SmartNICs and Data Processing Units (DPUs)

Bridging the Gap between Performance and Programmability

Sina Daneshgar & **Mohammadmohsen Abbaszadeh**

SinaDns & HisEgo

February 2026

Abstract

As cloud computing and data center workloads scale, the traditional CPU-centric architecture faces significant bottlenecks in networking and storage overhead. This project investigates the emergence of SmartNICs and Data Processing Units (DPUs) as a solution to these "infrastructure taxes." We provide a technical overview of DPU architectures, survey programming models such as P4 and eBPF/XDP, and present a simulation demonstrating the performance gains achievable through hardware offloading.

1 Introduction

The exponential growth of data center traffic has outpaced the gains in single-threaded CPU performance. Networking tasks such as encryption, load balancing, and packet inspection can consume up to 30% of host CPU cycles in modern cloud environments. SmartNICs and DPUs address this by moving these tasks to specialized networking hardware.

2 The Infrastructure Tax

In modern hyperscale data centers, a significant portion of compute resources is dedicated to management and utility tasks rather than user applications. This overhead, often termed the "Infrastructure Tax," stems from several key domains:

- **Virtualization**: The management of virtual switches (e.g., Open vSwitch) and encapsulation protocols like VXLAN or Geneve.
- **Storage**: Protocol translation for networked storage, such as NVMe over Fabrics (NVMe-oF).
- **Security**: Implementing distributed firewalls, micro-segmentation, and wire-speed encryption (TLS/IPsec).

When these tasks are executed on the host CPU, they compete for cache, memory bandwidth, and execution cycles, leading to "application stall" and reduced return on investment for hardware assets.

3 Evolution and Architecture

The network interface has evolved from a simple transport bridge into a complex compute node:

1. **Legacy NICs**: Fixed-function ASICs for L2/L3 transport with minimal offload capabilities (e.g., Checksum offload).
2. **SmartNICs**: Programmable devices based on FPGAs or SoCs, capable of offloading specific pipeline stages like OVS datapath or encryption.
3. **DPUs**: Fully integrated "Data Center on a Chip" that functions as an independent compute subsystem.

3.1 DPU Internal Architecture

A modern DPU (such as the NVIDIA BlueField-3 or AMD Pensando) typically consists of four primary components:

- **General-Purpose Compute Cluster**: Usually several high-performance ARM or RISC-V cores. These cores run a full Linux distribution, allowing the DPU to handle control-plane tasks and complex management logic.
- **Programmable Network Engine**: A high-throughput pipeline designed for packet parsing, match-action processing, and flow tracking at line rates (100–400 Gbps).
- **Hardware Acceleration Engines**: Dedicated silicon for computationally expensive operations, including:
  - **Cryptography**: AES-GCM and SHA acceleration for WireGuard, IPsec, and TLS.
  - **Storage**: VirtIO-blk/VirtIO-net acceleration and compression engines for storage disaggregation.
  - **Timing**: Support for Precision Time Protocol (PTP) for high-frequency trading and telecommunications.
- **Dedicated Memory**: On-board RAM (typically 16–32 GB) to store flow tables, connection states, and local application data without taxing the host's memory.

4 Programming Paradigms

The shift toward DPUs necessitates flexible programming models that can leverage hardware acceleration without sacrificing developer productivity.

4.1 P4: The Domain-Specific Language for Data Planes

P4 (*Programming Protocol-independent Packet Processors*) is a language designed to specify how packets are processed by the data plane. Key features include:

- **Protocol Independence**: The device is not hard-coded for specific protocols (e.g., IPv4, MPLS). Instead, the programmer defines custom header formats and parsers.

- **Match-Action Pipelines**: Logic is structured into tables that match specific header fields and execute actions (e.g., encapsulate, drop, increment TTL).

Our repository provides a sample P4_16 implementation in `examples/02-p4` which demonstrates basic Layer 2 forwarding logic on a programmable pipeline.

4.2 eBPF and the eXpress Data Path (XDP)

While P4 targets the hardware pipeline, eBPF (extended Berkeley Packet Filter) allows for safe, high-performance execution of code within the Linux kernel. XDP is a specific hook for eBPF that processes packets at the earliest possible point in the software stack:

- **Kernel Bypass without Bypass**: XDP runs before the `sk_buff` allocation, achieving performance comparable to DPDK while maintaining the security and usability of the Linux kernel.
- **Practical Application**: In `examples/03-ebpf-xdp`, we showcase a filter capable of dropping ICMP traffic at the driver level, significantly reducing the overhead compared to user-space firewalls.

