现场可编程逻辑门阵列智能网卡的研究综述

尹宇枫

(华中科技大学 计算机科学与技术学院,武汉市 430000)

摘 要 5G 的到来，对软件定义网络（SDN）和网络功能虚拟化（NFV）提出了更加迫切的需求。现有的网络基础设施可支撑千万人沟通，但不能支持实时沟通的数十亿物联网设备，这就需要运营商对网络进行虚拟化，构建更加灵活的网络基础设施。但是，软件解决方案并不能提供足够的网络可靠性和服务质量，而具有高级可编程功能 SmartNIC，将在虚拟化网络中扮演非常重要的角色。Smart NIC 能够提升应用程序和虚拟化性能，实现软件定义网络（SDN）和网络功能虚拟化（NFV）的诸多优势，将网络虚拟化、负载均衡和其他低级功能从服务器 CPU 中移除，确保为应用提供最大的处理能力。与此同时，智能网卡还能够提供分布式计算资源，使得用户可以开发自己的软件或提供接入服务，从而加速特定应用程序。Smart NIC 即智能网卡，其核心是通过 FPGA 协助 CPU 处理网络负载，编程网络接口功能。FPGA（Field－Programmable Gate Array），即现场可编程门阵列，它是在 PAL、GAL、CPLD 等可编程器件的基础上进一步发展的产物。Smart NIC 可以将代码从 CPU 引入网卡，显著加速用于安全应用的加密/解密或用于深度包检测（DPI）应用程序，降低 CPU 负载，并且支持灵活的网络可编程性。超融合架构数据中心中，Smart NIC 为 SDN 和虚拟化应用程序提供硬件加速与网络接口紧密结合，并可分布在大型服务器网络中，减小 CPU 负载，提供额外的边缘计算能力，加速特定应用和虚拟化功能，并且通过正确的语言和工具链支持，为用户提供应用加速即服务的附加价值。

关键词 智能网卡；网络虚拟化；SoC；FPGA；远程直接内存访问

1 引言

智能网卡是时代技术发展的需求。5G 的到来，对软件定义网络（SDN）和网络功能虚拟化（NFV）提出了更加迫切的需求。现有的网络基础设施可支撑千万人沟通，但不能支持实时沟通的数十亿物联网设备，这就需要运营商对网络进行虚拟化，构建更加灵活的网络基础设施。但是，软件解决方案并不能提供足够的网络可靠性和服务质量，而具有高级可编程功能 SmartNIC，将在虚拟化网络中扮演非常重要的角色。Smart NIC 能够提升应用程序和虚拟化性能，实现软件定义网络（SDN）和网络功能虚拟化（NFV）的诸多优势，将网络虚拟化、负载均衡和其他低级功能从服务器 CPU 中移除，确保为应用提供最大的处理能力。与此同时，智能网卡还能够提供分布式计算资源，使得用户可以开发自己的软件或提供接入服务，从而加速特定应用程序。

Smart NIC 即智能网卡，其核心是通过 FPGA 协助 CPU 处理网络负载，编程网络接口功能。FPGA（Field－Programmable Gate Array），即现场可编程门阵列，它是在 PAL、GAL、CPLD 等可编程器件的基础上进一步发展的产物。

随着越来越多的应用程序使用微服务架构部署，云服务提供商 CSP 正试图降低 RPC 尾部延迟。分布式应用程序通过将计算划分为同时执行的细粒度任务，获得了在云数据中心的商用服务器上运行的高性能（例如，视频编码、视频压缩和人脸识别）。这些应用程序通常将远程过程调用（RPC）请求从根节点扇出到大量叶节点，并在多个层中扇出。通常情况下，服务级别性能受到单个叶的 RPC 尾部延迟的限制。因此，如果能够减少（甚至限定）RPC 尾部延迟，分布式应用程序将运行得更快。这里将介绍 NanoPU。

许多不同的云和数据中心应用已被证明受益于将计算卸载到可编程 NIC 上，但是，没有一个“银弹”卸载方式可以在所有情况下提高性能。相反，不同的应用程序将指定自己的卸载链，然后运营商将这些链与基础设施相关的卸载合并，并在其可编程 NIC 上运行它们。为了实现这一目标，提出了一种新的可扩展的高性能可编程多租户网络网卡 PANIC。

