

---

**| RESEARCH ARTICLE**

## **Architectural Paradigm Shifts in Data Center Servers: A Review of Workload Offloading, Security Isolation, and the Rise of Infrastructure Processing Units (IPUs)**

**Ishaan Gupta**

*Software Engineering Manager, AI Infrastructure, Google Inc (Sunnyvale, USA)*

**Corresponding Author:** Ishaan Gupta, **E-mail:** [guptaishaan@google.com](mailto:guptaishaan@google.com)

---

**| ABSTRACT**

In hyperscale cloud computing, general-purpose central processing units (CPUs) face a fundamental scalability bottleneck driven by the cessation of Dennard scaling, decelerating Moore's Law, and exponential growth in network I/O bandwidth (scaling from 100 Gbps to 400–800 Gbps). As distributed microservices, disaggregated NVMe storage fabrics, and massive machine learning clusters expand, host CPUs are increasingly burdened by the "datacenter tax" - consuming 20% to 35% or more of total compute cycles on software-defined networking (SDN), storage virtualization, packet cryptography, and hypervisor management. Concurrently, multi-tenant co-location introduces microarchitectural cache contention, tail latency inflation, and side-channel vulnerability surfaces. To address these limitations, modern server architecture has bifurcated into application compute domains and dedicated Infrastructure Processing Units (IPUs) and Data Processing Units (DPUs). This review article provides a comprehensive synthesis of hardware offloading and workload isolation paradigms across five distinct evolutionary generations - spanning legacy NICs, TCP Offload Engines (TOEs), advanced SmartNICs, FPGA accelerators, and sovereign System-on-Chip (SoC) IPUs and DPUs. We examine state-of-the-art hyperscale implementations, focusing on Google's Titanium architecture, the custom Intel IPU (Mount Evans / E2000) deployed across Google Compute Engine C3 instances, the hardware-accelerated Falcon transport protocol, and Google's open-source Packet Security Protocol (PSP). Furthermore, we provide an extensive comparative analysis of leading commercial architectures - including Intel IPU, NVIDIA BlueField-3, AMD Pensando Elba, and AWS Nitro - and evaluate the emerging transition toward autonomous, "CPU-less" server topologies enabled by Compute Express Link (CXL) coherent fabrics.

**| KEYWORDS**

Infrastructure Processing Unit (IPU), Data Processing Unit (DPU), Datacenter Tax, Hardware Offload, Workload Isolation, SmartNIC, Falcon Transport Protocol, Google Titanium, Zero-Trust Architecture, CXL Disaggregation.

**| ARTICLE INFORMATION**

**ACCEPTED:** 15 August 2026

**PUBLISHED:** 26 September 2026

**DOI:** 10.32996/jcsts.2026.8.9.14

---

### **1. Introduction**

Hyperscale cloud data centers are undergoing a structural architectural transformation driven by the physics of semiconductor scaling and the rapid acceleration of network line rates. For decades, server architectures followed a strictly CPU-centric paradigm: general-purpose host CPUs executed both customer application workloads and the underlying cloud infrastructure management stack. However, while single-thread CPU performance scaling has slowed dramatically following the collapse of Dennard scaling [1], datacenter interconnect bandwidth has scaled by orders of magnitude - progressing from 10 Gbps and 40 Gbps to 100 Gbps, 200 Gbps, 400 Gbps, and 800 Gbps [1], [3].

At 100 Gbps, a continuous stream of minimum-sized 64-byte Ethernet frames yields an arrival rate of 148.8 million packets per second (Mpps), giving a 3.0 GHz processor core only 20.1 clock cycles to process each frame. At 400 Gbps, this arrival window shrinks to 5.04 clock cycles, and at 800 Gbps it collapses to approximately 2.5 clock cycles (840 ps). Given that a single uncached DRAM access requires 60 to 80 ns (180–240 cycles), executing virtual switching, overlay encapsulation, firewall policy lookups, and cryptographic verification on host CPU cores via software drivers is mathematically and microarchitecturally unsustainable.

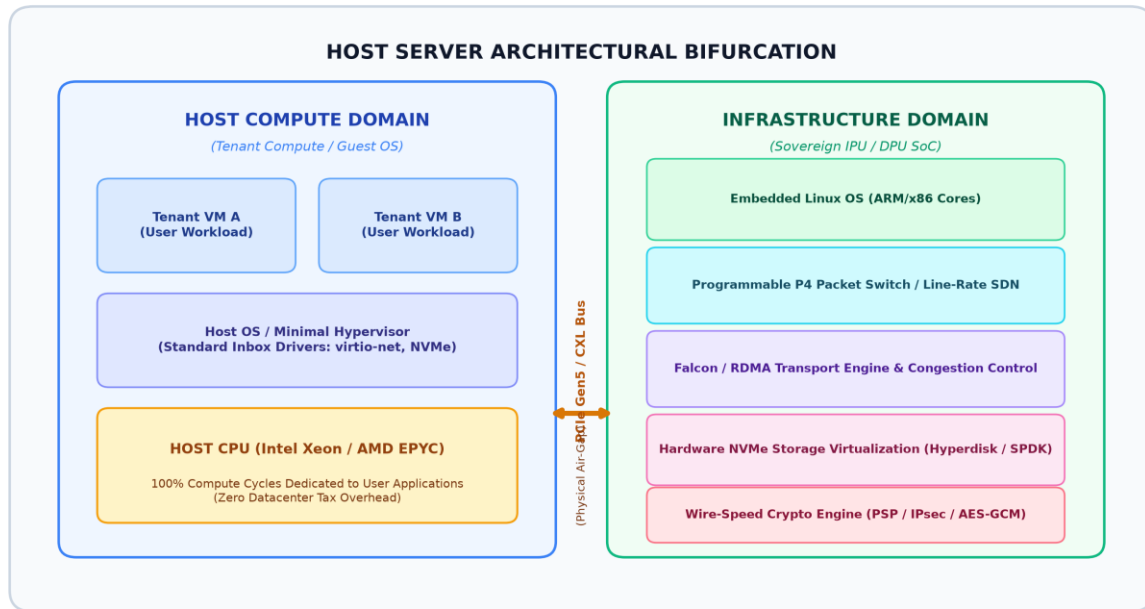


Figure 1: Architectural bifurcation of modern servers into host tenant compute domains and sovereign Infrastructure Processing Unit (IPU) domains across a PCIe Gen5/CXL bus.

This structural mismatch generates what is widely characterized across industry and academic literature as the **"datacenter tax"** [6], [8], [10]. In traditional virtualized and containerized environments, cloud providers must reserve 20% to 35%+ of available server CPU cores solely to run infrastructure overhead: virtual routing (e.g., Open vSwitch), software NVMe storage initiators (e.g., SPDK), transport protocols, security filtering, and hypervisor management [6], [10]. Beyond the direct loss of billable compute capacity, in-band infrastructure execution causes severe microarchitectural contention in shared Last-Level Caches (LLC), Translation Lookaside Buffers (TLB), and memory buses, degrading predictability and inflating tail latencies for latency-sensitive distributed workloads [2], [6], [11].

To eliminate this bottleneck, cloud providers and semiconductor manufacturers have introduced dedicated **Infrastructure Processing Units (IPUs)** and **Data Processing Units (DPUs)** [4], [5], [18], [19]. An IPU is an autonomous System-on-Chip (SoC) combining high-performance programmable network pipelines, multi-core embedded processors, dedicated hardware storage and cryptographic acceleration engines, and high-speed PCIe/CXL interconnects. By migrating the entire infrastructure datapath and control plane to an isolated hardware platform, the host CPU is fully unburdened and restored to executing tenant workloads.

