USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

ZOC: Elastic and Cost-Efficient Virtual SmartNIC Architecture for Cloud Physical Machines

Naixuan Guan, Xiaokang Hu, Yisheng Xie, Xishi Qiu, Chaojie Liu, Yuchao Cao, Banghao Ying, Dianchen Tian, Yu Zhou, Yangzeyu Zhang, Hujun Ge, Yibin Shen, and Jiesheng Wu, *Alibaba Cloud Computing*

https://www.usenix.org/conference/nsdi26/presentation/guan-naixuan

This paper is included in the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation.

May 4–6, 2026 • Renton, WA, USA

ISBN 978-1-939133-54-0

Open access to the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation is sponsored by

جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

ZOC: Elastic and Cost-Efficient Virtual SmartNIC Architecture for Cloud Physical Machines

Naixuan Guan, Xiaokang Hu*, Yisheng Xie, Xishi Qiu, Chaojie Liu, Yuchao Cao,
Banghao Ying, Dianchen Tian, Yu Zhou, Yangzeyu Zhang, Hujun Ge, Yibin Shen, Jiesheng Wu
Alibaba Cloud Computing

Abstract

DPU-based SmartNICs are reshaping cloud infrastructure by offloading core I/O functions, enabling cloud physical machines that support both bare-metal and virtualized environments. However, large-scale deployments face practical challenges—including operational inconsistency across heterogeneous fleets, limited resource elasticity, and performance bottlenecks caused by slow-path processing. We present ZOC, a software-defined virtual SmartNIC architecture that transforms general-purpose servers equipped with commodity NICs into full-featured cloud physical machines. ZOC delivers compatibility with existing infrastructure and provides a user experience on par with DPU-based nodes—without requiring any specialized hardware. At its core, ZOC integrates a passthrough NIC and dynamically provisioned host resources into a dedicated service VM, forming an elastic and programmable acceleration platform. Built upon this foundation, it introduces an efficient cross-domain device abstraction that exposes standardized I/O devices to the host, enabling seamless access to cloud storage and VPC networks. Extensive evaluation shows that ZOC achieves superior cost efficiency, eliminates slow-path bottlenecks, and provides high maintainability. Deployed in a major cloud, ZOC serves diverse public and private cloud services as a production-ready complement to existing DPU-based solutions.

1 Introduction

The rise of SmartNICs—particularly data processing units (DPUs)—is fundamentally reshaping cloud infrastructure by offloading core I/O functions [26, 72, 74]. A typical DPU is implemented as a System-on-Chip (SoC) that integrates a general-purpose processing subsystem, dedicated hardware accelerators, and a high-speed NIC to enable high-performance, low-latency I/O processing [38]. This architectural shift provides not only classic virtualized deployments but also a new paradigm: running cloud services directly on DPU-based bare-metal servers to deliver direct access to physical resources, full virtualization support, and strong isolation from noisy neighbors [16, 42, 46, 70].

However, from the perspective of cloud service providers (CSPs), relying solely on DPUs poses practical challenges in production. First, legacy datacenter servers and customer-owned private-cloud hardware often lack DPU integration, causing operational fragmentation. Second, DPUs provide fixed hardware resources, yet cloud workloads exhibit diverse and dynamic I/O demands; provisioning for peak loads typically results in underutilization [25]. Finally, certain data-plane operations—such as state transitions or remote procedure calls (RPCs)—must traverse the DPU's slow path on its general-purpose processing subsystem [38]. Given the limited CPU and memory resources in this subsystem, such operations can easily saturate its capacity, creating a bottleneck for services with high slow-path demands.

Prior work on SmartNICs has largely focused on offloading diverse workloads or improving the efficiency of offloaded execution [45, 47, 57, 58, 62, 66]. A recent attempt, vDUSE [68], implemented a software-defined data path to expose virtio devices [63] to the bare-metal host, partially addressing the elasticity challenge. However, it introduces tight OS coupling across abstraction layers and lacks native support for the NVMe protocol. In this paper, we present ZOC, an architecture that transforms general-purpose servers equipped with only commodity NICs into full-featured cloud physical machines. Like DPU-based nodes, ZOC supports both bare-metal and virtualized service models, but without their practical limitations in production environments.