传输协议，连同网络堆栈的其余部分，传统上是在软件中运行的。尽管努力提高性能和效率，软件网络堆栈往往消耗 30-40%的 CPU 时间，跟上当

今数据中心的应用程序需求。随着数据中心转向 100 Gbps 以太网，软件网络堆栈的 CPU 利用率变得越来越高。因此，多家供应商开发了完全在网络接口卡（NIC）上运行的硬件网络堆栈。但是，这些 NIC 上只实现了两种主要传输协议，它们都是硬连接的，并且只能由供应商修改：RoCE 用于远程直接内存访问（RDMA），使用 DCQCN 进行拥塞控制，使用简单的返回 N 方法进行可靠的数据传输。TCP 协议。一些供应商将自己选择的 TCP 变体卸载到 NIC 上，以便直接通过 socket API 使用（TCP 卸载引擎）或启用 RDMA（iWARP）。CP 在各种网络中多年来的优化列表证明了传输协议对可编程性的需求。这里提出 TONIC 主要研究如何使硬件传输协议可编程。

网络接口控制器（NIC）是计算机与网络进行交互的关口。网卡在软件栈和网络之间形成一个桥梁，即定义了网络接口。网络接口的功能以及这些功能的实现都在迅速发展变化。这些变化是由不断增长的带宽/速率和支持高性能分布式计算和虚拟化的 NIC 功能的双重需求推动的，不断增长的带宽导致许多 NIC 功能必须用硬件而不是软件来实现。另一方面，为了实现高级的协议和网络体系结构，需要新的网络功能，例如对多个队列的精确传输控制。为了满足新的网络协议和架构的开放式开发平台的需求，开发人员设计开发一个基于 FPGA 的高性能开放源码 NIC 原型平台 Corundum。

2 原理和优势

2.1 NanoPU

云服务提供商 CSP 正试图降低 RPC 尾部延迟。分布式应用程序通过将计算划分为同时执行的细粒度任务，获得了在云数据中心的商用服务器上运行的高性能（例如，视频编码、视频压缩和人脸识别）。这些应用程序通常将远程过程调用（RPC）请求从根节点扇出到大量叶节点，并在多个层中扇出。通常情况下，服务级别性能受到单个叶的 RPC 尾部延迟的限制。因此，如果能够减少（甚至限定）RPC 尾部延迟，分布式应用程序将运行得更快。CSP 正试图通过引入专门的 NIC 硬件来解决这个问题，比如，NIC 具有快速的 RDMA 和运行低延迟微服务的驻留在 NIC 内的 CPU 核。一般，微服务的调用需要 5-10 毫秒，因此只有当发送它的计算时间超过 10 毫秒时才值得调用。NanoPU 的目标是实现高效的亚微秒级 RPC，它可以在服务器上以不到 1ms 的通信开销调用。一个关键指标是线-线延迟，定义为从 RPC 请求消息的第一比特到达 NIC，直到处理的 RPC 响应的第一比特离开 NIC 的时间。典型的中位数线线延迟约为 850ns。NanoPU 目标是将中位数和尾数都减少到 100ns 以下，使运行“nanoService”变得有价值；短的 RPC 需要不到 1ms 的工作时间。

2.2 PANIC

网络线速处理与 CPU 产生和消耗数据的速率之间的差距正迅速扩大。可编程（“智能”）NIC 可以帮助克服这个问题。有许多不同类型的功能可以卸载到可编程 NIC 上。这些功能卸载加速了网络堆栈所有不同层的计算，可以减少通用 CPU 的负载，减少延迟，并提高吞吐量。许多不同的云和数据中心应用已被证明受益于将计算卸载到可编程 NIC 上，但是，没有一个“银弹”卸载方式可以在所有情况下提高性能。相反，不同的应用程序将指定自己的卸载链，然后运营商将这些链与基础设施相关的卸载合并，并在其可编程 NIC 上运行它们。为了实现这一目标，提出了一种新的可扩展的高性能可编程多租户网络网卡 PANIC，它支持多种不同类型的卸载，并将它们组合成独立的卸载功能链。

