



# 专用数据处理器 (DPU)

Technical White Paper of Data Processing Unit

## 技术白皮书

第1.0版

2021年10月

### 主编单位

中国科学院计算技术研究所

### 联合编写发布单位

中科驭数（北京）科技有限公司

计算机体系结构国家重点实验室

中国计算机学会（CCF）集成电路设计专业组



**INTENTIONALLY LEFT BLANK**

## 编委会

顾问：孙凝晖、刘明、李晓维、胡伟武、陈云霁

主编：鄢贵海

编委（按姓氏拼音排序）：崔慧敏、陈岩、陈世敏、樊海爽、孔浩、卢文岩、  
廖云坤、李明、孙伟、吴婧雅、袁晓飞、赵巍岳、张宇、张宇军

## 特别致谢

本白皮书编写过程中还得到了北京市科学技术委员会、中关村科技园区管理委员会、中国移动物联网有限公司、曙光信息产业有限公司、浪潮集团、龙芯中科、华泰证券、中泰证券、金证股份、上交所技术有限公司、英特尔（中国）有限公司、赛灵思电子科技公司、东方国信科技股份有限公司、首都在线、迈普通信技术股份有限公司、360数科等机构和上下游企业的支持与协助，特此表达衷心的感谢！

## 版权声明

本白皮书版权属于主编和联合编写发布单位，并受法律保护。转载、摘编或利用其它方式使用本白皮书文字或者观点应注明“来源：专用数据处理器（DPU）技术白皮书，中国科学院计算技术研究所，鄢贵海等”。违反上述声明者，版权方将追究其相关法律责任。

# 目录

内容提要 .....	7
1. DPU技术发展概况 .....	9
1.1. 什么是DPU .....	9
1.2. DPU的发展背景 .....	12
1.3. DPU发展历程 .....	16
1.4. DPU与CPU、GPU的关系 .....	19
1.5. DPU的产业化机遇 .....	21
2. DPU特征结构 .....	23
2.1. DPU是以数据为中心IO密集的专用处理器 .....	23
2.2. 计算平台从“计算为中心”转向“数据为中心” .....	25
2.3. 一种DPU参考设计 .....	29
2.4. DPU具备的主要功能 .....	34
3. DPU应用场景 .....	35
3.1. 应用场景一：网络功能卸载 .....	35
3.2. 应用场景二：存储功能卸载 .....	51
3.3. 应用场景三：安全功能卸载 .....	55
4. DPU软件栈五层模型 .....	58
4.1. 软件栈开发面临的挑战 .....	58
4.2. DPU异构计算架构五层开发模型 .....	58
4.3. 典型软件框架案例 .....	63
5. 业界产品概要介绍 .....	70
5.1. NVIDIA BLUEFIELD .....	70
5.2. INTEL IPU (MOUNT EVANS) .....	73
5.3. MARVELL OCTEON .....	77
5.4. FUNGIBLE DPU .....	81
5.5. 中科驭数 K2 DPU .....	85
6. DPU发展展望 .....	92

# 内容提要

DPU (Data Processing Unit) 是新近发展起来的一种专用处理器。2020年NVIDIA公司发布的DPU产品战略中将其定位为数据中心继CPU和GPU之后的“第三颗主力芯片”，掀起了一波行业热潮。DPU的出现是异构计算的一个阶段性标志。与GPU的发展类似，DPU是应用驱动的体系结构设计的又一典型案例；但与GPU不同的是，DPU面向的应用更加底层，类型也更多样。DPU要解决的核心问题是基础设施的“降本增效”，即将“CPU处理效率低下、GPU处理不了”的负载卸载到专用DPU，提升整个计算系统的效率、降低整体系统的总体拥有成本（TCO）。新一代的DPU不仅可以作为运算的加速引擎，还具备控制平面的功能，能够运行Hypervisor，更高效的完成网络虚拟化、IO虚拟化、存储虚拟化等任务，彻底将CPU的算力释放给应用程序。DPU的出现也许是体系结构朝着专用化路线发展的又一个里程碑。

本白皮书将重点分析DPU产生的背景、技术特征、软硬件参考架构，应用场景、并对目前已经公布的DPU产品做简要的比较分析，为后续DPU技术发展提供必要的参考。本文的大体结构如下：

第一部分介绍DPU的技术发展概况，首先对DPU做了一个基本的定义，然后阐述了DPU发展的背景，并简要介绍DPU发展的历程，DPU在现有计算生态中的角色，最后以DPU的产业化机遇作为总结。

第二部分详细说明DPU的特征结构，对DPU的定位做了进一步阐述，然后提出一种通用的DPU的结构模型。

第三部分介绍DPU的应用场景，本文总结了三大应用场景：网络功能卸载、存储功能卸载、安全功能卸载，这也是DPU目前最重要的三个应用方向。

第四部分提出DPU开发的五层参考模型，包括设备层、操作层、计算引擎层、应用服务层和业务开发层，既体现了DPU开发过程中的软硬协同，也充分继承了通用软件栈的分层结构。

第五部分概要介绍目前行业的已经发布或已经披露的DPU产品，虽然其中绝大部分尚未到批量应用的阶段，各个竞品的优缺点也尚未得到市场的充分验证，但是对于后续DPU研发具有重要的参考价值。

第六部分展望未来DPU发展，并作为全文的总结。

# 1. DPU技术发展概况

## 1.1. 什么是DPU

DPU (Data Processing Unit) 是以数据为中心构造的专用处理器，采用软件定义技术路线支撑基础设施层资源虚拟化，支持存储、安全、服务质量管理等基础设施层服务。2020年NVIDIA公司发布的DPU产品战略中将其定位为数据中心继CPU和GPU之后的“第三颗主力芯片”，掀起了一波行业热潮。DPU的出现是异构计算的一个阶段性标志。与GPU的发展类似，DPU是应用驱动的体系结构设计的又一典型案例；但与GPU不同的是，DPU面向的应用更加底层。DPU要解决的核心问题是基础设施的“降本增效”，即将“CPU处理效率低下、GPU处理不了”的负载卸载到专用DPU，提升整个计算系统的效率、降低整体系统的总体拥有成本（TCO）。DPU的出现也许是体系结构朝着专用化路线发展的又一个里程碑。

### 1.1.1. 关于DPU中“D”的解释

DPU中的“D”有三种解释：

(1) Data Processing Unit, 即数据处理器。这种解释把“数据”放在核心位置，区别于信号处理器、基带处理器等通信相关的处理器对应的“信号”，也区别于GPU对应的图形图像类数据，这里的“数据”主要指数字化以后的各种信息，特别是各种时序化、结构化的数据，比如大型的结构化表格，网络流中的数据包，海量的文本等等。DPU就是处理这类数据的专用引擎。

(2) Datacenter Processing Unit, 即数据中心处理器。这种解释把数据中心作为DPU的应用场景，特别是随着WSC (Warehouse-scale Computer) 的兴起，不同规模的数据中心成为了IT核心基础设施。目前来看，DPU确实在数据中心中使用前景非常广阔。但是计算中心的三大部分：计算、网络、存储，计算部

分是CPU占主导，GPU辅助；网络部分是路由器和交换机，存储部分是高密度磁盘构成的的RAID系统和SSD为代表非易失性存储系统。在计算和网络中扮演数据处理的芯片都可以称之为Datacenter Processing Unit，所以这种说法相对比较片面。

(3) Data-centric Processing Unit，即以数据为中心的处理器。Data-centric，即数据为中心，是处理器设计的一种理念，相对于“Control-centric”即控制为中心。经典的冯诺依曼体系结构就是典型的控制为中心的结构，在冯诺依曼经典计算模型中有控制器、计算器、存储器、输入和输出，在指令系统中的表现是具有一系列非常复杂的条件跳转和寻址指令。而数据为中心的理念与数据流(Data Flow)计算一脉相承，是一种实现高效计算的方法。同时，现在试图打破访存墙(Memory Wall)的各种近存(Near-memory)计算、存内(In-memory)计算、存算一体等技术路线，也符合数据为中心的设计理念。

以上三种关于“D”的解释，从不同角度反映DPU的特征，都有一定的可取之处，笔者认为可以作为不同的三个维度来理解DPU的内涵。

### 1.1.2. DPU的作用

DPU最直接的作用是作为CPU的卸载引擎，接管网络虚拟化、硬件资源池化等基础设施层服务，释放CPU的算力到上层应用。以网络协议处理为例，要线速处理10G的网络需要的大约4个Xeon CPU的核，也就是说，单是做网络数据包处理，就可以占去一个8核高端CPU一半的算力。如果考虑40G、100G的高速网络，性能的开销就更加难以承受了。Amazon把这些开销都称之为“Datacenter Tax”，即还未运行业务程序，先接入网络数据就要占去的计算资源。AWS Nitro产品家族旨在将数据中心开销(为虚拟机提供远程资源，加密解密，故障跟踪，安全策略等服务程序)全部从CPU卸载到Nitro加速卡上，将给上层应用释放30%的原本用于支付“Tax”的算力！

DPU可以成为新的数据网关，将安全隐私提升到一个新的高度。在网络环境下，网络接口是理想的隐私的边界，但是加密、解密算法开销都很大，例如国密标准的非对称加密算法SM2、哈希算法SM3和对称分组密码算法SM4。如果用CPU来处理，就只能做少部分数据量的加密。在未来，随着区块链承载的业务的逐渐成熟，运行共识算法POW，验签等也会消耗掉大量的CPU算力。而这些都可以通过将其固化在DPU中来实现，甚至DPU将成为一个可信根。

DPU也可以成为存储的入口，将分布式的存储和远程访问本地化。随着SSD性价比逐渐可接受，部分存储迁移到SSD器件上已经成为可能，传统的面向机械硬盘的SATA协议并不适用于SSD存储，所以，将SSD通过本地PCIe或高速网络接入系统就成为必选的技术路线。NVMe (Non Volatile Memory Express) 就是用于接入SSD存储的高速接口标准协议，可以通过PCIe作为底层传输协议，将SSD的带宽优势充分发挥出来。同时，在分布式系统中，还可通过NVMe over Fabrics (NVMe-oF) 协议扩展到InfiniBand、Ethernet、或Fibre channel节点中，以RDMA的形式实现存储的共享和远程访问。这些新的协议处理都可以集成在DPU中以实现对CPU的透明处理。进而，DPU将可能承接各种互连协议控制器的角色，在灵活性和性能方面达到一个更优的平衡点。

DPU将成为算法加速的沙盒，成为最灵活的加速器载体。DPU不完全是一颗固化的ASIC，在CXL、CCIX等标准组织所倡导CPU、GPU与DPU等数据一致性访问协议的铺垫下，将更进一步扫清DPU编程障碍，结合FPGA等可编程器件，可定制硬件将有更大的发挥空间，“软件硬件化”将成为常态，异构计算的潜能将因各种DPU的普及而彻底发挥出来。在出现“Killer Application”的领域都有可能出现与之相对应的DPU，诸如传统数据库应用如OLAP、OLTP，5G边缘计算，智能驾驶V2X等等。

## 1.2. DPU的发展背景

DPU的出现是异构计算的又一个阶段性标志。摩尔定律放缓使得通用CPU性能增长的边际成本迅速上升，数据表明现在CPU的性能年化增长（面积归一化之后）仅有3%左右<sup>1</sup>，但计算需求却是爆发性增长，这几乎是所有专用计算芯片得以发展的重要背景因素。以AI芯片为例，最新的GPT-3等千亿级参数的超大型模型的出现，将算力需求推向了一个新的高度。DPU也不例外。随着2019年我国以信息网络等新型基础设施为代表的“新基建”战略帷幕的拉开，5G、千兆光纤网络建设发展迅速，移动互联网、工业互联网、车联网等领域发展日新月异。云计算、数据中心、智算中心等基础设施快速扩容。网络带宽从主流10G朝着25G、40G、100G、200G甚至400G发展。网络带宽和连接数的剧增使得数据的通路更宽、更密，直接将处于端、边、云各处的计算节点暴露在了剧增的数据量下，而CPU的性能增长率与数据量增长率出现了显著的“剪刀差”现象。所以，寻求效率更高的计算芯片就成为了业界的共识。DPU芯片就是在这样的趋势下提出的。

### 1.2.1. 带宽性能增速比 (RBP) 失调

摩尔定律的放缓与全球数据量的爆发这个正在迅速激化的矛盾通常被作为处理器专用化的大背景，正所谓硅的摩尔定律虽然已经明显放缓，但“数据摩尔定律”已然到来。IDC的数据显示，全球数据量在过去10年年均复合增长率接近50%，并进一步预测每四个月对于算力的需求就会翻一倍。因此必须要找到新的可以比通用处理器带来更快算力增长的计算芯片，DPU于是应运而生。这个大背景虽然有一定的合理性，但是还是过于模糊，并没有回答DPU之所以新的原因是什么，是什么“量变”导致了“质变”？

<sup>1</sup> Hennessy J, Patterson D. Computer Architecture, Sixth Edition: A Quantitative Approach, Chapter 1, Morgan Kaufmann. 2019

从现在已经公布的各个厂商的DPU架构来看，虽然结构有所差异，但都不约而同强调网络处理能力。从这个角度看，DPU是一个强IO型的芯片，这也是DPU与CPU最大的区别。CPU的IO性能主要体现在高速前端总线（在Intel的体系里称之为FSB，Front Side Bus），CPU通过FSB连接北桥芯片组，然后连接到主存系统和其他高速外设（主要是PCIe设备）。目前更新的CPU虽然通过集成存储控制器等手段弱化了北桥芯片的作用，但本质是不变的。CPU对于处理网络处理的能力体现在网卡接入链路层数据帧，然后通过操作系统（OS）内核态，发起DMA中断响应，调用相应的协议解析程序，获得网络传输的数据（虽然也有不通过内核态中断，直接在用户态通过轮询获得网络数据的技术，如Intel的DPDK，Xilinx的Onload等，但目的是降低中断的开销，降低内核态到用户态的切换开销，并没有从根本上增强IO性能）。可见，CPU是通过非常间接的手段来支持网络IO，CPU的前端总线带宽主要是要匹配主存（特别是DDR）的带宽，而不是网络IO的带宽。

相较而言，DPU的IO带宽几乎可以与网络带宽等同，例如，网络支持25G，那么DPU就要支持25G。从这个意义上讲，DPU继承了网卡芯片的一些特征，但是不同于网卡芯片，DPU不仅仅是为了解析链路层的数据帧，而是要做直接的数据内容的处理，进行复杂的计算。所以，DPU是在支持强IO基础上的具备强算力的芯片。简言之，DPU是一个IO密集型的芯片；相较而言，DPU还是一个计算密集型芯片。

进一步地，通过比较网络带宽的增长趋势和通用CPU性能增长趋势，能发现一个有趣的现象：带宽性能增速比（RBP，Ratio of Bandwidth and Performance growth rate）失调。RBP定义为网络带宽的增速比上CPU性能增速，即 $RBP = BW\ GR / Perf.\ GR$ 。如图1-1所示，以Mellanox的ConnectX系列网卡带宽作为网络IO的案例，以Intel的系列产品性能作为CPU的案例，定义一个新指标“带宽性能增速比”来反应趋势的变化。

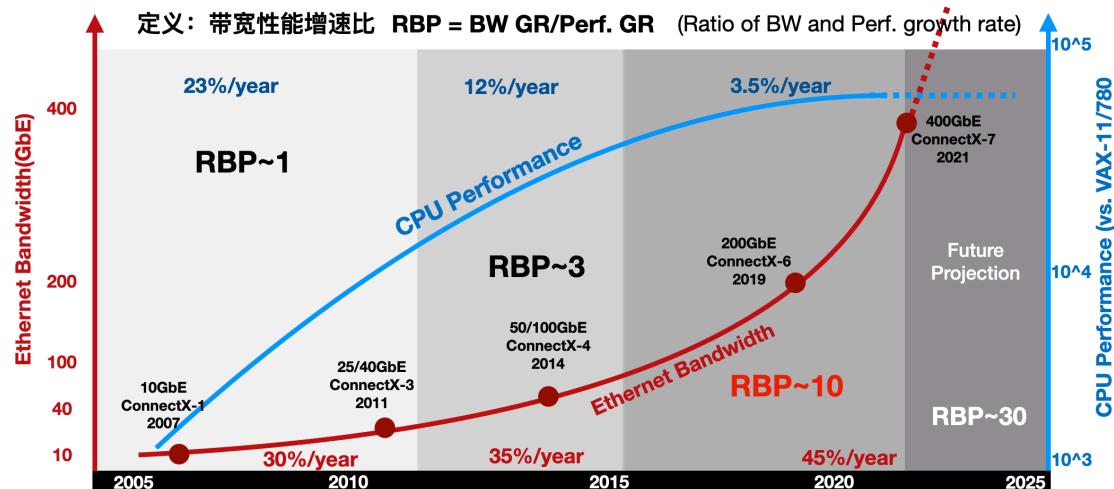


图1-1 带宽性能增速比 (RBP) 失调

2010年前，网络的带宽年化增长大约是30%，到2015年微增到35%，然后在近年达到45%。相对应的，CPU的性能增长从10年前的23%，下降到12%，并在近年直接降低到3%。在这三个时间段内，RBP指标从1附近，上升到3，并在近年超过了10！如果在网络带宽增速与CPU性能增速近乎持平， $RGR \sim 1$ ，IO压力尚未显现出来，那么当目前RBP达到10倍的情形下，CPU几乎已经无法直接应对网络带宽的增速。RBP指标在近几年剧增也许是DPU终于等到机会“横空出世”的重要原因之一。

### 1.2.2. 异构计算发展趋势的助力

DPU首先作为计算卸载的引擎，直接效果是给CPU“减负”。DPU的部分功能可以在早期的TOE (TCP/IP Offloading Engine) 中看到。正如其名，TOE就是将CPU的处理TCP协议的任务“卸载”到网卡上。传统的TCP软件处理方式虽然层次清晰，但也逐渐成为网络带宽和延迟的瓶颈。软件处理方式对CPU的占用，也影响了CPU处理其他应用的性能。TCP卸载引擎 (TOE) 技术，通过将TCP协议和IP协议的处理进程交由网络接口控制器进行处理，在利用硬件加速为网络时延和带宽带来提升的同时，显著降低了CPU处理协议的压力。具体有三个方面的优化：1) 隔离网络中断，2) 降低内存数据拷贝量，3) 协议解析

硬件化。这三个技术点逐渐发展成为现在数据平面计算的三个技术，也是DPU普遍需要支持的技术点。例如，NVMe协议，将中断策略替换为轮询策略，更充分的开发高速存储介质的带宽优势；DPDK采用用户态调用，开发“Kernel-bypassing”机制，实现零拷贝（Zero-Copy）；在DPU中的面向特定应用的专用核，例如各种复杂的校验和计算、数据包格式解析、查找表、IP安全（IPSec）的支持等，都可以视为协议处理的硬件化支持。所以，TOE基本可以被视为DPU的雏形。

延续TOE的思想，将更多的计算任务卸载至网卡侧来处理，促进了智能网卡（SmartNIC）技术的发展。常见的智能网卡的基本结构是以高速网卡为基本功能，外加一颗高性能的FPGA芯片作为计算的扩展，来实现用户自定义的计算逻辑，达到计算加速的目的。然而，这种“网卡+FPGA”的模式并没有将智能网卡变成一个绝对主流的计算设备，很多智能网卡产品被当作单纯的FPGA加速卡来使用，在利用FPGA优势的同时，也继承了所有FPGA的局限性。DPU是对现有的SmartNIC的一个整合，能看到很多以往SmartNIC的影子，但明显高于之前任何一个SmartNIC的定位。

Amazon的AWS在2013研发了Nitro产品，将数据中心开销（为虚机提供远程资源，加密解密，故障跟踪，安全策略等服务程序）全部放到专用加速器上执行。Nitro架构采用轻量化Hypervisor配合定制化的硬件，将虚拟机的计算（主要是CPU和内存）和I/O（主要是网络和存储）子系统分离开来，通过PCIe总线连接，节省了30%的CPU资源。阿里云提出的X-Dragon系统架构，核心是MOC卡，有比较丰富的对外接口，也包括了计算资源、存储资源和网络资源。MOC卡的核心X-Dragon SOC，统一支持网络，IO、存储和外设的虚拟化，为虚拟机、裸金属、容器云提供统一的资源池。

可见，DPU其实在行业内已经孕育已久，从早期的网络协议处理卸载，到后续的网络、存储、虚拟化卸载，其带来的作用还是非常显著的，只不过在此之前DPU“有实无名”，现在是时候迈上一个新的台阶了。

### 1.3. DPU发展历程

随着云平台虚拟化技术的发展，智能网卡的发展基本可以分为三个阶段（如图1-2所示）：

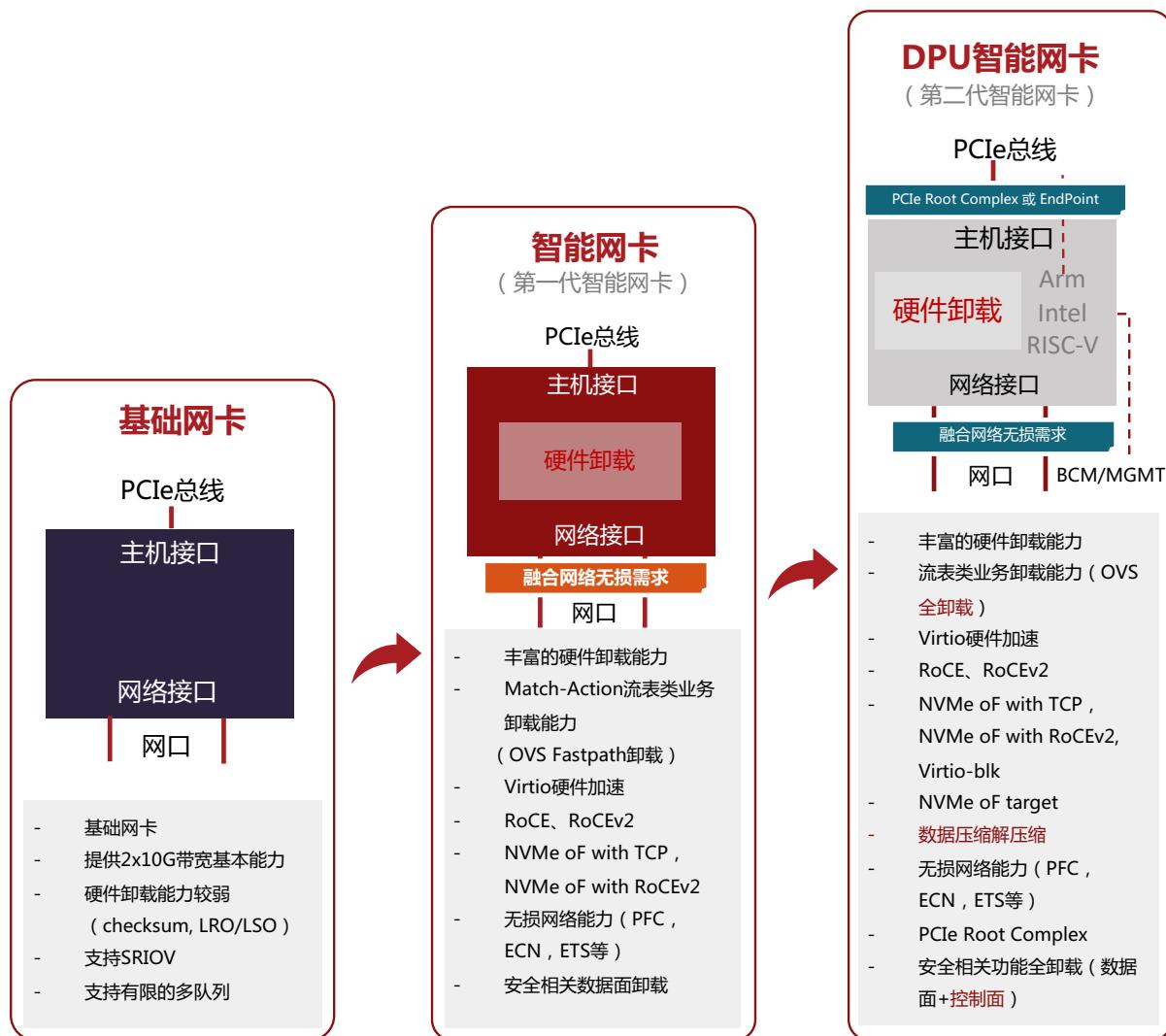


图1-2 智能网卡发展的三个阶段

## 第一阶段：基础功能网卡

基础功能网卡（即普通网卡）提供2x10G或2x25G带宽吞吐，具有较少的硬件卸载能力，主要是Checksum, LRO/LSO等，支持SR-IOV，以及有限的多队列能力。在云平台虚拟化网络中，基础功能网卡向虚拟机（VM）提供网络接入的方式主要是有三种：由操作系统内核驱动接管网卡并向虚拟机（VM）分发网络流量；由OVS-DPDK接管网卡并向虚拟机（VM）分发网络流量；以及高性能场景下通过SR-IOV的方式向虚拟机（VM）提供网络接入能力。

## 第二阶段：硬件卸载网卡

可以认为是第一代智能网卡，具有丰富的硬件卸载能力，比较典型的有OVS Fastpath硬件卸载，基于RoCEv1和RoCEv2的RDMA网络硬件卸载，融合网络中无损网络能力（PFC, ECN, ETS等）的硬件卸载，存储领域NVMe-oF的硬件卸载，以及安全传输的数据面卸载等。这个时期的智能网卡以数据平面的卸载为主。

## 第三阶段：DPU智能网卡

可以认为是第二代智能网卡，在第一代智能网卡基础上加入CPU，可以用来卸载控制平面的任务和一些灵活复杂的数据平面任务。目前DPU智能网卡的特点首先是支持PCIe Root Complex模式和Endpoint模式，在配置为PCIe Root Complex模式时，可以实现NVMe存储控制器，与NVMe SSD磁盘一起构建存储服务器；另外，由于大规模的数据中心网络的需要，对无损网络的要求更加严格，需要解决数据中心网络中Incast流量、“大象”流等带来的网络拥塞和时延问题，各大公有云厂商纷纷提出自己的应对方法，比如阿里云的高精度拥塞控制（HPCC, High Precision Congestion Control），AWS的可扩展可靠数据报（SRD, Scalable Reliable Datagram）等。DPU智能网卡在解决这类问题时将会引入更为先进的方法，如Fungible的TrueFabric，就是在DPU智能网卡上的新式解决方案。还有，业界提出了Hypervisor中的网络，存储和安全全栈卸载的发展方