The key insight behind ZOC is to construct a software-defined virtual SmartNIC by combining a passthrough NIC with dynamically provisioned host resources. These resources are encapsulated in a dedicated service virtual machine (VM), fully isolated from the host OS. Within this VM, existing infrastructure components—such as control-plane and data-plane stacks—can be reused with high compatibility. Critically, the service VM draws from the host's abundant resource pool, enabling fine-grained elasticity. Operators can either pre-

*Corresponding Author

allocate resources or hot-plug CPU and memory on demand, enabling ZOC to scale with workload demands and avoid performance bottlenecks. ZOC supports two running modes: a default performance mode with dedicated CPU cores, and an optional zero-cost mode that schedules the service VM's virtual CPUs (vCPUs) on otherwise-idle CPU cores.

ZOC introduces an efficient cross-domain device abstraction that exposes I/O devices from the service VM to the host, enabling seamless access to cloud storage and Virtual Private Cloud (VPC) networks. This abstraction comprises three components: (1) an I/O Proxy in the service VM for device emulation (virtio or NVMe), (2) a Virtual Bus on the host that exports standardized device interfaces, and (3) an I/O Link that bridges the semantic gap between them. The I/O Link provides memory-mapped I/O (MMIO) forwarding for control-plane operations, while supporting DMA-capable data transfers and low-latency interrupt delivery for the data plane. These I/O devices can be either directly probed by native host drivers to serve applications running on the host, or claimed via the mainstream VFIO framework [40] for passthrough to VMs.

We have implemented ZOC on both Intel and AMD server platforms using a variety of commodity NICs. Evaluation shows that ZOC delivers scalable I/O performance through elastic allocation of host resources. For workloads with low-to-moderate I/O demands, ZOC achieves better cost efficiency than the modern BlueField-3 DPU [54]. For I/O-intensive workloads dominated by slow-path operations, ZOC scales to higher throughput as more host CPU cores are allocated, surpassing the DPU's performance ceiling. In zero-cost mode, ZOC incurs only modest memory overhead while delivering predictable I/O performance and imposing bounded latency interference on co-located cloud workloads. ZOC has been deployed at a major CSP, demonstrating end-to-end compatibility, operational efficiency, and high maintainability in production.

In summary, ZOC serves as a practical and deployable complement to existing DPU-based solutions, offering CSPs the following key benefits:

1. Enables large fleets of legacy servers—lacking SmartNICs—to support cloud bare-metal execution at no additional hardware cost.

2. Unifies software stacks across DPU-based and DPU-less environments, particularly in private clouds.

3. Delivers superior cost efficiency for workloads with low-to-moderate I/O demands, where fixed DPU resources would otherwise remain underutilized.

4. Overcomes the slow-path bottlenecks inherent in the general-purpose processing subsystem of DPUs by leveraging elastic host resources.

2 Background

2.1 From basic NIC to DPU

Modern commodity NICs sustain line-rate throughput—from tens to hundreds of gigabits per second—using multi-queue architectures and receive-side scaling. Most mainstream NICs [34] support standard hardware offloads such as TCP segmentation offload and checksum computation, reducing host CPU overhead for basic packet processing. Advanced variants [49] further integrate RDMA, enabling low-latency, kernel-bypass data transfers for distributed systems.

SmartNICs extend basic NICs by integrating specialized or programmable processing units to offload complex host tasks [66]. Among them, data processing units (DPUs)—SmartNICs integrating a SoC—have become central to modern cloud infrastructure [8, 30]. A typical DPU combines a multi-core CPU complex, local DRAM, and high-speed I/O interfaces, enabling it to run a dedicated OS independently of the host. To further accelerate the data plane, many designs augment the SoC with hardware accelerators such as ASICs and FPGAs. Recent years have seen rapid adoption of commercial DPUs, including NVIDIA BlueField [14], Intel IPU [52], and AMD Pensando [3]. Concurrently, major CSPs have developed custom in-house DPUs to meet specialized requirements—examples include Alibaba Cloud CIPU [30], AWS Nitro [8], and Azure Boost [12]. By offloading critical infrastructure functions—such as storage, networking, and security—to the DPU, these platforms improve system efficiency and strengthen resource isolation.

2.2 Foundational Cloud Resources

CSPs offer services across multiple abstraction levels: infrastructure (IaaS), platform (PaaS), software (SaaS), and function (FaaS). At the infrastructure level, cloud storage and VPC networks serve as the foundational substrate for all higher-layer services.

Cloud storage systems deliver remote block storage to compute nodes over low-latency networks [72], with recent advances matching local disks in latency and throughput [28, 43]. Beyond performance, they offer several key advantages [7, 21, 22, 59], including elastic capacity, high availability and durability (e.g., "nine 9s"), and native VM live migration support. On compute nodes, cloud storage is exposed via standard block interfaces—either classic virtio-blk [63] or modern NVMe [67]. The adoption of NVMe-based cloud disks marks a significant evolution, enabling advanced features such as namespaces, multipath I/O, shared access, and atomic operations [9, 22, 67].

