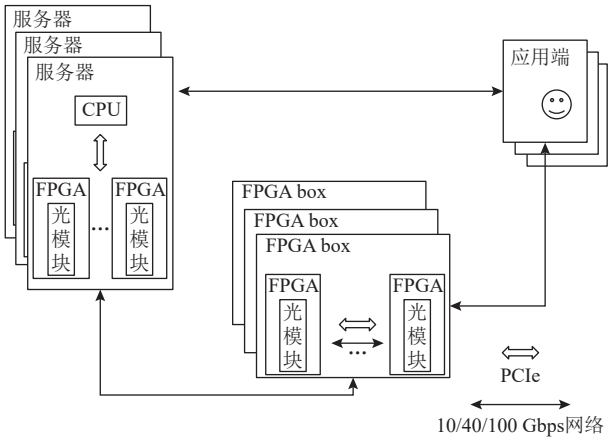


Fig. 1 Illustration of ISOF architecture

图 1 ISOF 架构示意图

络进行互连. 从平台管理的角度, FPGA 设备作为独立的计算资源节点组成计算子网, 整个子网内由多个计算节点组成, 可以按用户需求完成逻辑子网划分, 节点之间支持自定义的高可靠轻量级传输协议, 实现服务器节点与 FPGA 加速计算单元之间以及 FPGA 加速计算单元之间的数据传输和控制指令的通信等.

为了减轻 CPU 的负担, 充分提高平台整体性能, ISOF 平台架构采用了与 CPU 解耦的实现方式. ISOF 通过解耦的方式, 将待加速的网络数据, 直接通过数据中心网络传输到 FPGA 加速设备上, 数据无需经过 CPU 转发. 此设计在一定程度上降低了平台延迟和 CPU 负载, 进而可将 CPU 用于其他更适合的计算场景, 比如网页或者流媒体服务等更需要复杂逻辑处理的应用. 另外, 服务器端可以采用相对低端的 CPU 维护和管理进一步降低系统成本, 具体设计细节将在之后章节中介绍. 此外应用于服务器或者客户端的 vFPGA(virtual FPGA)设备驱动, 实现了完全兼容 PCIe 驱动功能. 上层应用程序支持包括 OpenCL 和 RTL(register transfer level)两种模式, 系统可以通过原有的 API 接口与 FPGA 加速设备进行数据交互, 也可采用远程映射的方式, 将分配的 FPGA 加速器映射到本地, 应用程序可以通过设备节点发现设备, 实现本地设备无差别访问.

ISOF 系统具备多种适用方式, 不同的方式应用于不同的场景之中, 主要包含通过 PCIe 的本地计算方式, 把解耦设备作为独立设备使用, 以及把远程加速设备通过远程映射到本地作为本地扩展设备全局应用. 同样借助于 ISOF 系统, 用户也可以通过云端模式在本地编译和执行远程计算流程. 图 2 表示根据应用需求划分的 3 种不同的应用场景. 图 2(a)表示传统场景, 应用端请求流经过服务器主机侧通过 PCIe 数据流与计算单元交互; 图 2(b)表示脱离主机模式的远程映射应用场景, 应用数据流可以绕开主机侧, 直接与计算单元交互, 实现低成本、低延时扩展计算单元, 提高大规模应用卸载的潜力优势; 图 2(c)表示传统与解耦设备协同共存的应用场景, 可以实现传统模式与可重构资源的组合优势.

3 实现平台设计

本节介绍系统总体的实现方案设计, 首先概述了 ISOF 系统平台整体组成设计, 随后详细介绍各组成部分包括底层协议和硬件等实现方案的细节设计, 最后介绍了应用开发设计部分.

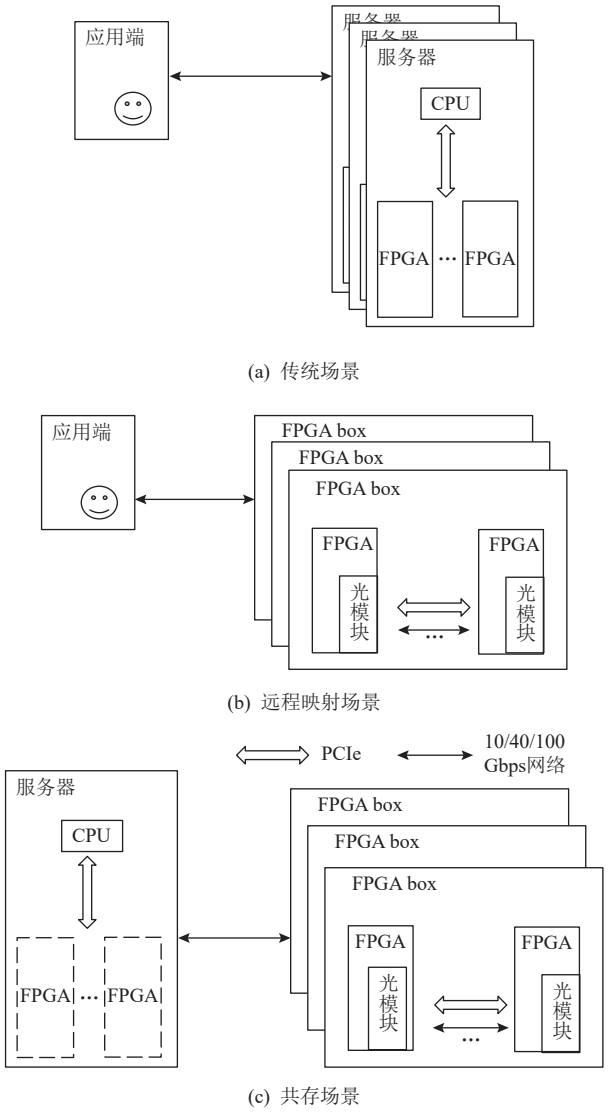


Fig. 2 ISOF usage scenarios

图 2 ISOF 使用场景

3.1 组成设计

ISOF 平台主要实现模块, 包括服务器节点上运行的逻辑控制模块、FPGA 上运行的加速计算模块等. 服务器节点的逻辑控制模块, 利用可扩展的编程框架提供的 API 接口实现, FPGA 上利用 RTL 实现的加速计算模块, 以及利用编程框架提供的 IP 层控制接口完成并行计算任务. 依据主机侧和设备侧视角, 模块对应细节将在之后不同部分介绍, 其核心设计包括如下:

1)设备管理模块, 用于获取每个 FPGA 加速设备管理 ID. 通过识别每个 FPGA 加速设备的 IP 地址信息分配本地的 ID, 并依据本地 ID 进行设备节点管理; 通过 LINUX 内核驱动的字符设备 API 创建设备节点, 分配设备号标识远端 FPGA 加速设备来管理 FPGA

计算映射资源.

2)设备模拟模块,用于模拟 PCIe 设备相关接口. ISOF 平台将所有 PCIe 的对 BAR(base address register)地址的操作,全部进行网络数据包的格式化转换,实现寄存器操作、DMA(direct memory access)数据流操作、中断操作迁移到网络接口,上层 OpenCL 或者 RTL 应用无需知道 FPGA 加速设备是否通过网络映射,为上层应用提供无感知的操作.

3)映射模块,用于建立服务器和各个 FPGA 加速设备之间映射关系. 其过程是首先通过创建管理 ID、加速设备 IP 地址、虚拟 ID 三者之间的路由匹配表关系,并且将这些信息进行封装,之后分发给所有 FPGA 加速资源,最后完成 FPGA 加速设备之间的关系映射.