向，以Intel为代表提出了IPU，将基础设施的功能全部卸载到智能网卡中，可以全面释放之前用于Hypervisor管理的CPU算力。

## 未来的DPU智能网卡硬件形态

随着越来越多的功能加入到智能网卡中，其功率将很难限制在75W之内，这样就需要独立的供电系统。所以，未来的智能网卡形态可能有三种形态：

(1) 独立供电的智能网卡，需要考虑网卡状态与计算服务之间低层信号识别，在计算系统启动的过程中或者启动之后，智能网卡是否已经是进入服务状态，这些都需要探索和解决。

(2) 没有PCIe接口的DPU智能网卡，可以组成DPU资源池，专门负责网络功能，例如负载均衡，访问控制，防火墙设备等。管理软件可以直接通过智能网卡管理接口定义对应的网络功能，并作为虚拟化网络功能集群提供对应网络能力，无需PCIe接口。

(3) 多PCIe接口，多网口的DPU芯片。例如Fungible F1芯片，支持16个双模PCIe控制器，可以配置为Root Complex模式或Endpoint模式，以及8x100G网络接口。通过PCIe Gen3 x8接口可以支撑8个Dual-Socket计算服务器，网络侧提供8x100G带宽的网口。

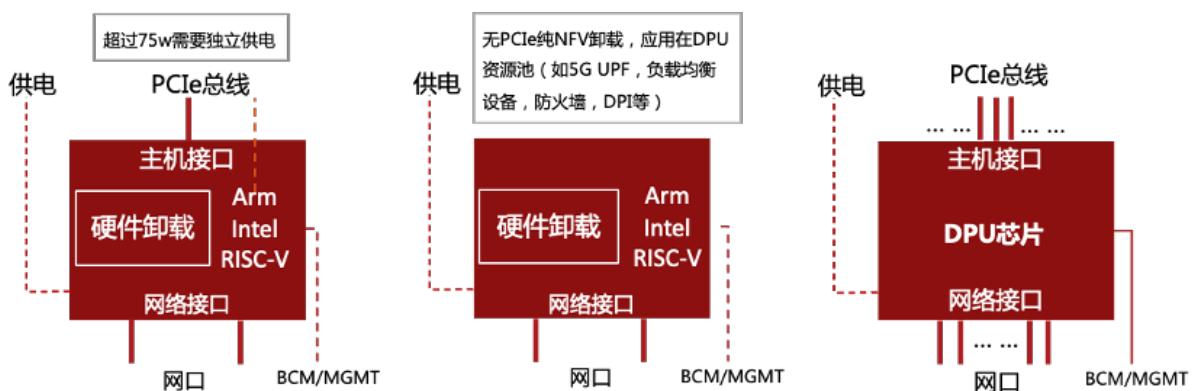


图1-3 未来智能网卡的硬件形态

DPU作为一种新型的专用处理器，随着需求侧的变化，必将在未来计算系统中成为一个重要组成部分，对于支撑下一代数据中心起到至关重要的作用。

## 1.4. DPU与CPU、GPU的关系

CPU是整个IT生态的定义者，无论是服务器端的x86还是移动端的ARM，都各自是构建了稳固的生态系统，不仅形成技术生态圈，还形成了闭合价值链。

GPU是执行规则计算的主力芯片，如图形渲染。经过NVIDIA对通用GPU(GPGPU)和CUDA编程框架的推广，GPU在数据并行的任务如图形图像、深度学习、矩阵运算等方面成为了主力算力引擎，并且成为了高性能计算最重要的辅助计算单元。2021年6月公布的Top500高性能计算机(超级计算机)的前10名中，有六台(第2、3、5、6、8、9名)都部署有NVIDIA的GPU。

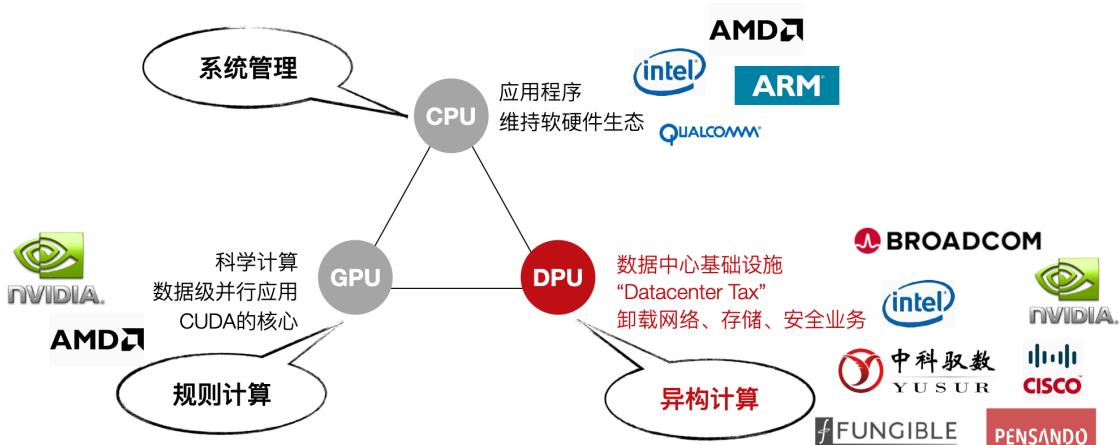


图1-4 未来算力生态（相关厂商为不完全列举，仅做为示意参考）

数据中心与超级计算机不同，后者主要面向科学计算，如大飞机研制，石油勘探、新药物研发、气象预报、电磁环境计算等应用，性能是主要指标，对接入带宽要求不高；但数据中心面向云计算商业化应用，对接入带宽，可靠性、灾备、弹性扩展等要求更高，与之相适应发展起来的虚拟机、容器云、并

行编程框、内容分发网等等技术，都是为了更好的支撑上层商业应用如电商、支付、视频流、网盘、办公OA等。但是这些IaaS和PaaS层的服务开销极大，Amazon曾公布AWS的系统开销在30%以上。如果需要实现更好的QoS，在网络、存储、安全等基础设施服务上的开销还会更高。

这些基础层应用类型与CPU架构匹配程度不高导致计算效率低下。现有的CPU的架构有两个大类：多核架构（数个或数十几个核）和众核架构（数百个核以上），每种架构支持唯一的规范通用指令集之一，如x86、ARM等。以指令集为界，软件和硬件被划分开来分别独立发展，迅速的催生了软件产业和微处理器产业的协同发展。但是，随着软件复杂度的上升，软件的生产率（Productivity）得到更多的重视，软件工程学科也更加关注如何高效地构建大型软件系统，而非如何用更少的硬件资源获得尽可能高的执行性能。业界有个被戏称的“安迪比尔定律”，其内容是“What Andy gives, Bill takes away”，安迪（Andy）指英特尔前CEO安迪·格鲁夫，比尔（Bill）指微软前任CEO比尔·盖茨，意为硬件提高的性能，很快被软件消耗掉了。

正如CPU在处理图像处理时不够高效一样，现在有大量的基础层应用CPU处理起来也比较低效，例如网络协议处理，交换路由计算，加密解密，数据压缩等这类计算密集的任务，还有支持分布式处理的数据一致性协议如RAFT等。这些数据或者通过从网络IO接入系统，或者通过板级高速PCIe总线接入系统，再通过共享主存经由DMA机制将数据提供给CPU或GPU来处理。既要处理大量的上层应用，又要维持底层软件的基础设施，还要处理各种特殊的IO类协议，复杂的计算任务让CPU不堪重负。

这些基础层负载给“异构计算”提供了一个广阔的发展空间。将这些基础层负载从CPU上卸载下来，短期内可以“提质增效”，长远来看还为新的业务增长提供技术保障。DPU将有望成为承接这些负载的代表性芯片，与CPU和GPU优势互补，建立起一个更加高效的算力平台。可以预测，用于数据中心的

DPU的量将达到和数据中心服务器等量的级别，每年千万级新增，算上存量的替代，估算五年总体的需求量将突破两亿颗，超过独立GPU卡的需求量。每台服务器可能没有GPU，但必须有DPU，好比每台服务器都必须配网卡一样。

## 1.5. DPU的产业化机遇

数据中心作为IT基础设施最重要的组成部分在过去10年成为了各大高端芯片厂商关注的焦点。各大厂商都将原有的产品和技术，用全新的DPU的理念重新封装后，推向了市场。

NVIDIA收购Mellanox后，凭借原有的ConnectX系列高速网卡技术，推出其BlueField系列DPU，成为DPU赛道的标杆。作为算法加速芯片头部厂商的Xilinx在2018年还将“数据中心优先（Datacenter First）”作为其全新发展战略。发布了Alveo系列加速卡产品，旨在大幅提升云端和本地数据中心服务器性能。2019年4月，Xilinx宣布收购Solarflare通信公司，将领先的FPGA、MPSoC和ACAP解决方案与Solarflare的超低时延网络接口卡（NIC）技术以及应用加速软件相结合，从而实现全新的融合SmartNIC解决方案。Intel 2015年底收购了Xilinx的竞争对手——Altera，在通用处理器的基础上，进一步完善硬件加速能力。Intel 2021年6月新发布的IPU产品（可以被视为Intel版本的DPU），将FPGA与Xeon D系列处理器集成，成为了DPU赛道有力的竞争者。IPU是具有强化的加速器和以太网连接的高级网络设备，它使用紧密耦合、专用的可编程内核加速和管理基础架构功能。IPU提供全面的基础架构分载，并可作为运行基础架构应用的主机的控制点，从而提供一层额外防护。几乎同一时间，Marvell发布了OCTEON 10 DPU产品，不仅具备强大的转发能力，还具有突出的AI处理能力。

在同一时期，一些传统并不涉足芯片设计的互联网厂商，如海外的Google、Amazon，国内的阿里巴巴等巨头纷纷启动了自研芯片的计划，而且研发重点都是面向数据处理器的高性能专用处理器芯片，希望以此改善云端的服

务器的成本结构，提高单位能耗的性能水平。数据研究预测DPU在云计算市场的应用需求最大，且市场规模随着云计算数据中心的迭代而增长，到2025年单中国的市场容量都将达到40亿美元的规模<sup>2</sup>。

---

<sup>2</sup> 中国数据处理器行业概览系列短报告(二): 数据处理器在数通市场的应用, 头豹行业研读 | 2021/06

## 2. DPU特征结构

### 2.1. DPU是以数据为中心IO密集的专用处理器<sup>3</sup>

从应用特征来看，可以把应用分为“IO密集型”和“计算密集型”两类，如图2-1纵轴所示。IO密集型应用，通常体现为较高的输入和输出带宽，数据直接来自于IO，数据通常具备流式特征，数据局部性不显著，如果处理性能与带宽匹配，片上缓存的作用就可以弱化。例如处理路由转发、数据加密、压缩等。计算密集型应用，体现为较高的计算密度，通常浮点性能突出，数据来自主存，数据局部性显著，复用性高，主存的大小对于问题求解的性能有直接影响。例如求解线性代数方程组，大规模神经网络训练、推理等。

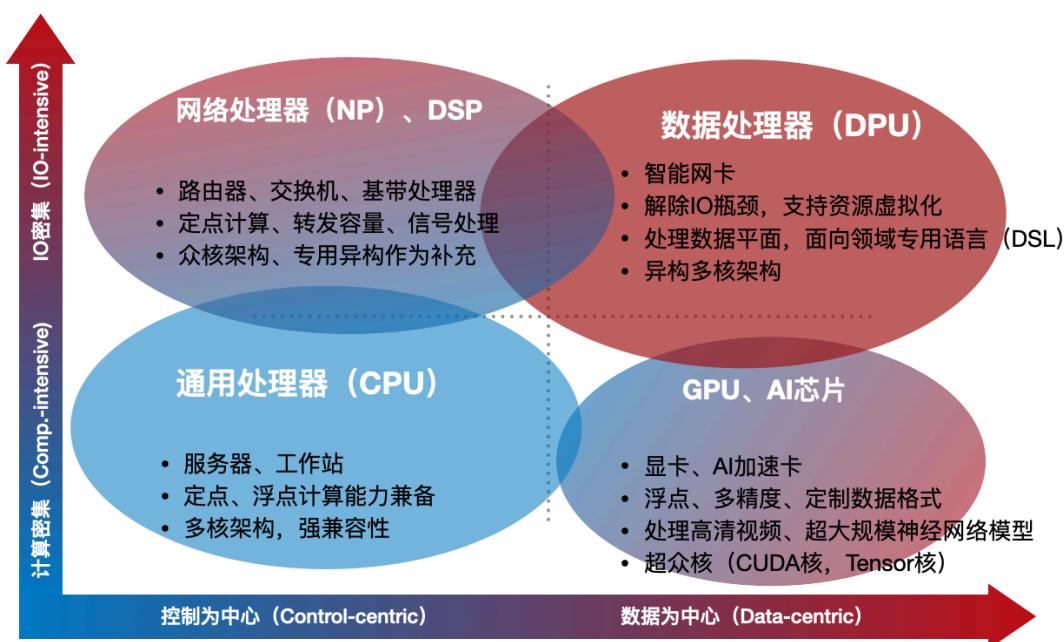


图2-1 不同类型的处理器的特征结构

一个处理器芯片是“IO密集”还是“计算密集”只部分决定了芯片的结构特征，并不能完全定义芯片的主体架构。无论是IO密集，还是计算密集，即可以以通用CPU为核心构造主体计算架构，也可以以专用加速器为核心构造主体

<sup>3</sup> “DPU：以数据为中心的专用处理器”，鄢贵海等，中国计算机学会通讯，2021年，第17卷第10期

计算架构。前者可称之为以控制为中心（control-centric）的模式，后者称之为以数据为中心的模式（data-centric）的模式。控制为中心的核心是实现“通用”，数据为中心的核心是通过定制化实现“高性能”。以应用特征和架构特征这两个维度粗略划分处理器芯片类型分布，如图2-1所示。

通用CPU是偏向于控制为中心结构，理论上看就是要“图灵完备”，要支持完备的指令集，通过编程指令序列来定义计算任务，通过执行指令序列来完成计算任务，因此具备极其灵活的编程支持，可以任意定义计算的逻辑实现“通用”——这也是CPU最大的优势。同时，为了提高编程的开发效率，降低编译器复杂度，缓存管理和细粒度并行度的开发通常都是由硬件来完成。类似的，还有大量的用于各种嵌入式、移动设备的微控制器MCU，并不强调高带宽，也是以控制为中心的结构。NP，DSP也是偏向于基于通用处理器来做专用化扩展，但是非常注重高通量的性能属性。例如，NP要支持数Tbps的转发带宽，所以大体可以视为控制为中心、但是IO密集的处理器类型。

GPU是以数据为中心的结构，形式上更倾向于专用加速器。GPU的结构称之为数据并行（data-parallel）结构，优化指令并行度并不是提升性能的重点，通过大规模同构核进行细粒度并行来消化大的数据带宽才是重点。例如，最新的NVIDIA TITAN RTX GPU有4608个CUDA核、576个Tensor核，而且单片GPU通常配置数十GB的超大显存。同时缓存管理多采用软件显示管理，降低硬件复杂度。这类超众核结构是以数据为中心、执行计算密集型任务的代表性架构。

DPU也偏向于数据为中心的结构，形式上集成了更多类别的专用加速器，让渡一定的指令灵活性以获得更极致的性能。但是与GPU不同，DPU要应对更多的网络IO，既包括外部以太网，也包括内部虚拟IO，所以DPU所面临的数据并行更多可能是数据包并行，而不是图像中的像素、像块级并行。而且DPU也会配置少数通用核（如ARM，MIPS）来处理一定的控制面的任务，运行轻量级操作系统来管理DPU上的众多的异构核资源，所以体现了一定“通用”性，但

性能优势主要不源于这些通用核，而是大量专用计算核。早期的一些网络处理器采用过类似Tile64的通用众核结构，以增加核的数量来应对多路处理的数据，实现并发处理，但单路延迟性能通常都比较差。因此，DPU更偏向于以数据为中心，执行IO密集任务。

DPU是软件定义的技术路线下的重要产物。在软件定义网络中，将数据面与控制面分离是最核心的思想。DPU被定义为强化了数据面性能的专用处理器，配合控制面的CPU，可以实现性能与通用性的更佳的平衡。

## 2.2. 计算平台从“计算为中心”转向“数据为中心”

### 2.2.1. 大规模云计算网络面对的挑战

(1) 网络拥塞和长尾延迟。数据中心网络最真实的性能指标之一是“尾部延迟”，即流量最慢百分位数所经历的延时水平。数据中心网络拥塞的一个典型原因是Incast流量，而Incast流量在数据中心网络中是不可避免的。现在常见的解决拥塞的方法，如基于优先级的流量控制（PFC），显示拥塞通知（ECN），数据中心量化拥塞通知（DCQCN）和数据中心TCP连接等，这些方案并没有完全解决拥塞问题，如突发流量导致的拥塞，同时也带来一系列新的挑战，如PFC pause帧风暴等。

(2) 负载不均衡。在数据中心网络中等价多路径（ECMP）路由是一种典型的流量转发负载均衡方式，它能够保证同一数据流跨网络Fabric转发时根据特定元组信息选择的路径相同，不同数据流选择的路径根据哈希的结果选择不同路径，从而实现不同流量之间的负载均衡和更高的带宽利用率。但是，当数据中心主要流量是“大象”流和“老鼠”流时，ECMP负载均衡的数据转发方式就可能存在问题是。如大数据处理和存储数据同步等工作负载需在数据中心网络中创建大量持续的数据流，这些数据流与对时延相对敏感的短期流共存。多个

“大象”流的哈希冲突会导致数据中心网络中某些路径上流量出现不均甚至拥塞，从而导致拥塞路径上的“老鼠”流受影响出现更长延迟和丢包。

(3) 故障检测和恢复效率。大规模数据中心通常是通过冗余备份来实现网络高可靠性，即当某一网络路径出现故障时，通过协议或者网络端口状态自动将故障路径上的业务流量切换到其它正常路径上。在数据中心这种Spine-Leaf的网络架构下，服务器的网卡如何能够快速检测到故障，进而规避流量通过故障路径，决定了数据中心业务的恢复效率。

(4) 不可靠传输。像RoCEv2这样的网络协议是基于网络无损的假设而设计的。然而在数据中心网络中，丢包是不可避免的。目前大部分解决方案都是通过在网络设备上实现复杂的协议或者遥测能力，来加固网络或者识别数据中心的网络状况，进而避免应用程序吞吐量和性能下降。如果数据包重传发生在传输层，这可能会显著提高性能。不幸的是UDP缺乏重传机制，而TCP又过于复杂，头部拥塞和尾部延迟很难预测。

### 2.2.2. 数据为中心的系统架构特征

数据中心是云计算时代最重要的基础设施之一，为云提供强大的计算和存储能力。近年来业界对数据中心网络性能以及虚拟化能力的需求日益增长，并伴随着后摩尔时代的到来，传统的以CPU为中心的计算机体系结构，其计算能力已经无法支撑网络带宽的增长速度，会造成网络拥塞，以单节点能力为上限的资源分配方式无法满足云资源需求，运维成本和影响也日益增加，主要表现在如下几个方面：

- 以计算为中心的数据中心，所有数据都需要先被送到CPU。CPU的计算包含了应用的计算，通信的计算和存储的计算等，每一类计算都需要独占CPU资源。CPU已然成为数据中心释放算力的瓶颈。

- 以CPU为中心的计算机体系结构，通过内部系统总线的通信方式，单节点服务器上的总线能力和PCIe插槽的数量是固定的，导致单节点上支持的专用算力芯片如GPU和FPGA加速卡等也受限于总线能力和PCIe插槽数量。
- 以CPU为中心的计算机体系结构，单节点的专用算力资源如GPU和FPGA加速卡等无法被其它节点使用，这也导致了专用算力资源无法按需扩展，预留专用算力资源浪费的问题，使得虚拟化用户层的资源管理异常困难。
- 在运维方面，由于单节点上资源是耦合的，故障维修需对节点断电，使得整个节点对外不可用，业务的高可靠性难以保障，维修流程非常复杂。

鉴于当前以计算为中心的数据中心体系中所存在的局限，一个宏观趋势是数据中心的体系架构将会从以计算为中心转向以数据为中心。后者相对于前者具有显著的优势。以CPU为中心的通信场景里，由于数据转发都需要经过CPU（如虚拟机场景——CPU收包转给CPU，大数据场景——CPU收包转给GPU），CPU的处理时延通常会比较高，约30至40微秒，另外，由于CPU数据转发能力和CPU数量的限制，会在数据通道上成为系统的瓶颈，使得虚拟机CPU或者GPU的算力无法释放，甚至在触及性能瓶颈时出现严重的丢包现象，进一步恶化网络传输。以数据为中心的架构，数据转发通道不再需要CPU参与，DPU智能网卡可以直接将数据送达GPU和虚拟机CPU，不需要基于CPU的软件转发，时延上可以提升到3至4微妙，典型的通信时延可以从30至40微妙降低到3至4微妙，有近10倍的性能提升。另外，由于DPU智能网卡硬件线速转发，避免了性能瓶颈和丢包。

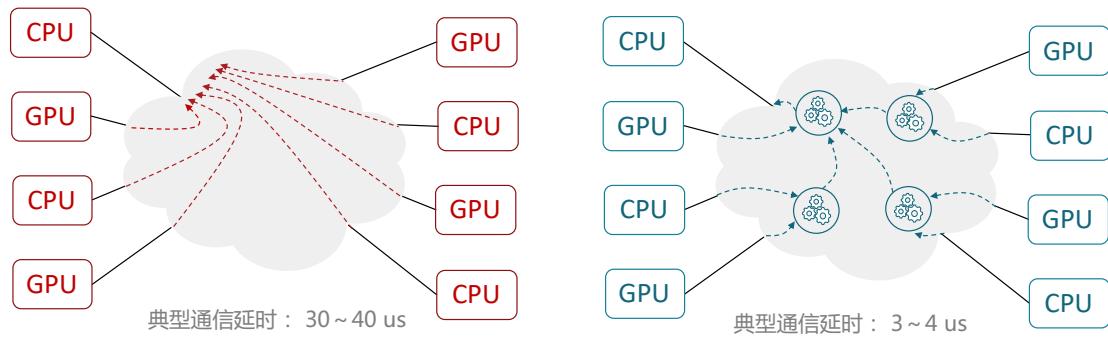


图2-2 数据中心网络通信延时

以数据为中心的数据中心架构，至少具备以下几个特点：

- (1) 分散 (Disaggregated) 的计算架构。计算结构内部资源不是紧耦合。通过DPU智能网卡提供的网络能力，对系统总线数据做转换传输、延伸和分发，使得CPU、GPU、存储和加速器等可以在不同的位置或物理机上，通过数据中心网络对数据分发和调度，使得各资源互相协同工作。
- (2) 虚拟化资源池 (Resource Pooling) 。以降低TCO为目的，对CPU、GPU、存储和加速器等资源按需分配，同时降低资源维护和管理调度的复杂度，集中的资源池化势在必行。同时，通过虚拟化资源池的方式向用户提供各种资源，利用资源伸缩性，使得提供给用户侧的资源上限更大。
- (3) 可伸缩性 (Scalability) 。资源池化后，资源的维护和管理更容易，整体资源库存情况更清晰，从数据中心运营的角度，库存的扩展也更容易。
- (4) 可组合性 (Composability) 。数据中心的网络能力使得服务资源的接入更加灵活。对于新加入资源池的资源（如专用FPGA加速器），可以通过对DPU智能网卡做软件定义网络，打通数据中心的数据传输通路，让新加入的资源加入到云计算资源系统中。

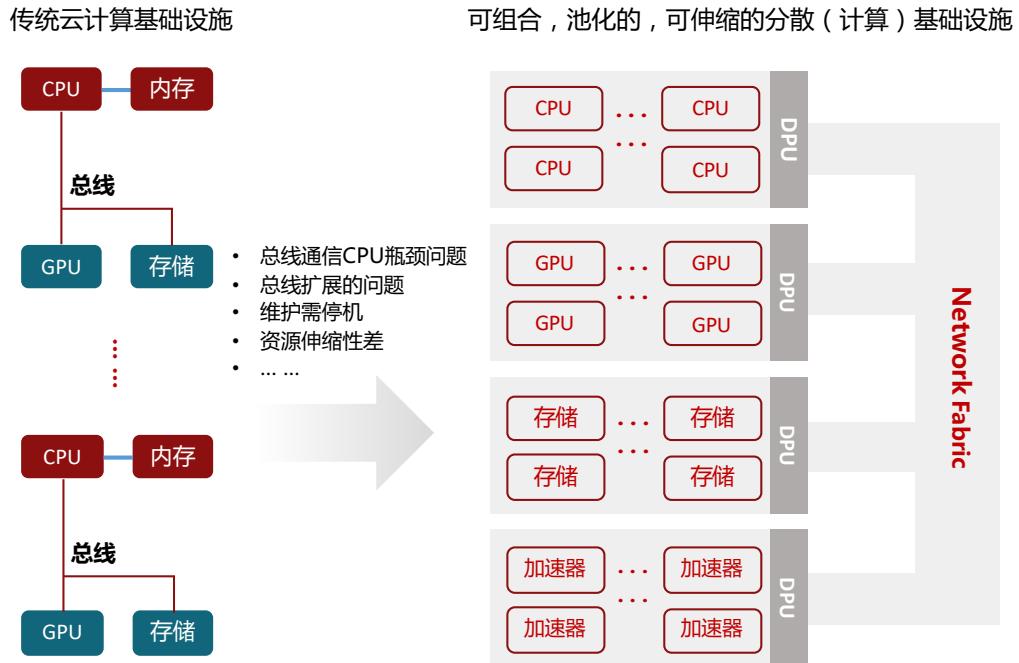


图2-3 传统云计算基础设施与以数据为中心的数据中心架构对比

### 2.3. 一种DPU参考设计

为了满足“数据为中心”的设计理念，本节给出一个通用的DPU参考设计。目前DPU架构的演化比较快，DPU既可以呈现为一个被动设备作为CPU的协处理器，也可以作为一个主动设备，承接Hypervisor的一些功能。尤其是容器技术、虚拟化技术的广泛采用，DPU的角色已经不仅仅是一个协处理器，而是呈现出更多的HOST的特征，比如运行Hypervisor，做跨节点的资源整合，为裸金属虚拟机提供虚拟网络，数据安全，热迁移等支撑。宏观来看，DPU架构至少可以分为以下几个核心组成部分：