2.3 TONIC

传输协议，连同网络堆栈的其余部分，传统上是在软件中运行的。尽管努力提高性能和效率，软件网络堆栈往往消耗 30-40%的 CPU 时间，跟上当今数据中心的应用程序需求。随着数据中心转向 100 Gbps 以太网，软件网络堆栈的 CPU 利用率变得越来越高。因此，多家供应商开发了完全在网络接口卡（NIC）上运行的硬件网络堆栈。但是，这些 NIC 上只实现了两种主要传输协议，它们都是硬连接的，并且只能由供应商修改：RoCE：RoCE 用于远程直接内存访问（RDMA），使用 DCQCN 进行拥塞控制，使用简单的返回 N 方法进行可靠的数据传输。TCP 协议：一些供应商将自己选择的 TCP 变体卸载到 NIC 上，以便直接通过 socket API 使用（TCP 卸载引擎）或启用 RDMA（iWARP）。

然而，这些协议只使用了过去几十年中提出的用于可靠传输和拥塞控制的无数可能算法中的一小部分。例如，最近的研究表明，低延迟数据中心网络可以显著受益于接收端驱动的传输协议，这在当前的硬件堆栈中不是一个可选项。为了在

Microsoft 数据中心部署 RoCE NIC，运营商需要修改数据传输算法以避免网络中的活锁，但必须依赖 NIC 供应商来进行更改。已经提出了其他算法来改进 RoCE 的简单可靠传输算法。TCP 在各种网络中多年来的优化列表证明了传输协议对可编程性的需求。TONIC 主要研究如何使硬件传输协议可编程。即使 NIC 供应商开放了硬件编程接口，在高速硬件中实现传输协议也需要大量的专业知识、时间和精力。为了满足 100Gbps 的处理速度，传输协议应该每隔几纳秒生成并传输一个数据包。它应该能够处理超过 1000 个活动流。

2.4 Corundum

网络接口控制器（NIC）是计算机与网络进行交互的关口。网卡在软件栈和网络之间形成一个桥梁，即定义了网络接口。网络接口的功能以及这些功能的实现都在迅速发展变化。这些变化是由不断增长的带宽/速率和支持高性能分布式计算和虚拟化的 NIC 功能的双重需求推动的，不断增长的带宽导致许多 NIC 功能必须用硬件而不是软件来实现。另一方面，为了实现高级的协议和网络体系结构，需要新的网络功能，例如对多个队列的精确传输控制。为了满足新的网络协议和架构的开放式开发平台的需求，设计开发一个基于 FPGA 的高性能开放源码 NIC 原型平台 Corundum，能够运行至少 94Gbps，是完全开源的，连同它的驱动程序，可以使用在一个完整的网络协议栈。这种设计既具有好的移植型又紧凑，支持许多不同的设备，同时也为进一步定制（即使在较小的设备上）留下了充足的硬件资源。Corundum 模块化设计和可扩展性允许协同优化的硬件/软件解决方案在系统中开发和测试高级网络应用程序。

3 研究进展

3.1 NanoPU

NIC 具有快速的 RDMA 和运行低延迟微服务的驻留在 NIC 内的 CPU 核。一般，微服务的调用需要 5-10 毫秒，因此只有当发送它的计算时间超过 10 毫秒时才值得调用。NanoPU 的目标是实现高效的亚微秒级 RPC，它可以在服务器上以不到 1ms 的通信开销调用。一个关键指标是线-线延迟，定义为从 RPC 请求消息的第一比特到达 NIC，直到处理的 RPC 响应的第一比特离开 NIC 的时间。典型的中位数线线延迟约为 850ns。NanoPU 目标是将中位数和尾数都减少到 100ns 以下，使运行“nanoService”变得有价值；短的 RPC 需要不到 1ms 的工作时间。

RPC 尾延迟高的原因:

1. 关键路径上的内存和缓存层次结构。

   CPU 的网络堆栈使用内存作为工作空间来保存和处理数据包。这必然会干扰应用程序的内存访问，引发资源争用，从而导致较差的 RPC 尾部延迟。此外，如果一个数据包通过 PCIe 传输到 DRAM，它在到达几百纳秒后才对 CPU 可用。使用直接缓存访问技术（如 DDIO 和 DCA），这一点得到了减少，但数据包仍必须经过网络堆栈的许多层，并且上下文切换需要额外的延迟（1–5ms），包括内存拷贝、TLB 刷新、虚拟内存管理和缓存替换。