平台的互连模块扩展升级互连机制,用以提高远程映射性能,通过采用高速互连机制将交互数据抽象成配置指令和数据搬移 2 类. ISOF 平台基于 Intel 的 OpenCL BSP(board support package),在 FPGA 异构加速器内增加网络接口的访问通路,将通过 PCIe 接口完成的配置和数据传输通过网络接口实现,FPGA 收到指令帧后,按照帧内容配置寄存器,完成各项功能配置,FPGA 收到数据帧后,按照帧内容存储数据完成应用任务. 该互连协议不限定于底层的传输协议,其可基于 UDP 协议或者直接基于以太网 MAC.

3.2 设备可扩展层

为了让主机定位和识别设备,另外为了提高各种异构计算单元互连扩展能力,需要一种基于网络协议的识别查询和互连方法. ISOF 设计中包含了节点管理模块和协议扩展模块. 节点管理模块的主要功能就是识别查询扩展的异构计算设备,包括设备类型、设备地址、设备 ID 管理、物理设备与虚拟设备的映射关系维护等,设计网络互连协议将在此后的章节中介绍. 主机可以通过查询节点管理模块以获得允许与其通信的实体列表,并返回所有设备所需具体配置. 以上设计方式满足了目标设计中的原则 4, 5.

设备可扩层的执行流程可以总结为:云端 FPGA 加速设备上电后默认向服务器发送 DHCP(dynamic host configuration protocol)服务请求,服务器获取请求后为每个 FPGA 加速设备分配 IP 地址,并将 FPGA 加速设备的相关信息保存到服务器的配置文件当中,主机通过查询服务器配置文件获取相关设备列表、IP 等信息,并对每个设备分配虚拟 ID 以建立映射表. 节点模块收集上述所有信息后,依据虚拟 ID 号在设备下创建具体的设备节点.

3.3 系统虚拟层

为了让主机上层的 OpenCL 应用可以无感知地操作远端的 FPGA 加速设备,需要在驱动层实现虚拟化的 PCIe 的 BAR 地址操作到数据包操作的转换,这种转换可以实现 OpenCL 的 MMD(memory mapped device)层的所有寄存器操作、DMA 数据流操作、中断数据等的接收全部兼容,上层 OpenCL 完全不清楚底层具体的硬件细节. 驱动层模拟块设计的主要功能包含了 PCIe 总线数据兼容、单寄存器读写操作、多寄存器连续读写操作、DMA 数据传输和中断数据处理等. 此种设计方式,满足了目标设计中的原则 1, 2, 3.

系统虚拟层的实现细节流程可以总结为:MMD 层调用和 PCIe 总线相关的参数,模块返回 FPGA 加速设备的 PCIe 总线数据. 对于寄存器或者 DMA 的写操作,模块首先会获取主机当前环境与 FPGA 加速设备相对应的以太网设备;然后依据节点管理模块收集的所有 FPGA 加速设备的配置信息,按照之后的高速传输协议进行 UDP 的组包操作;之后通过以太网设备将数据发送给远端的 FPGA 加速设备完成写操作. 对于读操作模块检测以太网接收数据包,按照高速传输协议进行 UDP 的解包操作完成数据接收. 对于中断的处理,FPGA 加速设备会主动发送一个特征包,当模块解析特征包后,会通过一个信号机制传输给上层的 MMD.

3.4 高速传输协议

人工智能应用对通信有着多种需求,比如基于文本的模型相比 CNN 更占用带宽[6],而且面对不同需求的多样化应用延迟和带宽,流量负载不均衡等也会导致链路拥塞和丢包[25]. 为了实现节点间的高速互连,也为了给应用提供充分和稳定的带宽基础,ISOF 平台设计了应用于硬件卸载平台的轻量级链路传输协议 iDMA 和改进的 RDMA 协议 iRDMA,通过采用网络接口的方式与数据中心网络互连,此种设计方式满足了目标设计中的原则 1, 2, 3. 整个协议分为:1)轻量级的高速链路无状态传输协议 iDMA,其主要基于 MAC 层的数据传输和 IP 层的远程配置与部署,其数据报文格式如图 3 所示,主要实现 box 服务器和机柜级间计算单元的数据传输与配置功能,另外实现设备单元的远程映射功能;2)基于 UDP 的节点间高速稳定传输协议 iRDMA,其实现节点间的数据高速传输,以及丢包重传和流控机制.