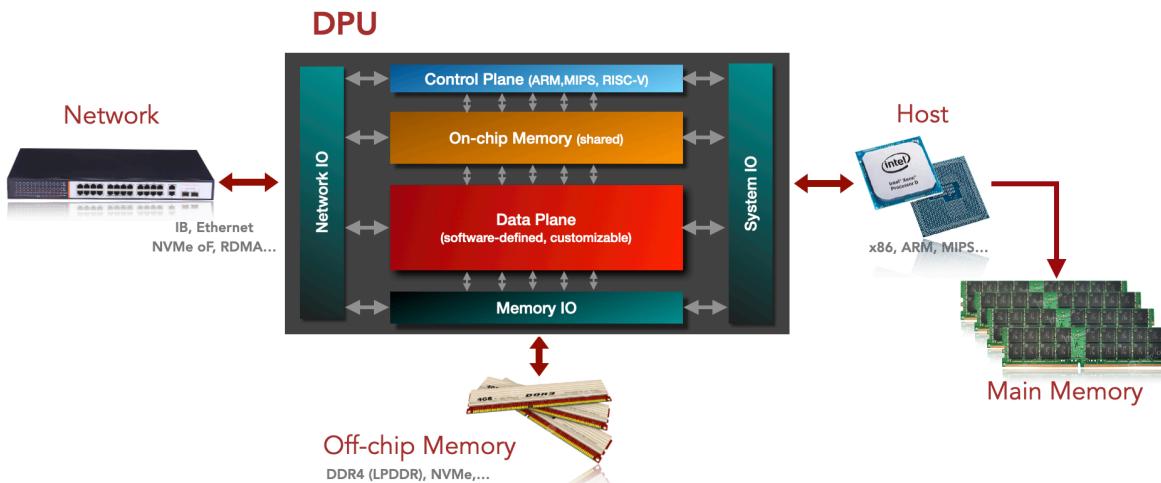


图2-4 DPU架构参考设计

### 2.3.1. 控制平面

负责管理、配置，通常由通用处理器核来实现。控制平台负责DPU设备运行管理，以及计算任务和计算资源配置。运行管理通常包含设备的安全管理和实时监控两个主要功能。在安全管理方面支持支持如信任根、安全启动、安全固件升级以及基于身份验证的容器和应用的生命周期管理等。在设备实时监控方面，对DPU各子系统、数据平面中各处理核动态监测，实时观察设备是否可用、设备中流量是否正常，周期性生成报表，记录设备访问日志和配置修改日志。

计算任务和计算资源配置方面，根据计算任务实施配置数据平面中处理单元间的通路，以及各处理单元参数。根据资源利用情况实时进行任务调度以及在计算单元的映射和部署。同时DPU上层同时会支持多个虚拟机，控制平面在任务部署时还要进行虚拟设备的管理，考虑虚拟机间数据和资源隔离，记录运行状态协助虚拟机热迁移。最后，当DPU集成第三方计算平台，如GPU、FPGA等，还需要参与部分卸载任务调度。

由于控制平面任务多样，灵活性要求较高，算力要求较低，通常由通用处理器核来实现，比如ARM、MIPS等核心。为便于用户统一管理和配置DPU设

备，提供较好的可编程性，通常会运行标准Linux应用程序。并且控制平面与数据平面数据交互驱动程序需要进行深度优化，来提升控制平面与数据平面有效地交互，任务调度效率。

### 2.3.2.IO子系统

主要分为三个大类：

(1) 系统IO，负责DPU和其他处理平台（如X86、ARM处理器、GPU、FPGA等）或高速外部设备（如SSD）的集成。系统IO通常传输数据量较大对带宽有着极高的要求，因此多基于PCIe来实现。系统IO接口分为两大类：EP (Endpoint) 类和RC (Root Complex) 类。

EP类接口负责将DPU作为从设备与X86、ARM等处理平台相连接。为了充分利用DPU上的内部资源，此类接口要支持强大的硬件设备虚拟化功能，比如SR-IOV和VirtIO。并且可以灵活地支持多种类型的设备，如NIC、Storage、Compute设备等。

RC类接口负责将DPU作为主设备与加速平台（如GPU、FPGA）或外设（SSD）相连接。通过此种方式将部分数据处理卸载到第三方加速平台GPU、FPGA中处理，通常数据量较大，需要支持较强的DMA方案。

(2) 网络IO，负责DPU与高速网络相连接，主要是以太网或者FC为主。为了能应对急剧增加的网络带宽，DPU中通常辅以专门的网络协议处理核来加速网络包的处理。包括L2/L3/L4层的ARP/IP/TCP/UDP网络协议处理、RDMA、数据包交换协议、基本网络虚拟化协议等，可以实现100G以上的网络包线速处理。

(3) 主存IO，负责缓存网络IO和系统IO输入输出数据，以及数据平面中间数据结果。也可作为共享内存，实现不同处理核之间的数据通信。目前主存IO主要包含DDR和HBM接口类型，两类接口，DDR可以提供比较大的存储容

量，可以提供512GB以上的存储容量；HBM可以提供比较大的存储带宽，可以提供500GB/s以上的带宽。两种存储接口相结合可以满足不同存储容量和带宽的需求，但是需要精细的数据管理，这块也是DPU设计中比较有挑战的。

### 2.3.3.数据平面

主要负责高速数据通路的功能单元的集成，通常集成多个处理核。数据平面的功能主要分为五类：

- 1) 高速数据包处理，主要对接收到的网络数据包进行如OvS（开放式虚拟交换机）解析、匹配和处理，以及RDMA远程数据传输加速等操作，和之前的网络处理器NP功能类似，但是在性能上有更高的要求，处理带宽数线速要达到100G、200G甚至400G。同时，在进行有状态数据处理时也有着更高的要求，如TCP协议，要求硬件记录各连接信息，并能实现多连接间无缝切换。
- 2) 虚拟化协议加速，支持SR-IOV、VirtIO 和PV(Para-Virtualization)等虚拟化。支持网络虚拟化VxLAN、Geneve Overlay卸载和VTEP等协议卸载。
- 3) 安全加密，在线IPSec和TLS加密加速，以及多种标准加解密算法和国密算法。并且对于安全算法的处理性能有较高的要求，要达到网络线速，从而不影响其它正在运行的加速操作。
- 4) 流量压缩，对网络数据包，或者要存储的数据，进行实时地数据压缩/解压缩处理，压缩过程中还要完成地址的转换和重映射等操作。或者在线完成数据流变换处理，如面向多媒体流、CDN(内容分发网络)和4K/8K IP视频的“Packet Pacing”流量整形加速等。
- 5) 其他算法加速。除了上述网络、安全协议外还要支持NVMe等存储协议，业务相关的处理卸载也呈增长趋势，如大数据分析SQL加速。

### 2.3.4. DPU设计的关键

数据平面是整个DPU设计的关键，也是DPU设计中最有挑战的模块。主要面临四个挑战：

- 1) 数据中心的工作负载复杂多样，数据平面支持的处理核种类要足够多，不仅包括网络、存储、安全和虚拟化等基础设施服务，另外业务相关的处理也在加速向DPU平台卸载。
- 2) 高并发性数据处理，数据中心承载的业务多且复杂，多虚拟机多种类业务并发要求数据平面集成足够数量的核心，规模要达到几百个核心规模。随着数据中心数据量的不断增加，对处理性能提出越来越多的挑战，DPU数据平面在处理核规模上要具有非常强的可扩展性。
- 3) 复杂的片上互联系统，随着DPU数据平面处理核数量的增加，再加之高并发处理线程运行，同时还要兼顾好数据平面数据处理的灵活，这就要求处理核之间的数据交互既要灵活又要兼顾高带宽。处理核之间的数据互联，以及核间的数据一致性成为另一设计难题。
- 4) 高效简易的编程方式，数据中心业务的复杂多变决定了DPU数据平台可编程性的硬性需求。一方面要兼顾计算效率，必须直观表达出并发处理任务，充分利用计算资源。另一方面要兼顾DPU的易用性，尽量采用高级语言进行编程，易于设计、开发和维护。

总之，DPU数据平面需要一种大规模敏捷异构的计算架构。这一部分的实现也处在“百家争鸣”的阶段，各家的实现方式差别较大，有基于通用处理器核的方式，有基于可编程门阵列FPGA的方式，也有基于异构众核的方式，还有待探索。

## 2.4. DPU具备的主要功能

- (1) 算力卸载。将存储协议和安全协议的封装与解封装等不适合CPU做的事物卸载到智能网卡或DPU上，节约更多CPU算力支撑更多的应用业务。
- (2) 虚拟网络控制面隔离。DPU需要独立的网络控制平面，从主机中完全隔离，可以服务于各种资源池能力的组合和伸缩。
- (3) 主机侧总线通信的延展。在分散的计算架构中，连接不同资源池的方式将从原来的系统总线承载，转变成总线——网络——总线的方式，DPU不仅需要提供PCIe Endpoint的能力，还需要提供PCIe Root Complex的能力。
- (4) 网络侧无损网络传输。网络侧提供针对Incast流量模型和“大象”流影响的拥塞控制机制和增强的负载均衡能力，降低长尾时延，提供更可靠更高效的传输网络。
- (5) 精细化的测量和故障检测能力。精细化的流量测量支撑精细化的故障检测能力，让流量数据更透明。

### 3. DPU应用场景

#### 3.1. 应用场景一：网络功能卸载

网络功能卸载是伴随云计算网络而产生的，主要是对云计算主机上的虚拟交换机的能力做硬件卸载，从而减少主机上消耗在网络上的CPU算力，提高可售卖计算资源。

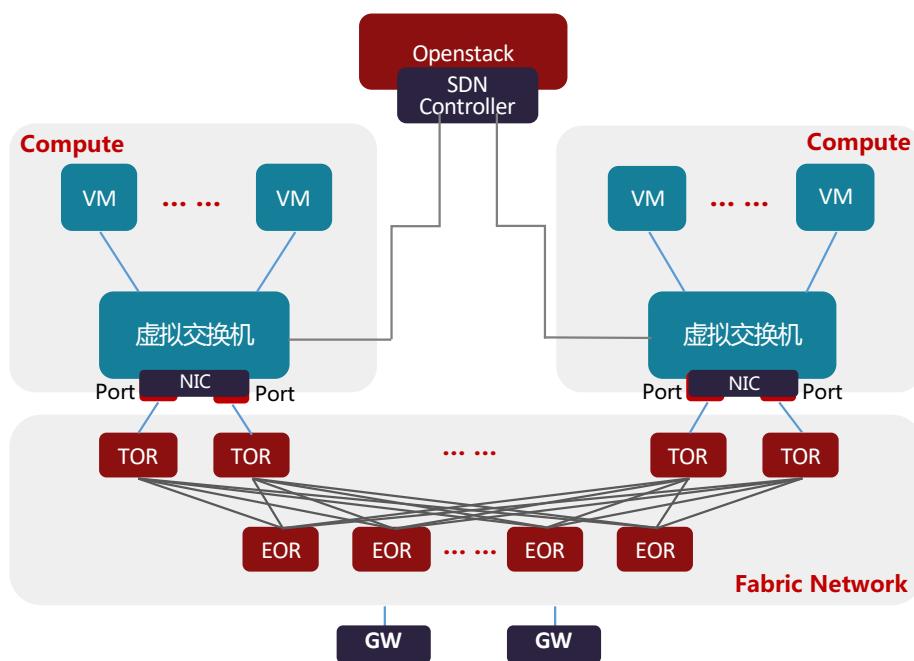


图3-1 云计算网络架构

目前除了公有云大厂采用自研云平台，绝大部分私有云厂商都使用开源的OpenStack云平台生态。在OpenStack云平台中，虚拟交换机通常是Open vSwitch，承担着云计算中网络虚拟化的主要工作，负责虚拟机（VM）与同主机上虚拟机（VM）、虚拟机（VM）与其它主机上虚拟机（VM）、虚拟机（VM）与外部的网络通信。虚拟交换机与网关路由器（GW）通常由同一SDN控制器来管理控制，为租户开通VPC网络以及和外部通信的网络。主机与主机间的网络通常是Underlay网络，是由TOR/EOR构建的Spine-Leaf结构的Fabric Network。虚拟机（VM）与虚拟机（VM）通信的网络是Overlay网络，是承载在

Underlay网络构建的VxLAN, NVGRE或Geneve隧道之上的。通常VxLAN, NVGRE或Geneve的隧道端点（VTEP）在虚拟交换机和网关路由器（GW）上。也有部分对网络性能要求比较高的场景，采用SR-IOV替代虚拟交换机，VF直通到虚拟机（VM）内部，这样就要求隧道端点（VTEP）部署在TOR上，TOR与网关路由器（GW）创建隧道，提供Overlay网络服务。虚拟交换机的场景是最通用的应用场景，所以，虚拟交换机的技术迭代也直接影响着虚拟化网络的发展。

### 3.1.1. 虚拟化网络功能（Virtual Network Function）

行业内主流的Hypervisor主要有Linux系统下的KVM-Qemu, VMWare的ESXi, 微软Azure的Hyper-V, 以及亚马逊早期用的Xen（现在亚马逊已经转向KVM-Qemu）。KVM-Qemu有着以Redhat为首在持续推动的更好的开源生态，目前行业内90%以上的云厂商都在用KVM-Qemu作为虚拟化的基础平台。

在KVM-Qemu这个Hypervisor的开源生态里，与网络关系最紧密的标准协议包括virtio和vhost, 以及vhost衍生出来的vhost-vdpa。Virtio在KVM-Qemu中定义了一组虚拟化I/O设备，和对应设备的共享内存的通信方法，配合后端协议vhost和vhost-vdpa使用，使虚拟化I/O性能得到提升。

#### （1）内核虚拟化网络（vhost-net）

在虚拟化网络的初期，以打通虚拟机（VM）间和与外部通信能力为主，对功能诉求远高于性能，虚拟交换机OVS（Open vSwitch）的最初版本也是基于操作系统Linux内核转发来实现的。vhost协议作为控制平面，由Qemu做代理与主机kernel内的vhost-net通信，主机内核vhost-net作为virtio的backend，与虚拟机（VM）内部的virtio NIC通过共享内存的方式进行通信。实现了虚拟机（VM）间和虚拟机（VM）与外部的通信能力，但是内核转发效率和吞吐量都很低。目前在虚拟化网络部署中已经很少使用。

## (2) 用户空间DPDK虚拟化网络 (vhost-user)

随着虚拟化网络的发展，虚拟机（VM）业务对网络带宽的要求越来越高，另外，英特尔和Linux基金会推出了DPDK（Data Plane Development Kit）开源项目，实现了用户空间直接从网卡收发数据报文并进行多核快速处理的开发库，虚拟交换机OVS将数据转发平面通过DPDK支持了用户空间的数据转发，进而实现了转发带宽量级的提升。OVS-DPDK通过将virtio的backend实现在用户空间，实现了虚拟机（VM）与用户空间OVS-DPDK的共享内存，这样虚拟机（VM）在收发报文时，只需要OVS-DPDK将从网卡收到的报文数据写入虚拟机（VM）的内存，或从虚拟机（VM）内存将要发送的报文拷贝到网卡DMA的内存，由于减少了内存拷贝次数和CPU调度干扰，提升了转发通道的整体效率和吞吐量。

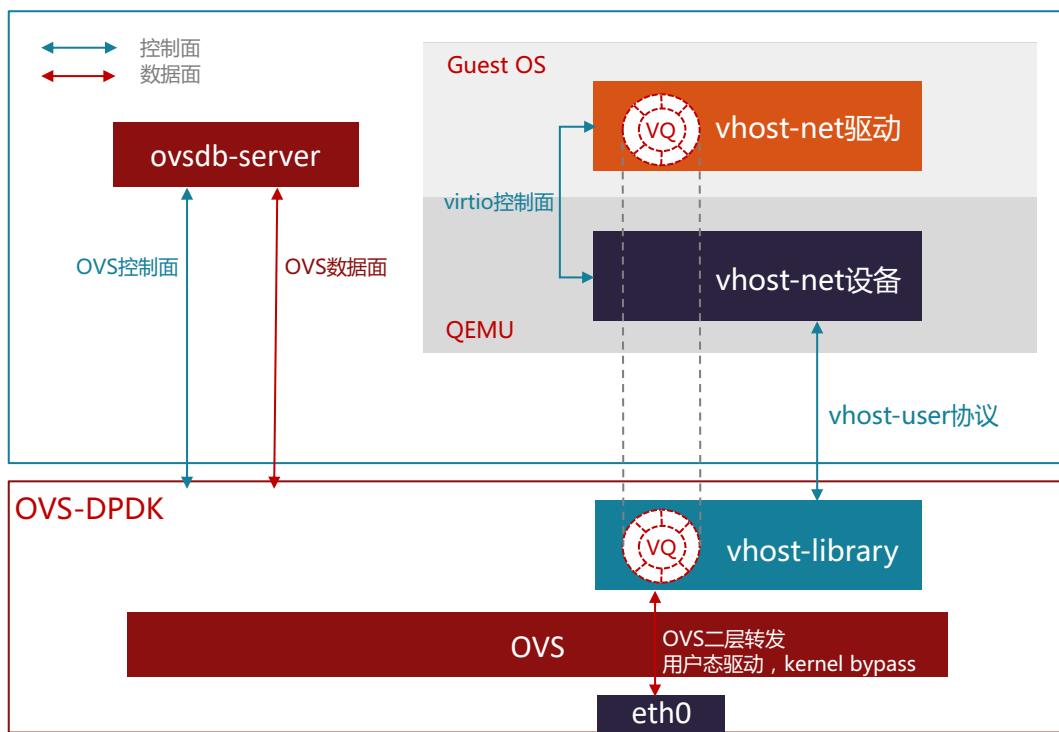


图3-2 基于DPDK的虚拟化网<sup>4</sup>

<sup>4</sup> A journey to the vhost-users realm <https://www.redhat.com/en/blog/journey-vhost-users-realm>

OVS-DPDK相对转发能力有所提高，但也存在新的问题。首先，目前大多数服务器都是NUMA（多CPU）结构，在跨NUMA转发时性能要比同NUMA转发弱。而物理网卡只能插在一个PCIe插槽上，这个插槽只会与一个NUMA存在亲和性，所以OVS-DPDK上跨NUMA转发的流量不可避免。第二，在虚拟机（VM）与OVS-DPDK共享内存时，初始化的队列数量通常是与虚拟机（VM）的CPU个数相同，才能保证虚拟机上每一个CPU都可以通过共享内存收发数据包，这样导致不同规格的虚拟机（VM）在OVS-DPDK上收发队列所接入的CPU是非对称的，在转发过程中需要跨CPU转发数据。最后，OVS-DPDK在转发数据时，不同虚拟机（VM）的流量由于CPU瓶颈导致拥塞，会随机丢包，无法保障租户带宽隔离。

### (3) 高性能SR-IOV网络 (SR-IOV)

在一些对网络有高性能需求的场景，如NFV业务部署，OVS-DPDK的数据转发方式，无法满足高性能网络的需求，这样就引入的SR-IOV透传(passthrough)到虚拟机(VM)的部署场景。

在SRIOV passthrough的场景下，虚拟机(VM)可以获得与裸金属主机上相拟的网络性能。但是，仍然存在两个限制：

(1) SRIOV VF passthrough到VM后，VM的迁移性会受限，主要原因在于SRIOV这种passthrough I/O借助了Intel CPU VT-d (Virtualization Technology for Directed I/O) 或AMD的IOMMU (I/O Memory Management Unit) 技术，在VM上VF网卡初始化的时候，建立了Guest虚拟地址到Host物理地址的映射表，所以这种“有状态”的映射表在热迁移的过程中会丢失。

(2) 由于SRIOV VF passthrough到VM，而SRIOV PF直接连接到TOR上，在这种部署环境中虚拟机(VM)对外的网络需要自定义，如需要像OVS-DPDK那样自动开通网络，则需要将TOR加入SDN控制器的管理范畴，由SDN控制器统一管控，这样做通常会使网络运营变的非常复杂。

针对上面第二个问题，Mellanox最早提出在其智能网卡上支持OVS Fastpath硬件卸载，结合SRIOV VF passthrough到VM一起使用，提供临近线速转发的网络能力，解决了虚拟机（VM）租户网络自动化编排开通的问题。

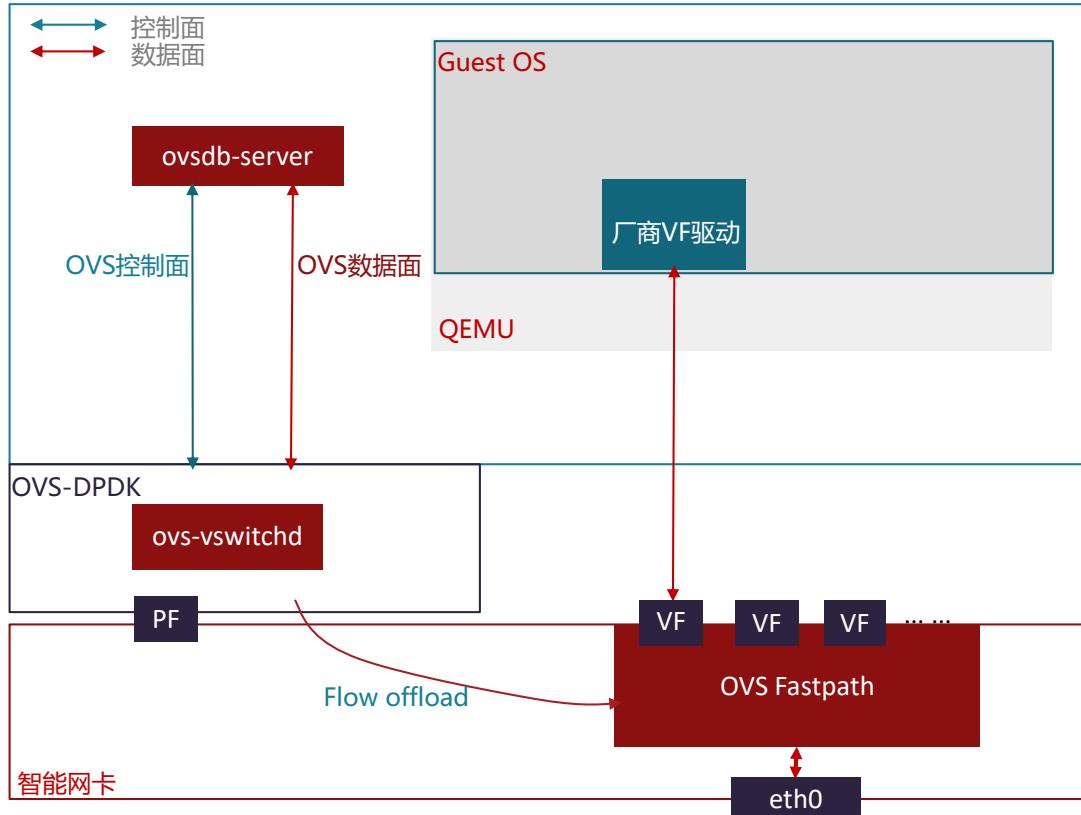


图3-3 OVS Fastpath硬件卸载虚拟化网络

在OVS Fastpath卸载后，OVS转发报文时，数据流首包仍然做软件转发，在转发过程中生成Fastpath转发流表并配置到硬件网卡上，这个数据流的后续报文则通过硬件直接转发给虚拟机（VM）。由于早期的Mellanox智能网卡还没有集成通用CPU核，OVS的控制面依然在物理主机上运行。

#### (4) Virtio硬件加速虚拟化网络 (vDPA)

为了解决高性能SRIOV网络的热迁移问题，出现了很多做法和尝试，尚未形成统一的标准。在Redhat提出硬件vDPA架构之前，Mellanox实现了软件vDPA（即VF Relay）。理论上讲，Mellanox的软件vDPA并不能算是vDPA，其实就是将数据在用户空间的virtio队列和VF的接收队列做了一次数据Relay。Redhat提出

的硬件vDPA架构，目前在DPDK和内核程序中均有实现，基本是未来的标准架构。Qemu支持两种方式的vDPA，一种是vhost-user，配合DPDK中的vDPA运行，DPDK再调用厂商用户态vDPA驱动；另一种方式是vhost-vdpa，通过ioctl调用到内核通用vDPA模块，通用vDPA模块再调用厂商硬件专有的vDPA驱动。

1) 软件vDPA：软件vDPA也叫VF relay，由于需要软件把VF上接收的数据通过virtio转发给虚拟机（VM），如Mellanox在OVS-DPDK实现了这个relay，OVS流表由硬件卸载加速，性能上与SR-IOV VF直通（passthrough）方式比略有降低，不过实现了虚拟机（VM）的热迁移特性。

2) 硬件vDPA：硬件vDPA实际上是借助virtio硬件加速，以实现更高性能的通信。由于控制面复杂，所以用硬件难以实现。厂商自己开发驱动，对接到用户空间DPDK的vDPA和内核vDPA架构上，可以实现硬件vDPA。目前Mellanox mlx5和Intel IFC对应的vDPA适配程序都已经合入到DPDK和kernel社区源码。