This review article provides a comprehensive examination of the architectural evolution, design principles, and system-level trade-offs of modern IPUs and DPUs. We synthesize empirical findings from seminal cloud deployments, evaluate hardware offload mechanics across networking, storage, and security, contrast leading commercial silicon architectures, and outline the trajectory toward autonomous, "CPU-less" cloud architectures.

## 2. The Datacenter Tax and the Imperative for Hardware Offloading and Isolation

### 2.1. Quantifying the Datacenter Tax in Hyperscale Deployments

In classical virtualization architectures, the host CPU functions as a shared resource executing both revenue-generating guest virtual machines (VMs) and non-revenue cloud platform software. The platform stack encompasses:

1. **Software-Defined Networking (SDN):** Flow classification, packet parsing, header manipulation (VXLAN, Geneve), and access control lists (ACLs).
2. **Storage Virtualization:** Emulating local block devices over distributed remote fabrics (NVMe-over-Fabrics, SPDK drivers).
3. **Transport Protocol Processing:** Managing TCP/IP state machines, window sizing, timers, and congestion control.
4. **Data Encryption:** Executing authenticated cryptographic ciphers (e.g., AES-GCM) for data-in-transit protection.
5. **Host Management and Telemetry:** Collecting hardware health metrics, resource accounting, and orchestrating live migration.

As reported in large-scale studies by Google [6], Microsoft [8], and AWS [10], dedicating host CPU cores to these tasks severely degrades overall datacenter efficiency. Software switching stacks such as Open vSwitch (OVS) running on host cores can consume up to 8–16 dedicated cores per dual-socket node under heavy network load [8]. When polling frameworks (e.g., DPDK) are deployed to sustain higher packet rates, designated CPU cores run at 100% duty cycles regardless of instantaneous traffic, permanently stranding valuable compute resources [7].

The Total Cost of Ownership (TCO) inefficiency can be modeled by evaluating the ratio of infrastructure cores to total available cores across a datacenter fleet:

$$TCO\_Inefficiency = (Cores\_Infrastructure / Cores\_Total) \times (CapEx\_Host + OpEx\_Power)$$

When infrastructure tasks consume 25% of a 128-core server, 32 high-performance cores are unavailable for tenant monetization. At the scale of hundreds of thousands of servers, this overhead represents hundreds of millions of dollars in capital expenditure and megawatts of wasted electrical power.

## 2.2. Microarchitectural Contention, Cache Pollution, and Tail Latency

Beyond core subtraction, in-band execution inflicts indirect performance penalties on co-located guest applications through shared microarchitectural resource contention:

1. **Last-Level Cache (LLC) Thrashing:** High-bandwidth packet streams continuously cycle packet ring descriptors and frame buffers through the host processor's L3 cache. In software-based network pipelines, this continuous memory streaming evicts the working sets of tenant applications, increasing L3 cache miss rates and stalling execution pipelines [6], [7].
2. **Translation Lookaside Buffer (TLB) and Context-Switch Overhead:** Interleaving hypervisor network processing with guest computation triggers frequent kernel context switches, causing TLB invalidations and pipeline flushes [6].
3. **Interrupt Processing and Receive Livelock:** Under bursty traffic, high-frequency network interrupts monopolize core execution, leading to "receive livelock" where the host CPU spends all available cycles handling interrupts without making progress on application-level processing [17].

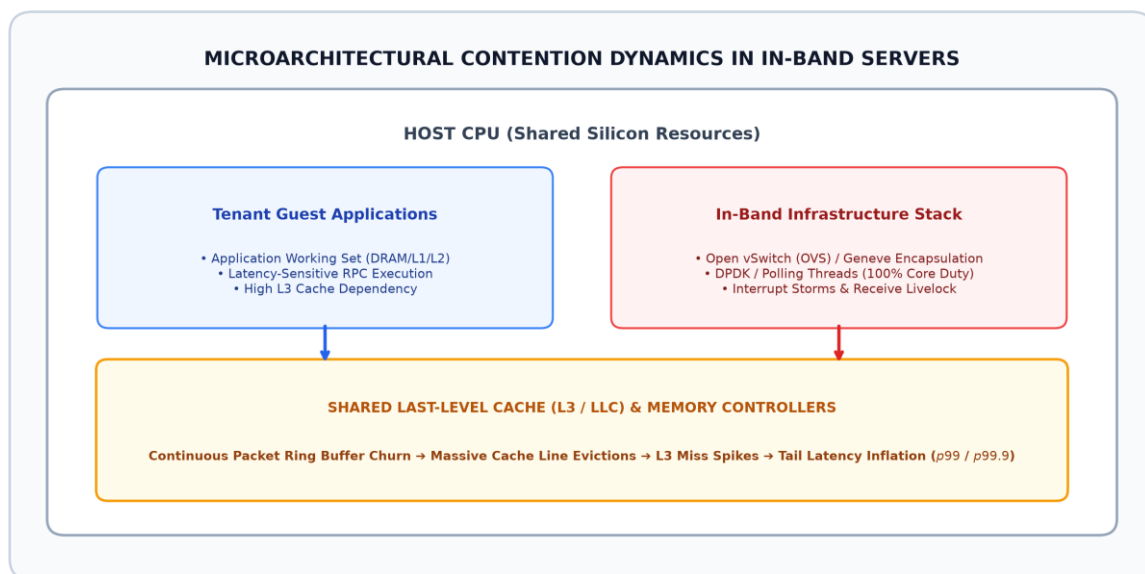


Figure 2: Microarchitectural resource contention in in-band servers: network packet churning through the shared Last-Level Cache (LLC) degrades tenant application cache locality.

In scale-out distributed systems - such as partitioned key-value stores, distributed query engines, and synchronized deep learning training loops - microarchitectural stalls compound across RPC call trees. As demonstrated by Dean and Barroso [2], an operation requiring fan-out to 100 backend servers has its overall response latency dictated by the slowest 99th percentile ( $p_{99}$ ) response. When host CPUs experience interrupt delays and scheduling jitter,  $p_{99}$  and  $p_{99.9}$  tail latencies inflate by orders of magnitude [2], [11], [16].

| Architectural Vector                               | In-Band Software Execution (Host CPU)                        | Hardware IPU / DPU Offload                                    | Synthesized Architectural Impact (Published Literature)               |
|----------------------------------------------------|--------------------------------------------------------------|---------------------------------------------------------------|-----------------------------------------------------------------------|
| <b>Host CPU Utilization for Infrastructure</b>     | 20% – 35%+ of total server cores [6], [8], [10]              | < 1% (Zero cores dedicated to switching/crypto) [4], [10]     | Recovers 20%–35% of total server compute capacity for guest workloads |
| <b>Small-Packet Forwarding Rate</b>                | 10 – 30 Mpps (Software OVS / Polling DPDK) [7], [8]          | 100 – 400+ Mpps (Hardware P4/ASIC pipeline) [4], [18]         | 5x to 15x increase in packet processing throughput                    |
| <b>Remote Block Storage Tail Latency (\$p99\$)</b> | 350 – 800 $\mu$ s (Software initiator + host jitter) [6]     | 60 – 120 $\mu$ s (Hardware NVMe-oF emulation) [4], [5]        | 4x to 8x reduction in storage tail latency                            |
| <b>RPC Tail Latency Jitter (\$p99.9\$)</b>         | 1.5 – 5.0 ms (Interrupt delays, context switching) [2], [11] | 50 – 150 $\mu$ s (Hardware transport & rate control) [3], [4] | 10x to 30x reduction in extreme latency jitter                        |
| <b>L3 Cache Contention</b>                         | High (Packet buffers churn through shared LLC) [6]           | Isolated (DMA directly to designated guest buffers via DDIO)  | Eliminates infrastructure-induced cache eviction                      |
| <b>Security &amp; Trust Boundary</b>               | Shared Hypervisor / Kernel boundary [8], [10]                | Physical Hardware Air-Gap across PCIe bus [4], [10]           | Complete isolation of cloud control plane from tenant host            |

Table 1: Synthesized comparison of system performance and architectural characteristics: in-band host execution vs. dedicated IPU/DPU offloading based on published hyperscale literature [2]–[8], [10], [11], [18].