VPC services [10, 20] provide secure, isolated overlay networks over shared physical infrastructure using network tunneling. Each tenant defines a VPC and associates it with virtual network interfaces—typically implemented via virtio-

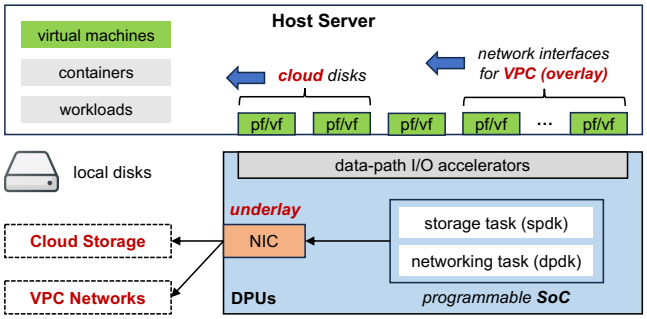


Figure 1: DPU-based cloud physical machine

net [63]. VPCs enable seamless network extension across machines within and between geographically distributed datacenters [56]. Beyond isolation and connectivity, modern VPC implementations support multipath routing, path health monitoring, and fast failover to ensure high availability and service continuity [20]. As the de facto standard for cloud networking, VPCs natively integrate with managed services such as NAT gateways, load balancers, and DDoS protection systems, simplifying application deployment and operations.

2.3 Cloud Physical Machine

A cloud physical machine leverages DPU-based offloading to deliver efficient access to foundational cloud storage and VPC networks. It natively supports bare-metal and VM-centric service models through a unified device abstraction, as illustrated in Figure 1 [8, 30, 74].

The DPU offloads storage and networking tasks, collaborating with integrated data-path accelerators, to expose high-performance I/O devices to the host as standard PCI functions: physical functions (PFs) or virtual functions (VFs). These devices are directly usable by bare-metal hosts or assignable to VMs, enabling seamless integration with existing virtualization stacks. I/O requests can be forwarded through the DPU's internal NIC to remote infrastructure services for processing. Notably, the host OS or VMs interact solely with the VPC overlay, while the DPU transparently utilizes the underlay network for efficient data transport.

Cloud workloads have conventionally been deployed in VMs to access foundational cloud resources. With the advent of DPU-based cloud physical machines, services can now run directly on the bare-metal host. This new deployment model is increasingly adopted across diverse workloads, including database systems [16, 71], distributed file systems [42], network functions [70], and serverless platforms [1, 46]. The shift is primarily driven by the following key advantages [50, 74, 76]. First, services on the host gain direct access to physical resources, achieving bare-metal performance. Second, full virtualization capabilities are preserved: workloads can enforce flexible isolation via on-demand instantiation of secure containers or lightweight VMs (e.g., Firecracker [1], RunD [46]). Finally, dedicated hardware eliminates noisy neighbors, ensuring predictable performance in multi-tenant clouds.

3 Motivation

3.1 DPU Challenges in Production

In production cloud environments, we find that relying solely on DPUs to enable cloud physical machines incurs practical limitations and operational challenges.

Heterogeneous fleets. DPUs are not uniformly available across datacenters, leading to inconsistencies in the development and deployment of cloud services. We identify two key scenarios that exemplify this challenge.

Existing legacy servers. Despite the rise of SmartNICs, CSPs often operate large fleets of legacy general-purpose servers equipped only with basic NICs. Repurposing these servers for bare-metal use is challenging: retrofitting with DPUs incurs substantial costs beyond hardware acquisition—requiring upgrades to rack power supplies to support higher power draw, as well as enhancements to top-of-rack switches and routers to avoid bandwidth bottlenecks. Moreover, compatibility issues with older server platforms can limit full utilization of SmartNIC capabilities.

Customer-owned infrastructure in private clouds. In private cloud deployments, a common model involves customers providing their own physical servers, while the CSP deploys and manages cloud services on them. The decision to include SmartNICs rests with the customer, typically driven by cost or procurement policies. According to operational data from a major CSP, over 50% of newly procured compute nodes in private cloud environments lack DPU integration. Consequently, CSPs cannot directly reuse the mainstream software stack developed for DPU-based nodes in public cloud, resulting in operational fragmentation and degraded service consistency.