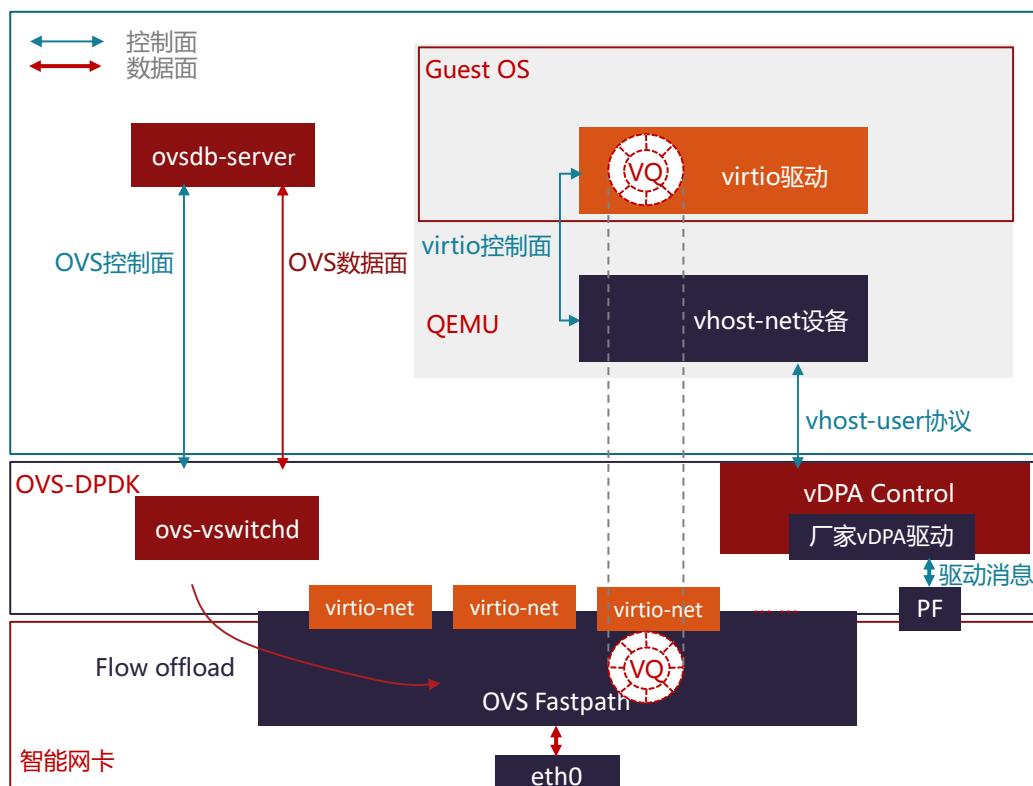


图3-4 virtio硬件加速虚拟化网络

在硬件vDPA场景下，通过OVS转发的流量首包依然由主机上的OVS转发平面处理，对应数据流的后续报文由硬件网卡直接转发。

后来在Bluefield-2上，由于集成了ARM核，所以NVIDIA在与UCloud的合作中，将OVS的控制面也完全卸载到网卡到ARM核上，这样主机上就可以将OVS完全卸载到网卡上。

### 3.1.2. 网络功能虚拟化 (Network Function Virtualization)

伴随着越来越多的业务上云，一些原来运行在专用设备或者特定主机上的网络产品也开始重视上云后的按需扩缩容能力，所以出现了网路功能虚拟化(NFV)产品。NFV产品主要以虚拟机(VM)或者容器(Container)的形态部署到云计算平台上，对外提供对应的网络功能，如Load Balance, Firewall, NAT, vRouter, DPI和5G边缘计算UPF等。这些NFV产品之前全部基于DPDK在X86 CPU上运行，由于CPU算力上限问题，通常难以提供对应网络带宽的吞吐能力。DPU智能网卡的出现，为NFV加速提供了资源和可能。

#### (1) 5G 边缘计算UPF (User Plane Function)

5G垂直行业对5G网络提出了更高的要求，如大带宽，高可靠，低时延，低抖动等，这对用户面数据转发的核心5G UPF，提出了更高的实现要求。尤其在边缘网络，交互式VR/AR在大带宽的要求下，还需要低时延，从而提高业务的用户体验；而车路协同系统等，对高可靠和低时延低抖动的要求更高，这是保障车路协同能够实时做出正确决策的关键；一些工业控制实时自动化应用，也是要求视频数据实时传输到服务端，并通过视频识别等功能实时做出控制指令下发等等。这些典型的应用，都对边缘计算中5G UPF提出了更严苛的要求。

在5G边缘计算中，未来的主要应用场景包括：增强型视频服务，监测与追踪类服务，实时自动化，智能监控，自动机器人，危险和维护传感，增强现实，网联车辆和远程操控等。对应的5G技术指标和特征也比较明显。首先，超

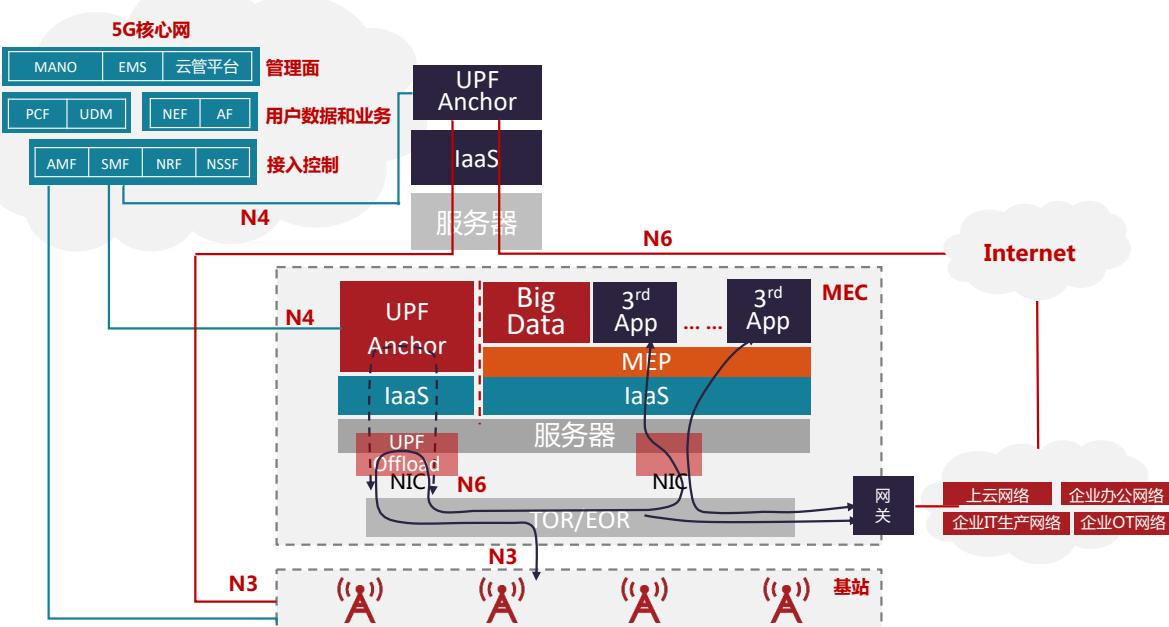
大带宽 (eMBB, Enhanced Mobile Broadband) 要求单用户峰值带宽可达 20Gbps，用户体验数据速率在100Mbps；其次，超密连接 (mMTC, Massive Machine Type Communication) 要求每平方公里设备数量在10k到1M个终端，流量密度10Mbps每平方米；最后，超低时延 (uRLLC, Ultra Reliable Low Latency Communication) 要求时延在1~10ms，高可靠性达到99.999%。



图3-5 5G业务网络特征

传统的5G UPF通常由软件实现，运行在X86 CPU上，虽然在吞吐上可以通过增加CPU来实现更大能力，但是，时延和抖动通常都会比较高，很难稳定支撑低时延低抖动业务。对于超大带宽，超低时延的需求，这类数据转发业务更适合在硬件中实现，硬件实现更容易做到低时延低抖动和大带宽。

5G UPF业务模型复杂，需要选择性卸载转发，将高可靠低时延低抖动业务要求的用户会话卸载到硬件中。如图3-5所示，数据流首包通过软件转发，然后将对应的流表卸载到智能网卡，后续报文通过智能网卡硬件转发，这样可以在5G边缘计算中，提供低时延低抖动和超大带宽网络能力的同时，还能降低边缘计算的整体功耗。

图3-6 5G边缘计算 UPF硬件卸载方式<sup>5</sup>

## (2) 智能DPI (Deep Packet Inspection)

DPI不论在运营商网络还是互联网数据中心，都是重要的配套设备。而且DPI功能是很多网络功能产品的基础功能，如IPS/IDS，5G UPF，DDoS防攻击设备等，具有重要的产品价值。DPI具有高新建、高并发、高吞吐等特点，使得性能成为虚拟化部署的瓶颈。通过DPU智能网卡实现DPI流量卸载，性能可以提升在30%以上。

<sup>5</sup> 5G时代工业互联网边缘计算网络白皮书 <http://www.ecconsortium.org/Uploads/file/20201209/1607521755435690.pdf>

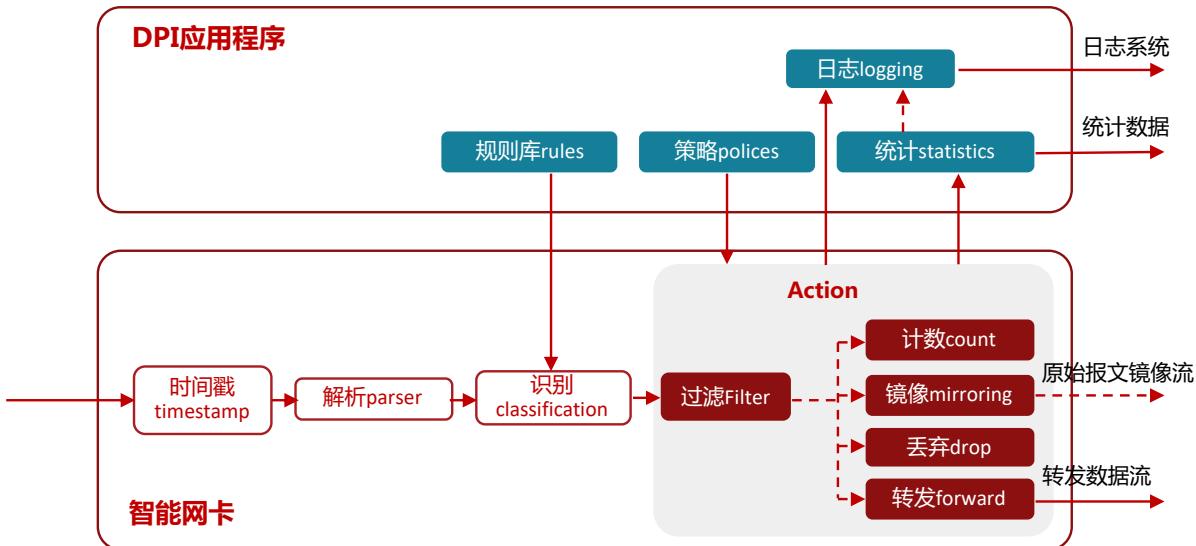


图3-7 智能DPI硬件卸载

智能DPI基于智能网卡卸载DPI流量，软件规则库下发到硬件形成流识别的规则，再将软件策略下发到硬件形成对应的动作。这样数据流进入网卡通过打时间戳进行硬件解析，匹配识别规则库，根据匹配的规则寻找对应的策略，根据策略进行计数，镜像，丢弃和转发等行为。通过硬件卸载DPI流量，不仅可以节省虚拟化DPI的CPU，还可以提升其吞吐量，同时降低了整体TCO。一些公有云的运维系统也是通过DPI功能做流日志，除了运维使用以外，还对用户提供对应的流日志服务。

### 3.1.3. 云原生网络功能

#### (1) 云原生网络架构

云原生，从广义上来说，是更好的构建云平台与云应用的一整套新型的设计理念与方法论，而狭义上讲则是以docker容器和Kubernetes（K8S）为支撑的云原生计算基金会（CNCF）技术生态堆栈的新式IT架构。对比虚拟机，容器应用对磁盘的占用空间更小，启动速度更快，直接运行在宿主机内核上，因而无Hypervisor开销，并发支持上百个容器同时在线，接近宿主机上本地进程的性

能，资源利用率也更高。以K8S为代表的云原生容器编排系统，提供了统一调度与弹性扩展的能力，以及标准化组件与服务，支持快速开发与部署。

容器平台包括容器引擎Runtime（如containerd, cri-o等），容器网络接口（CNI，如calico, flannel, contiv, cilium等）和容器存储接口（CSI，如EBS CSI, ceph-csi等）。

云原生平台可以部署在裸金属服务器上，也可以部署在虚拟机（VM）上。通常为了追求更高的性能，云原生平台会部署在裸金属上。如果考虑故障后更容易恢复，通常会部署在虚拟机（VM）上。

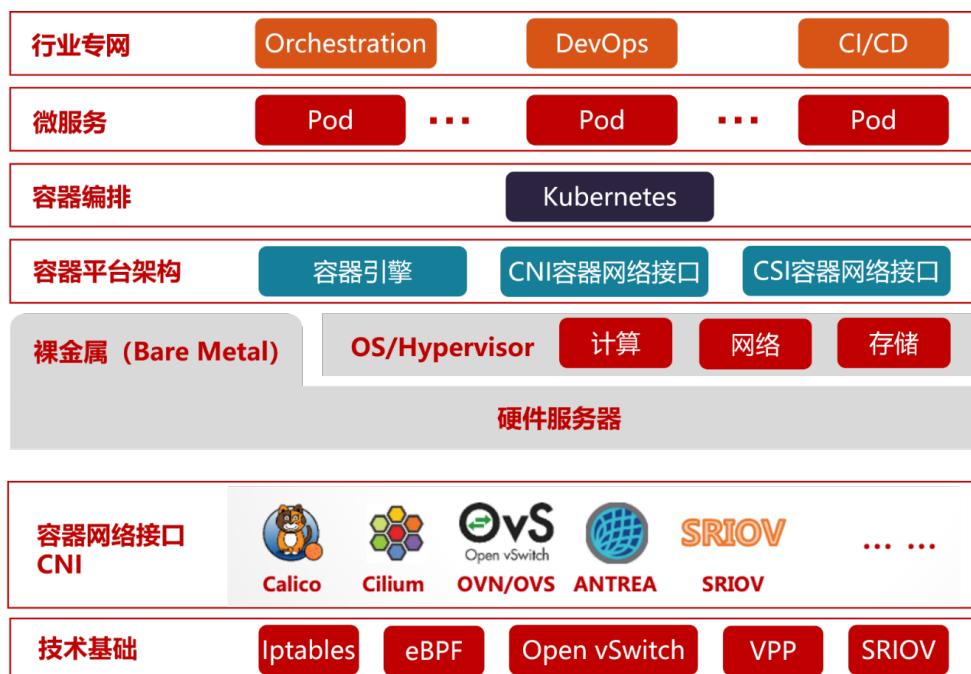


图3-8 云原生网络架构介绍

云原生对于网络的需求，既有基础的二三层网络联通，也有四至七层的高级网络功能。二三层的网络主要是实现K8S中的CNI接口，具体如calico, flannel, weave, contiv, cilium等。主要是支持大规模实例，快速弹性伸缩，自愈合，多集群多活等。四至七层网络功能，主要是服务网格（Service Mesh）。服务网格的本质是提供安全、可靠、灵活、高效的服务间通信。服务网格还提

供了一些更加高级的网络功能，如有状态的通信，路由限流，灰度流量切换，熔断监控等。

## (2) eBPF的硬件加速

eBPF是一项革命性的技术，可以在Linux内核中运行沙盒程序，而无需重新编译内核或者加载内核模块。在过去几年，eBPF已经成为解决以前依赖于内核更改或者内核模块的问题的标准方法。对比在Kubernetes上Iptables的转发路径，使用eBPF会简化其中大部分转发步骤，提高内核的数据转发性能。Cilium是一个基于eBPF实现的开源项目，提供和保护使用Linux容器管理平台部署的应用程序服务之间的网络和API连接，以解决容器工作负载的新可伸缩性，安全性和可见性要求。Cilium超越了传统的容器网络接口（CNI），可提供服务解析，策略执行等功能，实现了组网与安全一体化的云原生网络。Cilium数据平面采用eBPF加速，能够以Service/pod/container为对象进行动态地网络和安全策略管理，解耦控制面等策略管理和不断变化的网络环境，具有应用感知能力（如https，gRPC等应用），从而实现对流量的精确化控制。同时它的状态通过K-V数据库来维护实现可扩展设计。

基于eBPF的Cilium已经被证明是Kubernetes云原生网络的最佳实践，国内阿里云和腾讯云已在自己的公有云中引用了Cilium构建自己的云原生平台。

为进一步提升性能，Netronome将eBPF路径上的部分功能卸载到硬件网卡，如XDP和Traffic Classifier cls-bpf，实现了对于用户无感知的eBPF卸载加速。硬件卸载的eBPF程序可以直接将报文送到任意内核eBPF程序，eBPF中map的维护对于用户程序和内核程序是透明的。

eBPF程序的编译需要在生成内核微码的基础上，加入编译硬件可识别的微码程序，并将对应的硬件微码程序装载到网卡中。从而实现eBPF硬件卸载。

### (3) 云原生Istio服务网格

Istio是CNCF主推的微服务框架，实现了云原生四至七层网络能力。Istio在数据平面通过Sidecar对流量进行劫持，实现了无代码侵入的服务网格。控制平面组件中，pilot负责下发控制，mixer收集运行的状态，citadel则负责安全证书方面的处理。

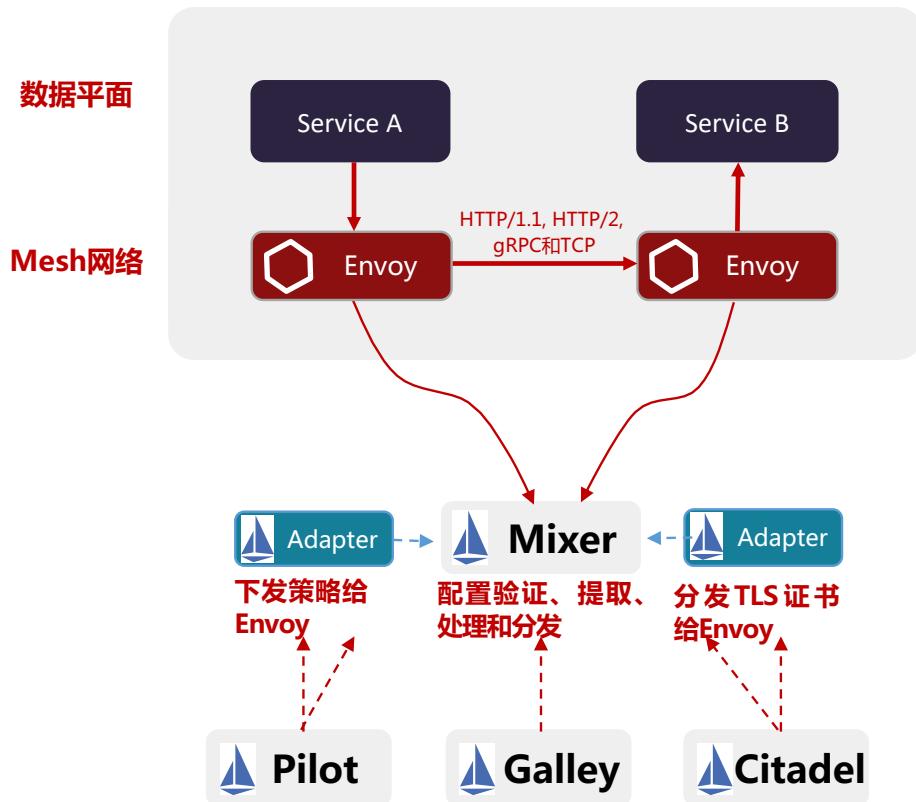


图3-9 服务网格Istio架构<sup>6</sup>

在Istio中，原生的七层代理使用的是Envoy，Envoy提供了动态服务发现，负载均衡的能力，同时支持TLS连接，可以代理HTTP/1.1 & HTTP/2 & gRPC等协议。同时支持熔断器，流量拆分，故障注入等能力和丰富的度量指标。另外，阿里还提出了MOSN作为Istio的七层代理，这里不再介绍细节。由于引入七层代理后，增加数据转发时延。Broadcom尝试在Stingray DPU智能网卡将Envoy以网关模式卸载到智能网卡的ARM核上，用了八个ARM核中的六个来做Envoy

<sup>6</sup> Istio <https://istio.io>

卸载支持100G网卡上的流量。这种方式是否能够适用于当前Sidecar上代理模式的Envoy，还有待探索，另外，编排和管理方式也需要进一步调研。

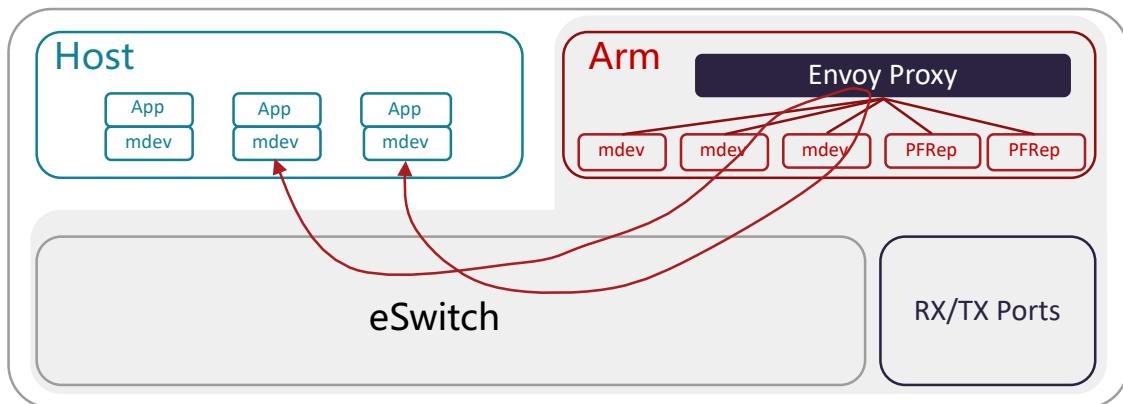


图3-10 Envoy卸载实例<sup>7</sup>

OVS硬件卸载的加速方式主要由OVN-Kubernetes和Antrea CNI做控制面。由于OVS的数据面在OpenStack云平台上已经有了硬件卸载的先例，所以，在云原生网络再做硬件卸载，基本上没有技术障碍，只需要增加NAT处理能力即可。但是，这种硬件卸载方式，由于没有涉及到七层代理，所以不可避免的还是会将报文上送到内核转发，进而在数据转发性能上的提升相对有限。

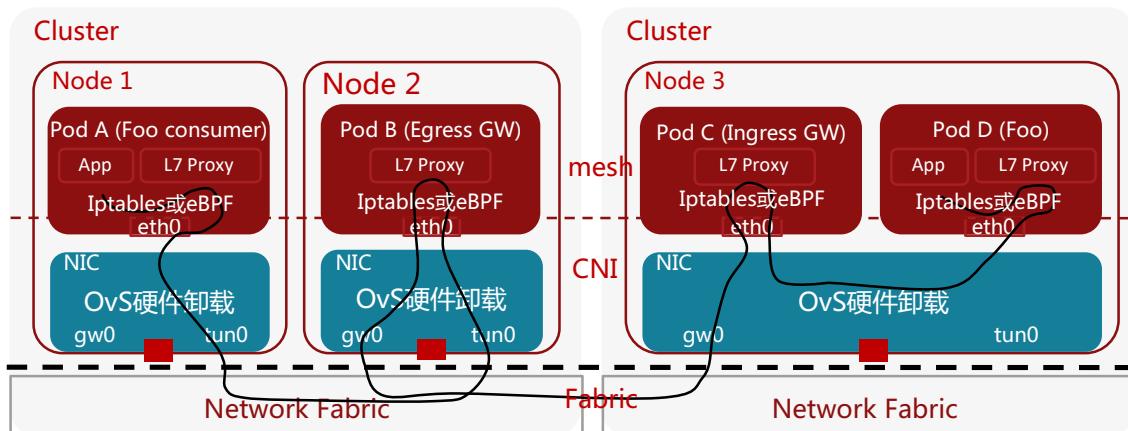


图3-11 OVS硬件加速<sup>7</sup>

总体而言，在云原生网络中智能网卡的应用场景还需要进一步探索，目前尚无一个相对完美的能够解决服务网格性能和时延问题的方案。

<sup>7</sup> Taking Control of your SmartNIC  
<https://legacy.netdevconf.info/0x14/pub/slides/39/Netdev%200x14%20--%20Taking%20Control%20of%20your%20SmartNIC%20v1.pdf>

### 3.1.4. RDMA网络功能

#### (1) RDMA网络功能介绍

面对高性能计算、大数据分析和浪涌型IO高并发、低时延应用，现有TCP/IP软硬件架构和应用高CPU消耗的技术特征根本不能满足应用的需求。这主要体现在处理时延过大——数十微秒，多次内存拷贝、中断处理，上下文切换，复杂的TCP/IP协议处理，以及存储转发模式和丢包导致额外的时延。而RDMA通过网络在两个端点的应用软件之间实现Buffer的直接传递，相比TCP/IP，RDMA无需操作系统和协议栈的介入，能够实现端点间的超低时延、超高吞吐量传输，不需要网络数据的处理和搬移耗费过多的资源，无需OS和CPU的介入。RDMA的本质实际上是一种内存读写技术。

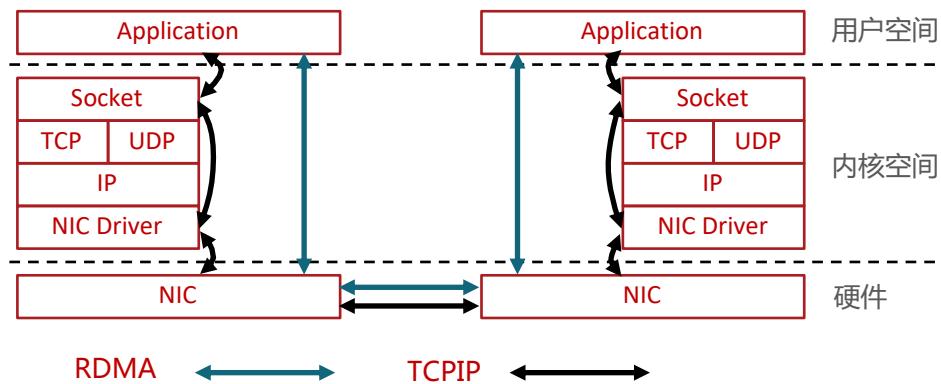


图3-12 对比RDMA和TCP/IP收发数据

RDMA和TCP/IP网络对比可以看出，RDMA的性能优势主要体现在：

- (1) 零拷贝——减少数据拷贝次数，由于没有将数据拷贝到内核态并处理数据包头部到过程，传输延迟会显著减少。
- (2) Kernel Bypass和协议卸载——不需要内核参与，数据通路中没有繁琐的处理报头逻辑，不仅会使延迟降低，而且也节省了CPU的资源。