### 2.3. The Isolation Deficit and Zero-Trust Multi-Tenancy

In multi-tenant cloud environments, sharing the host CPU between customer applications and infrastructure management creates significant security risks:

1. **Microarchitectural Side Channels:** Speculative execution vulnerabilities (e.g., Spectre, Meltdown, L1TF, Downfall) allow malicious guest VMs to potentially infer data from co-located tenant VMs or the host hypervisor by measuring timing variations in microarchitectural caches.
2. **Hypervisor Escape Vulnerabilities:** Software hypervisors maintain complex virtualization codebases. A vulnerability in hypervisor device emulation (e.g., QEMU virtio handlers) can permit an attacker to compromise the host kernel.
3. **Bare-Metal Multi-Tenancy Challenges:** Offering high-performance bare-metal servers traditionally required granting customers physical host root privileges, making it impossible to safely run cloud management agents, VPC network virtualization, and storage encryption software on the same host CPU.

IPUs resolve this isolation deficit by establishing a **hardware-enforced physical air-gap**. The sovereign IPU sits on an independent SoC connected to the host CPU across the PCIe or CXL bus. The host CPU is treated as completely untrusted. Even if a tenant achieves root privileges or compromises the host kernel, they cannot access the IPU's internal memory, cryptographic keys, VPC routing tables, or cloud control planes.

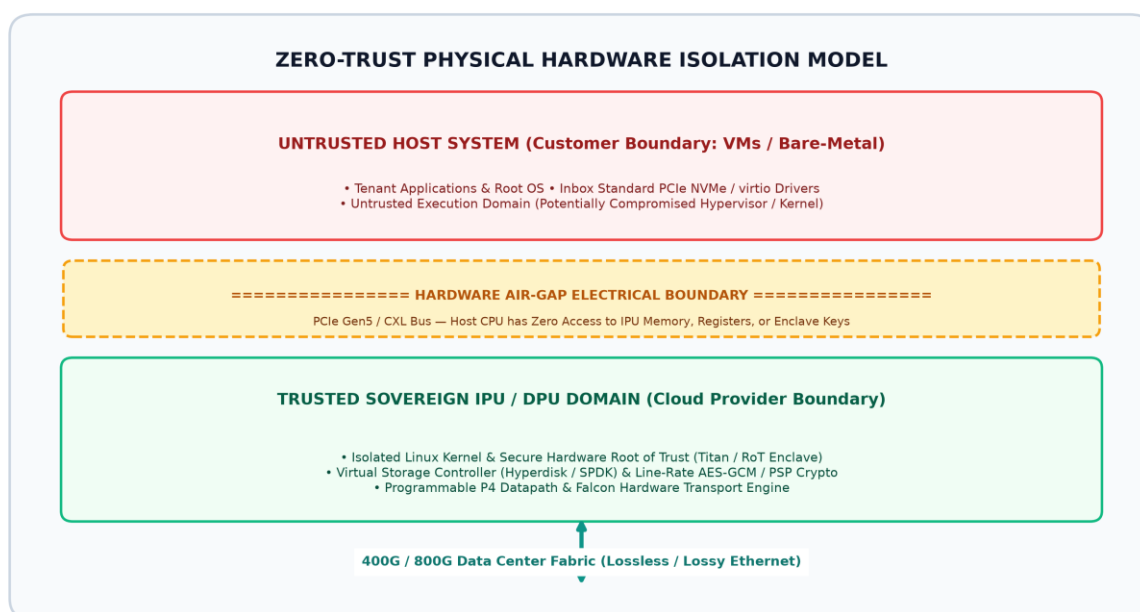


Figure 3: Zero-trust physical hardware isolation model: establishing a hardware-enforced electrical boundary across the PCIe/CXL bus between untrusted host instances and trusted sovereign IPUs.

### 3. The Role of Infrastructure Processing Units (IPUs) in Workload Offloading

#### 3.1. Defining the IPU / DPU Architectural Construct

An IPU or DPU is an advanced heterogeneous System-on-Chip comprising five core hardware subsystems:

1. **High-Speed Network Pipeline:** Multi-hundred-gigabit (100G–800G) MAC/PHY interfaces coupled with programmable packet parsing, match-action tables (MAT), and traffic shaping engines.
2. **Embedded Processing Complex:** A cluster of multi-core 64-bit processors (typically 8 to 32 ARM Neoverse or Intel x86 cores) operating an independent embedded Linux operating system and dedicated DRAM subsystem.
3. **Programmable Flow Engines:** Specialized hardware pipelines programmed via domain-specific languages (e.g., P4 [9]) or vendor runtime environments (e.g., DOCA [14]) to execute line-rate packet parsing, table lookups, and stateful flow tracking.
4. **Hardware Storage and Cryptographic Accelerators:** Dedicated fixed-function silicon blocks providing wire-speed symmetric cryptography (AES-GCM-128/256), CRC/checksum calculation, compression, and NVMe controller emulation.
5. **Host Interconnect and Root Complex Interface:** High-bandwidth PCIe (Gen4/5/6) and Compute Express Link (CXL [15]) controllers supporting Single Root I/O Virtualization (SR-IOV) and Scalable I/O Virtualization (SIOV).

#### 3.2. Core Infrastructure Offload Subsystems

##### 3.1. Software-Defined Networking (SDN) Offload

Modern cloud VPCs implement complex overlay networks (e.g., Geneve, VXLAN) with distributed firewalls, connection tracking, and load balancing. In an IPU-based architecture, flow definitions are programmed into hardware match-action tables. Inbound packets are parsed, decapsulated, filtered against security rules, and DMA-transferred directly into guest VM memory buffers without traversing the host kernel network stack [4], [8].

##### 3.2. Storage Virtualization and NVMe-over-Fabrics (NVMe-oF)

Disaggregated storage architectures decouple compute nodes from storage pools. Traditionally, host CPUs executed software storage clients (e.g., SPDK NVMe-oF initiators), consuming significant memory bandwidth and core cycles on queue management and block translation.