2. 次优调度。

   当 RPC 数据包到达时，它必须被分派到核心进行网络堆栈处理；当完成时，RPC 请求消息被转发到工作核心进行处理。在每个步骤中，软件核心选择算法选择核心；线程调度器决定处理何时开始。这两种算法都需要内核和 NIC 频繁访问内存，需要内存总线、PCIe 和缓存的介入。已经证明，这些算法位于关键的处理路径上，并试图降低处理时间。然而，这些软件调度器的粒度本质上受到执行核心间同步（例如发送和接收中断）所需开销的限制。因此，每 5ms 进行一次以上的调度决策变得不切实际。

以前许多减少 RPC 开销的尝试包括低延迟和无损交换机，减少了网络层的数量，以及专门的库。目前最快的方法是部署专用 NIC 和交换机硬件，但这些都很难编程。能否设计一个易于编程的 CPU 内核，同时能够以绝对最小的开销和尾部延迟服务 RPC 请求？

虽然 NanoPU 不是第一个尝试减少这些处理的延迟，但描述了第一个将 RPC 尾部延迟最小化的完整设计。nanoPU 的目标是通过使用流水线、可编程硬件，将数据直接放置到 CPU 内核中，完全绕过内存和缓存层次结构，最大限度地减少在每一层上花费的时间。

NanoPU 可以看作是一个在线处理模型，除了常

规处理，为未来的 CPU 核心优化亚微秒 RPC 服务。或者，NanoPU 可以被认为是一种新的特定于领域的“NanoService”处理器，设计用于安装在 SmartNIC 上，或者作为一个独立的集群来服务亚微秒级的 RPC。例如，构建一个单芯片 512 核 NanoPU 是可行的，类似于 Celerity，有 100GE 接口，以持续 10Tb/s 的速度每秒服务超过 5 亿个 RPC。这样的设备可以从根本上提高大型分布式应用程序的性能。NanoPU 是一种 NIC 和 CPU 联合设计，最大限度地减少 RPC 尾部延迟。主要包括：1）NIC 中的专用内存层次结构，直接连接到 CPU 寄存器文件；2）低延迟硬件传输逻辑、核心选择和线程调度；3）通过限制消息处理时间来限制尾部延迟。

在 NanoPU 中，第一，需要最小化从 RPC 请求包通过以太网到达到它在运行线程中开始处理的时间。NanoPU 通过用硬件替换软件线程调度器和核心选择器（又称负载均衡器）；完全绕过 PCIe、主内存和缓存层次结构；将 RPC 数据直接放入 CPU 寄存器文件；用硬件中的可靠传输层替换主机网络软件堆栈来实现这一点，向 CPU 传递完整的 RPC 消息。第二，需要尽量减少网络拥塞。NanoPU 在硬件上实现了 NDP（使用可编程 P4 流水线），减少了拥塞并提高了 incast 性能。第三，需要在硬件上通过流水线头和传输层处理、线程调度和核心选择来最大化 RPC 吞吐量。NanoPU 在 NIC 中包含 P4 PISA 流水线，并行处理多个数据包，并重新组装 RPC 消息。最后，大型分布式应用程序的性能通常受到 RPC 尾部延迟的限制；因此，在处理 RPC 时，需要最小化尾部延迟。NanoPU 提供了第一个确定性尾延迟 RPC 服务，保证了一致的 RPC 请求将在到达 NIC 的 1ms 内完成服务。

利用 200Gb/s 网络接口及扩展 RISC-V Rocket 核心的 NanoPU 开源原型，在 AWS F1 FPGA 实例上通过可重复的周期精确模拟进行评估。测试得到：1）线-线延迟仅为 65ns（没有以太网 MAC 和串行 I/O 的情况下为 13ns），2）每个核心的吞吐量为 200Gb/s，比最新技术快 2.5 倍，3）NIC 中的处理速度为 350 Mpkts/s（包括传输和核心选择逻辑），比 Shinjuku ， Shenango 和 eRPC 软件解决方案快 50 倍，4）硬件抢占式线程调度，在高负载下使 99% 的尾部延迟低于 2:1ms。