按照业务流程 iRDMA 协议帧分为指令帧和数据帧,其数据报文格式如图 4 所示,部分字段含义如表 1 所示. FPGA 设备收到指令帧后直接配置寄存器,

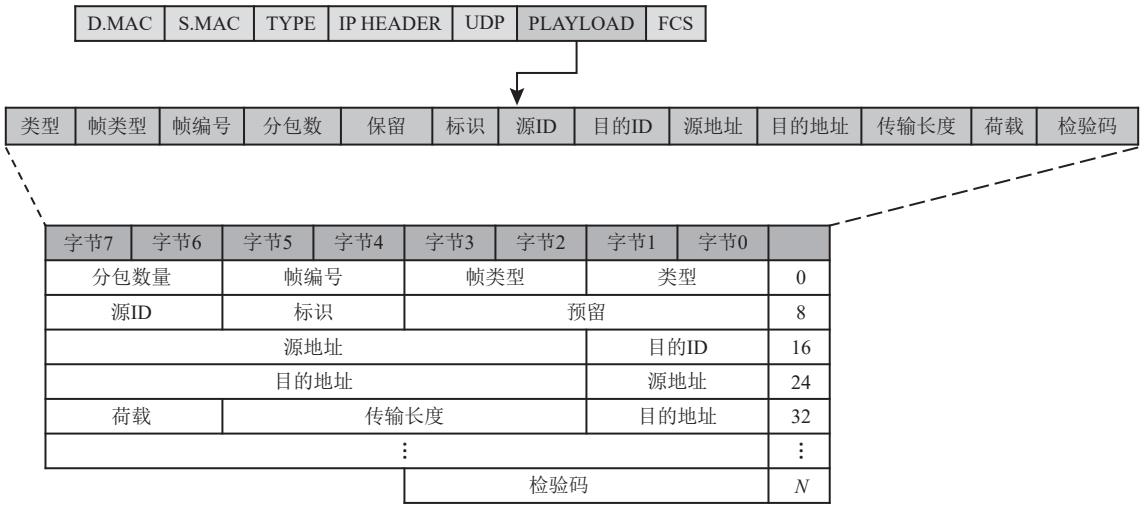


Fig. 3 iDMA protocol frame

图 3 iDMA 协议帧

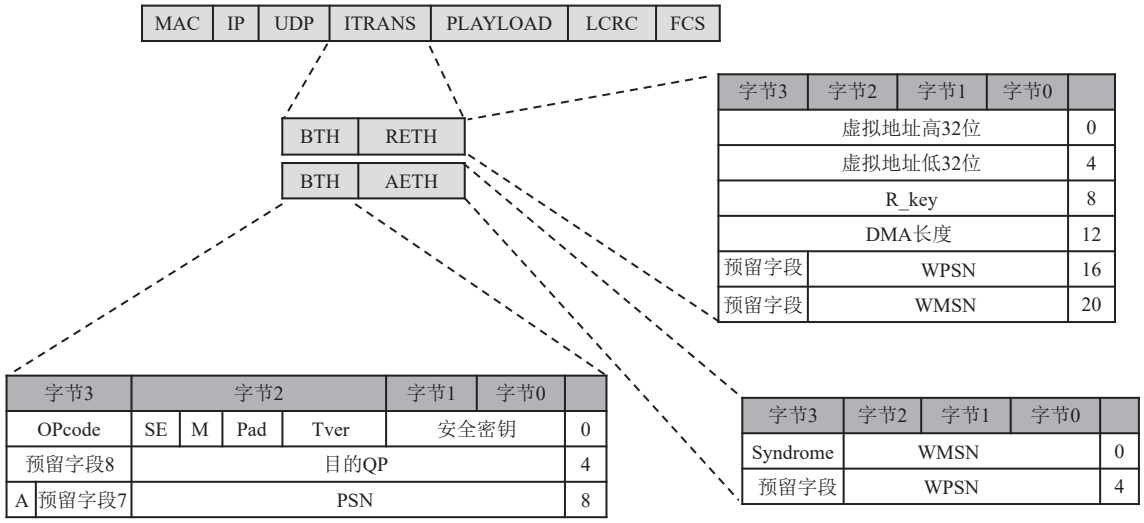


Fig. 4 iRDMA protocol frame

图 4 iRDMA 协议帧

Table 1 iRDMA Protocol Fields

表 1 iRDMA 协议字段

字段	字段含义
SE	重发请求指示
M	路径迁移指示
Pad	补充字节数，载荷 4B 对齐发送
TVer	BTH 的版本信息，默认为 0
A	需要返回应答包指示
R_key	与存储的地址空间对应
WPSN	写（读）操作包序列号
WMSN	写（读）消息序列号
Syndrome	操作成功指示

完成各项功能配置. 在数据帧的传输过程中，按照帧的数据字段，由数据传输发起方进行组包和分片，设备收到数据帧后，按帧内容传输、存储数据. 设备在收到协议包后，默认情况下是按照回复帧进行回复，但若帧格式里有控制字段，可进行配置控制，选择是否对协议包进行回复.

iRDMA 协议包含基本传输头部（base transport header, BTH）和读写相关的扩展传输头部（RDMA extended transport header, RETH），以及用于 ACK（acknowledge character）包的附加传输字段扩展传输标头（ACK extended transport header, AETH）. 为了兼容现有网络及成本等原因，在基于以太网的数据中心中，RoCE（RDMA over converged Ethernet）成为基于以太网数据中心中部署 RDMA 的规范方法. RoCE 需要一个无损网络，对于丢包十分敏感，并要求网络是不丢包的才能获得良好的性能. 为了获得更好的性能，需

要通过在网络中启用优先级流控制 PFC(priority flow control)来防止丢包发生[26]. 但 PFC 带来了许多问题,问题的范围从轻微(例如不公平和队头阻塞)到严重的拥塞扩散,甚至是网络死锁[26]. 由于 RDMA 零拷贝设计和网卡内部资源限制,丢包重传在现有商用网卡上很难实现,为了更符合实际网络环境应用,iRDMA 基于自研的可重构硬件平台设计,采用选择性丢包重传恢复机制,接收方不丢弃无序数据包,发送方选择性地重传丢失的数据包,另外 iRDMA 协议的流控采用了限制端到端飞行中数据包数量的机制来防止拥塞的发生.

3.5 可重构硬件网络加速平台

本文研究提出一种基于 FPGA 的可重构硬件网络加速平台,硬件架构及其系统配套程序如图 5(a)(b)所示,用以实现设备间通信的高性能、灵活性和适用性. 作为承载 ISOF 系统所有的控制和数据传输之间硬件级的高性能设备,把控制层和数据层从 CPU 端解绑. FPGA 网络加速栈由静态区和用户应用加速引擎组成. 静态部分包括控制面及数据面 2 部分,控制面主要是执行主机或者用户侧下发的控制命令,上报 FPGA 本地的信息,完成与主机的交互处理,以及配合主机完成路由、流量控制、授权等管理以及资源复用等功能. 数据面主要是各类物理接口包括 10 Gbps 和 100 Gbps MAC 以及 PCIe 通道和 iRDMA 协议支持等. FPGA 加速引擎可通过 PCIe 接口传输给加速单元进行本地加速,也可以通过经 PCIe 到 MAC 接口输出给其他设备卡进行远程加速;动态可重配置加速单元通过 PR(partial reconfiguration)技术实现远程动态重配置;路由转发模块的转发路由表软件可配置,FPGA 内部通过查找路由转发表实现路由选择. 此种设计方式,满足了目标设计中的原则 1, 2, 3, 5.

基于现有软硬件创新的独立可重构硬件平台其好处在于:1)拥有硬件级别优化控制和数据路径,并最大限度地减少软件干预,这样可以显著降低设备间通信的延迟,并减少主机端 CPU 的利用率. 通过在硬件层面实现控制和数据传输的优化,可以提高系统的性能和效率. 2)支持使用标准设备控制器,这意味着系统可以使用标准的设备控制器来实现独立的、可分解的设备编排. 这样可以更经济高效且灵活地进行设备间通信,而且可以与广泛的现成设备进行兼容. 3)支持可重构的近数据处理单元,在系统中增加一个可重构的数据处理单元用于直接通信. 这样可以为现有的服务器应用程序提供更多的机会,使

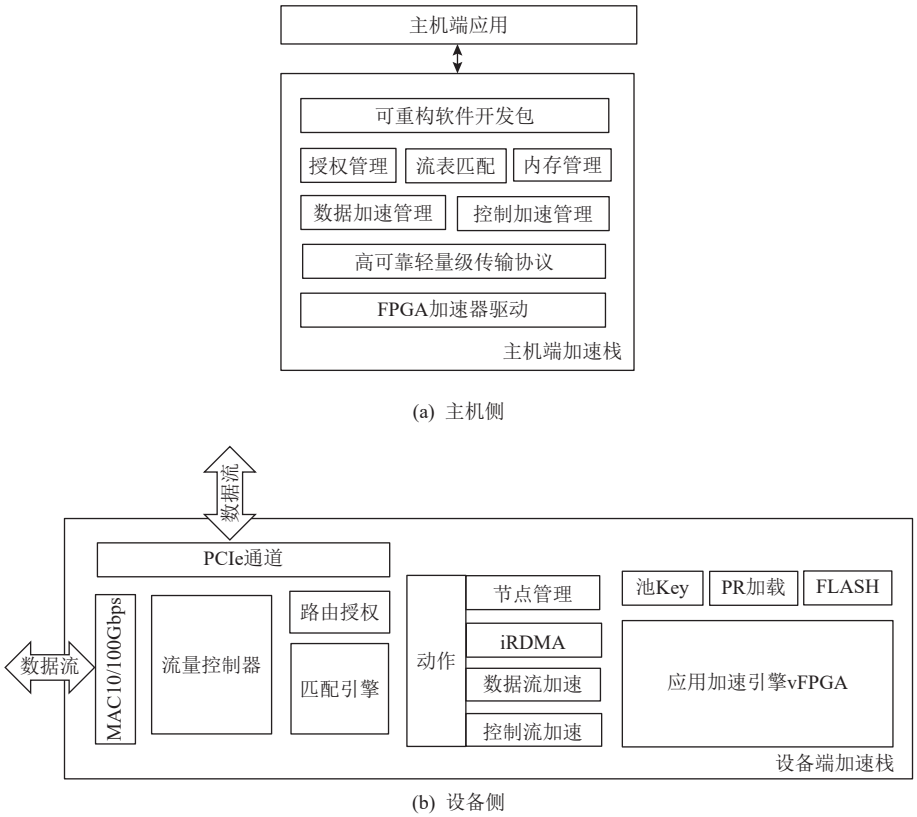


Fig. 5 Schematic diagram of hardware and system architecture

图 5 硬件与系统架构示意图

其能够更有效地进行直接通信，而无需经过主机端的干预.