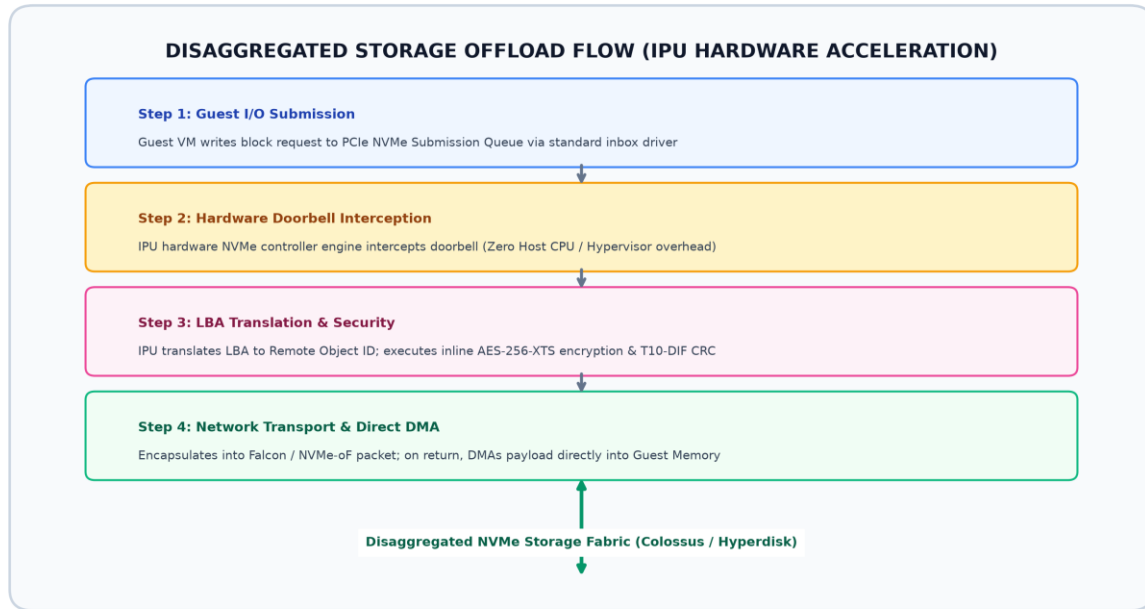


*Figure 4: Disaggregated NVMe storage offload datapath: IPU hardware controller emulation intercepts PCIe submission doorbells, performs line-rate crypto, and transfers data via direct memory access (DMA).*

An IPU resolves this by presenting physical PCIe NVMe controllers directly to the host CPU:

1. The guest OS utilizes standard, inbox NVMe drivers to submit read/write commands to PCIe Submission Queues.
2. The IPU's hardware NVMe emulation engine intercepts doorbells with zero host hypervisor intervention.
3. Embedded storage logic maps Logical Block Addresses (LBAs) to remote disaggregated storage object identifiers.
4. Dedicated hardware blocks perform line-rate AES-256-XTS encryption and data integrity verification (T10-DIF).
5. Requests are encapsulated into transport packets (Falcon or NVMe-oF over RDMA/TCP) and transmitted across the fabric.
6. When read data returns, the IPU DMAs payload bytes directly into the guest VM's target memory buffer and posts a completion entry.

### 3.3. Zero-Trust Security and Cryptographic Isolation

IPUs provide line-rate cryptographic acceleration for network transit encryption protocols - including IPsec, MACsec, and Google's open-source Packet Security Protocol (PSP) [20]. Because encryption keys are stored exclusively inside the IPU's secure memory enclave and managed by an embedded Hardware Root of Trust (RoT), keys remain inaccessible to the host CPU, eliminating exposure to host-side compromises.

## 4. Architectural Taxonomy and Evolution: From Legacy NICs to SmartNICs, DPUs, and IPUs

### 4.1. Evolutionary Taxonomy of Network Interfaces (Gen 1 to Gen 5)

The network interface has evolved from a passive packet serializer into an autonomous, distributed compute platform across five distinct architectural generations [8], [10], [18]:

1. **Gen 1 (1980s–1990s): Legacy NICs.** Fixed-function ASICs performing physical layer encoding, framing, and MAC-layer serialization. Host CPUs executed the full TCP/IP networking stack in software.
2. **Gen 2 (Late 1990s–2000s): Stateless Offload NICs and TOEs.** Introduced stateless hardware assists, including TCP/UDP checksum offload, Large Send Offload / TCP Segmentation Offload (TSO/LSO), and Receive Side Scaling (RSS) to distribute packet processing across multiple CPU cores.
3. **Gen 3 (2010s–Present): Advanced SmartNICs (e.g., Mellanox ConnectX-4/5/6/7).** High-speed ASICs (25G–400G) providing hardware-accelerated flow switching (ASAP<sup>2</sup>), RoCEv2 (RDMA over Converged Ethernet), and SR-IOV virtualization. These devices accelerate data planes but remain fundamentally dependent on host CPUs for control plane execution.
4. **Gen 4 (Mid-2010s–2020): FPGA-Based Programmable Accelerators (e.g., Microsoft Catapult / Azure SmartNIC).** Combined reconfigurable FPGA logic with network MACs, enabling dynamic updates to SDN pipelines [8]. However, FPGAs suffered from higher cost, higher power consumption, lower clock frequencies, and complex RTL programming cycles compared to standard ASICs.

5. **Gen 5 (2021–Present): Sovereign IPU and DPUs (Intel E2000 Mount Evans, NVIDIA BlueField-3, AMD Pensando Elba, AWS Nitro).** Autonomous, multi-core SoCs featuring independent operating systems, P4/C programmable packet engines, hardware NVMe controller emulation, line-rate cryptographic engines, and emerging CXL coherent memory support.

| Evolutionary Dimension               | Gen 1: Legacy NIC   | Gen 2: Stateless Offload / TOE    | Gen 3: Advanced SmartNIC (e.g., ConnectX-6/7) | Gen 4: FPGA SmartNIC (e.g., Azure SmartNIC) | Gen 5: Sovereign IPU / DPU (Intel E2000, BF-3, Elba) |
|--------------------------------------|---------------------|-----------------------------------|-----------------------------------------------|---------------------------------------------|------------------------------------------------------|
| <b>Primary Architectural Role</b>    | Frame Serialization | Stateless Checksum & Segmentation | Line-Rate Flow Switching & RDMA Transport     | Programmable Flow Acceleration              | Sovereign Infrastructure Compute & Isolation         |
| <b>Interface Line Rates</b>          | 10 Mbps – 100 Mbps  | 1 Gbps – 10 Gbps                  | 25 Gbps – 400 Gbps                            | 40 Gbps – 100 Gbps                          | 200 Gbps – 800 Gbps+                                 |
| <b>Embedded Compute Core Complex</b> | None                | None                              | None (Microcontrollers for ASIC state)        | Embedded Soft Cores / Few ARM Cores         | 8 – 32 Coherent 64-bit ARM / x86 Cores               |
| <b>Onboard Operating System</b>      | None                | None                              | None (Host OS Driver dependent)               | Lightweight RTOS / Host Managed             | Independent Sovereign Linux Operating System         |
| <b>Datapath Programmability</b>      | Fixed Hardware      | Fixed ASIC Registers              | ASIC Match-Action Flow Tables                 | RTL (Verilog / VHDL) / High-Level Synthesis | P4, DOCA C/C++, DPDK, SPDK, eBPF                     |
| <b>Storage Virtualization</b>        | None                | None                              | Hardware NVMe-oF Target offload               | Basic Virtio Acceleration                   | Full PCIe NVMe Controller Emulation & Mapping        |
| <b>Security &amp; Trust Boundary</b> | Host Bound          | Host Bound                        | Host Bound (ASIC driver interface)            | Host Bound Control Plane                    | Complete Hardware Air-Gap & Enclave RoT              |
| <b>Management Plane Execution</b>    | In-Band (Host OS)   | In-Band (Host OS)                 | In-Band (Host CPU Core Overhead)              | In-Band / Hybrid Management                 | Out-of-Band (Host CPU completely untrusted)          |

Table 2: Five generations of network interface and infrastructure processing architectures.