## (2) RDMA硬件卸载方式

原生RDMA是IBTA (InfiniBand Trade Association) 在2000年发布的基于 InfiniBand的RDMA规范；基于TCP/IP的RDMA称作iWARP，在2007年形成标准；基于Ethernet的RDMA叫做RoCE，在2010年发布协议，基于增强型以太网并将传输层换成IB传输层实现；在2014年，IBTA发布了RoCEv2，引入IP解决扩展性问题，可以跨二层组网，引入UDP解决ECMP负载分担等问题。

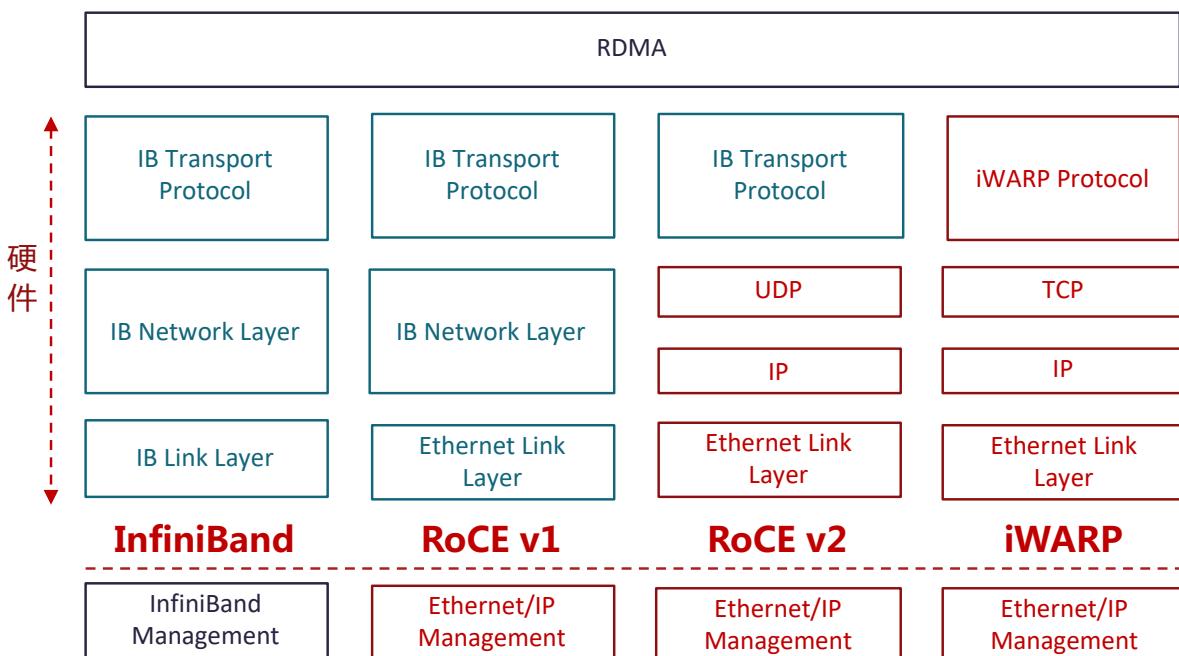


图3-13 RDMA协议栈介绍<sup>8</sup>

InfiniBand是一种专为RDMA设计的网络，从硬件级别保证可靠传输。全球HPC高算系统TOP500大效能的超级计算机中有相当多套系统在使用InfiniBand Architecture (IBA)。最早做InfiniBand的厂商是IBM和HP，现在主要是NVIDIA的Mellanox。InfiniBand从L2到L4都需要自己的专有硬件，成本非常高。

iWARP直接将RDMA实现在TCP上，优点就是成本最低，只需要采购支出iWARP的NIC即可以使用RDMA，缺点是性能不好，因为TCP协议栈本身过于重

<sup>8</sup> RoCE vs. iWARP A Great Storage Debate  
<https://www.snia.org/sites/default/files/ESF/RoCE-vs.-iWARP-Final.pdf>

量级，即使按照iWARP厂商的通用做法将TCP卸载到硬件上实现，也很难超越IB和RoCE的性能。

RoCE (RDMA over Converged Ethernet) 是一个允许在以太网上执行RDMA的网络协议。由于底层使用的以太网帧头，所以支持在以太网基础设施上使用RDMA。不过需要数据中心交换机DCB技术保证无丢包。相比IB交换机时延，交换机时延要稍高一些。由于只能应用于二层网络，不能跨越IP网段使用，市场应用场景相对受限。

RoCEv2协议构筑于UDP/IPv4或UDP/IPv6协议之上。由于基于IP层，所以可以被路由，将RoCE从以太网广播域扩展到IP可路由。由于UDP数据包不具有保序的特征，所以对于同一条数据流，即相同五元组的数据包要求不得改变顺序。另外，RoCEv2还要利用IP ECN等拥塞控制机制，来保障网络传输无损。RoCEv2也是目前主要的RDMA网络技术，以NVIDIA的Mellanox和Intel为代表的厂商，均支持RoCEv2的硬件卸载能力。

## 3.2. 应用场景二：存储功能卸载

### 3.2.1. NVMe-oF硬件加速

NVMe over Fabric (又名NVMe-oF) 是一个相对较新的协议规范，旨在使用NVMe通过网络结构将主机连接到存储，支持对数据中心的计算和存储进行分解。NVMe-oF协议定义了使用各种通用的传输协议来实现NVMe功能的方式。

在NVMe-oF诞生之前，数据存储协议可以分为三种：

- (1) iSCSI：是一种基于IP的存储网络标准，在TCP/IP网络上通过发送SCSI命令来访问块存储服务。
- (2) 光纤通道 (Fibre Channel)：是一种高速的数据传输协议，提供有序无损的块数据传输。主要用于关键高可靠要求的业务上。

(3) SAS (Serial Attached SCSI) : 一种点对点串行协议，通过SAS线缆传输数据。

上述数据存储协议，在当今数据爆发的时代，已经无法满足大数据量的传输。NVMe-oF的出现，不仅解决了上述协议的性能瓶颈问题，它还允许组织为高度分布式、高度可用的应用程序实施横向扩展的存储。通过将NVMe协议扩展到SAN设备，NVMe-oF提高了CPU的使用效率，同时提高了服务器和存储应用程序之间的连接速度。

NVMe-oF主要支持三大类Fabric传输选项，分别是FC、RDMA和TCP，其中RDMA支持InfiniBand、RoCEv2和iWARP。

NVMe-oF/FC和第六代FC可以共存于同一基础设施中，避免了数据中心的叉车升级。但是，NVMe-oF/FC不具有软件定义存储的能力。

NVMe-oF/RDMA利用了RDMA网络的优势，是理想的Fabric，提供了低延迟、低抖动和低CPU使用率低传输层协议，可以最大限度利用硬件加速，避免软件协议栈开销。同时，由于RDMA是一种内存读写技术，可以应用在众多场景中，如GPUDirect Storage的应用场景。

NVMe-oF/TCP利用了TCP协议的可靠性传输的特点，以及TCP/IP网络的通用性和良好的互操作性，可以完美的应用于现代数据中心网络。在相对性能要求不是非常高的场景，NVMe-oF/TCP可作为备选。

NVMe支持Host端（Initiator或Client）和Controller端（Target或Server），目前DPU智能网卡硬件加速的场景中，包括如下四中情况：

(1) 普通智能网卡硬件加速NVMe-oF Initiator。智能网卡支持NVMe-oF/TCP和NVMe-oF/RoCEv2作为Initiator，通过硬件卸载NVMe-oF/TCP或NVMe-oF/RoCEv2，用于计算和存储之间，来达到较高性能。

(2) 支持GPUDirect Storage的智能网卡加速NVMe-oF Initiator和Target。

GPUDirect Storage是NVIDIA提出的GPU可以绕过CPU直接访问存储磁盘的技术，RDMA技术是GPUDirect Storage的基础。这类网卡可以通过硬件卸载NVMe-oF/RDMA来实现GPU与远端存储服务的直接访问。常见的如NVMe-oF/RDMA IB和NVMe-oF/RoCEv2。

(3) 智能网卡硬件加速NVMe-oF Target。该场景主要是通过智能网卡提供PCIe Root Complex能力和NVMe-oF Controller端的硬件卸载加速，来实现NVMe存储服务器。如Broadcom Stingray PS1100R是这个场景的代表之一。

(4) DPU芯片硬件加速NVMe-oF Target。该场景是通过DPU芯片提供多个PCIe Root Complex通道以及多个100Gbps的网卡实现的超大吞吐的存储服务器。Fungible FS1600 12x100Gbps带宽吞吐的存储服务器是这个场景的典型代表。

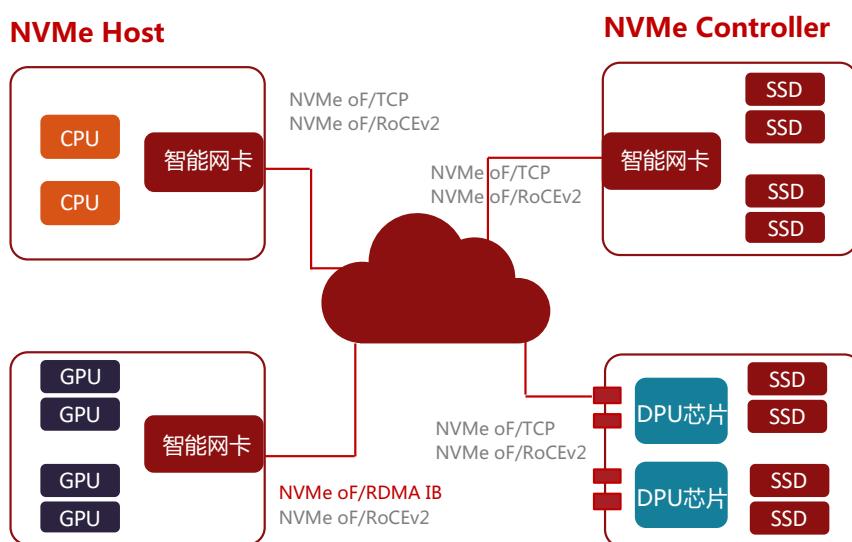


图3-13 NVMe-oF硬件卸载方式

OpenStack从Rocky版本已经支持了NVMe-oF，通过OpenStack Cinder通过消息在NVMe-oF Target上来创建，查询和删除卷等，OpenStack Nova在主机上通过NVMe-oF Initiator发现NVMe-oF存储设备，并将存储设备信息传递给Hypervisor来实现虚拟机挂载磁盘。另外，OpenStack集成Ceph做块存储和对象

存储已经非常成熟，Ceph的后端存储也渐渐的从使用本地磁盘的方式转向远端NVMe存储，这样NVMe-oF为Ceph存储服务提供了容量可伸缩的能力。

### 3.2.2. Virtio-blk硬件加速

基于virtio的virtio-blk是KVM-Qemu虚拟化生态中的虚拟化块存储的一种实现方式，利用了virtio共享内存的机制，提供了一种高效的块存储挂载的方法。Guest OS内核通过加载virtio-blk驱动，实现块存储的读写，无需额外的厂家专用驱动。Virtio-blk设备在虚拟机以一个磁盘的方式呈现，是目前应用最广泛的虚拟存储控制器。

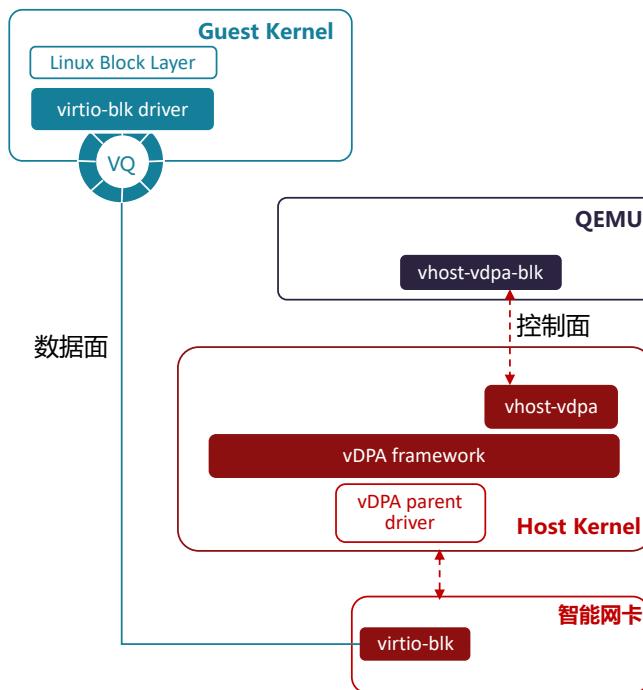


图3-15 基于vDPA架构的virtio-blk硬件卸载<sup>9</sup>

由于virtio机制通过硬件实现加速已经是通用做法，所以利用这个优势，virtio-blk卸载到硬件，已经是必然趋势。在智能网卡中，将virtio-blk到后端映射

<sup>9</sup> Vdpa-blk: Unified Hardware and Software Offload for virtue-blk, Stefano Garzarella, Redhat  
[https://static.sched.com/hosted\\_files/kvmforum2021/b1/KVMForum\\_2021\\_vdpa\\_blk\\_Stefano\\_Garzarella.pdf](https://static.sched.com/hosted_files/kvmforum2021/b1/KVMForum_2021_vdpa_blk_Stefano_Garzarella.pdf)

到如NVMe-oF的远端磁盘上，这样相比较当前virtio-blk的用法，不需要在主机系统中挂载很多的远端NVMe磁盘，由智能网卡直接完成映射，更加安全。

在2021年KVM论坛会议中，Redhat提出统一软硬件卸载virtio-blk方案，正式将virtio-blk加入vDPA框架，同virtio-net公用相同的框架，来完成硬件卸载控制平面。

### 3.3. 应用场景三：安全功能卸载

#### 3.3.1. 硬件信任根

硬件信任根在安全领域是其它安全功能的基础，主要表现如下方面：

- (1) 硬件信任根 (Root-Of-Trust)：硬件信任根提供更离散的密钥生成算法，并且与主机操作系统相隔离，可以做到硬件防破解。硬件信任根实现私有密钥存储，可以反克隆和签名。通过硬件信任根认证授权实现访问受控。
- (2) 加密解密 (Encryption/Decryption)：数据加密解密算法完全卸载到硬件网卡，无需主机CPU资源，效率更高更可靠。可以实现通用加密算法和国密算法等。
- (3) 密钥证书管理 (KMS)：密钥证书管理卸载到智能网卡，与主机系统相隔离；支持多种密钥交换算法，如D-H密钥交换等。
- (4) 动态数据安全 (Secure Data-in-Motion)：利用硬件级加解密算法，对传输通道上的数据做加解密处理，如IPSec和TLS等。硬件处理可以实现更高吞吐量。
- (5) 静态数据安全 (Secure Data-at-Rest)：在存储服务中，永久存盘的数据需要进行加密，防止被窃取，硬件级数据加解密在存储服务中可以提供更高效的数据读取，并保证数据安全。

(6) 流日志和流分析 (Flowlog)：流分析和流日志监控，对数据中心流量做精细监控，有效识别，可以及时识别DDoS攻击，并做出响应。

### 3.3.2. 安全服务应用

在安全领域，还有很多的安全功能产品，如NGFW，WAF，IPS/IDS，DDoS防御设备等。随着云和虚拟化技术的发展，越来越多的安全功能产品的实现方式转为虚拟化方式，并通过云平台来部署管理。这些安全功能产品由于部署在数据中心流量的主要路径上，转发性能对整体网络的吞吐量和时延具有重要的影响。基于X86的软件实现方式，需要大量CPU资源来处理对应的业务逻辑，性能上的瓶颈已经愈发明显。通过智能网卡对这些安全功能产品做硬件加速，已经是必然趋势。

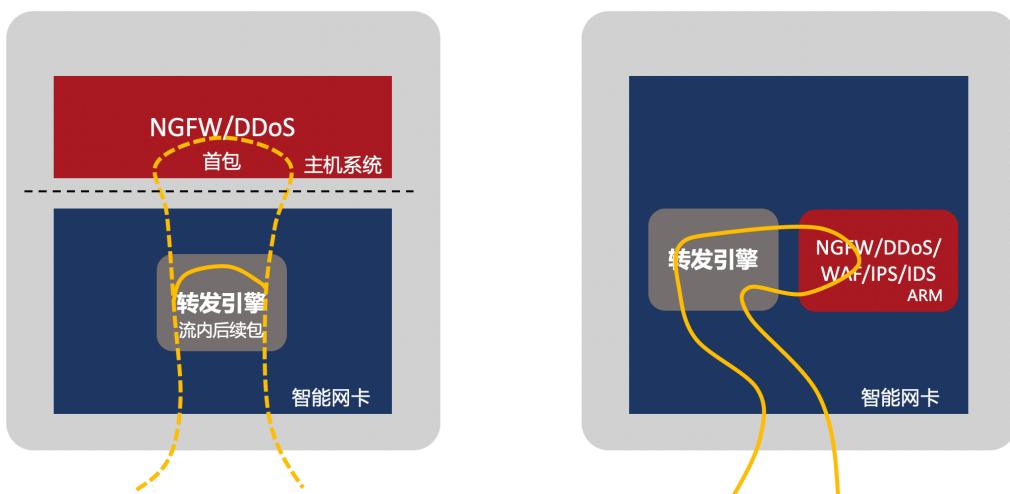


图3-16 安全服务应用硬件卸载

由于安全功能产品对报文处理的深度不同，有些只需要在二至四层处理，有些则需要在七层进行处理，所以在智能网卡的卸载方式上，也存在不同。如NGFW和DDoS等设备，可以通过流表卸载的方式，对流量进行拦截，来加速运行在主机系统中的安全服务应用。如IPS/IDS等，需要对报文内容做深度检测，则可以通过in-line的方式将数据深度检测功能卸载到智能网卡的CPU上，这时需要智能网卡的CPU具有较强的性能。

### 3.3.3. 隔离网络虚拟化

在传统的网卡上做云平台虚拟化，Hypervisor以及对应的虚拟化网络的实现，都是在主机操作系统上实现的。这样如果黑客如果攻陷了Hypervisor并拿到主机操作系统的root权限，就可以通过篡改虚拟化网络配置，来对租户网络进行攻击，甚至可以渗透到其它计算节点，进行更大范围的攻击。

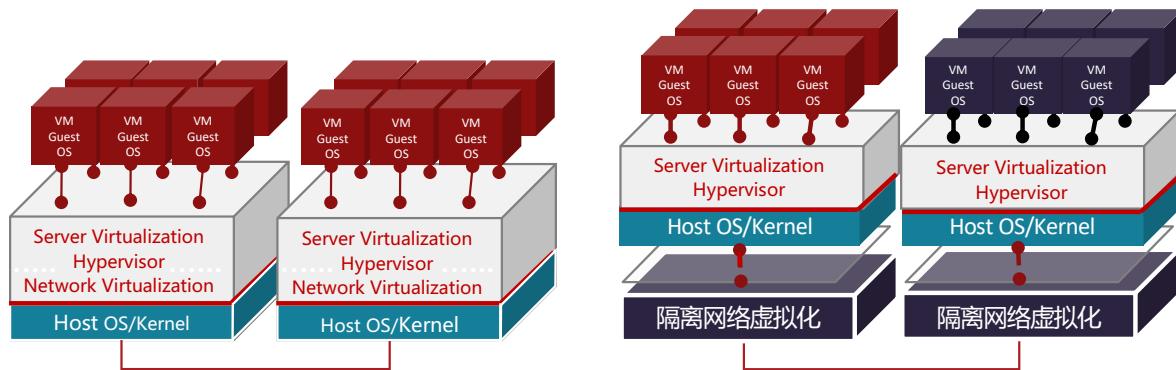


图3-17 隔离虚拟化网络<sup>10</sup>

引入DPU智能网卡之后，将虚拟化网络的控制平面完全卸载到智能网卡上，与主机操作系统相隔离。即使黑客攻陷了Hypervisor，获取了主机操作系统的root权限，也无法篡改虚拟化网络的配置，这样可以将黑客的攻击范围限制在主机操作系统上，不会影响到虚拟化网络以及其它主机。进而达到了安全隔离的效果。

<sup>10</sup> 隔离网络虚拟化 <https://www.oracle.com/cn/security/cloud-security/isolated-network-virtualization/>

## 4. DPU软件栈五层模型

### 4.1. 软件栈开发面临的挑战

DSA架构和XPU芯片的兴盛在给解决算力问题带来新机遇的同时，也给软硬件开发带来了新的挑战。与传统的以CPU为核心的应用开发模式相比，异构计算的开发难度较高。异构计算的一个特点是“异构”，即缺少“通用性”，需要开发人员同时深入了解多种处理器的开发和调优方式，给软硬件开发带来了更高的复杂度。异构计算的另一个特点是“软硬协作”，它需要开发人员将软件硬件作为一个整体来架构和开发，给当下计算生态中软硬界限分明、技术栈分层的开发模式带来颠覆性挑战，尤其是当DPU兴起之后的“多PU”“共存(CPU+DPU+GPU/XPU)时代，如何协同调度好各个处理器编程框架(如GPU的CUDA、FPGA的OpenCL/HLS等)，使其发挥最大效用，并且构建一个面向开发人员友好的协同软件开发生态，是迫切需要解决的问题。

针对这个问题，根据职责分层、按功能抽象的思路，提出一个针对DPU芯片应用场景的异构计算五层架构模型。该模型定义了在异构计算场景下的通用的开发架构模式，以此来降低包括DPU芯片在内的异构计算应用研发的难度，提高开发、维护和迭代效率。

### 4.2. DPU异构计算架构五层开发模型

一般说来，异构计算的核心目的是解决特定应用场景下算力不足的问题，并且大幅度提升整体系统的计算性能。在整体架构上，它的分层逻辑从应用场景出发，通过上层的需求来定义下层的功能，而每一层是对特定功能的抽象与封装。在定义每一层功能时，要达到以下几个目标：

- 各层职责单一

- 层间边界清晰
- 层内功能实现独立
- 灵活易扩展

基于上述目标，将一个异构计算的系统抽象为五层（如图4-1所示），自下而上分别是：1) DSA设备层 (DSA Device Layer) , 2) DSA操作层 (DSA Operating Layer) , 3) 计算引擎层 (Scheduling Operating Layer) , 4) 应用服务层 (Application Service Layer) 和5) 业务开发层 (Business Development Layer) , 详述如下。

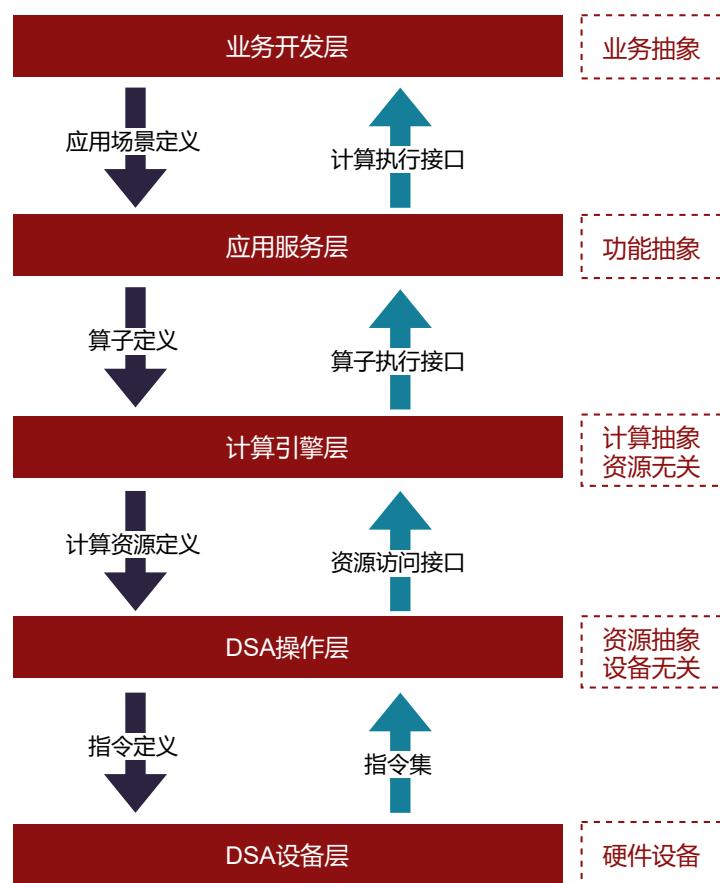


图4-1 异构计算系统抽象模型

## 4.2.1.DSA设备层

DSA设备层代表的是执行异构计算的DSA处理器以及集成了该处理器的硬件设备，例如，以DPU或GPU为处理器的异构计算设备。异构计算设备需要具备以下两个核心能力：1) 提供支持专用计算操作的指令集（Instruction Set），2) CPU或其他DSA设备的标准通信接口，如PCIe数据传输标准。

## 4.2.2. DSA操作层

DSA操作层是对DSA处理器的指令集的管理以及基础开发平台的整合，该层完成了对硬件资源的抽象，从而使上层软件对底层设备透明；DSA操作层是对DSA设备层计算设备的抽象和计算资源的封装，是软件与硬件、逻辑与物理的边界。它基于如DPU芯片等DSA处理器提供的指令集，以更加抽象和编程友好的方式对上层提供了异构计算开发和访问的软件接口、以及设备监控管理的接口。该层内部有四个必要的模块，分别是设备驱动器，指令集管理器，资源访问接口，开发和管理平台。

- 设备驱动器：设备驱动器是硬件设备的软件抽象，它基于操作系统标准的驱动框架及PCIe协议，实现了对计算设备的物理访问，主要包括设备处理器的指令执行和设备存储的读写。
- 指令集管理器：指令管理器的作用是对计算设备所提供的指令集进行统一管理，通过对指令集的封装及组合，提供更加友好的编程接口。
- 资源访问接口：基于设备驱动器和指令集管理器的功能，该模块完成对整个计算资源访问的抽象和封装，它以编程接口的方式对上层提供资源访问入口，服务于上层计算逻辑和控制逻辑的执行。

- **开发和管理平台**: 除了上述运行时所需的能力外, 还需要针对开发人员提供友好的编程工具, 如指令集编译工具、监控管理工具、日志工具、异构计算卡模拟器等。

### 4.2.3. 计算引擎层

计算引擎层是对计算逻辑的封装, 为上层提供通用的计算能力。与DSA操作层的对计算资源封装不同, 计算引擎层是对计算逻辑的封装, 它基于DSA操作层提供的资源访问接口, 根据上层应用层软件对算力的需求, 提供了可复用的算子集合及执行接口。