3.6 应用端接口

考虑到系统级原型验证涉及到很多软硬件层次代码，ISOF 系统将全栈的代码仓库组织起来，基于以上软硬件基础构建支持用户使用的云平台. 构建配套应用于异构并行计算平台的加速编程接口，屏蔽平台底层的技术复杂性，提供统一的加速平台模型和接口，使客户端在本地或服务器端不修改用户程序的情形下统一调用，自动执行编译和测试相关程序. 接口设计满足了目标设计中的原则 4，5. 为了支持用户底层无差异化的接口调用，平台实现的函数如表 2~4 所示. 基于总体及接口等设计，该体系结构具有以下属性：1）具有统一的命名方案，可以处理和访问所有资源，而不考虑其位置；2）无论传输彼端是什么，以及端点驻留在哪里，配置与通信都有一个全局统一的传输接口；3）提供路由和资源复用的底层网络服务. 通过这样的体系结构，FPGA 用户应用程序可以使用具有资源全局名称的通用编程接口，轻松访问数据中心内的各种资源集，支持计算资源调度，支持计算任务到异构计算设备的映射和云平台的构建，用户可以在云平台上灵活获取计算资源、存储资源和应用等服务.

Table 2 Initialize Interface

表 2 初始化接口

初始化 API	功能
Icloud_init	实现 FPGA 云编程接口，初始化 FPGA 云平台软件内部资源
Icloud_close	云编程接口关闭，释放 FPGA 云平台软件内部资源
icfReadReg_n	读取 FPGA 多个寄存器内容
icWriteReg_n	向 FPGA 寄存器填写内容

Table 3 Configuration Interface

表 3 配置接口

配置 kernel API	功能
ifcConfigKernel	完成将每个输入数据的加 1 操作后将数据放到定义参数 *result addr* 中
ifeStartKernel	启动对应设备 ID 的 FPGA 的 kernel 功能
icfStartLoadPR	通过配置相关寄存器，进行 PR（partial reconfiguration）加载
icfFlashPcieMem	烧写指定的格的文件到 FPGA 设备
icfLoadApplicationImage	加载 FPGA applicaton 文件

Table 4 Data Interface

表 4 数据接口

数据传输 API	功能
ifcHostAddrData	实现 FPGA 云编程接口，初始化 FPGA 云平台软件内部资源
ifcFpgaSwFpgaDatauint8	通过 RDMA 网络接口在 2 块 FPGA 设备之间传输数据
ifcConfiglkl	将 FPGA 的 kerel 计算加 1 的数据传输到目的 FPGA 的数据存放地

3.7 开发与映射框架

ISOF 平台将有关的软硬件以高内聚、低耦合的原则组织成一个集成的环境. 用户仅需要向平台的接口文件提供关键配置参数，支持从软硬件上开发和设备映射的全系统敏捷过程，以上设计方式满足了目标设计中的原则 4，5.

3.7.1 ISOF 应用实现流程

ISOF 用户应用的实现包括 RTL 和 OpenCL 两种方式，其应用流程如图 6 所示，对于 RTL 版本开发，其执行流程为：①应用调用 DHCP scan 接口获取 FPGA 设备 IP 信息. ②调用 iRDMA 协议接口发送数据. ③通过以太网将数据发送到远端 FPGA 设备.

对于 OpenCL 版本开发，其执行流程为：

④应用软件调用 OpenCL runtime 函数接口. ⑤接口向下调用 HAL（hardware abstraction layer）库. ⑥库函数调用 MMD 接口函数. ⑦MMD 调用 Virtual kernel 提供驱动接口. ⑧驱动调用 Virtual user 用户接口函数. ⑨之后调用 iRDMA 协议接口发送数据. ⑩通过以太网将数据发送到远端 FPGA 设备.

3.7.2 ISOF 设备匹配与远程映射机制

ISOF 的核心功能包括 FPGA 的独立性和对 FPGA 的远程控制能力. 实现以上功能，首先可以解耦 FPGA 与 CPU 的位置耦合，同时实现 FPGA 成为独立的设备；然后在数据中心中闲置的 FPGA 设备，通过将加速设备抽象为网络节点，就可以通过远程控制使用的方式实现不同服务器间的资源共享. 因而远程控制也是数据中心中包括 FPGA 等各类异构硬件实现资源池及构建云级平台的基础之一. 特别在一些边缘计算场景中[27-28]，借助于 FPGA 远程映射，可有更好的远程部署、重配置计算和服务聚合能力.

ISOF 中远程映射平台结合软硬件协同设计，在基于可靠传输网络下可将分散式 FPGA 的计算资源按需映射到服务器中，另外增加了匹配引擎用于设备端所接受封装的数据的匹配. 通过以上方式可以完成主机和 FPGA 加速设备以及 FPGA 加速设备和 FPGA 加速设备之间的通信. 整体流程逻辑包括：首先远端服务器根据所需控制或者所分配的 FPGA 资源，创建本地虚拟驱动；然后待加速应用可以在不更改实现逻辑的情况下利用原有平台接口与虚拟驱动进行数据交互，虚拟驱动根据配置将参数与运行数据转换为网络协议，利用可靠网络将数据传输到

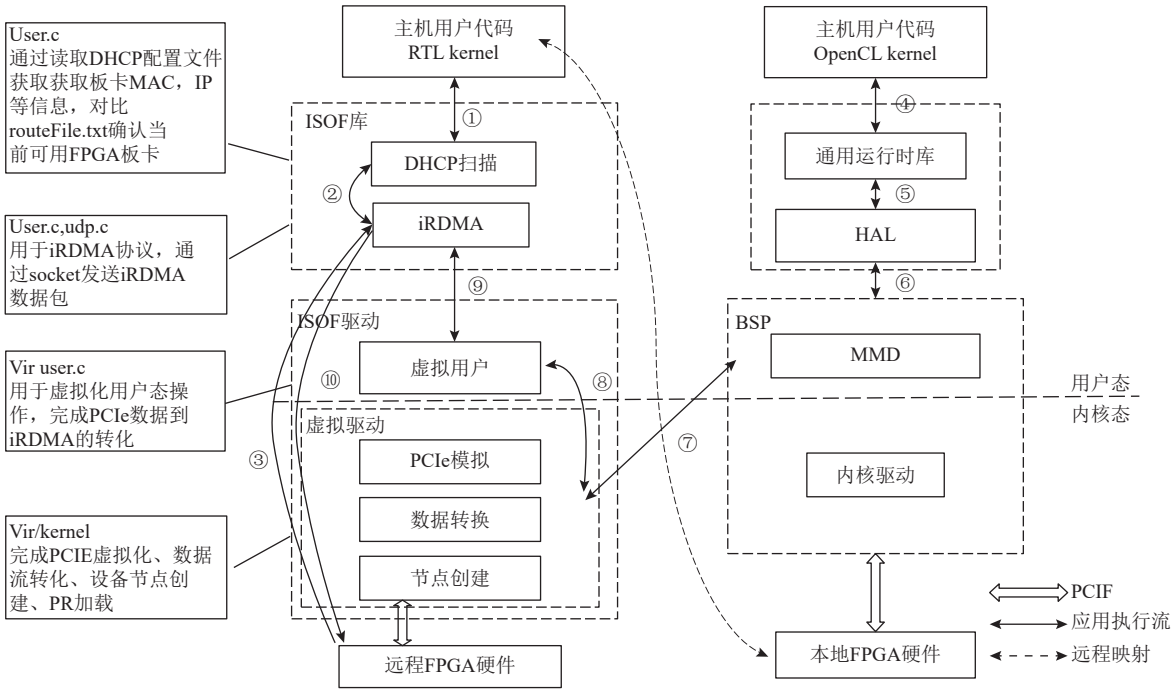


Fig. 6 Flow chart of application

图 6 应用流程图

FPGA 加速设备；最后由 FPGA 加速设备通过对互联协议的解析完成对 FPGA 的寄存器配置及数据搬移，从而实现 FPGA 上的应用逻辑部署，最终处理完成的数据通过网络返回到主机请求端. 上述方式将依赖于 PCIe 的控制流以及数据流迁移到了网络上，使得 FPGA 加速器作为独立计算单元，可在无本地 CPU 的参与下，映射 FPGA 运算单元到远端服务器中. 而没有加速设备的客户端也可调用远加速设备完成 AI 等运算，实现 CPU 与设备的解耦.