#### 4.2. SmartNICs vs. Sovereign IPU/DPUs: The Architectural Divide

A clear architectural boundary separates Advanced SmartNICs (e.g., Mellanox ConnectX-6/CX-7) from Sovereign IPU/DPUs (e.g., Intel IPU E2000, NVIDIA BlueField-3, AMD Pensando Elba):

| Architectural Dimension       | Advanced SmartNIC (e.g., Mellanox ConnectX-7)              | Sovereign IPU / DPU (e.g., Intel IPU E2000, BlueField-3) |
|-------------------------------|------------------------------------------------------------|----------------------------------------------------------|
| Execution Paradigm            | Fixed-Pipeline Hardware Acceleration Engine                | Autonomous Distributed Heterogeneous System-on-Chip      |
| Host System Dependency        | Dependent (Host OS must run drivers and control plane)     | Autonomous (Boots independently, out-of-band management) |
| Control Plane Location        | Executes on Host CPU (Consumes host cores & memory)        | Executes on IPU Embedded CPU Complex (Host unburdened)   |
| Operating System              | No Onboard OS (Hardware Firmware Execution Only)           | Independent Embedded Linux Kernel / OS Platform          |
| Bare-Metal Security Trust     | Partial (Host root user can reflash or alter device state) | Absolute (Host CPU has zero access to IPU OS or memory)  |
| Local Storage Stack Execution | Limited (Relies on host software for complex translation)  | Complete (Executes local SPDK, mapping tables, caching)  |
| Programmability Scope         | ASIC table entry insertion via host APIs                   | Arbitrary C/C++, Rust, Python, P4, and eBPF runtimes     |

Table 3: Technical contrast between Advanced SmartNICs and Sovereign IPUs/DPUs.

While advanced SmartNICs deliver outstanding wire-speed switching and RDMA performance, they remain peripheral devices requiring host CPU cycles to manage connections, route exceptions, and configure control tables. In contrast, an IPU functions as a sovereign computing node that happens to share a chassis and a PCIe/CXL bus with the host processor.

5. Hyperscale Innovations: Google Titanium, Intel Mount Evans, Falcon, and Packet Security Protocol (PSP)  
5.1. The Titanium Infrastructure Architecture

Google developed **Titanium**, a custom-silicon ecosystem designed to offload, accelerate, and isolate hypervisor management, networking, storage, and security across Google Cloud [4], [5].

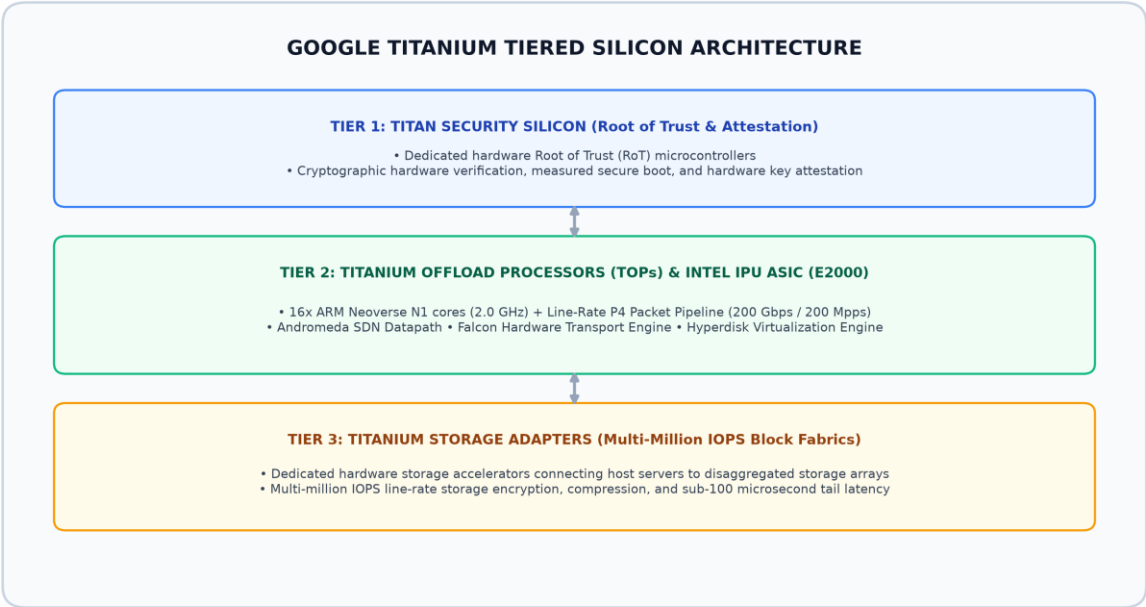


Figure 5: Google Titanium multi-tiered custom silicon architecture spanning Titan Root of Trust microcontrollers, Titanium Offload Processors / Intel IPU ASICs, and Titanium Storage Adapters.

Titanium operates as a tiered hardware architecture:

1. **Titan Security Silicon:** Hardware Root of Trust microcontrollers providing cryptographic identity, signed firmware verification, and attestation.
2. **Titanium Offload Processors (TOPs) and Custom IPU:** Custom ASICs executing the Andromeda SDN datapath, Falcon hardware transport protocol, and Hyperdisk storage virtualization.
3. **Titanium Storage Adapters:** Dedicated storage processors bridging host servers to Google’s disaggregated storage arrays (Colossus).

## 5.2. Intel Infrastructure Processing Unit (IPU) E2000 (Mount Evans)

Co-developed by Intel and Google, the **Intel IPU E2000** (codenamed *Mount Evans*) represents a premier implementation of a sovereign cloud SoC [4], [18]:

- **Compute Complex:** 16 64-bit ARM Neoverse N1 cores interconnected via a coherent mesh running at up to 2.0 GHz with dedicated system memory controllers.
- **Programmable Packet Processing:** A line-rate P4-programmable pipeline supporting 200 Gbps network bandwidth and up to 200 Mpps packet processing.
- **Hardware Storage Engine:** An integrated NVMe emulation engine exposing physical PCIe functions to the host while translating I/O to network storage.
- **Crypto Engine:** Line-rate AES-GCM and SHA cryptographic engines supporting wire-speed encryption at 200 Gbps.

## 5.3. Performance Realization: Google Compute Engine C3 Virtual Machines

In Google Compute Engine C3 instances, pairing 4th Gen Intel Xeon Scalable processors (Sapphire Rapids) with the custom Intel IPU E2000 ASIC achieves major system-level gains over prior software-managed C2 instances [4]:

| Architectural Dimension            | Prior Generation (C2 Instance: In-Band Software) | C3 Instance (Custom Intel IPU E2000)      | Published Architectural Impact [4], [5]                  |
|------------------------------------|--------------------------------------------------|-------------------------------------------|----------------------------------------------------------|
| <b>Infrastructure Architecture</b> | In-Band Software Hypervisor (KVM/OVS)            | Dedicated Intel IPU E2000 (Mount Evans)   | Complete physical isolation of infrastructure stack      |
| <b>Maximum Network Bandwidth</b>   | Up to 100 Gbps (Host-processed)                  | Up to 200 Gbps (Line-rate hardware)       | 2x Bandwidth Scale                                       |
| <b>Packet Forwarding Capacity</b>  | ~30 Mpps (Software bound)                        | Up to 200 Mpps (Hardware P4 engine)       | 6.6x Packet Processing Throughput                        |
| <b>Transport Protocol Layer</b>    | Software TCP / Host Kernel Stack                 | Hardware-Accelerated Falcon Transport     | Sub-microsecond tail latency and hardware retransmission |
| <b>Storage Architecture</b>        | Persistent Disk (Software SPDK client)           | Hyperdisk (IPU Hardware NVMe-oF Engine)   | Up to 500,000 IOPS per VM                                |
| <b>Storage Latency (\$p99\$)</b>   | ~400 – 850 $\mu$ s                               | < 90 – 100 $\mu$ s                        | Up to 4x–8x reduction in storage tail latency            |
| <b>Host CPU Core Allocation</b>    | Cores shared between guest and I/O               | 100% of guest vCPUs dedicated to workload | Complete elimination of host datacenter tax              |

Table 4: Comparative architectural characteristics: Google Compute Engine C2 vs. C3 with Custom Intel IPU [4], [5].