Limited resource elasticity. Cloud workloads exhibit diverse and dynamic I/O demands: some are compute-intensive with modest I/O requirements, while others are highly I/O-constrained, demanding extreme storage or network throughput. Many services further experience pronounced load fluctuations—such as e-commerce platforms during sales events—resulting in sharp peaks and troughs in I/O activity.

However, DPUs offer fixed hardware resources, including CPU cores, DRAM capacity, and FPGA/ASIC accelerators, which cannot be elastically scaled. To meet service-level objectives, providers typically provision for peak loads. Analysis of a production cluster at a major CSP reveals that, under this provisioning strategy, DPU I/O processing capacity remains below 33% utilization for over 99% of the observed period [25]. This chronic underutilization reveals a fundamental mismatch between static hardware allocation and dynamic workload demands, leading to poor cost efficiency.

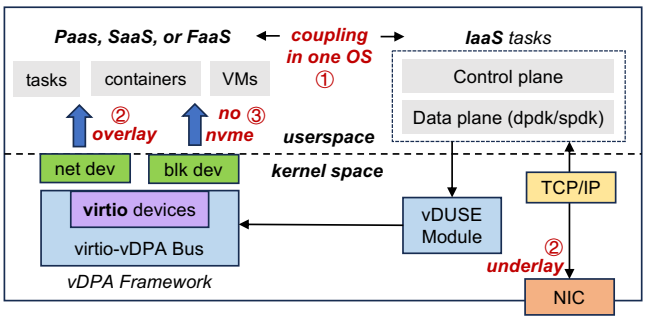


Figure 2: vDUSE and its limitations

Slow-path processing bottlenecks. A typical DPU employs data-plane accelerators to offload fast-path traffic, while delegating necessary slow-path operations to its general-purpose processors [38]. In VPC networks, even with hardware-accelerated fast paths, a subset of traffic—such as state transitions and exception handling—still traverses these processors [44]. The bottleneck is more severe in cloud storage: despite advanced, production-grade architectures like Solar [50] that offload functional units (e.g., CRC, encryption, and packet generation) to FPGA, every block I/O request still requires CPU cycles to process service headers, RPC metadata, and congestion control logic.

However, resources in the DPU's general-purpose processing subsystem are both constrained and precious. For instance, the BlueField-3 DPU provides at most 16 ARM cores and 32 GB of DRAM [54]. Consequently, slow-path processing can become a performance bottleneck for certain cloud services. Network function workloads, for example, may manage millions of concurrent connections [70], generating sustained slow-path traffic that can saturate the general-purpose CPU or memory capacity.

3.2 Limitations of Existing Approaches

On one hand, much prior work on SmartNICs [37, 45, 47, 50, 57, 58, 62, 66] focuses on *what* to offload and *how* to optimize offload efficiency—yet provides little support for addressing the practical challenges outlined above in production cloud environments. On the other hand, I/O virtualization research [31, 41, 69, 75] has largely remained confined to hypervisor-based VM environments, offering minimal support for bare-metal execution models.

The userspace vDPA device (vDUSE) [68] implements a software-defined data path to expose virtio devices to the host server, as illustrated in Figure 2. It can partially alleviate the elasticity challenge, yet several limitations impede its practical adoption in production environments.

First, storage and networking tasks—being part of the IaaS layer—run in the same OS as higher-layer cloud services (e.g., PaaS). This tight OS coupling across service abstraction layers is inherently conflict-prone: the IaaS stack requires a stable, vendor-controlled kernel with low-level tuning, whereas higher-layer services demand flexibility in kernel versions and runtime configurations (e.g., memory and I/O parameters). In practice, kernel isolation is essential for multi-team collaboration, enabling independent development, versioning, and maintenance.

Second, OS coupling complicates native VPC support for workloads on the host. Different service layers require logically isolated networks to enable independent addressing of their distributed machines. In practice, the IaaS layer relies on the underlay network for direct connectivity and performance, while higher-layer services use VPC overlays for flexibility and reliability. Co-locating both models in a single OS results in complex, error-prone network configurations and security vulnerabilities.

Finally, vDUSE is built on the virtio data path acceleration framework, which restricts it to virtio-based devices. However, many cloud services depend on advanced NVMe features—such as multi-attached shared disks for database systems [16]—that fall outside the virtio model.