4 验证平台的分析与评估

本节从理论分析和实验分析对 ISOF 系统的扩展性和通信交互性能等方面进行验证. 首先理论分析平台在扩展性和通信控制方面的优势，然后通过几组实验验证总体方案的可行性和性能.

4.1 理论分析

假设将计算处理过程简化为无损模式的 M/M/S 模型[29]，分析 ISOF 扩展模式的系统收益. 为了简化推导过程，用 1 个节点代表 1 个计算事务的服务平台，$\rho=\lambda/S\mu$，其中：λ为客户端请求到达率；μ为单位时间的平均服务速率；S 为服务平台数；ρ平均每个服务平台的服务强度. ρ值不得大于 1，否则计算节点会发生阻塞，计算队列会无限制排队下去，系统无法进入稳态[30].

平均队长L_S为系统内客户端请求的数学期望，包括正被服务的顾客和排队等待服务的客户端，其计算方式为

$$L_S=\sum_{j=0}^{\infty}jp_j=S\rho+\rho/(1-\rho)^2p_S, \tag{1}$$

其中p_j表示系统处于稳态时有j个顾客的概率.

z_c为系统成本，其计算方式为

$$z_c=c_S\mu+c_wL_S=c_S\mu+c_w\lambda/(\mu-\lambda), \tag{2}$$

其中c_w为每个客户端请求在系统中逗留单位时间的固定成本. c_S为服务 1 个顾客时单位时间内的服务成本.

为简化推导过程，假设以相同的计算单元扩展，客户端请求到达率λ和平台服务速率μ不受系统中现有服务数量的限制，则可在此引入增益因子$\theta(0<\theta<1)$，其代表 CPU 解耦方式扩展计算节点相比扩展服务器扩展模式所带来的低成本和减少 CPU 参与带来的收益效果，则$c_S=\theta c_S$为 ISOF 系统服务 1 个顾客时单位时间内的服务成本，再带入总费用可得：在同样算力单元最优服务率下的 ISOF 系统，z_c小于原系统z_c.

假设将计算处理过程简化为 M/M/1/k 损失模型，分析系统通信收益，k为通信系统容量，则系统收益为

$$z_r=\lambda/(1-p_k)E-c_S\mu, \tag{3}$$

其中E为给每个客户端的固定收益，$p_k(0<p_k<1)$为因系统容量k导致的损失概率，因此在丢包后续重传模式下系统收益z_r会减小. 引入$\eta(\eta>0)$表示非流控模式下所带来的通信节点服务强度增大程度，则$\lambda=$

$\lambda(1+\eta)$, 这就使得ρ更容易进入大于 1 的非稳态状态, 队列就会无限变长导致系统可靠性大幅度下降.

4.2 验证平台评估

承载 ISOF 系统的硬件平台为双路 Intel Xeon CPU, E5-2640V4-Xeon2.4, 内存大小为 256 GB. 可重构设备硬件平台为本团队提供 IEIT F10A 和 F20A, 以及 Intel Rush Creek PAC FPGA 可重构硬件设备, 网络设备采用了 Mellanox ConnectX-4 Lx 10 Gbps 和 Mellanox ConnectX-5, 此外开发了 Linux 内核模块和用户级库等. 本节将 ISOF 应用于现有的设备协作下的设备间通信、设备与主机间通信, 以及基于此平台的应用验证等.

ISOF 部分硬件实物测试环境及部分实验效果如图 7 所示. 首先, 为了验证平台的可行性及扩展适配效果, 我们基于现有的一些开源模型做了一组实验. 如图 7 中所示, 通过运行 ResNet50 模型图像识别与分类, 最终结果证明了平台在单引擎和独立多引擎互连运行的可行性, 单点 FPGA box 池化服务器原型达到了最大 32 个单元的扩展. 此外通过 SqueezeNet 模型拆分的图像识别, 验证在 IOSF 平台上多引擎独立计算单元互连通信协作的可行性. 另外通过 TensorFlow 推理 ResNet50 模型得到节点间的吞吐达到 92%, 节点间通信延时为 1.36 μs, 计算单元的弹性扩容时间约为 6.94 s.

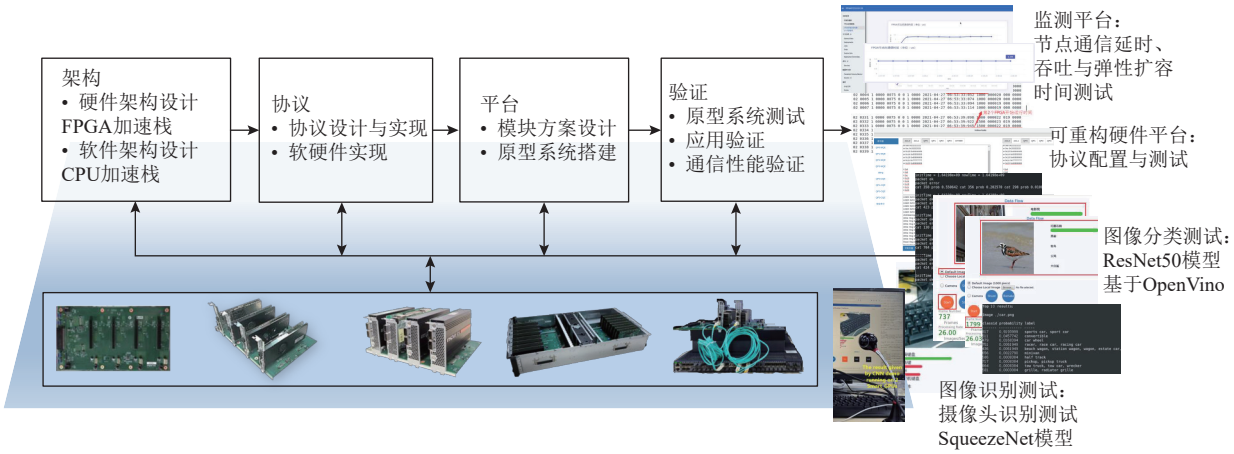


Fig. 7 Hardware test environment and part of the experimental effect diagram

图 7 硬件测试环境及部分实验效果图

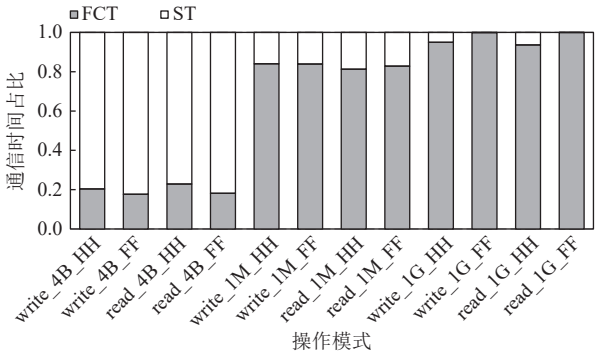
流完成时间(flow completion time, FCT)是衡量数据中心网络传输性能的重要指标之一[31], 表 5 是