3.2 PANIC

现有的 NIC 设计可分为以下几类，每类都有关键限制：

1. 卸载流水线 NIC。

   在流水线中部署多个功能卸载，以使数据包能够由一系列功能处理。功能链可以在这些 NIC 中修改，但是需要大量的时间和开发人员的努力来进行 FPGA 合成，并且慢的卸载会导致数据包丢失或头（HOL）阻塞。
2. Manycore NIC。

   跨多个嵌入式CPU 内核对数据包进行负载平衡，然后 CPU 内核根据不同的卸载需要控制数据包的处理。这些设计都存在性能问题，因为嵌入式 CPU 内核会增加数十微秒的额外延迟。此外，现有的多核心 NIC 都没有提供隔离竞争租户的性能机制。此外，如果工作集不适合内核的缓存，许多内核 NIC 的性能可能会显著降低。
3. RMT NIC。

   在 NIC 可重构匹配+操作（RMT）流水线上使用，以实现 NIC 卸载。RMT 流水线支持的卸载类型是有限的，因为每个流水线级必须能够在每个时钟周期处理一个新的数据包。

PANIC 克服了现有网卡设计的主要局限性。PANIC 从最近关于可重构（RMT）交换机的工作中获得了灵感。PANIC 的设计利用了三个关键原则：

1. 卸载应该是独立的。潜在有用的卸载集是多样的和巨大的，跨越了网络堆栈的所有层。因此，可编程 NIC 应该能够支持硬件 IP 核和嵌入式 CPU 作为卸载。
2. 数据包调度、缓冲和负载平衡应集中化，以获得最佳性能和效率。因为分散决策和每次卸载队列可能会导致较差的尾部响应延迟，以及由于负载不平衡而导致较差的缓冲区利用率。
3. 由于中小型无阻塞结构的成本相对于 NIC 整体而言较小，因此卸载应该由无阻塞/低超额订阅交换结构连接，以实现卸载的灵活链接。

遵循这些设计原则，PANIC 做出了三个关键贡献：1）一种新的可编程 NIC 设计，其中不同的功能卸载连接到一个非阻塞交换结构，功能链由可编程 RMT 流水线编排；2）一种新的混合推拉调度器和负载均衡器，具有优先级感知丢包功能，3）对网卡上可编程交换和调度的成本进行分析，发现它

们相对于网卡整体而言成本较低。

PANIC NIC 有四个组件：

1. RMT 交换流水线
2. 交换结构
3. 中央调度器
4. 独立的计算单元

RMT 流水线提供可编程的功能链编排。一个高性能的互连，使可编程功能链线速处理。中央调度器提供隔离、缓冲区管理和负载平衡。独立的计算单元可以是硬件加速器或嵌入式内核，不需要以线速率运行。

为了评估 PANIC 的可行性，进行了 ASIC 分析和 FPGA 原型实验。FPGA 原型能够以 100Gbps 的速率执行动态卸载链，并在各种链配置下实现纳秒级（<0.8ms）的数据包调度和负载均衡。经验表明，与最先进的基于流水线的设计相比，PANIC 可以更好地处理多租户隔离和线速功能卸载。在小型试验台上的端到端实验表明，PANIC 可以在 100Gbps 下实现动态带宽分配和优先分组调度。总的来说，包括 8*8 的 PANIC 组件只消耗了 Xilinx UltraScale Plus FPGA 上可用的总逻辑区域（LUT）的 11.27%。

3.3 Tonic

高速网卡上的传输协议可以进行编程，而不会让用户暴露于高速硬件编程的全部复杂性。主要基于两个主要因素：首先，可编程传输逻辑是实现灵活的硬件传输协议的关键。传输协议的实现执行若干功能，例如连接管理、数据缓冲区管理和数据传输。然而，它的中心责任，关键创新点，是决定要传输的数据段（数据传递）和时刻（拥塞控制），称为传输逻辑。因此，高速网卡上可编程传输协议的关键是允许用户修改传输逻辑。其次，可以利用传输逻辑中的常见模式来创建可重用的高速硬件模块。尽管它们在应用程序级 API（例如，TCP 的套接字和字节流抽象与 RDMA 的基于消息的谓词 API）以及连接和数据缓冲区管理方面存在差异，但传输协议有几个共同的模式。例如，不同的传输协议使用不同的算法来检测丢失的数据包。然而，一旦一个数据包被丢弃，可靠的传输协议将其重传优先于发送一个新的数据段。在拥塞控制中，给定由控制环路确定的参数（例如，拥塞窗口和速率），只有几种常见的方法可以计算流在任何时候可以传输多少字节。这使能够为硬件中的传输逻辑设计一个有效的“模板”可以用一个简单的 API 编程。