#### 算子抽象

算子定义为实现某一特定功能或算法的函数或独立的服务, 它是对计算逻辑的抽象。例如, 根据指定条件对数据进行过滤的函数可以是一个算子, 称它为“过滤算子”。在标准的数据库软件中, 它的算子有Scan, Join, GroupBy等, 而在网络处理软件中, 特有的算子会是以BSD Socket, NVMe等标准服务的形式呈现。由于异构计算的“异构”特性, 每个算子在不同设备上的具体实现有所不同, 所以针对每一个支持的算子, 都要有多种不同设备平台上的实现, 如ScanOnDPU、ScanOnGPU。

#### 计算优化器

异构计算追求的是计算性能的提升, 相应的需要一个计算优化器来对上层的计算请求做优化。它的基本策略是根据应用场景、上下文、数据规模等因素来动态的选择最优的算子实现进行计算。

### 4.2.4. 应用服务层

应用服务层是数据处理的应用服务软件, 也是算力的需求侧。应用服务层代表具有通用功能的软件系统, 这些软件系统可以利用计算引擎提供的算子进

行异构计算，从而达到计算性能提升的目的。常见的应用层软件系统有分布式计算领域的Spark, Flink, Hadoop；数据库领域的PostgreSQL, MySQL；分布式网络中的gRPC, Network Gateway, Nginx；以及存储服务中的Ceph等等，基本上通用服务型的系统都属于该层的范畴。

在实际开发中，针对应用服务层中的软件，需要解决以下几个关键问题：

- 性能瓶颈的识别：通常应用软件的性能瓶颈会在高并发、大吞吐的情况下出现，这些瓶颈一般源于CPU计算资源的竞争、CPU计算性能的不足、网络传输的延迟以及磁盘I/O的延迟等。识别出应用软件的性能瓶颈是算力卸载的第一步。
- 异构计算的有效性边界：在定位到软件的性能瓶颈后，需要从中识别出哪些是可以通过异构计算来解决的。通常，CPU成为瓶颈的原因会有两类，一类是算力的问题，另一类是算法的问题。针对算力的问题，可以通过异构计算来解决，而算法的问题则不然。
- 算子的高效调用：算子是异构计算的执行单元，只有把算子集成到应用软件的执行路径中，算力卸载才算完成。考虑到性能的优化，还需要根据实际情况优化算子的执行策略，例如，数据在主机端与设备端内存之间的数据拷贝策略、各算子执行序列的编排策略等等。
- 应用软件的向前兼容性：在整合应用软件与异构计算的算子时，要确保应用软件的向前兼容性，以保证应用服务层的软件迭代对正在运行的上层业务系统是透明的，从而提高整个架构的稳定性与可维护性。

#### 4.2.5. 业务开发层

业务开发层是在某特定领域的业务系统。业务开发层是最贴近实际业务场景的软件系统，通常它是针对某个特定行业的具体业务需求定制的软件系统，如金融行业的交易系统，互联网行业的数据分析系统等等。整个异构计算架构

本质上就是解决业务层的性能瓶颈问题，所以在实际开发过程中，应该从业务端出发，寻找要解决的根本问题，然后驱动整个异构系统的构建。同时，整体架构也要保证底层构建对具体的业务系统完全透明，达到对各行业业务软件系统的无缝支撑和业务逻辑开发的隔离。

## 4.3. 典型软件框架案例

### 4.3.1. NVIDIA DOCA软件框架

#### (1) 设计目的

NVIDIA BlueField DPU的核心目的是解决数据中心中基于CPU计算的基础设施的算力问题，主要涉及网络、存储、安全及基础设施管理这几个方面。而DOCA是为了将BlueField DPU提供的硬件能力做软件抽象和封装，以SDK Library的形式提供友好的可编程接口，提高应用开发的效率。

#### (2) 技术细节

DOCA架构如图4-2所示，DOCA有三个模块，分别是：

(1) DOCA drivers: 是对DPU硬件资源的低层次封装，其提供的low-level API是对硬件卸载能力的访问，主要包含网络卸载、安全卸载、存储卸载等算力的访问，同时支持DPDK, RDMA, Virtio-net（网络虚拟化），Virtio-blk（存储虚拟化），PCIe等通用能力。

(2) DOCA libs: 是基于DOCA drivers为上层应用封装的high-level API, 这些API是面向应用层所需的功能，例如面向网络应用的Flow, Data Integrity, UPF(User Plane Function), VNF(Network Functions Virtualization); 面向存储应用的SPDK; 面向安全应用的DPI(Deep Packet Inspection), Host introspection等。

(3) DOCA services: 封装了基础设施的控制和管理功能，如DPU设备的管理，SDN（Software-Defined Network）的控制接口，存储管理，Network Telemetry等。

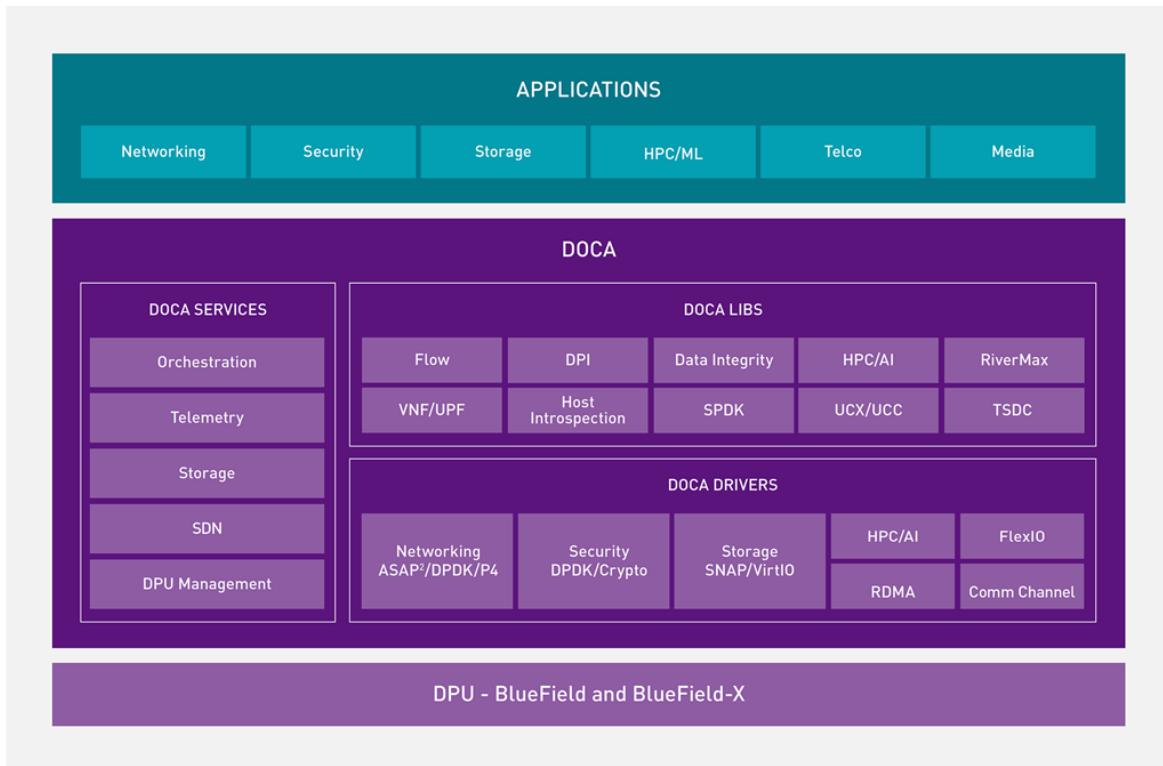


图4-2 DOCA软件栈<sup>11</sup>

### (3) 功能分析

从纵向的功能角度来看，DOCA SDK提供了网络加速、安全加速、存储加速和基础设施管理加速的功能，相关的技术细节如下：

- (1) 网络加速SDK：支持ASAP2 (Accelerated Switching and Packet Processing) SDN、VirtIO、OVS (open virtual switching)、P4 编程、RDMA。
- (2) 安全加速SDK：支持Inline encryption、DPI (Deep Packet Inspection)、TLS、IPSec。

<sup>11</sup> <https://developer.nvidia.com/zh-cn/networking/docta>

(3) 存储加速SDK：支持SPDK、VirtIO、NVMe-oF、数据加解密和数据压缩等。

(4) 基础设施管理加速SDK：支持DPU management、Traffic telemetry、Packet filtering。

### 4.3.2. Intel OneAPI软件框架

#### (1) 设计目的

OneAPI设计的目的是通过实现一个跨平台，开放的，标准的通用编程模型及接口来提高在异构加速器架构下的开发效率。OneAPI本身并不提供DPU设备，而是构建一个软件框架来整合业界现有的异构计算设备<sup>11</sup>。

#### (2) 技术细节

OneAPI抽象出两层APIs，分为L0 (Low-level) API 和L1 (High-Level) API。其结构如图4-3所示。

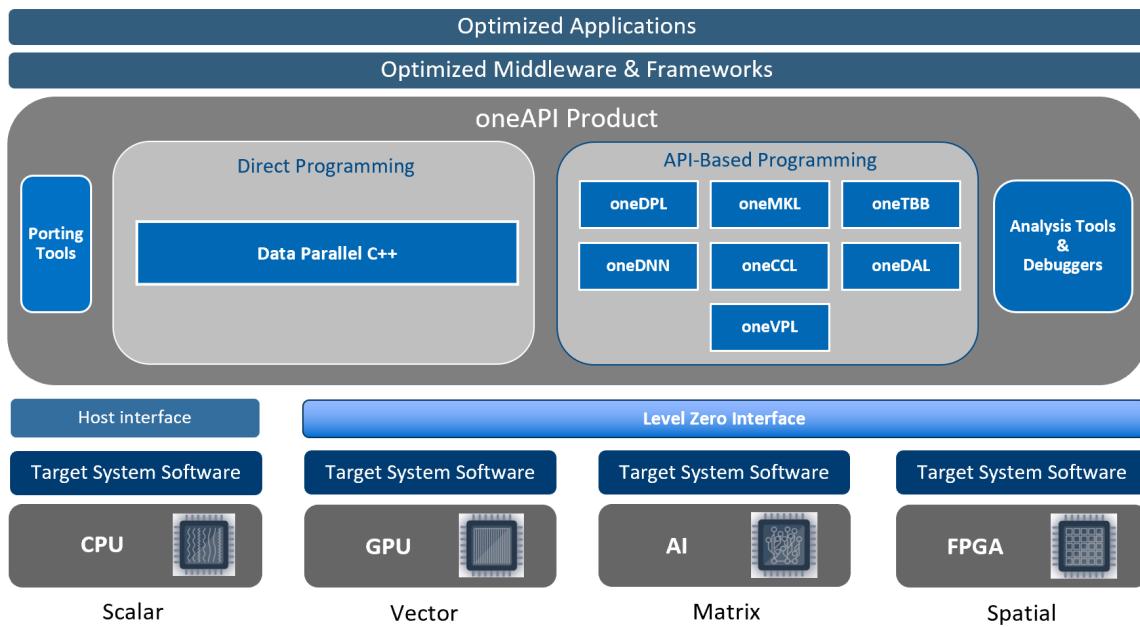


图4-3 OneAPI软件栈<sup>12</sup>

<sup>12</sup><https://spec.oneapi.io/level-zero/latest/core/INTRO.html>

L0 API: OneAPI 的L0 API整合并封装了业界流行的异构计算设备（如 GPU, AI, FPGA等）及其软件框架（如CUDA, OpenCL），并提供统一的针对硬件资源访问的low-level API。这样，不同的硬件平台对上层应用开发来说是透明的，从而起到了跨平台的作用。

L1 API: OneAPI的L1 API提供了一系列针对特定应用场景的High-level API，这些API主要服务于机器学习、数据分析、并行计算、视频处理等特定领域，有 Deep Learning API, Data Science API, Data Analysis API, Multiple Thread API, Video Processing API等。

基于上述两层的抽象和封装，上层应用程序可以利用L1 API进行特定领域的数据处理加速，也可以直接访问L0 API进行编程来服务更加广泛的场景。

### 4.3.3. 中科驭数 HADOSTM 软件框架

驭数DPU的软件框架是基于上述五层模型构建的，整体架构如图4-4所示：

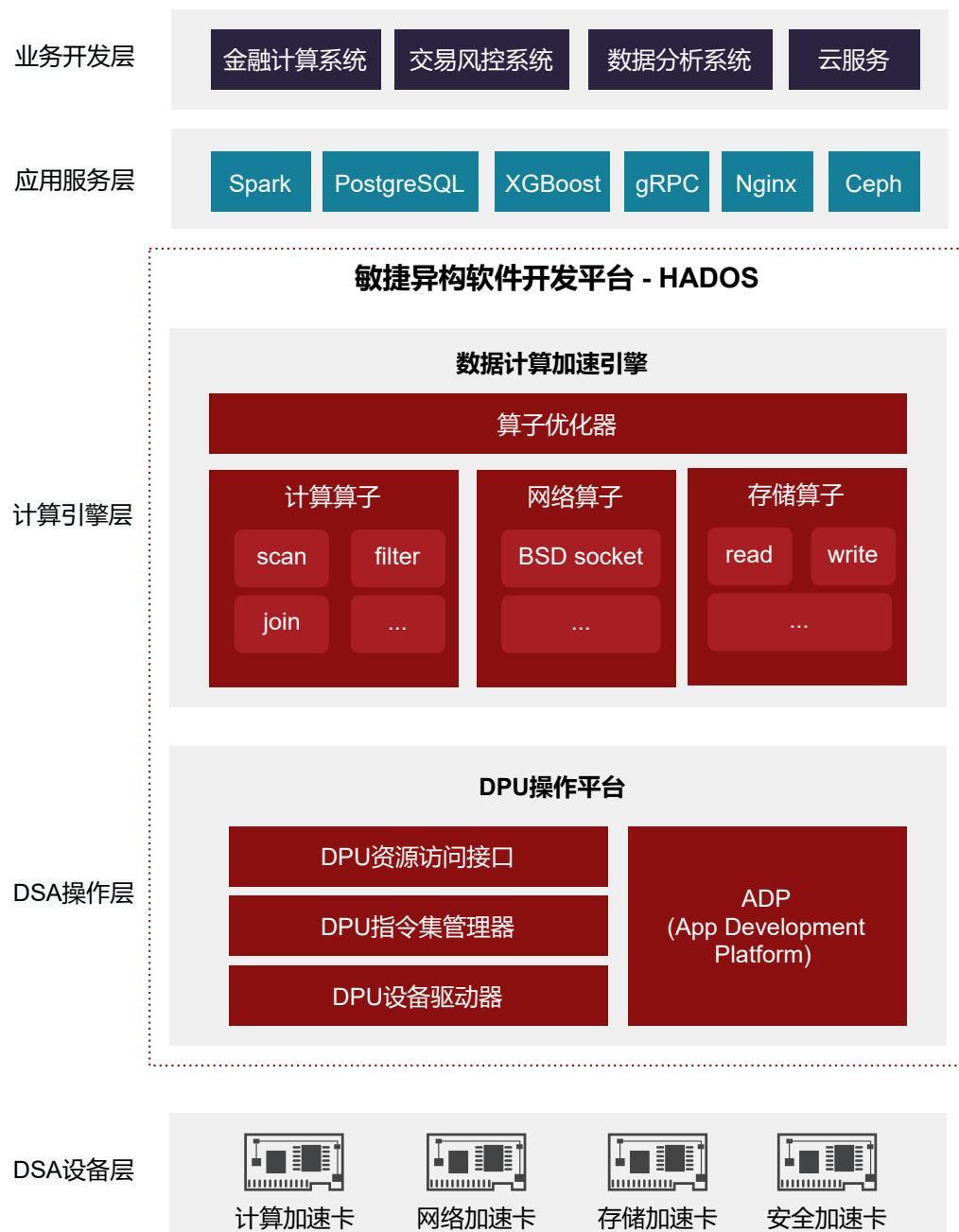


图 4-4 叟数异构计算系统架构

DAS设备层集成了驭数自研DPU芯片加速卡设备，分别是针对数据计算、软件定义网络、软件定义存储以及安全相关的硬件设备。这些硬件设备在物理上是由驭数DPU芯片、内存、存储、I/O通道和KPU指令集等共同构成；DSA操作层是一个为支撑基于DPU设备开发的软件操作平台，即DPU操作平台，该平台提供了DPU资源访问、调度、监控和管理的功能，同时也提供了完备友好

的开发工具库；计算引擎层为针对数据计算应用软件封装的一个通用的数据计算加速引擎，它基于DPU操作平台提供的计算能力，实现了硬件卸载的算子及相应算子的软件优化器，这些算子分为计算类算子、网络类算子和存储类算子；应用服务层为使用驭数DPU进行加速的数据处理平台、网络服务平台和存储服务平台，如大数据领域的Spark，数据库领域的PostgreSQL，机器学习领域的XGBoost，微服务架构中负责远程通信的gRPC框架，Web服务平台Nginx以及分布式存储平台Ceph等；业务开发层为驭数数据计算加速产品所服务的业务系统，主要来自对计算、网络、存储性能敏感的领域，如金融计算、交易、数据分析、云服务等。在上述五层架构中，DPU操作平台与数据计算加速引擎共同构建了驭数为支撑DPU异构计算而打造的专用软件框架，即敏捷异构软件开发平台，Heterogenous Agile Developing & Operating System，简称“HADOS™”。

#### 4.3.4.DOCA，OneAPI与HADOS比较分析

整体来看，这三款软件框架的出发点是相似的，都是在异构计算的架构下提供一套通用且编程友好的软件开发框架。它们在设计这套软件框架时都采用了分层抽象的思路，目的是使各层职责清晰，对上层提供足够简单且功能完备的接口。

具体到功能实现上，它们都是在网络、计算、存储、安全这些范畴内做了相应的支持，不过由于它们待解决的问题及根本目的存在差异，所以在架构设计和功能实现上的侧重点有所不同。简单来说，DOCA侧重点是支持基于Blue Field DPU的数据中心基础设施服务加速的场景；OneAPI侧重点是构建设备无关的资源访问库和编程接口；HADOS侧重点是支持基于Yusur DPU的网络、计算、存储服务加速的场景。

表4-1针对这三款软件框架在设计目的、五层架构模型及各自特点上做了详细比较。

表4-1 DOCA, OneAPI与HADOS比较分析

对比维度		NVIDIA DOCA 架构	Intel OneAPI 架构	Yusur HADOS 架构
设计目的		配合 Blue Field DPU 实现对数据中心基础设施计算的加速	构建通用的异构计算编程模型与接口	为支撑 Yusur DPU 异构计算而打造的专用的软件框架
五层模型比较	DSA 设备层	Blue Field DPU 设备 (数据中心基础设施加速设备)	不提供设备，而是整合业界现有设备	基于 Yusur DPU 的网络加速设备，数据计算加速设备，存储加速设备
	DSA 操作层	通用：支持 RDMA, DPDK, PCIe 等； 网络：支持 ASAP <sup>2</sup> , SDN, OvS, Virtio-net, P4 programing 等； 存储：NVMe-oF, SPDK, Virtio-blk, 数据加解密和数据压缩等； 安全：Encryption、DPI、TLS、IPSec 等； 基础设施管理：支持 DPU management, Traffic telemetry, Packet filtering 等。	不直接实现操作层软件，而是整合其它平台的操作层 Libraries (如 CUDA, OpenCL 等)，构建统一的 L0 API 对设备层资源的访问进行封装。	通用：支持 LightningDMA <sup>TM</sup> , RDMA, DPDK, PCIe, SR-IOV 等； 网络：支持 SDN, OvS, Virtio-net 等； 存储：NVMe-oF, SPDK, Virtio-blk 等； 安全：Encryption、DPI、TLS、IPSec 等； 计算：Data computation offloading 开发工具：Compiler, DeepInsight <sup>TM</sup> (硬件日志), Monitor, IDE, DPUSimulator, DPU Profiler&Debugger
	计算引擎层	无	构建面向深度学习，数据分析，并行计算，视频处理等具体场景的 L1 API，包括 Deep Learning API、Data Science API、Data Analysis API、Multiple Thread API、Video Processing API 等。	网络加速引擎：整合并支持 BSD Socket, gPRC 等服务的网络传输加速。 计算加速引擎：整合并支持 Spark、Flink、Database 等计算平台的算子加速。 存储加速引擎：整合并支持 Ceph 等存储服务的数据读写加速。
	应用服务层	面向数据中心的基础设施应用。	面向深度学习，数据分析，并行计算，视频处理等相关应用。	面向网络服务，存储服务，大数据计算，数据库计算等相关应用。
	业务开发层	服务于数据中心的业务领域。	服务于深度学习，数据分析，并行计算，视频处理等业务场景。	服务于数据中心，金融计算，数据分析，机器学习等业务领域。
特点	侧重于网络加速卸载，主要服务于数据中心的基础设施应用的性能提升。除了通用的网络、存储、安全的支持外，DOCA 特别的针对基础设施管理提供了支持。	不提供硬件设备，为业界现有的异构计算设备构建统一的软件库和 APIs。	除了支持通用的网络，安全，存储加速外，还支持大数据和数据库的硬件卸载和加速。 抽象层次更高，在计算引擎层针对网络服务，数据计算平台提供更加友好的 API，并和业界受欢迎的应用平台做了集成。	

## 5. 业界产品概要介绍

### 5.1. NVIDIA BlueField

NVIDIA推出的BlueField系列DPU，在支持网络处理、安全和存储功能的同时，实现网络虚拟化、硬件资源池化等基础设施层服务。BlueField DPU既是一个承担高带宽（100Gbs/200Gbs/400Gbs）的网络处理器，同时也是一个独立的嵌入式处理器，它管理着众多加速器引擎，比如加密解密、正则表达式匹配以及存储加速等等。BlueField DPU也可以通过ARM核运行嵌入式Linux系统，处理一定控制面的任务，具有一定的通用能力。

BlueField DPU 可加速数据中心常用的 SDN、NFV、OVS、Overlay 网络（例如 VXLAN）、网络地址转换 (NAT)、自动负载平衡、细粒度流量管理和内容分发网络。DPU 支持网络包处理语言 P4，提供 DPU 数据路径加速器中的编程能力。将这些需要占用CPU资源的网络处理卸载到DPU中，使释放后的CPU资源注重用户业务处理。

BlueField DPU 将存储与计算隔离，支持加速软件定义的弹性存储、NVMe over Fabrics、分布式纠错和数据压缩。DPU 实现 NVMe SNAP 技术可向主机提供远程块存储，等同于本地 NVMe 块存储，具有低延迟、高吞吐量和高 IOPS 等特性。

BlueField DPU 通过硬件卸载实现安全服务，其中包括防火墙、微分段、使用透明 IPSec 和 TLS 进行动态数据在线加密以及入侵保护。

对于基础设施层管理，与在每台服务器上运行代理的传统管理不同，BlueField DPU 通过 Netflow 等方式，可以监测到网络流量的变化，分析网络拥塞原因等。DPU 通过这些方式，增加 CPU 用于业务处理的时间，实现对服务器的管理。

### 5.1.1. BlueField 系列硬件架构

NVIDIA 2020年推出BlueField-2系列 DPU，并计划在2022年推出性能更强的BlueField-3 DPU。

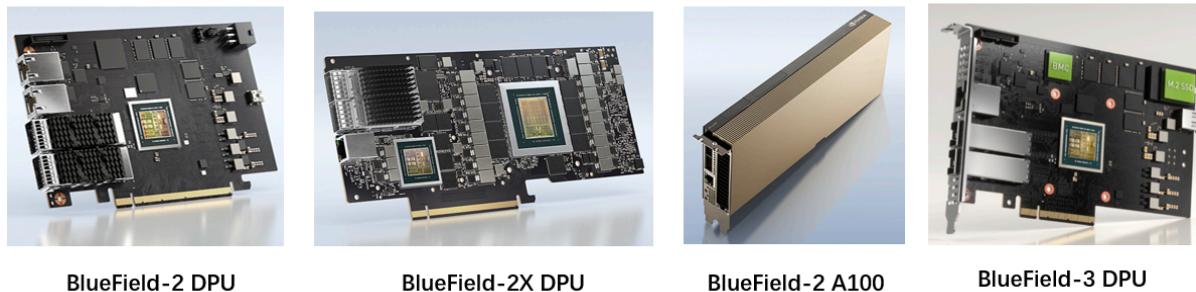
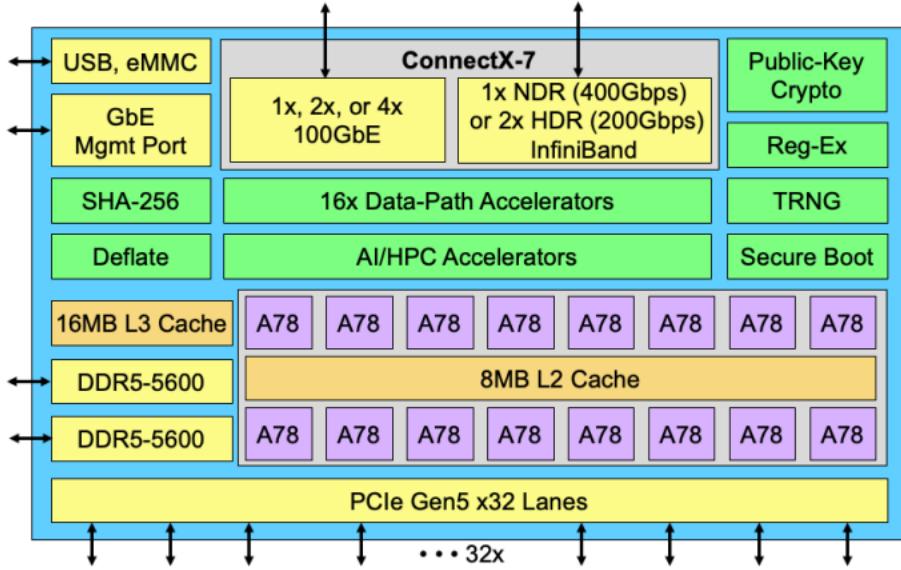


图5-1 BlueField 系列DPU<sup>13</sup>

BlueField-2 DPU具有ConnectX-6的网络处理功能，可支持高速以太网（200Gb/s）或 InfiniBand两种接口，八个ARM核，高带宽DRAM和PCIe交换机，通过高速Mesh网络连接在一起。包含支持网络、存储、加密、流媒体等计算的专用加速器，同时具有面向安全、虚拟化、硬件隔离和远程管理的功能。