Table 5 Time for Flow Completion

表 5 流完成时间 μs

测试内容	FCT
write_4B	10
read_4B	4
write_1KB	10
read_1KB	4
write_1MB	103
read_1MB	94
write_10MB	939
read_10MB	905
write_100MB	9 126
read_100MB	8 962
write_1GB	91 121
read_1GB	89 198

10 Gbps 版本下的流完成时间. 为了证明 IOSF 的通信效果, 本文首先在 10 Gbps 的版本上测试了不同 FPGA 数据流和软件各自占用的时间, 从图 8 中可以看出当数据量小时, 数据经过设备处理, 从设备穿透到主机的过程占用时间较高; 当数据量足够大时, 设备到主机的过程时间占比较小, 同时也可以看出即使是大数据量读写, 数据穿透主机的过程仍占据着较大的延时 ST(software time). 这个原因在于控制通路发送端和接收端创建通信所需各种资源与软件数据包解析所产生的处理时间有所关联. 但是通过对比大量数据在主机侧读写及数据直接在设备端交互, 最终结果仍可以看出 FPGA 设备独立进行数据交互的方式, ST 等方面所带来的延时影响几乎可以忽略不计. 以上可以证明采用 CPU 解耦架构实现的机器与设备分离方式, 在硬件级别达到优化控制和数据路径, 并最大限度地减少软件干预, 可以显著降低设备间通信的延迟和主机端的资源占用, 也侧面证明了当前计算平台采用独立架构扩展模式提高的必要性.



注：HH 表示主机侧与设备侧通信的数据经过 FPGA 设备处理后在主机端内存交互，FF 表示通过设备卡直接通信数据在 FPGA 内存中进行交互.

Fig. 8 Communication time proportion test

图 8 通信时间占比测试

为了测试系统实际带宽效果，图 9 所示的结果为 100 Gbps 版本在 0 丢包的环境下的 FCT 测试，可以看出平台实现了良好带宽效果，最高带宽达到了 90 Gbps 以上.

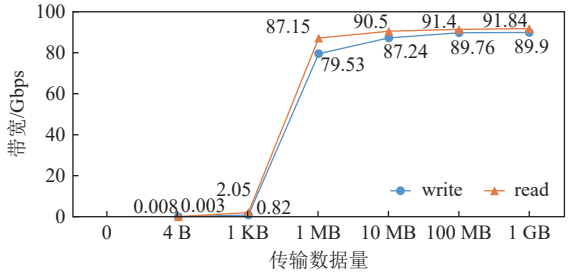


Fig. 9 Bandwidth test

图 9 带宽测试

为了根据实际情况测试网络的稳定性，在图 10 中增加不同丢包策略测试，采用了不同的丢包间隔测试网络与协议性能，图 10 代表不同丢包间隔下的流完成时间测试结果(1 000_1 代表每 1 000 个包丢 1 个，以此类推). 可以看出，基于不同的丢包间隔，丢包与不丢包的最终流传输完成时间基本无差异. 但是业界的公认以及本文研究的测试证明，同样测试情况下 1% 的丢包率会使得包括 Mellanox Connect-5 在内的 RDMA 网卡吞吐率下降为 0，这主要原因可能是由于 PFC 导致的重排机制和多流排队积压引起网络拥塞. 最终测试经验得出，当丢包率低于 2% 时，基于本文研究的网络加速硬件平台和网络协议能正常完成应用数据传输. 图 11 中所示是采用 10 Gbps 版本的 ISOF 的可重构硬件网络加速设备和 10 Gbps 的 Mellanox 网卡验证对比. 通过多流实验对比结果可以看出，不论是 1 MB 流还是 1 GB 流在不丢包时，传输性能基本接近，但是无论在读还是写的测试中采用 100 丢 1 和 10 丢 1 的丢包策略，本文研究的网络传输软硬件协议栈有明显较好的稳定性能. 另外图 12 中基于 perftest 的延迟测试结果显示，在正常通信情况下本文研究平台通信性能与 Mellanox 基本保持一致，但在现有服务器应用程序中增加可重构的近数据加速处理单元可以为通信提供更多的定制方式.

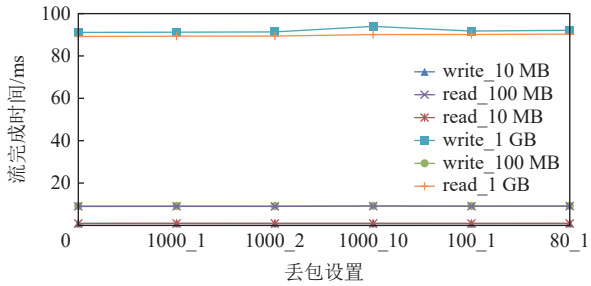
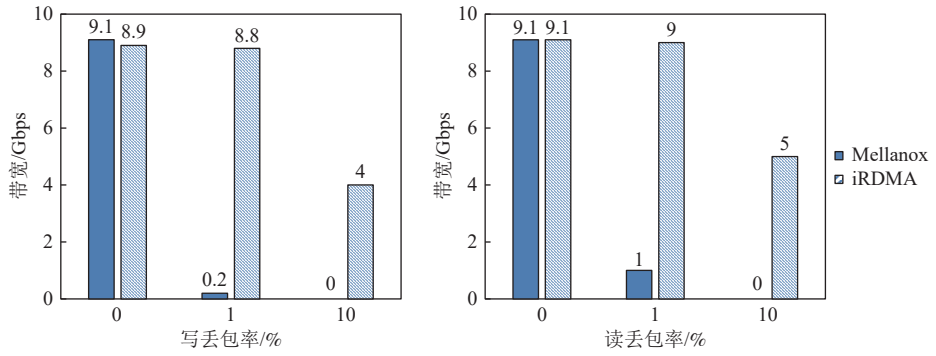