基于这些观点，设计并开发了 Tonic，这是一种可编程的硬件体系结构，可以使用简单的 API 实现各种传输协议的传输逻辑，同时支持 100Gbps 的数据速率。每个时钟周期，Tonic 都会生成下一段的地址以供传输。数据段由下游 DMA 管道从内存中提取，并由硬件网络堆栈的其余部分转换为完整的数据包。Tonic 将驻留在 NIC 上，取代传输协议硬件实现中的硬编码传输逻辑（例如，未来的 RDMA NIC 和 TCP 卸载引擎）。Tonic 为传输逻辑提供了一个统一的可编程体系结构，与不同传输协议的具体实现如何执行连接和数据缓冲区管理以及它们的应用程序级 API 无关。然而，将描述 Tonic 如何与传输层的其余部分进行通用接口，以及如何将其集成到 Linux 内核中，以使用 socketapi 与应用程序进行交互。

3.4 Corundum

现有 NIC 的网络接口功能设计关键在于硬件和软件之间进行功能划分。硬件 NIC 功能分为两大类。第一类由简单的卸载功能组成，这些功能可以从 CPU 中移除一些逐包的处理，例如校验和/哈希计算和分段卸载，这些功能使网络协议栈能够批量处理数据包。第二类包括必须在 NIC 上硬件中实现的功能，以实现高性能和公平性，这些特性包括流控制、速率限制、负载均衡和时间戳。

ASIC NIC：传统的 NIC 的硬件功能实现于专有的专用集成电路（ASIC）中。再加上规模经济，这使得 ASIC 网卡具有高性能低成本特点。然而，这些 ASIC 可扩展性是有限的，添加新硬件功能的开发成本高周期长。为了克服这些限制，开发了各种智能 NIC 和软件 NIC。智能网卡通常通过提供大量可编程处理核心和硬件原语，在网卡上提供强大的可编程性。这些资源可用于从主机上卸载各种应用程序、网络和虚拟化操作。然而，智能 NIC 不一定能很好地扩展到高性能，并且硬件功能可能受到限制。

软件 NIC：软件 NIC 通过在软件中实现网络功能，提供了最大的灵活性，绕过了大多数硬件卸载功能。因此，可以快速开发和测试新功能，但需要进行各种权衡，包括占用主机 CPU 周期和不一定支持线速处理。此外，由于软件固有的随机中断特性，开发需要精确传输控制的网络应用程序是不可行的。尽管如此，许多研究项目通过修改网络协议栈或使用内核旁路框架（如数据平面开发套件（DPDK））在软件中实现了新的 NIC 功能。

FPGA NIC：基于 FPGA 的 NIC 结合了基于

ASIC 的 NIC 和软件 NIC 的优点：能够以线速运行，提供低延迟和精确计时，同时具有相对较短的功能开发周期。高性能、专有、基于 FPGA 的 NIC 也已有很多。例如，阿里巴巴开发了完全定制的基于 FPGA 的 RDMA 专用 NIC，用于运行精确拥塞控制协议（HPCC）。商业产品也存在，包括 Exablaze 和 Netcope 等产品。

基于 FPGA 的数据包处理解决方案包括 Catapult，它实现了网络应用程序卸载，FlowBlaze，它在 FPGA 上实现了可重构的匹配动作引擎。然而，这些平台将标准 NIC 功能留给了一个单独的基于 ASIC 的 NIC，并且完全作为一个“bump-in-the-wire”模式来运行，不提供对 NIC 调度器或队列的显式控制。其他项目使用软件实现或部分硬件实现。Shoal 描述了一种网络体系结构，它使用自定义 NIC 和快速链路层交叉开关执行单元路由。Shoal 是在硬件中构建的，但是只使用合成流量进行评估，没有与主机连接。SENIC 描述了基于可伸缩 NIC 的速率限制。调度器的硬件实现是单独评估的，但系统级评估是在带有定制排队规程（qdisc）模块的软件中执行的。PIEO 描述了一个灵活的 NIC 调度器，它是在硬件中单独评估的。NDP 是数据中心应用程序的接收端驱动的拉模式传输协议。使用 DPDK 软件 NICs 和基于 FPGA 的交换机对 NDP 进行评估。Loom 描述了一种高效的 NIC 设计，并用 BESS 软件对其进行了评估。与基于 ASIC 的 NIC 类似，商用的基于 FPGA 的 NIC 往往是专有的，具有不能修改的基本“黑盒”功能。基本 NIC 功能的不能升级，严重限制了开发新网络应用程序的实用性和灵活性。