BlueField-2X DPU相比于BlueField-2 DPU，增加了对AI功能的支持，融合NVIDIA Ampere 架构的GPU并行处理能力与BlueField-2 DPU的数据处理能力。BlueField-3 DPU是第三代NVIDIA DPU。与BlueField-2 DPU相比，设计支持400Gb/s以太网或NDR InfiniBand网络连接，也可以卸载、加速和隔离软件定义的网络、存储、安全和管理功能，从而提高数据中心的性能、效率和安全性。

<sup>13</sup> <https://www.nvidia.cn/networking/products/data-processing-unit/>

图5-2 BlueField-3 架构图<sup>14</sup>

### 5.1.2.DOCA软件栈

NVIDIA提供专用软件开发平台DOCA，支持使用行业标准API在BlueField系列DPU上快速创建网络、存储、安全、管理以及AI HPC等应用和服务。利用DOCA可以使编程过程忽略硬件细节，同时保证兼容性。通过软硬件协同，发挥DPU的性能优势。

在DOCA和DPU上开发的应用程序会将数据中心基础服务与应用程序隔离开来（如图5-3所示），即卸载到DPU之中，使得CPU用于业务应用程序，提升应用程序的性能和效率。数据中心分能够划分为应用程序处理域以及一个独立的数据中心基础设施服务域，实现功能隔离。例如，如果主机遭受入侵，安全控制代理与被入侵主机之间的隔离层可防止攻击扩散至整个数据中心。

<sup>14</sup> DPU-Based Hardware Acceleration: A Software Perspective <https://resource.nvidia.com/en-us-linely-whitepaper>

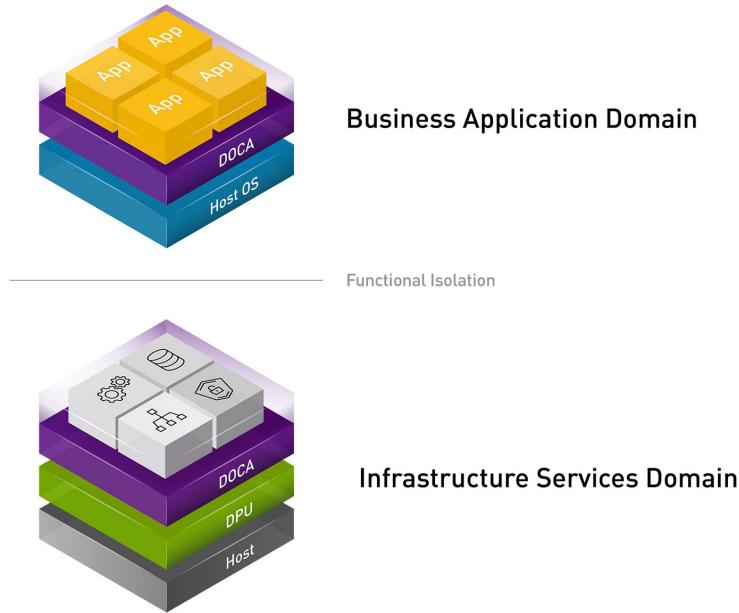


图5-3 应用程序与数据中心基础服务隔离<sup>11</sup>

DOCA详情设计参考第四章第三节技术介绍。

## 5.2. Intel IPU (Mount Evans)

IPU (Infrastructure Processing Unit) 是一种具有强化加速和以太网连接的网络设备，可使用紧密耦合的专用可编程内核来加速和管理基础设施<sup>15</sup>。IPU提供完整的基础设施卸载，并通过充当运行基础设施应用程序的主机的控制点来提供额外的安全层。通过使用IPU，可以从服务器卸载与运行基础设施任务相关的开销。云服务供应商（cloud service provider, CSP）软件在IPU本身上运行，而租户的应用程序在服务器CPU上运行。这不仅释放了服务器上的资源，同时优化了整体性能，而且为CSP提供了一个单独且安全的控制点。

IPU将基于硬件的数据路径（包括 FPGA）与处理器内核相结合，使得基础设施以硬件的速度处理，能够跟上不断提高的网络速度和软件实现控制平面

<sup>15</sup> 本节图片及技术规格出处均参考自Intel相关产品白皮书  
<https://www.intel.com/content/www/us/en/products/docs/programmable/ipu-based-cloud-infrastructure-white-paper.html>

功能的灵活性。IPU具有以下三个优势：基础设施功能和租户工作负载物理分离允许用户完全控制CPU；供应商可以将基础设施任务完全卸载给IPU，有助于提高CPU资源利用率，最大化收益；实现完全无磁盘服务器架构的云数据中心。

### 5.2.1. IPU硬件架构

随着基础设施和租户业务的物理分离，通过加速器可以有效地卸载基础设施功能，并将其转移到真正的无磁盘架构。Intel认为IPU将成为未来数据中心架构的核心组件，在2021年的Intel Architecture Day上，Intel推出了基于FPGA和ASIC的两种实现方式的产品。其中，Oak Springs Canyon和Arrow Creek是针对云和通信的基于FPGA的IPU产品，Mount Evans是基于ASIC的IPU产品。目前，相关产品的技术细节还未对外公布，仅简要概括其技术特点。

#### 基于FPGA的IPU

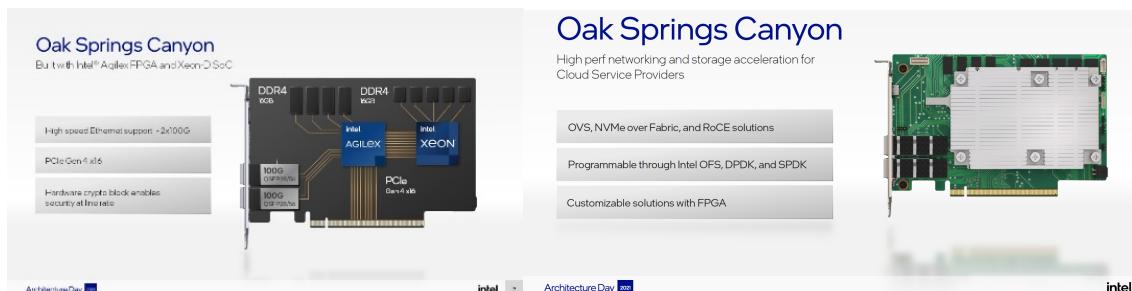
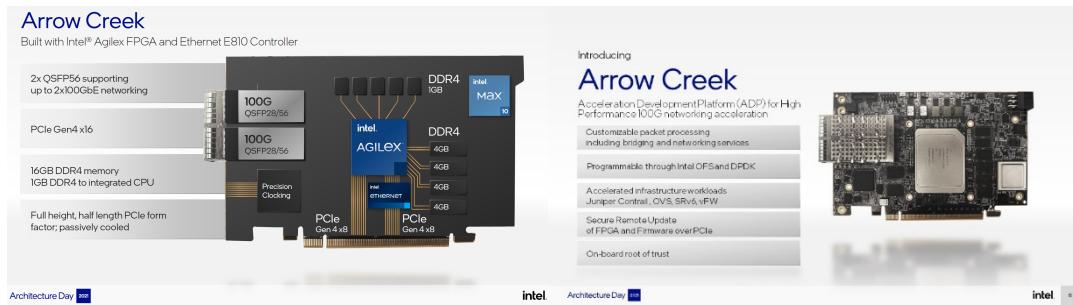


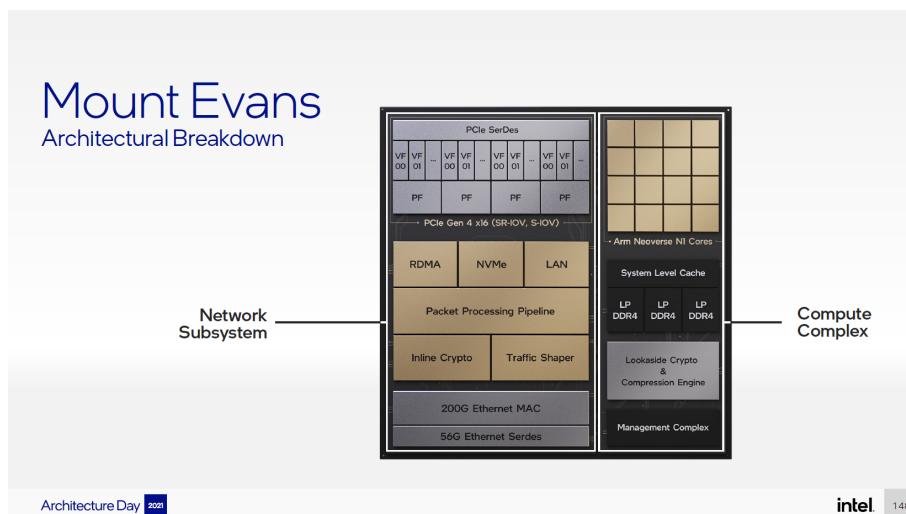
图5-4 Oak Springs Canyon IPU<sup>15</sup>

Oak Springs Canyon基于Agilex FPGA和Xeon-D CPU实现。其中，Agilex在功耗、效率和性能方面是业界领先的FPGA，与基于Xeon的服务器级CPU协同工作，能够支持2x 100G的工作负载。同时，该款产品还围绕x86优化了软件生态系统，使用Intel OpenFPGA堆栈（是一个可扩展源可访问的软件和硬件基础设施堆栈）。Oak Springs Canyon符合下一阶段100G CSP部署的需求，具有一个加固的加密块，允许以线速率性能确保基础设施流量、存储和网络的安全。

图5-5 Arrow Creek<sup>15</sup>

Arrow Creek是一个基于Agilex FPGA和E810 100G以太网控制器的加速开发平台。它建立在英特尔N3000 Pack的基础上，该Pack目前已部署在全球一些顶级通信服务提供商。Arrow Creek将帮助电信供应商提供灵活的加速工作负载，如Juniper Contrail、OVS和SRv6。

## 基于ASIC的IPU

图5-6 Mount Evans<sup>15</sup>

Mount Evans是Intel第一颗基于ASIC的IPU，通过PCIe最高可以链接四个Xeon处理器，并将其中的计算负载卸载至IPU中进行处理。Mount Evans具有Intel认为是该类中最好的包处理引擎，它支持大量现有用例，如vSwitch卸载、防火墙和虚拟路由，并为未来用例提供了重要的空间；通过扩展英特尔久经考验的高性能Optane NVMe控制器创建的，Mount Evans能够模拟NVMe设备；

Intel与CSP合作伙伴共同创新推进下一代可靠的传输协议，以解决有损网络上的长尾延迟问题。

Mount Evans分为网络系统和计算系统。

网络系统侧支持连接四台Xeons主机，带宽可达200Gb/s，利用ROCEv2技术实现RDMA协议；Intel的Optane衍生NVMe引擎将高性能NVMe设备公开给主机处理器，使基础设施提供商能够使用IPU来实现其存储协议，无论是通过Fabric的硬件加速NVMe还是计算系统的自定义软件后端；可编程包处理器为vSwitch卸载、防火墙、遥测功能等用例提供支持，同时可支持高达每秒2亿个包的性能；Mount Evans提供内联IPSec来保护通过网络发送的每个数据包。

计算系统建立在使用N1 Ares核心的ARM Neoverse架构上，并配有由三个LPDDR4控制器支持的大型系统级缓存，支持基于QAT技术的后备加密和压缩引擎；通过双核管理处理器提供了一个到平台和编排层的接口，支持健壮的系统可管理性。

## 5.2.2. 软件栈

第四章第三节中介绍了Intel提出的统一编程模型OneAPI，支持跨平台编程，如CPU、GPU、FPGA及专用加速器等硬件设备。截止目前披露的技术信息，Intel没有针对DPU提出专门的软件栈，但已实现对P4语言、DPDK等功能的软件支持。利用收购Barefoot获得的技术许可，Intel推崇了P4语言在业内的使用，并作为将网络数据平面编程集成到IPU上的标准框架，提高硬件的灵活性和可编程性。Intel还将扩展DPDK和SPDK等软件开发工具包，以利用IPU的数据和存储处理能力。

### 5.3. Marvell OCTEON

2021年6月28日Marvell发布了基于5nm工艺的OCTEON 10系列DPU，搭载了算力强劲的ARM Neoverse N2核，并且配备了多种硬件加速模块，包括加解密、包处理及人工智能推理加速器（如图5-7所示）<sup>16</sup>。同时为了提供用户友好的可编程接口，提高应用开发效率，Marvell配套硬件设计了包括DPDK、Marvell ML toolchain等在内的软件开发平台。

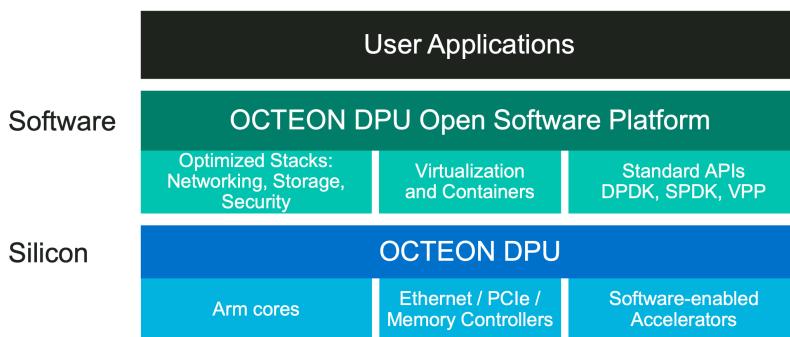


图5-7 Marvell OCTEON 10系列DPU<sup>16</sup>

OCTEON 10子产品包括CN103, CN106, CN106S 和DPU 400，不同子产品间的主要区别在于集成的ARM Neoverse N2核数，核数越多，性能越强的同时功耗也相对越大(不高于60W)。目前OCTEON 10 系列DPU产品形式主要为芯片及其配套开发平台(如图5-8)，开发平台预计于2021年第四季度面世。四款子产品仅公布了其核心技术特征，具体产品细节及形式尚未披露。

<sup>16</sup> Press deck of Octeon 10 DPU family.

<https://www.marvell.com/content/dam/marvell/en/company/media-kit/octeon-10/marvell-octeon-10-media-deck.pdf>



图5-8 OCTEON 10 DPU芯片(左)及其开发平台(右)<sup>17</sup>

### 5.3.1. OCTEON 10硬件架构

为了兼顾高性能和灵活性，将部分功能固化成硬件逻辑特性模块，如沿多核智能网卡技术路线发展而来的NIC子系统；同时也支持可编程芯片共性模块来处理其他计算，如ARM CPU。具体从硬件角度来看，OCTEON 10系列DPU整体可以分为高速I/O接口、计算核心和硬件加速逻辑三大部分(如图5-9)。

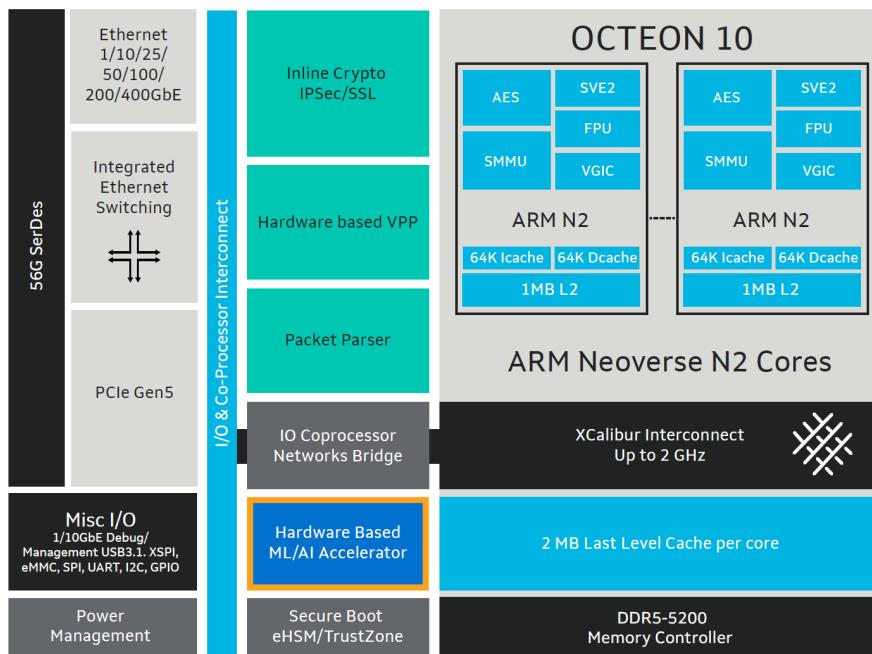


图5-9 OCTEON 10系列DPU框架<sup>18</sup>

<sup>17</sup> OCTEON 10 Press Kit. <https://www.marvell.com/company/media-kit/octeon-10-press-kit.html>

<sup>18</sup> Product Brief of OCTEON 10 DPU. <https://www.marvell.com/content/dam/marvell/en/public-collateral/embedded-processors/marvell-octeon-10-dpu-platform-product-brief.pdf>

OCTEON 10系列DPU配备了一系列业界先进的硬件设备及标准，如DDR5、PCIe Gen5和支持56G SerDes的高速以太网接口等，以支持高速率的数据传输。

## 计算核心

计算核心采用64位的ARM Neoverse N2核，至多可以同时部署36个，每个核心配备有1MB的L2 Cache和2MB的L3 Cache (Last Level Cache)。在OCTEON 10系列DPU发布仅两个月之前，N2核作为首款ARM v9架构CPU推出，相比于前代N1，性能有40%左右的显著提升，达到了业界最高的SPECint测评标准。N2核为OCTEON 10 系列DPU在许多方面的优势提供了强有力的支撑，如在高速I/O接口方面，N2核内部的CMN(Coherent Mesh network) 700提供了连接CPU和缓存、IO以及其他加速模块的高速互联解决方案，支持PCIe Gen 5以及DDR5等；除OCTEON 10专门设计的ML/AI硬件加速引擎外，N2核自身针对ML/AI任务有专门优化的向量指令，同时支持Bfloat 16，在进行机器学习推理时无需进行数据格式的转换，因此可以与ML/AI 加速器配合，共同承担ML/AI加速任务。由于N2核集成工艺的要求，OCTEON 10系列DPU采用5nm工艺。

## 硬件加速单元

硬件加速逻辑主要包括ML/AI加速引擎、包处理速率提升5倍的VPP (Vector Packet Processor)硬件加速器、1Tb速率的Switch、加解密模块、高度可编程化的包处理单元、Secure Boot、PUF (Physically unclonable function)和PCIe DMA 加速。其中ML/AI加速引擎是首次在DPU中设计实现，通过软硬件结合的方案，在恶意流量监测、QoS等具体案例中实现了比软件超百倍的性能提升。

具体来看，同常规神经网络加速器类似，ML/AI硬件加速模块不支持训练阶段，而将重点放在推理阶段（其结构如图5-10所示）。部署了一系列推理tile，包含私有SRAM（总计8MB）和支持INT8与FP16的MAC (Matrix-Multiply-Add Computation)，减少了网络节点间的数据搬运耗时，所消耗功耗低于2W。

目前未披露独立使用硬件加速模块或者N2核的案例，同时尚不清楚ML/AI加速器是否为一款通用的ML/AI加速器，或者是为采用不同算法的不同领域配置了不同的加速模块，以及是否可配置、可硬件编程。

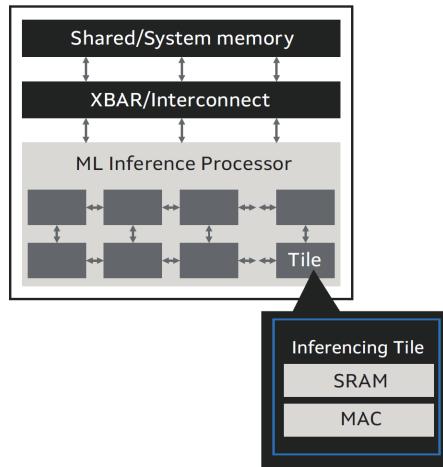


图5-10 OCTEON 10系列DPU中的ML/AI硬件加速模块<sup>16</sup>

### 5.3.2. 软件栈

Marvell基于开放标准和API，形成了一套多产品线统一的软件开发平台（如图5-11所示），支持软件在OCTEON和非OCTEON平台（包括x86）上的重用。核心平台基于标准Linux环境和用户空间DPDK，任何DPDK、Linux或控制平面应用程序都可以在基本SDK的基础上无缝编译，几乎不需要修改。在核心平台之上增加了其他扩展功能，对于特定应用程序，Marvell提供了针对OCTEON系列DPU预先优化的模块，以达到快速构建复杂应用程序的目的。该架构兼容裸金属和虚拟化云实现，无论是在裸金属、容器还是虚拟机上，都能以极快的速度运行控制平面和数据平面应用程序。

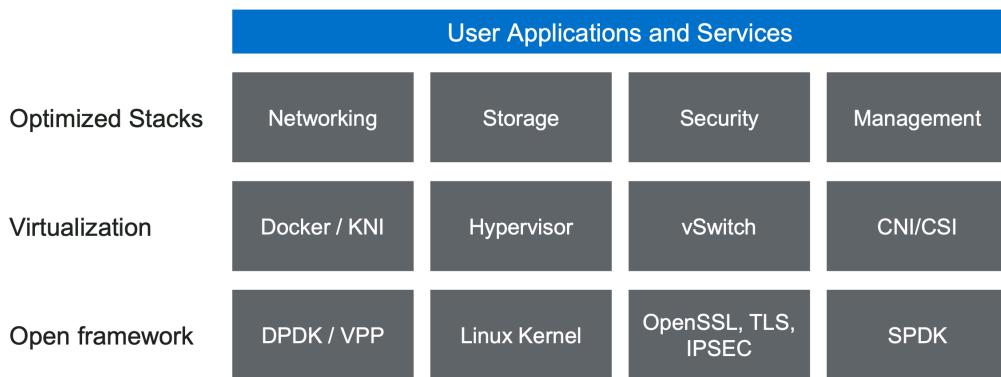


图5-11 MARVELL OCTEON 10系列DPU软件支持<sup>16</sup>

针对OCTEON 10系列DPU的ML/AI加速引擎，Marvell设计了新的机器学习软件套件，用于在ARM Neoverse N2 CPU和OCTEON 10 ML推理加速引擎上编译和执行机器学习模型，支持通用的机器学习格式和框架。Marvell ML工具链进行了平台针对性优化并集成到ML编译器框架（如TVM和GLOW）中，因而在这些框架中开发的机器学习模型代码可以很容易地被编译，然后部署到目标硬件上，或在Marvell虚拟平台上进行测试和调整。

## 5.4. Fungible DPU

针对以数据为中心（data-centric）应用的处理，Fungible研发了F1 DPU处理器和TrueFabric互联技术。TrueFabric是由Fungible首先提出的新型大规模数据中心网络互联标准，这种Fabric互联协议基于标准的UDP/IP/Ethernet协议栈构建。RoCEv2是一种当前数据中心网络中主流的互联网络协议，该协议同样基于UDP/IP/Ethernet搭建，对终端提供高性能的RDMA Read/Write服务，而TrueFabric对接入点提供高性能的Send/Receive服务。Fungible F1 DPU原生支持TrueFabric，因此F1 DPU可以用于大规模TrueFabric数据中心网络，不同类型的服务器都可以将Fungible DPU作为网络接入点。

### 5.4.1. F1 DPU硬件架构

图5-12是F1 DPU的架构示意图。在数据面设计上，F1集成了8个数据集群（Data Cluster），其中每个Data Cluster包括6个4线程的核心，通过将每个线程调度到专用的硬件加速单元，可以完成数据搬移、查找、安全、规约、保护以及分析的卸载加速任务。

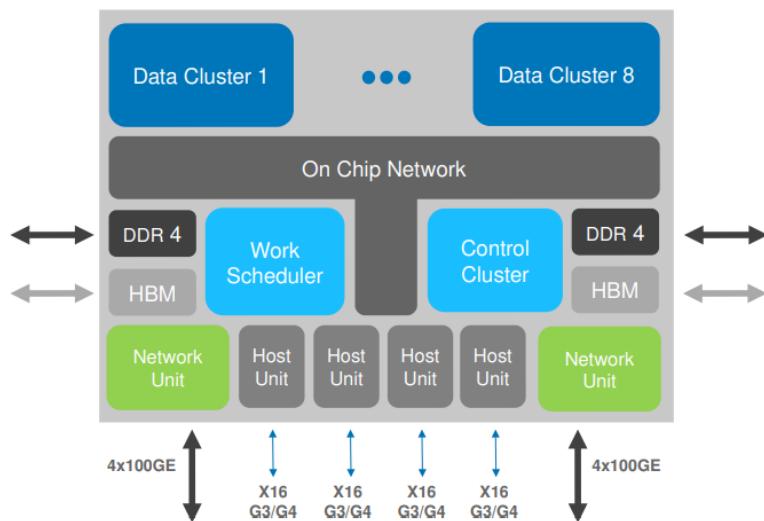


图5-12 Fungible F1 DPU硬件架构<sup>19</sup>

#### (1) 控制面

在控制面设计上，F1集成了1个控制集群（Control Cluster）用于执行控制面Linux，还包括一个工作调度器（Work Scheduler）。Control Cluster由CPU组成，包括4个双线程的核，支持安全启动、安全密钥生成，二进制签名和验证，集成了公钥（RSA和Elliptic Curve）加密模块、真随机数生成模块以及物理不可克隆模块，这些安全模块保证了控制平面的安全可信执行。在CPU处理器设计上，F1 DPU采用了MIPS-64指令集，9级流水线，双发射，4xSMT超线程，包括浮点计算单元FPU和向量单元SIMD。在性能上，对于data-centric应用时，F1 DPU 每个时钟周期执行的指令数（IPC）接近服务器通用CPU的最大值，同时提供完全

<sup>19</sup> The Fungible DPU™:A New Category of Microprocessor for the Data-Centric Era, HotChips 2020  
[https://www.hc32.hotchips.org/assets/program/conference/day2/HotChips2020\\_Networking\\_Fungible\\_v04.pdf](https://www.hc32.hotchips.org/assets/program/conference/day2/HotChips2020_Networking_Fungible_v04.pdf)