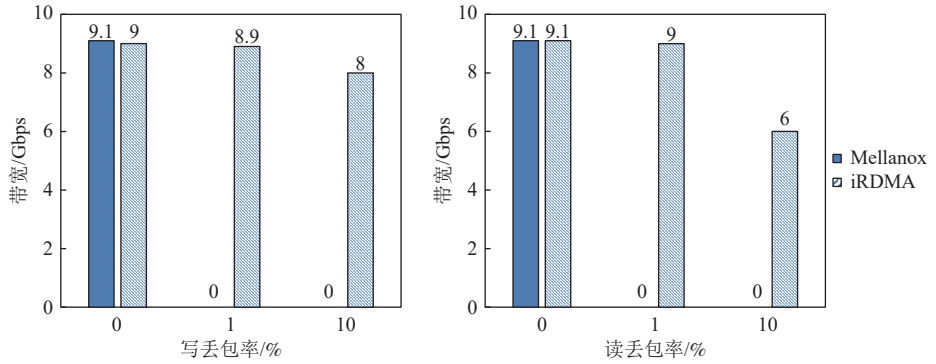
Fig. 10 Packet loss test

图 10 丢包测试

将 FPGA 等计算资源采用网络增量部署，一方面有利于系统扩容，为云级池化奠定技术基础，另外一方面可以加速数据中心的各类应用，比如网络功能和存储功能等. 为了验证 ISOF 原型系统应用加速效果，并探索通信方式改变带来的可能影响，在此实验中对通信加解密应用加速及通信效果对比验证. 因服务器通常与大量的客户端建立加密通信，需要经常进行非对称加密的计算，其中基于椭圆曲线算法的密钥协商协议称为 ECDH(elliptic curve Diffie-Hellman). 椭圆曲线加密算法的计算成本较高，通常由 CPU 来计算. 在本文中使用 X25519 来作为椭圆曲线加密算法的标准曲线，硬件配置及连接方式等如表 6 以及图 13 所示. 在此项实验中，FPGA 加速设备部署了 64 个 X25519 椭圆曲线乘法加速器 IP Core，分别使用 PCIe 和光纤网络与服务器进行连接. 使用设备分别与 8 192 个客户端进行 ECDH 过程，服务器端计算椭圆曲线乘法所花费的总时间. 该时间包含下载原始数据至 FPGA、FPGA 加速设备进行椭圆曲线乘法计算、从 FPGA 设备中读取结果数据 3 个过程. 最终结果如图 14 表示，比起纯 CPU 计算方式，使用 FPGA 加速设备在通过 PCIe 和光纤网络连接服务器时，分别能提供约 124 倍和 47 倍的椭圆曲线乘法加速效果. 对比服务器节点内 FPGA 计算方式，ISOF 光纤通信方式扩大了通信域，并且为应用大规模并行计算与通信提供支持. 最终实验结果表明，即使比起设备卡 PCIe 的带宽、交换机带宽成为瓶颈，以及链路延时加持的情况下，ISOF 面向云的方式，仍能在大量用户请求的情况下，以不超过 1 s 的总体加解密处理时



(a) 1 MB小包读写测试



(b) 1 GB大包读写测试

Fig. 11 Device transmission stability comparison test

图 11 设备传输稳定性对比测试

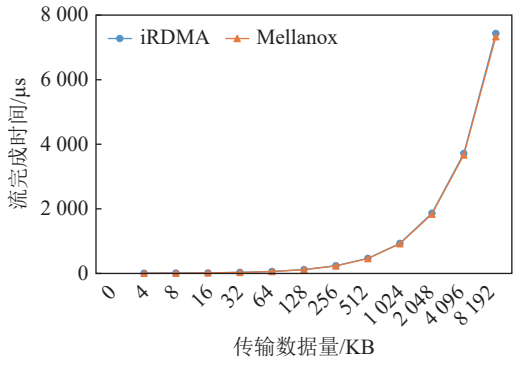


Fig. 12 Delay test comparison

图 12 延迟测试对比

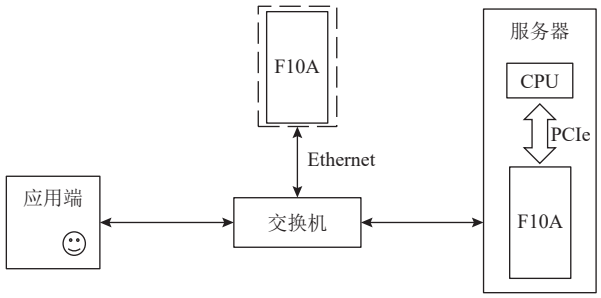


Fig. 13 Connection method

图 13 连接方式

Table 6 Hardware Configuration

表 6 硬件配置

硬件	配置
10 Gbps 光纤交换机	Ruijie S6220
CPU 型号	Intel Xeon E5-2650 v4@2.4GHz
FPGA 加速卡	型号：Inspur F10A FPGA 芯片：10AX115H3F34E2SG 连接方式：SFP+ 10Gb Ethernet /PCIe Gen3 x8

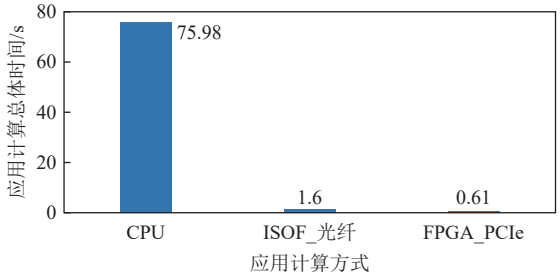


Fig. 14 Overall time comparison of application computing

图 14 应用计算总体时间对比

延增加，达到良好的系统性能. 可以推断，如果采用更大规模的网络级别扩展，并且扩大投入更高性能加速设备和交换节点，将会大大降低系统总体处理时延.

5 结 论

本文提出了一种ISOF计算系统级平台，充分开发利用FPGA灵活可重构特性，基于自研互连协议构建了支持底层高速传输的硬件平台以支持系统层面的硬件解耦和独立资源远程映射功能，并为上层应用提供丰富统一的开发接口以支持云平台的建设，为数据中心计算单元扩展架构和资源应用提供思路，并做了可行性和性能验证.

未来的工作包括研究支持多种异构计算单元高性能扩展和高速互连的平台架构，增加实现异构计算资源池化、存储资源池化和异构网络融合等功能，以及实现资源池化以按需供给、减少预判需求分配资源导致的资源浪费等，为人工智能时代计算平台基础设施的融合架构做前瞻性探索与研究.

作者贡献声明：王彦伟提出工作思路和指导意见，规划研究方案，完成轻量级协议设计及系统实现等；李仁刚制定研究方向，提供总体方案路线，指导整体研究路线的调整；徐冉构建并完善详细研究方案，完成实验方案设计以及系统与实验实现，撰写论文；刘钧锴完成传输协议设计、网络传输硬件平台实现和部分实验.

参 考 文 献

[1] Zhu Zongwei, Zhang Junneng, Zhao Jinjin, et al. A hardware and software task-scheduling framework based on CPU+FPGA heterogeneous architecture in edge computing[J]. IEEE Access, 2019, 7: 148975−148988

[2] Choi Y, Cong J, Fang Zhenman, et al. A quantitative analysis on microarchitectures of modern CPU-FPGA platforms[C/OL]//Proc of the 53rd Annual Design Automation Conf. New York: ACM, 2016 [2024-07-09]. https://dl.acm.org/doi/abs/10.1145/2897937.2897972

[3] Man Xingchen, Zhu Jianfeng, Song Guihuan, et al. CaSMap: Agile mapper for reconfigurable spatial architectures by automatically clustering intermediate representations and scattering mapping process[C]//Proc of the 49th Annual Int Symp on Computer Architecture. New York: ACM, 2022: 259−273

[4] Qi Le, Chang Yisong, Chen Yuxiao, et al. A system-level platform with SoC-FPGA for RISC-V hardware-software integration[J]. Journal of Computer Research and Development, 2023, 60(6): 1204−1215 (in Chinese)
(齐乐，常轶松，陈欲晓，等. 基于SoC-FPGA的RISC-V处理器软硬件系统级平台[J]. 计算机研究与发展，2023，60(6)：1204−1215)

[5] Zha Yue, Li Jing. Virtualizing FPGAs in the cloud[C]//Proc of the 25th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2020: 845−858

[6] Chung E, Fowers J, Ovtcharov K, et al. Serving DNNs in real time at datacenter scale with project brainwave[J]. IEEE Micro, 2018, 38(2): 8−20

[7] Suda N, Chandra V, Dasika G, et al. Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks[C]//Proc of the 2016 ACM/SIGDA Int Symp Field-Programmable Gate Arrays. New York: ACM, 2016: 16−25