## 5.4. Falcon: A Reliable, Low-Latency Hardware Transport Protocol

Traditional datacenter transport protocols face significant challenges in high-density environments: TCP incurs high software stack latency and coarse timeout granularities, while RoCEv2 relies on Priority Flow Control (PFC) which is susceptible to pause

frame deadlocks and head-of-line blocking. Building on principles from low-latency datacenter transport research such as Swift [3] and Homa [11], modern hyperscale infrastructure relies on hardware transport offload.

Google introduced **Falcon** [3], a hardware-accelerated transport protocol designed specifically for IPU integration:

1. **Lossy Fabric Operation:** Falcon operates resiliently over standard lossy Ethernet, eliminating the requirement for PFC pause frames.
2. **Nanosecond RTT Measurement:** Falcon measures round-trip time (RTT) at hardware granularity on every packet, allowing rapid rate adjustments before switch buffers fill.
3. **Hardware-Driven Fast Retransmission:** Drop detection and retransmission execute directly in silicon, recovering from lost packets within microsecond timeframes.
4. **Multipath Packet Spraying:** Falcon sprays individual packets across multiple Equal-Cost Multi-Path (ECMP) network paths to prevent core hash collisions, utilizing hardware reordering buffers on the receiving IPU to deliver in-order streams to host memory.
5. **Integrated Cryptographic Encapsulation:** Integrates hardware line-rate encryption via the Packet Security Protocol (PSP).

### 5.5. Packet Security Protocol (PSP) Hardware Cryptographic Offload

To protect transit traffic at hyperscale without incurring software overheads, Google developed and open-sourced the **Packet Security Protocol (PSP)** [20].

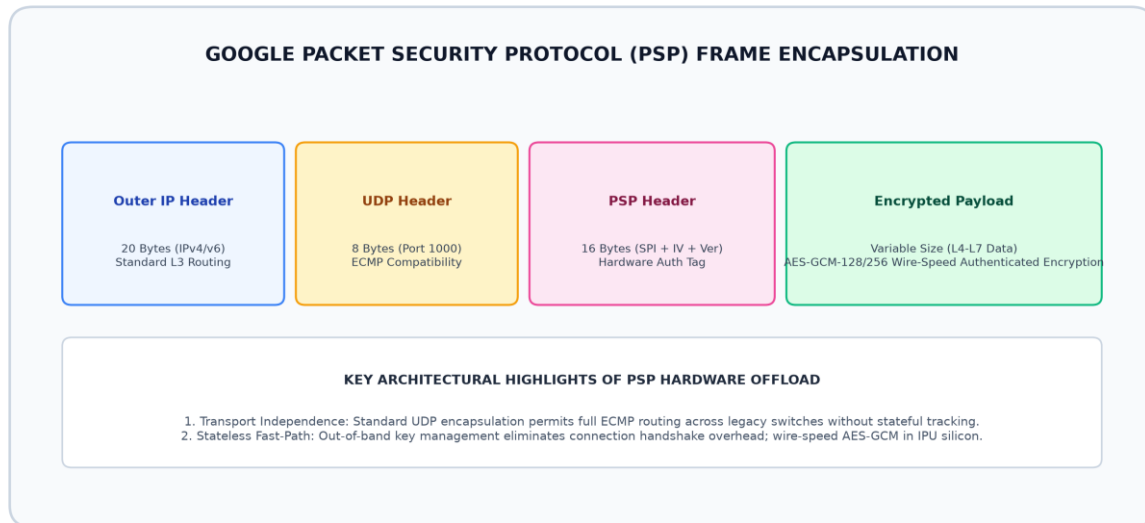


Figure 6: Google Packet Security Protocol (PSP) header format: standard UDP encapsulation with stateless SPI/IV metadata enabling line-rate hardware encryption.

Unlike TLS, which is connection-oriented and requires extensive state management in software, or IPsec, which introduces significant protocol complexity and MTU fragmentation issues, PSP was engineered specifically for line-rate hardware offloading:

1. **Transport Independence:** PSP encapsulates packets within standard UDP headers (typically destination port 1000), making encrypted traffic fully compatible with existing ECMP hashing and network switches.
2. **Stateless Security Parameters:** The PSP header contains a 32-bit Security Parameter Index (SPI), cryptographic Initialization Vector (IV), and authentication tag. Key exchange occurs out-of-band via cloud control planes, eliminating in-band connection handshake overhead.
3. **Hardware Line-Rate Performance:** IPU cryptographic engines decrypt and authenticate incoming packets and encrypt outgoing packets at 200 Gbps to 800 Gbps line rates with zero host CPU intervention.

### 5.6. High-Performance Computing (HPC) and MPI Collectives Acceleration

In distributed High-Performance Computing (HPC) and distributed machine learning training, collective communication operations (e.g., MPI\_Allreduce, MPI\_Bcast) require tight synchronization across thousands of compute nodes. In traditional systems, operating system jitter on a single core can delay collective barriers, degrading overall cluster scaling efficiency [6], [21]. Deploying IPUs paired with optimized communication libraries (such as the **Intel MPI Library** over Libfabric) offloads collective state tracking and communication interrupts to the IPU compute complex [21]. Host CPU cores compute without preemption, enabling scaling efficiencies exceeding 90% across massive node topologies.

## 6. Comprehensive Comparative Landscape: Commercial IPU, DPU, and SmartNIC Architectures

The semiconductor and cloud landscape features distinct architectural methodologies across major silicon vendors:

- **Intel IPU (Mount Evans / E2000):** Focuses on open standards, P4 pipeline programmability, and tight host coupling via CXL coherent memory links [4], [18].
- **NVIDIA BlueField-3 DPU:** Combines 16 64-bit ARM Neoverse V2 cores with ConnectX-7 network silicon and the DOCA software framework, targeting AI supercomputing clusters and enterprise zero-trust infrastructure [14].
- **AMD Pensando Elba DPU:** Integrates a programmable P4 packet processing pipeline with 16 ARM Cortex-A72 cores, targeting enterprise line-rate stateful firewalling, micro-segmentation, and high-density SDN switching [19].
- **AWS Nitro System:** Utilizes modular, custom-designed ASICs (Nitro Cards for VPC, EBS, and local NVMe) orchestrated by the Nitro Security Chip to power Amazon EC2 instances [10].