的硬件虚拟化支持，支持快速的线程切换，具备完整的缓存机制，提供全系统的一致性支持。在灵活性上，Control Cluster上集成的CPU给F1 DPU提供了完全的通用可编程能力，支持用户使用ANSI-C风格的C语言进行编程。

## (2) 网络I/O

在网络I/O处理上，F1 DPU提供了总共800G的带宽，实现了TCP/UDP、RDMA over TCP以及TrueFabric端点的卸载，支持前向错误纠正（Forward Error Correction, FEC）的MAC，集成了L2/L3/L4的数据包转发逻辑，支持诸如OvS等通用虚拟化协议卸载，IEEE1588精确时间校准协议（Precision Time Protocol, PTP），所有的数据包都可以使用AES-GCM算法进行线速加密。在网络数据路径的灵活性层面，用户可以使用P4语言对F1 DPU的数据包处理路径进行编程，实现对数据包的自定义解析，封装，解封装，以及收发协议的加速。

## (3) 主机交互

在和主机侧交互层面上，F1 DPU集成了64个PCIe Gen3或32个PCIe Gen4通道。值得注意的是，F1 DPU的PCIe通道可以用于16个双模式（RC或EP）的PCIe控制器，这种设计极大地提升了F1 DPU的互联灵活性。当F1 DPU作为X86或者ARM CPU的EP时，F1 DPU的PCIe控制器提供了64个PF（Physical Function），1024个VF（Virtual Function）以及细粒度QoS保证的硬件虚拟化支持；当F1 DPU作为RC时，可以将其与SSD、GPU以及FPGA进行互联。

在存储层面上，F1 DPU集成了总共8GB的高带宽内存（High Bandwidth Memory, HBM）以及高达512GB的双通道DDR4。在片上互联层面上，F1 DPU使用了片上网络在不同模块以及片外内存之间传递控制消息和数据。

### 5.4.2.FunOS软件栈

Fungible开发了FunOS，其结构示意图如图5-13所示。FunOS用于DPU数据路径的管理，给服务器CPU提供异步调用F1 DPU数据面加速器的编程接口，包

括网络、存储、虚拟化、安全以及数据分析功能函数，以PCIe VF的形式暴露给服务器Linux操作系统。在控制面的Linux操作系统中，运行网络、存储以及安全的控制面，并且作为服务器Hypervisor的Guest。

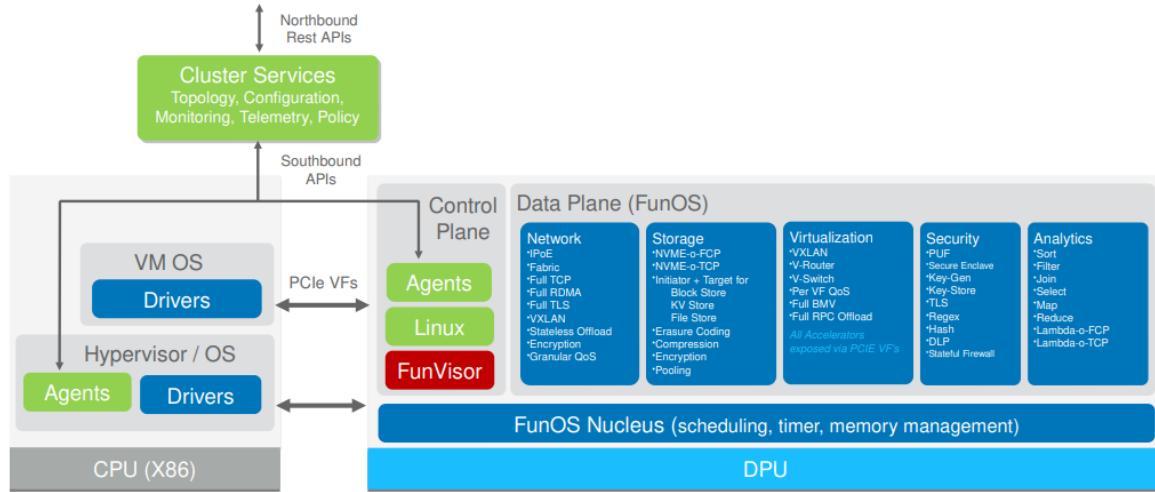


图5-13 Fungible F1 DPU软件架构<sup>19</sup>

FunOS提供的异步编程接口如图5-14所示。在传统的方式中，服务器端CPU CORE #1将一个任务调度到硬件加速器上执行的过程中，需要阻塞等待硬件加速器的执行完成；在FunOS提供的编程模型中，CORE #1可以在Job #1执行的过程中，切换上下文到Job #2，当Job #1在硬件加速器上执行完成后，在CORE #2上继续执行，这种异步的调用硬件加速器的方式提高了CPU利用率。

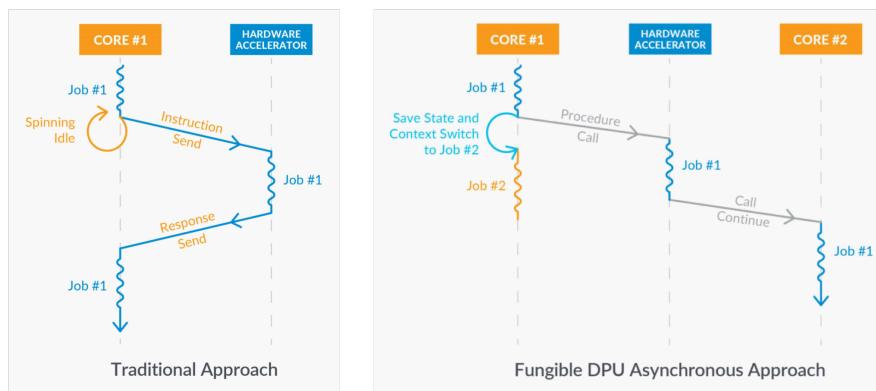


图5-14 Fungible DPU异步编程接口示意图<sup>20</sup>

<sup>20</sup> Tuesdays with FUNGIBLE CTO: Part2 – Fungible DPU: Putting the fast in the datapath  
<https://www.fungible.com/2020/08/18/fungible-dpu-putting-the-fast-in-the-datapath/>

## 5.5. 中科驭数 K2 DPU

### 5.5.1. 中科驭数KPU硬件架构

核处理器（Kernel Processing Unit，KPU）是中科驭数原创的软件定义计算架构，专为加速特定领域核心功能计算而设计的一种协处理器架构。KPU以功能核作为基本单元，直接对应用中计算密集型应用进行抽象核和高层综合，实现以数据为中心的架构“定制”。KPU具有超强异构核集成和调度能力，一颗KPU根据需求可以集成数十至数百个功能核。在运行机制上，KPU采用“数据驱动并行计算”的方式，运行过程中通过数据流来激活不同的功能核进行相应计算。通过软件定义的方式用户可以灵活的建立“功能核”与应用层运算之间的关系，从而实现“功能核”到运算需求的“一对一”服务，保证计算效率。且不同于FPGA在电路层的改造的性能牺牲，KPU的核心技术在功能核层，功能核来自于对于计算模式的抽象，并将其IP化。通过高层次综合，既实现了领域内硬件的统一，降低了规模限制的硬件成本和设计周期，又能通过软件编程实现不同功能的计算；特定需求只需要增删功能核的种类和数量即可。在整体计算效率提升百倍的前提下，仍然具有非常高的可扩展性和灵活性。

目前中科驭数已经完成了四类KPU芯片架构设计，1) KPU-Swift针对网络协议处理设计；2) KPU-Conflux针对时间序列/大数据分析设计；3) KPU-Trusy针对安全领域处理设计，4) KPU-FlexFlow针对智能计算设计。并在5个应用领域积累了80余类功能核。

中科驭数在2019年完成第一代KPU芯片K1流片，针对序列数据处理及数据库/大数据分析而设计。集成了序列卷积tsconv、序列滤波tsfir、序列距离tsdist、序列相似tsdtw等20多类功能核。相比于传统软件解决方案，基于驭数K1的加速

方案在数据库/大数据分析，以及时间序列处理等业务场景中可获得超2个数量级的性能提升。

### 5.5.2. 基于KPU架构的K2 DPU芯片

中科驭数基于软件定义KPU架构打造了K2 DPU芯片，以异构众核为基础实现了以数据为中心的高性能DPU架构。

在数据平面集成了四类KPU处理引擎，具有极强的数据处理能力。整个芯片设计以数据为中心，集成自研FlashNOC™流式片上互联架构，可实现数百个处理核互联，在2TB/S数据带宽下保证零阻塞数据传输。辅以超大容量片上共享缓存，可以实现数百核间高效数据交互。同时集成了面向数据对象的数据管理系统DOMS™智能数据管理系统保证了全系统处理核间数据一致性，从而可实现多数据流高通量并行处理。在网络方面集成了2路10/25/100GE接口，主存储集成了四路DDR4，带宽可达50GB/S，容量达128GB。在控制平面集成了四核ARM Cortex-A72通用处理器核，为用户提供以Linux操作系统为基础良好编程性的开发环境。

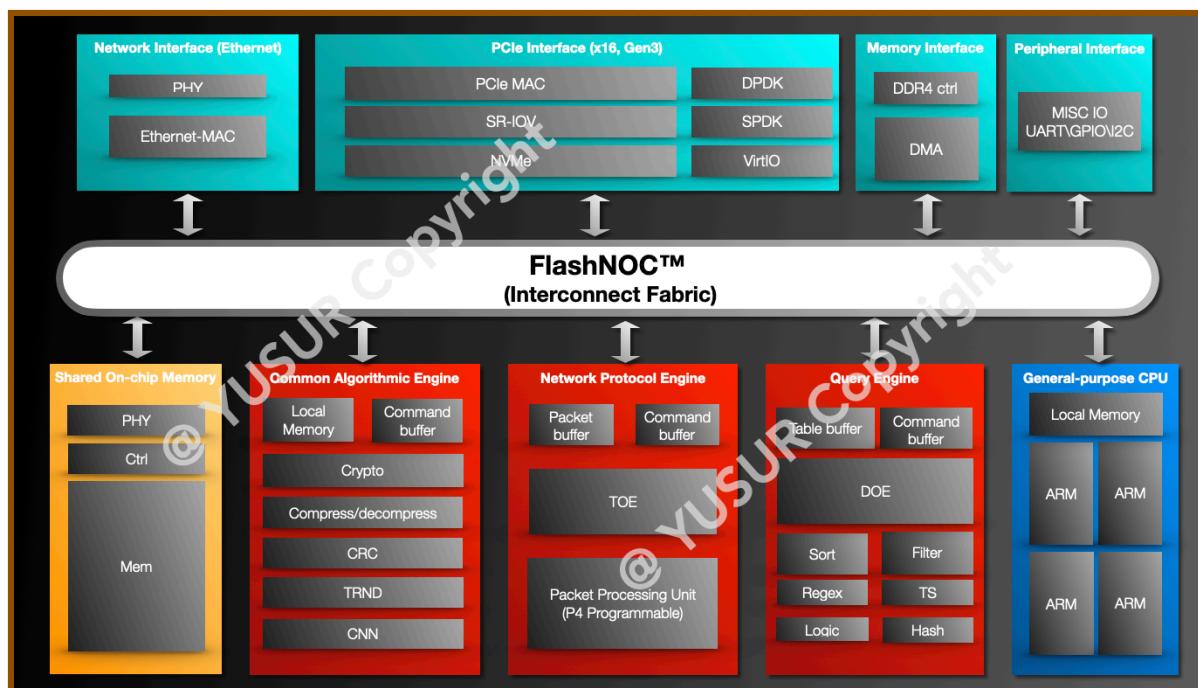


图5-15 K2 DPU顶层架构

## (1) 数据平面

数据平面基于驭数自研KPU处理架构实现，将从CPU上卸载下来的算法交由专用硬件加速模块DSA实现，以实现高性能和高可靠性。主要包括通用算法处理引擎、包数据引擎和Query处理引擎以及存储处理引擎。不同处理核之间借助了共享内存进行数据通信，各引擎之间不影响彼此之间的加速进程。

通用算法处理引擎是基于自研KPU-Trusty和KPU-FlexFlow架构实现的。主要算法包括通用加解密算法和国密算法，多种密钥交换算法，动态数据安全IPsec/TLS，ML算法和CNN等。

Query处理引擎基于自研KPU-Conflux架构实现，将OLAP数据库中最为核心的Sort、Filter、Join和Hash以及Regex/TS时间序列处理等操作分别进行了硬件加速，能够将SQL语句对应的Query Plan按具体组成部分直接映射到相应的硬件加速模块。可以用于日志分析监控，对数据中心流量做精细监控和有效识别，可以及时识别DDoS攻击。

网络协议处理引擎基于自研KPU-Swift架构实现，将L2/L3/L4层的ARP/IP/TCP/UDP网络协议处理、RDMA、数据包交换协议、基本网络虚拟化协议等从CPU上全部卸载，单kernel线速可以达到100G/S。使用P4实现网络数据平面可编程，兼容数据包处理，支持虚拟化网络协议和用于在整个网络中执行时序同步的1588协议。

## (2) IO系统

主存IO选用4通道DDR4，可以达到50GB/s的带宽。配以智能数据管理系统DOM<sup>TM</sup>（Data Object Management System）动态分配和回收策略，空间利用率高，实现了高效的内存管理。

系统IO方面，集成两组PCIe Gen3 x16，兼具RC/EP两种模式，灵活实现系统扩展和集成。EP类接口可用于X86/ARM主设备集成，RC类接口用于FPGA/GPU/SSD从设备扩展。在系统IO系统中集成驭数超低延时DMA方案

LightningDMA实现第三方加速平台X86/ARM、GPU、FPGA高效数据交换。系统IO中集成完备的SR-IOV和VirtIO的硬件设备虚拟化功能，并实现了超融合虚拟化方案，原生支持网络、存储、安全、计算等多类型设备，无需PCIe Switch切换，可实现资源动态分配，多设备共享，性能高，可扩展性好。

网络IO集成了2路 10/25/100GE接口，配以TOE专用引擎可实现64K超大数量连接，可实现400G线速网络包解析。

除此之外，还集成了MISC IO、UART、GPIO、I2C等丰富的外设接口，为开发调试提供便利。

### (3) 片上互联和数据管理

片上互联方面集成了RichFlow™和FlashNOC™两类互联架构。其中RichFlow是驭数提出的一种众核互联架构，可实现100s众核全互联，采用流式传输方式。RichFlow用于KPU处理引擎中各处理Kernel的互联，通过软件灵活配置实现多处理核组合，完成多种不同算法处理。FlashNOC是驭数提出的一种大数据流互联架构，可实现2TB/S带宽无阻塞数据交换，主要用于KPU处理引擎间，处理引擎和片上共享内存之间，以及处理引擎和主存之间的数据交换。

K2-DPU中集成了一套面向数据对象的数据管理系统DOMS™，主要负责两个功能：1) 存储管理，2) 数据对象管理。在存储管理上，DOMS动态对各存储（主存、片上共享存储）空间进行分配和回收，以实现高存储空间利用率。针对多主存通道进行存储带宽平衡，充分利用存储器带宽，可无缝扩展HBM和SSD等高带宽大容量存储。在数据对象管理上，动态对各数据对象记录和访问，实现全局数据一致性。DOMS分布在各处理系统和存储系统之中，通过私有协议确保百纳秒级数据一致性。

另外，在片上集成了128MB的超大容量共享存储，可提供1TB/S多核数据共享带宽，实现多核的高效数据交互。

#### (4) 控制平面

由于控制平面任务多样，灵活性要求较高，算力要求较低，驭数K2 DPU部署了4个ARM Cortex-A72通用处理器核。为便于用户统一管理和配置DPU设备，提供较好的可编程性，驭数K2基于标准Linux系统提供开发和维护环境。在控制平面上实现了驭数自研LightningDMA™底层驱动程序，可实现控制平面与数据平面超低延时交互。搭建了DPU KOS运行时系统，动态对DPU各资源进行监控。

针对多种应用框架进行了深度优化，并提供标准API接口，方便实现应用轻量化部署。其中包括BSD Socket, DPDK, SPDK, OvS等。除此之外，在控制平面上实现了完备的安全控制，如信任根，安全启动，安全固件升级等。

### 5.5.3. 中科驭数 HADOSTM 软件框架

HADOS (Heterogenous Agile Developing & Operating System) 是中科驭数推出的专用计算敏捷异构软件开发平台。基于自主研发的KOS (KPU Operating System) 和KLIB (KPU Kernel Library)，可支撑DPU芯片及其他异构计算硬件平台算力输出，可兼容标准软件应用生态，可大幅降低以KPU架构和DPU芯片为核心的应用软件开发难度，集合了中科驭数在领域专用计算架构、专用处理器研发及相关产品和应用经验。

基于HADOS软件开发平台，客户可以通过自有软件团队，快速开发DPU芯片等异构算力在软件定义网络、软件定义存储、虚拟化IO、安全、大数据运算等计算基础设施方向上的应用，大幅提升系统性能和效率，提升系统易用性。

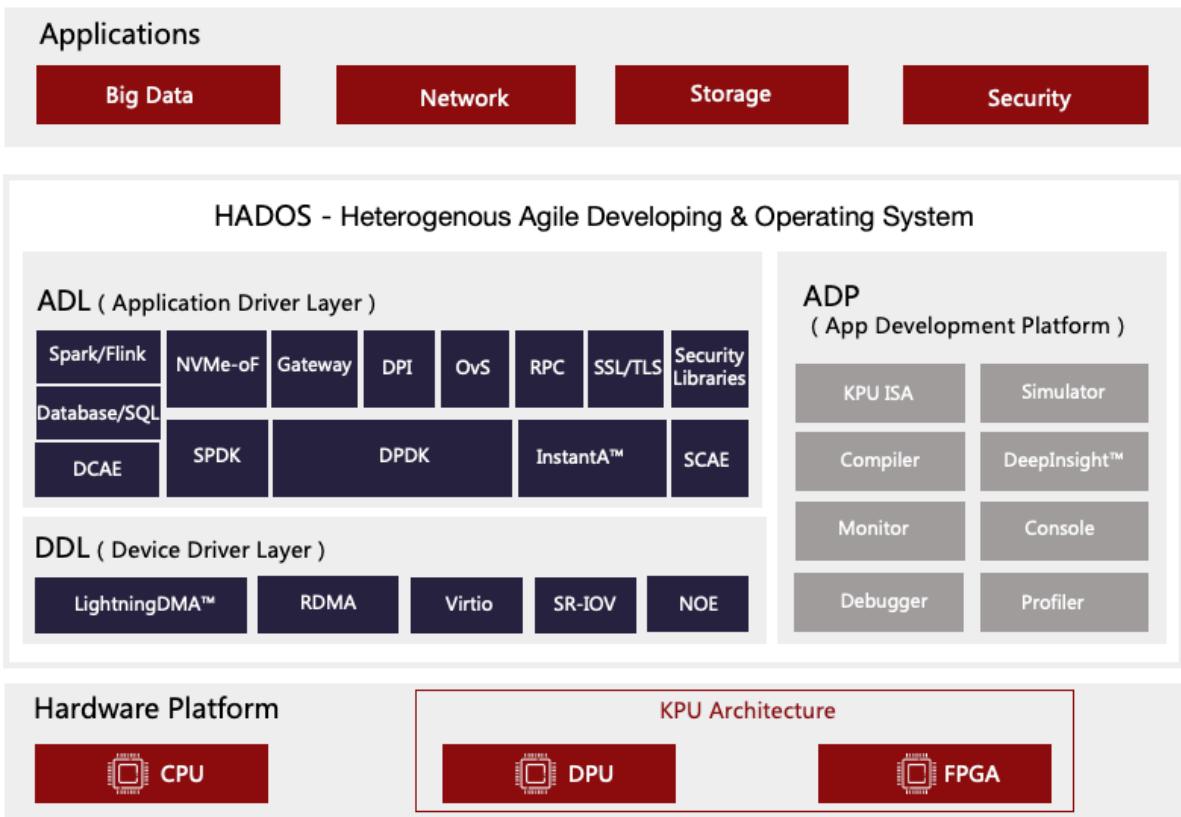


图5-16 HADOS™架构

如上图所示，HADOS的设计思路是以异构计算架构的五层模型为基础，实现了模型中DSA操作层和计算引擎层，并以SDK Library的形式为基于驭数DPU的软件应用开发提供友好的编程模型和接口。HADOS软件框架主要有三大核心组件，它们分别是设备驱动层DDL（Device Driver Layer），应用驱动层ADL（Application Driver Layer）和应用开发平台ADP（Application Development Platform）。

其中，DDL是五层模型中的DSA操作层，它通过对DPU设备的软件抽象，以low-level API的形式提供了DPU资源的访问能力，同时支持在网络、存储、安全和计算等领域的算力卸载，如网络领域的SDN, OvS, Virtio-net等；存储领域的NVMe-oF, SPDK, Virtio-blk等；安全领域的Encryption, DPI, TLS, IPSec等；计算领域的Data computation offloading等；以及各领域通用的LightningDMA™, RDMA, DPDK, PCIe, SR-IOV等。

ADL是五层模型中的计算引擎层，它基于DDL提供的算力卸载能力，结合应用软件的算力需求实现的高层次封装。针对不同的应用场景，ADL实现了相应的加速引擎组件，它们分别是：1) 网络加速引擎，支持BSD Socket (InstantA™)、分布式计算的RPC、SSL/TLS等服务的网络传输加速；2) 计算加速引擎 (DCAE, Data Computation Acceleration Engine) 支持Spark、Flink、Database等计算平台的算子加速；3) 存储加速引擎，支持Ceph等存储服务的数据读写加速；4) 安全加速引擎 (SCAE, Security Computation Acceleration Engine)，支持安全、加解密和可信根相关计算应用的加速。

ADP也是DSA操作层中不可或缺的一部分，它封装了开发和运维所必需的工具包，主要有Compiler，DeepInsightTM（硬件日志），Monitor，IDE，DPUSimulator，DPU Profiler&Debugger等。这些工具包在应用程序开发中的编码、编译、监控、日志、报警、诊断、测试等各个环节起到了重要的辅助作用。

## 6. DPU发展展望

工业和信息化部发布的《新型数据中心发展三年行动计划(2021-2023年)》中明确提出要加快提升算力算效水平，“推动CPU、GPU等异构算力提升，逐步提高自主研发算力的部署比例”，“加强专用服务器等核心技术研发”，“树立基于5G和工业互联网等重点应用场景的边缘数据中心应用标杆”等等。该行动计划也部分反映了DPU等新型算力芯片难得的历史发展机遇。虽然国内厂商在芯片产品化的环节还相比国外一线厂商还有差距，但是在DPU架构的理解上还是有独到的见解的，而且我国目前在数据中心这个领域，无论是市场规模还是增速，特别是用户数量，相较于国外都有巨大的优势。国内厂商有望充分利用这一“应用势能”，加快发展步伐，在DPU这个赛道与国外厂商逐鹿中原。

DPU的潜在市场非常巨大，预测到2025年仅中国市场就能达到每年40亿美元的规模，估计全球将超过120亿美元，但挑战与机遇并存。IaaS在国内云服务市场占比约60%，支撑了目前最重要的PaaS的容器云技术。未来几年，我国仍将维持IaaS为主的云计算结构，预计市场占比将逐上升到70%。目前要解决DPU标准化应用，还存在一定挑战。由于数据中心本身的复杂性，各大厂商一方面采用COTS组件来构建系统，追求低成本，一方面又设法分层服务化（IaaS, PaaS, SaaS），打造面向不同类型的客户的产品，但除此之外的所有技术实现几乎都是各家“八仙过海，各显神通”，如AWS有Nitro，阿里云有MOC。有的厂商强化IO能力、有的关注路由转发、有的重视存储卸载、有的关注安全加密——不一而足。例如各大公有云厂商、电信运营商等都有比较完整、也比较封闭的底层架构和应用生态。上层负载不同，必然对底层架构有各异的需求，这也许是目前DPU标准化面临的最大的挑战。

DPU作为一类专用处理器，与通用CPU的发展路径可能会有所不同。专用计算体系结构和通用计算体系结构的阵地是不同的，专用计算竞争的焦点是数据平面，而通用计算竞争的焦点是控制平面。专用计算好比是造赛车，目标就

是“快”，重点是根据赛道的类型来决定赛车的结构；通用计算好比是造民用军车，目标更加的多元化，不仅要兼顾不同路况下的可用性，还要考虑性价比、代际兼容性等等。所以，以通用CPU的标准来看待DPU可能并不合适，甚至会制约了专用DPU的发展。一个有商业价值的技术必须建立在“技术闭环”的基础上：锚定需求、研发、使用、反馈、再研发改进、再扩大使用范围……，即所谓“先垂直深耕，再水平扩展”的发展战略可能更适合DPU的发展。技术只有投入使用才能体现价值，有使用价值才有可能商业化，才能完成技术闭环到商业闭环的进化。技术闭环的形成需要集中火力打穿到应用才能铺就。碎片化并不是“专用”障碍，反而应该是专用技术路线充分利用的优势。

当然，传统的“one-size-fit-all”的ASIC商业模式，通过上量来摊薄芯片研发的巨额NRE成本本身还是有效的，所以专用DPU最终也要谋求“水平扩展”来覆盖更多的场景，还是要尽可能把各异的需求整合起来，并且适应不同厂商的数据中心架构，但这必将是一个长期而艰巨的任务。DPU肯定不算是一个“低垂的果实”，各个DPU厂商可能不能寄期望于当前“需求各异、体系封闭”的局面自发地在短期内变得“整齐划一，全面开放”，只能是在竞争合作的博弈过程中，逐渐满足越来越多的行业需求。放弃幻想，步步为营，“结硬寨，打呆仗”，这需要长期行业“Knowhow”的整合和持续的产品迭代。更需要上下游企业共同来构建良性、开放的生态环境，按照基础性技术研发的规律来研发DPU，面向网络、安全、存储、虚拟化等基础技术，划分好逻辑层次，利用好“软件定义”的思想，构造一个完整的DPU软硬件体系。而不是把DPU当成普通的算法加速器，只谋求解决一些碎片化的需求。

从目前行业的关注度来看，DPU带来的机遇已经基本形成共识，期待在这一趋势的驱使下，行业内的各个厂商协同起来，将DPU这一创新的产品早日赋能各行各业，成为新的生产力。

## FREE THE POWER OF DATA