[8] Zhang Jialiang, Li Jing. Improving the performance of OpenCL-based FPGA accelerator for convolutional neural network[C]//Proc of the 2017 ACM/SIGDA Int Symp on Field-Programmable Gate Arrays. New York: ACM, 2017: 25−34

[9] Tine B, Yalamarthy K P, Elsabbagh F, et al. Vortex: Extending the RISC-V ISA for GPGPU and 3D-graphics[C]//Proc of the 54th Annual IEEE/ACM Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2021: 754−766

[10] Caulfield A M, Chung E S, Putnam A, et al. A cloud-scale acceleration architecture[C/OL]//Proc of the 49th Annual IEEE/ACM Int Symp on microarchitecture (MICRO). Piscataway, NJ: IEEE, 2016 [2024-07-09]. https://ieeexplore.ieee.org/abstract/document/7783710

[11] Amazon Web Services EC2. FPGA hardware and software development kit[EB/OL]. [2023-01-28]. https://github.com/aws/aws-fpga

[12] Tarafdar N, Thomas L, Fukuda E, et al. Enabling flexible network FPGA clusters in a heterogeneous cloud data center[C]//Proc of the 2017 ACM/SIGDA Int Symp on Field-Programmable Gate Arrays. New York: ACM, 2017: 237−246

[13] Shu Ran, Cheng Peng, Chen Guo, et al. Direct universal access: Making data center resources available to FPGA[C]//Proc of the 16th USENIX Symp on Networked Systems Design and Implementation (NSDI 19). Berkeley, CA: USENIX Association, 2019: 127−140

[14] Yu Xiaoyu, Wang Yuwei, Miao Jie, et al. A data-center FPGA acceleration platform for convolutional neural networks[C]//Proc of the 29th Int Conf on Field Programmable Logic and Applications (FPL). Piscataway, NJ: IEEE, 2019: 151−158

[15] Choi Y K, Jason C, Fang Zheman, et al. In-depth analysis on microarchitectures of modern heterogeneous CPU-FPGA platforms [J]. ACM Transactions on Reconfigurable Technology and Systems, 2019, 12(1): 1−20

[16] Fleming K, Adler M. The LEAP FPGA Operating System[M]//FPGAs for Software Programmers. Berlin: Springer, 2016: 245−258

[17] Khawaja A, Landgraf J, Prakash R, et al. Sharing, protection, and compatibility for reconfigurable fabric with AmorphOS[C]//Proc of the 13th USENIX Symp on Operating Systems Design and Implementation (OSDI'18). Berkeley, CA: USENIX Association, 2018: 107−127

[18] Baxter R, Booth S, Bull M, et al. Maxwell-a 64 FPGA supercomputer[C]//Proc of the 2nd NASA/ESA Conf on Adaptive Hardware and Systems (AHS 2007). Piscataway, NJ: IEEE, 2007: 287−294

[19] Jeremy F, Kalin O, Michael P, et al. A configurable cloud-scale DNN processor for real-time AI[C/OL]//Proc of the 45th Annual Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2018[2024-07-09]. https://ieeexplore.ieee.org/abstract/document/8416814

[20] Ouyang J, Shiding L, Qi Wei, et al. SDA: Software-defined accelerator for large-scale DNN systems[C]//Proc of the 26th IEEE Hot Chips Symp (HCS). Piscataway, NJ: IEEE, 2014: 10–12

[21] Vesper M, Koch D, Vipin K, et al. JetStream: An open-source high-performance PCI express 3 streaming library for FPGA-to-Host and FPGA-to-FPGA communication[C/OL]//Proc of the 26th Int Conf on Field Programmable Logic and Applications (FPL). Piscataway, NJ: IEEE, 2016[2024-07-09]. https://ieeexplore.ieee.org/abstract/document/7577334

[22] Jacobsen, M, Richmond, D, Hogains, M, et al. RIFFA 2.1: A reusable integration framework for FPGA accelerators[J]. ACM Transactions on Reconfigurable Technology and Systems, 2015, 8(4): 1−23

[23] Zeke W, Zhang Shuhao, He Bingsheng, et al. Melia: A MapReduce framework on OpenCL-Based FPGAs[J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(12): 3547−3560

[24] Sharma D D, Blankenship R, Berger D S. An introduction to the compute express link (CXL) interconnect[J]. arXiv preprint, arXiv: 2306.11227, 2023

[25] Wang Fu, Yan Fulong, Xue Xuwei, et al. Traffic load balancing based on probabilistic routing in data center networks[C/OL]//Proc of the Int Conf on Optical Network Design and Modeling (ONDM). Piscataway, NJ: IEEE, 2020[2024-07-09]. https://ieeexplore.ieee.org/abstract/document/9133002

[26] Mittal R, Shpiner A, Panda A, et al. Revisiting network support for RDMA[C]//Proc of the 2018 Conf of the ACM Special Interest Group on Data Communication. New York: ACM, 2018: 313−326

[27] Biookaghazadeh S, Zhao Ming, Ren Fengbo. Are FPGAs suitable for edge computing[C]//Proc of the USENIX Workshop on Hot Topics in Edge Computing (HotEdge'18). Berkeley, CA: USENIX Association, 2018. https://www.usenix.org/conference/hotedge18/presentation/biookaghazadeh

[28] Belabed T, Coutinho M G F, Fernandes M A C, et al. User driven FPGA-based design automated framework of deep neural networks for low-power low-cost edge computing[J]. IEEE Access, 2021, 9: 89162−89180

[29] Ross S M. Introduction to Probability Models[M]. Amsterdam, Netherlands: Elsevier, 2014

[30] Duan Tiantian, Guo Y, Li Bo, et al. PieBridge: An on-demand scalable cross-chain architecture[J]. Journal of Computer Research and Development, 2023, 60(11): 2520−2533 (in Chinese)
(段田田, 郭仪, 李博, 等. PieBridge: 一种按需可扩展的跨链架构[J]. 计算机研究与发展, 2023, 60(11): 2520−2533)

[31] Zhang Fan, Hu Chengchen. Flow scheduling policy for burst traffic in data center networks[J]. Journal of Software, 2018, 28(s2): 81−89 (in Chinese)
(张帆, 胡成臣. 一种适用突发流量的数据中心网络流调度策略[J]. 软件学报, 2018, 28(s2): 81−89)



Wang Yanwei, born in 1985. PhD candidate, associate researcher. Member of CCF. His main research interests include computer architecture, heterogeneous acceleration, high-performance networks, and distributed computing platforms.

王彦伟, 1985 年生. 博士研究生, 副研究员. CCF 会员. 主要研究方向为计算机体系结构、异构加速、高性能网络、分布式计算平台.



Li Rengang, born in 1980. PhD candidate, senior engineer. Member of CCF. His main research interests include computer architecture, heterogeneous acceleration, high-performance networks, artificial intelligence, and high performance computing.

李仁刚, 1980 年生. 博士研究生, 正高级工程师. CCF 会员. 主要研究方向为计算机体系结构、异构加速、高性能网络、人工智能、高性能计算.



Xu Ran, born in 1988. PhD. His main research interests include computer architecture, heterogeneous acceleration, and high-performance networks.

徐　冉, 1988 年生. 博士. 主要研究方向为计算机体系结构、异构加速、高性能网络.



Liu Junkai, born in 1977. Master. His main research interests include network data transmission protocols and FPGA prototype verification.

刘钧锴, 1979 年生. 硕士. 主要研究方向为网络数据传输协议、FPGA 原型验证.