| Architectural Feature               | Intel IPU E2000 (Mount Evans) [4], [18]   | NVIDIA BlueField-3 DPU [14]             | AMD Pensando Elba DPU [19]               | AWS Nitro System (Gen 5) [10]                 |
|-------------------------------------|-------------------------------------------|-----------------------------------------|------------------------------------------|-----------------------------------------------|
| <b>Primary Architectural Target</b> | Hyperscale Cloud Infrastructure & Storage | AI Supercomputing, Enterprise DOCA      | Enterprise Security, SDN & Multi-Cloud   | Proprietary AWS EC2 Fleet Infrastructure      |
| <b>Embedded Compute Complex</b>     | 16x ARM Neoverse N1 (2.0 GHz)             | 16x ARM Neoverse V2 (Coherent Mesh)     | 16x ARM Cortex-A72 (High Frequency)      | Custom Proprietary Cores across Modular Cards |
| <b>Maximum Network Line Rate</b>    | 200 Gbps (Dual 100G / Single 200G)        | 400 Gbps (Dual 200G / Single 400G)      | 400 Gbps (Dual 200G / 1x400G)            | 100G – 400G+ Ethernet (ENA Architecture)      |
| <b>Packet Processing Datapath</b>   | Line-Rate P4 Programmable Pipeline        | ConnectX-7 Hardware Flow Steering       | P4-Programmable Packet Engine            | Modular Proprietary Nitro ASICs               |
| <b>Hardware Transport Engine</b>    | Falcon Transport & RoCEv2 Engine          | RoCEv2, InfiniBand, GPUDirect RDMA      | RoCEv2 & Custom Low-Latency Flow         | Scalable Reliable Datagram (SRD)              |
| <b>Software Development Model</b>   | P4, DPDK, SPDK, OPI Standard              | NVIDIA DOCA Framework, C/C++ SDK        | P4, Pensando Policy APIs                 | Internal AWS Microservices Architecture       |
| <b>Host System Interconnect</b>     | PCIe Gen4 x16 / CXL Ready                 | PCIe Gen5 x16 (Dual x8 or Single x16)   | PCIe Gen5 x16                            | High-Speed Proprietary PCIe Interfaces        |
| <b>Cryptographic Offload</b>        | Line-rate AES-GCM (200G), PSP, IPsec      | Line-rate IPsec, TLS 1.3, MACsec (400G) | Line-rate IPsec, PSP, Wireguard, AES-GCM | Nitro Security Chip & Inline Hardware Crypto  |
| <b>Storage Virtualization</b>       | Hardware NVMe Emulation, virtio-blk       | NVMe-oF SNAP Emulation, SPDK Target     | NVMe Controller Emulation, virtio-blk    | Nitro EBS Direct Hardware Virtualization      |

Table 5: Comprehensive architectural matrix: Major commercial Infrastructure Processing Units and Data Processing Units [4], [10], [14], [18], [19].

## 7. The Paradigm Shift: The Rise of CPU-Less Architectures and Autonomous IPU Computing

### 7.1. Inversion of the Server Hierarchy

Historically, data center architecture has been strictly **CPU-centric**: the host CPU initialized the system, controlled the PCIe hierarchy, managed physical memory, and configured peripheral devices. With modern IPUs integrating 16 to 32 64-bit cores, independent high-bandwidth DRAM channels, and direct network connectivity, the server hierarchy is undergoing an **inversion** [13]. Rather than functioning as a passive PCIe device, the IPU becomes the primary system orchestrator, treating the host CPU as a specialized compute engine alongside GPUs and accelerators.

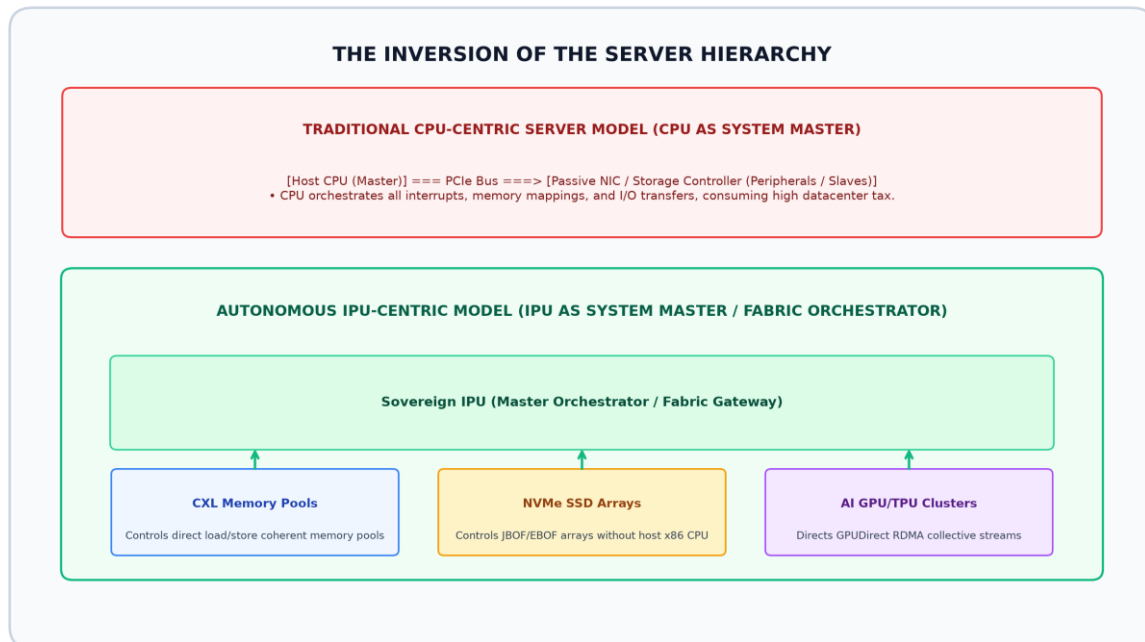


Figure 7: The structural inversion of server hierarchy from traditional CPU-centric systems to autonomous IPU-centric systems orchestrating CXL memory, NVMe SSD arrays, and AI accelerator clusters.

## 7.2. Emerging Autonomous Server Topologies

This architectural inversion enables new system topologies:

1. **Autonomous Storage Appliances (CPU-Less JBOF/EBOF)**: Modern disaggregated storage arrays eliminate general-purpose host CPUs entirely. IPUs connect directly to PCIe NVMe SSDs (E1.S/E3.S) through PCIe switches, executing SPDK storage targets and remote transport encapsulation on embedded cores, reducing system power by up to 40% [13].
2. **AI Cluster Interconnect Gateways**: In massive GPU/TPU training clusters, autonomous IPUs manage cross-rack GPUDirect RDMA routing, checkpoint synchronization, and in-network reduction operations without involving host CPUs.
3. **CXL Coherent Memory Pools**: Leveraging Compute Express Link (CXL 2.0/3.0) protocols (CXL.io, CXL.cache, CXL.mem) [15], IPUs attach directly to multi-terabyte pools of disaggregated DDR5 memory, dynamically managing multi-tenant memory allocation and caching.
4. **Autonomous Edge Micro-Nodes**: In 5G Open RAN environments, standalone IPUs operate as compact, self-contained micro-servers executing containerized edge functions within tight 35W–75W power envelopes.

## 8. Open Challenges, Security Implications, and Future Research Directions

### 8.1. Thermal, Power, and Form-Factor Constraints

Modern 400G and 800G IPUs operate within thermal design power (TDP) envelopes of 150W to 250W. Integrating these high-power SoCs into dense 1U and 2U rack servers requires advanced cooling architectures, such as direct-to-chip liquid cooling, and customized form factors (e.g., OCP NIC 3.0, PCIe add-in cards with specialized heatsinks). Optimizing energy efficiency (Joules per processed gigabit) remains a primary research priority.