5 Real-World Use Cases

DPUs enable architectural shifts that were previously inefficient or impossible using standard NICs.

5.1 Cloud Resource Disaggregation

In a traditional server architecture, storage and compute are tightly coupled within the same chassis. DPUs facilitate *disaggregation* by allowing storage to be pulled from a remote pool over the network while appearing as a local NVMe block device to the host OS. The DPU handles the NVMe-oF translation in hardware, minimizing latency and freeing the host CPU from storage stack processing.

5.2 Bare Metal as a Service (BMaaS)

Cloud providers utilize DPUs as an "out-of-band" management layer. By running the virtualization and security stack on the DPU (the "sidecar" model), providers can offer customers full "Bare Metal" access to the host CPU while maintaining strict isolation. Even if the guest OS is compromised, the DPU remains a secure enclave that the user cannot bypass.

6 Challenges and Limitations

Despite their benefits, the adoption of DPUs faces several hurdles:

- **Vendor Lock-in**: Unlike the CPU market, DPU programming environments are often fragmented. While P4 provides a degree of portability, many performance-critical features remain tied to vendor-specific SDKs (e.g., NVIDIA DOCA, Intel IPDK).
- **Power and Thermal management**: High-performance DPUs can consume between 75W and 120W. Integrating these into standard server slots requires careful consideration of the power delivery and cooling capacity of the data center.

- **Observability**: Debugging code running on a remote DPU cluster is significantly more complex than standard host debugging, requiring specialized tools for tracing and telemetry.

7 Performance Analysis

We developed a simulation to model the impact of DPU offloading on host CPU throughput. By shifting a fraction α of processing work to the DPU, the host can handle significantly higher packet rates.

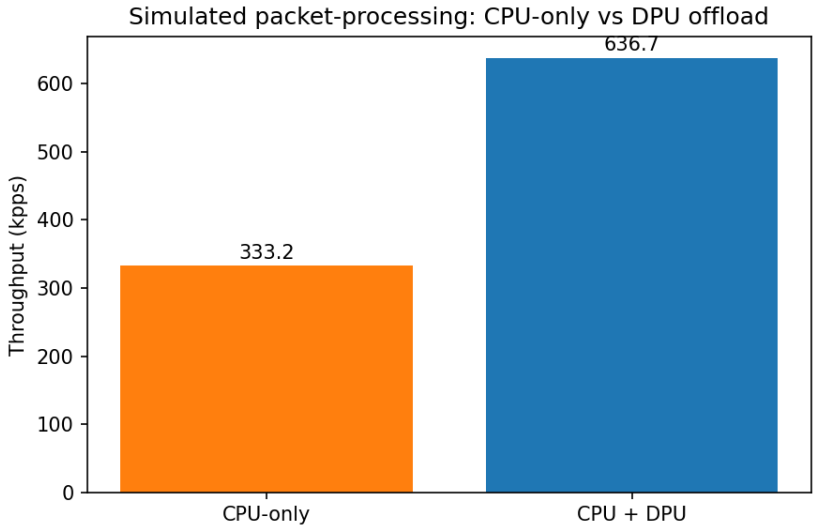

Figure 1: Simulated Packet Throughput: CPU-only vs. DPU-offloaded model.

8 Conclusion

The transition from fixed-function NICs to fully programmable Data Processing Units represents a fundamental shift in data center architecture. By offloading the "Infrastructure Tax" to specialized hardware, DPUs recover lost host CPU cycles for revenue-generating user applications. Our simulation confirms that even a partial offload of networking tasks can yield significant throughput gains. As programming standardizations like P4 and eBPF continue to mature, DPUs are poised to become the third pillar of compute alongside the CPU and GPU.

References

[1] P. Bosshart et al., "P4: programming protocol-independent packet processors," ACM SIGCOMM Computer Communication Review, vol. 44, no. 3, pp. 87-95, 2014.

[2] T. Høiland-Jørgensen et al., "The eXpress Data Path: Fast Programmable Packet Processing in the Operating System Kernel," Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies (CoNEXT), 2018.

[3] G. Liu et al., "A Survey of SmartNICs and DPUs: Architecture, Programming, and Applications," IEEE Communications Surveys & Tutorials, 2024.