商用高性能 DMA 组件，如 Xilinx XDMA 核心和 QDMA 核心，以及 Atomic Rules Arkville DPDK 加速核心，没有提供完全可配置的硬件来控制传输数据流。Xilinx DMA 内核是为计算卸载应用程序而设计的，因此提供了非常有限的排队功能，并且没有简便方法来控制传输调度。Xilinx QDMA 核心和 Atomic Rules Arkville DPDK 加速核心通过支持少量队列和提供 DPDK 驱动程序面向网络应用程序。然而，XDMA 核心支持的队列数量是的 2K 队列，Arkville 核心的队列达 128 个队列，这两个核心都没有提供精确控制数据包传输的简单方法。

NetFPGA 等开源项目只为基于 FPGA 的数据包处理提供了一个功能库和平台，并不是专门为 NIC 开发而设计的。此外，NetFPGA NIC 参考设计使用了 Xilinx XDMA 内核，它不是为网络应用而设计的。将 NetFPGA 板的参考 NIC 设计中的 Xilinx-XDMA 内核替换为 Corundum，可以获得更强大、更灵活的原型平台。

Corundum 别于所有这些项目，因为它是完全开源的，可以运行在一个标准的主机网络栈，在实际的物理链路。它提供了数千个传输队列和可扩展的传输调度程序，用于流的细粒度控制。这就为开发结合了硬件和软件功能的网络应用程序提供了一个强大而灵活的开源平台。

4 总结与展望

随着应用性能高要求和虚拟交换机性能限制的矛盾日益突出，使用智能网卡来代替传统网卡，成为 ICT 行业的研究热点。云计算的两大特性是虚拟化和资源池化，智能网卡则加速了基础设施的虚拟化和资源池化。随着数据中心内部数据和带宽的爆炸式增长，传统软件实现虚拟化以及数据处理的方式遇到了一系列问题。智能网卡的出现，将网络、存储以及安全任务从 CPU 卸载到了硬件进行加速，解决了在大带宽场景下 CPU 的消耗以及延迟和抖动等问题。目前，市场上主流的智能网卡硬件形式主要有四种，NP 架构、通用 ASIC 架构(内嵌 ARM)、FPGA+SoC 架构、SoC 和定制化 ASIC 架构，产品形态包括单卡、双卡、OCP 卡。几种架构没有优劣势区分，差异在于性能可编程、功耗和成本之间的平衡，用户基于这几个方面的需求做选择。在智能网卡解决方案上可采取了 FPGA+CPU 架构。这个架构有它的独到之处：一是高性能，FPGA 提供了接近 ASIC 的处理能力。二是软硬件全可编程，产品设计更灵活，更能满足客户业务的实际演进。”将 IO 设备硬件虚拟化，在网络、存储、安全方面做到了硬件加速卸载，用户可以基于智能网卡逐步去做管理平面卸载。

参 考 文 献

[1] Ibanez S , Mallery A , Arslan S , et al. The nanoPU: Redesigning the CPU-Network Interface to Minimize RPC Tail Latency[J]. 2020.

[2] Lin J , Patel K , Stephens B E , et al. PANIC: a high-performance programmable NIC for multi-tenant networks. 2020.

[3] Mina Tahmasbi Arashloo, Alexey Lavrov, Manya Ghobadi, et al.Enabling Programmable Transport Protocols in High-Speed NICs. 2020.

[4] Forencich A , Snoeren A C , Porter G , et al. Corundum: An Open-Source 100-Gbps NIC[C]// 2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). IEEE, 2020.

[5] Grant S , Yelam A , Bland M , et al. SmartNIC Performance Isolation with FairNIC: Programmable Networking for the Cloud[C]// SIGCOMM '20: Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication. ACM, 2020.