### 8.2. Firmware Security and Attack Surface Expansion

Because IPUs have direct DMA access to host physical memory and full control over network interfaces, compromised IPU firmware would represent a catastrophic security failure. Mitigating this risk requires hardware-anchored roots of trust (e.g.,

Google Titan, AWS Nitro Security Chip), cryptographically signed firmware verification, isolated secure execution enclaves, and formal verification of IPU management microcode.

### 8.3. Software Standardization and the OPI Project

The rapid emergence of proprietary vendor frameworks (NVIDIA DOCA, Intel IPU SDK, AMD Pensando APIs, AWS Nitro microservices) risks software ecosystem fragmentation. To establish open standards, the Linux Foundation and Open Compute Project established the **Open Programmability Infrastructure (OPI) Project** [12]. OPI standardizes Northbound gRPC and Protobuf interfaces across networking, storage, and telemetry, enabling cloud orchestration platforms (e.g., Kubernetes) to manage heterogeneous IPU seamlessly.

### 8.4. Coherent Interconnects (CXL) Integration

Bridging IPU memory and host memory through cache-coherent fabrics remains a vibrant area of systems research. As CXL 3.0 fabrics mature, IPU will participate directly in hardware cache-coherency domains, enabling zero-copy data exchanges between host applications and network pipelines without DMA serialization overheads [15].

## 9. Conclusion

The bifurcation of server architecture into host compute domains and sovereign Infrastructure Processing Units represents one of the most critical structural shifts in computer systems over the past decade. As Dennard scaling terminates and network bandwidth surges toward 800 Gbps, dedicated IPU and DPU silicon resolves the datacenter tax - recovering 20% to 35%+ of server CPU capacity while enforcing hardware-isolated zero-trust security boundaries.

Hyperscale innovations such as Google's Titanium architecture, Intel's Mount Evans E2000 IPU, the Falcon hardware transport protocol, and open standards like the Packet Security Protocol (PSP) demonstrate the substantial latency, throughput, and efficiency advantages of specialized infrastructure silicon. As high-speed Compute Express Link (CXL) fabrics and open frameworks like OPI mature, IPU will continue transitioning from peripheral offload accelerators into the primary system orchestrators of next-generation cloud and artificial intelligence infrastructure.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest.

**Publisher's Note:** All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors, and the reviewers.

## References

- [1] D. Reinsel, J. Gantz, and J. Rydning, "Data Age 2025: The Digitization of the World from Edge to Core," International Data Corporation (IDC) White Paper, Tech. Rep. US44413318, Nov. 2018.
- [2] J. Dean and L. A. Barroso, "The Tail at Scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, Feb. 2013.
- [3] G. Kumar, N. Dukkipati, K. Jang, H. M. G. Wassel, X. Wu, B. Montazeri, Y. Wang, D. G. Springborn, C. Alfeld, M. Ryan, D. Wetherall, and A. Vahdat, "Swift: A Low-Latency Congestion Control for Data Centers," in *Proc. ACM SIGCOMM Conference*, 2020, pp. 514–528.
- [4] Intel Corporation, "Intel and Google Cloud Announce Co-Designed Infrastructure Processing Unit," Intel Newsroom Press Release and Architecture Brief, Oct. 2022. [Online]. Available: <https://www.intel.com/content/www/us/en/newsroom/news/intel-google-cloud-co-design-ipu.html>
- [5] Google LLC, "Google Infrastructure Security Design Overview," Google Whitepaper, 2020. [Online]. Available: <https://cloud.google.com/docs/security/infrastructure/design-overview>
- [6] M. Marty, C. de Kruijf, R. Adriaens, C. Alfeld, S. Babu, B. Bingham, P. Dahl, et al., "Snap: A Microkernel Approach to Host Networking at Google Scale," in *Proc. 27th ACM Symposium on Operating Systems Principles (SOSP)*, 2019, pp. 399–413.
- [7] A. Belay, G. Prekas, A. Klimovic, S. Grossman, C. Kozyrakis, and E. Bugnion, "IX: A Protected Dataplane Operating System for High Throughput and Low Latency," in *Proc. 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2014, pp. 49–65.
- [8] D. Firestone, A. Putnam, S. Mundkur, et al., "Azure Accelerated Networking: SmartNICs in the Public Cloud," in *Proc. 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2018, pp. 51–66.
- [9] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, et al., "P4: Programming Protocol-Independent Packet Processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, Jul. 2014.
- [10] Amazon Web Services, "The Security Design of the AWS Nitro System," AWS Whitepaper, Amazon.com, Inc., Seattle, WA, Nov. 2021. [Online]. Available: <https://docs.aws.amazon.com/whitepapers/latest/security-design-of-aws-nitro-system/>
- [11] M. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, "Homa: A Receiver-Driven Low-Latency Transport Protocol Using Network Priorities," in *Proc. ACM SIGCOMM Conference*, 2018, pp. 221–235.
- [12] Linux Foundation, "Open Programmability Infrastructure (OPI) Project Architecture Specification," *Linux Foundation Networking Standard*, Release 1.2, 2023. [Online]. Available: <https://opiproject.org>

- [13] S. Ibanez, G. Antichi, and N. McKeown, "The Case for Computing Without CPU in the Cloud Data Center," in *Proc. 22nd ACM Workshop on Hot Topics in Networks (HotNets)*, 2023, pp. 183–190.
- [14] NVIDIA Corporation, "NVIDIA DOCA Software Framework Architecture Guide," Santa Clara, CA, Tech. Doc. v2.5, 2024. [Online]. Available: <https://docs.nvidia.com/doca/>
- [15] Compute Express Link Consortium, "Compute Express Link (CXL) Specification Revision 3.1," Standard Specification, Beaverton, OR, Nov. 2023.
- [16] J. Ousterhout, A. Gopalan, A. Gupta, A. Kejriwal, C. Lee, B. Montazeri, D. Ongaro, S. J. Park, H. Qin, S. Rosenblum, S. Rumble, R. Stutsman, and S. Yang, "The Case for Low-Latency Serialization," in *Proc. 14th USENIX Workshop on Hot Topics in Operating Systems (HotOS)*, 2013.
- [17] J. Mogul and K. K. Ramakrishnan, "Eliminating Receive Livelock in an Interrupt-Driven Kernel," *ACM Transactions on Computer Systems (TOCS)*, vol. 15, no. 3, pp. 217–252, Aug. 1997.
- [18] Intel Corporation, "Intel Infrastructure Processing Unit (IPU) Architecture Overview: Architecture and Programming Model," Intel White Paper, Document Number 738491-002, 2023.
- [19] AMD Pensando Systems, "AMD Pensando Elba Enterprise DPU Architecture Specification," Advanced Micro Devices, Tech. Rep., 2023.
- [20] Google LLC, "PSP: Packet Security Protocol Specification and Cryptographic Hardware Offload," Open-Source Specification, GitHub Repository google/psp, 2022. [Online]. Available: <https://github.com/google/psp>
- [21] M. J. Koop, T. Jones, and D. K. Panda, "Reducing Operating System Jitter with Hardware-Assisted MPI Offload," in *Proc. IEEE International Conference on Cluster Computing (CLUSTER)*, 2008, pp. 248–255.