3.3 Opportunities

Service VM as a solution. Encapsulating storage and networking processing within a dedicated service VM offers a promising path toward decoupling infrastructure functions from the host OS. With guest cooperation, a range of paravirtualized techniques [17] can be leveraged to minimize the virtualization overhead incurred by the service VM. Yet, realizing efficient I/O requires solving the *semantic mismatch*: how to project high-level device abstraction from the service VM to the host in a performant and scalable manner. ZOC addresses this by emulating DPU-equivalent I/O abstractions and enabling key functions for virtual devices: MMIO, DMA transfers, and interrupts.

Elastic resources from the host. Cloud servers are typically equipped with hundreds of CPU cores and terabytes of DRAM. Moreover, the turbo frequency of server CPUs far exceeds that of DPU SoCs, and core counts per socket continue to scale, reaching up to 256 logical cores in Intel Granite Rapids [35]. Leveraging the host's abundant computing resources, ZOC enables dynamic CPU and memory allocation to overcome the rigidity of DPUs with fine-grained resource elasticity.

Flexibility in idle resource harvesting. The service VM can be assigned dedicated CPU cores to guarantee I/O performance. Alternatively, through dynamic vCPU scheduling, it can opportunistically utilize otherwise-idle computing resources from hosted cloud workloads. This is motivated by the widespread prevalence of underutilized CPU capacity in cloud datacenters [48, 65]. Leveraging the vCPU scheduling flexibility, ZOC introduces an optional zero-cost running mode, further avoiding the cost of reserved cores.

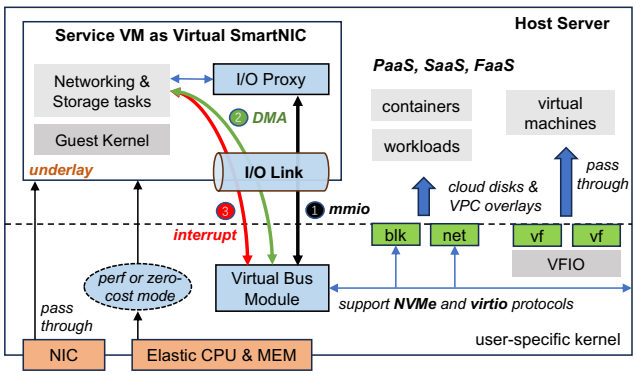


Figure 3: Overview of ZOC architecture

4 Design

ZOC transforms general-purpose servers equipped with only basic NICs into full-featured cloud physical machines. It delivers a consistent user experience—functionally and operationally equivalent to that of DPU-based platforms—while avoiding their limitations in production environments.

4.1 Architectural Overview

The architecture of ZOC is illustrated in Figure 3. At its core, ZOC instantiates a *service VM* that combines a passthrough physical NIC with dynamic computing resources (CPU, memory) provisioned from the host, forming a software-defined virtual SmartNIC. The use of the service VM enables both operational consistency and resource elasticity. It supports two running modes: a default performance mode using dedicated cores, and an optional zero-cost mode leveraging idle cycles.

Built upon the service VM, an efficient cross-domain device abstraction is provided through three key components.

- *I/O Proxy* (in the service VM): prepares I/O device emulation with either virtio or NVMe protocol; collaborates with the networking and storage infrastructure to access foundational cloud resources and build the I/O data plane.
- *Virtual Bus* (in the host): exposes standardized I/O device interfaces to the host server, which can be driven by native host drivers to serve host workloads or containers, and can also be claimed via the mainstream VFIO framework for direct device passthrough to VMs.
- *I/O Link* (between them): bridges the semantic gap between the above two execution domains by providing essential functions, including (1) cross-domain communication to forward MMIO messages for device discovery or configuration, (2) DMA capability for zero-copy I/O data transfer, and (3) low-latency interrupt delivery to enable fast I/O signaling.

4.2 Service VM

Operational consistency: The service VM provides DPU-like isolation by running in a dedicated OS fully decoupled from the host. The passthrough NIC enables efficient, transparent access to the underlay network. This architecture aims to deliver two key dimensions of consistency. First, the majority of existing infrastructure components running on the DPU can be reused within the service VM. Second, cloud disks and VPC networks are seamlessly exposed to the host, providing consistent user experience for cloud workloads that utilize DPU-based machines.

Resource elasticity: Leveraging the host's abundant resource pool, the service VM can be elastically configured—from lightweight instances (e.g., 2 vCPUs, 8 GB RAM) for low-I/O services to scaled-up setups (e.g., 16 vCPUs, 64 GB RAM) for high-throughput, concurrent workloads. Crucially, ZOC supports runtime resource adjustment via mature CPU and memory hot-plugging, enabling seamless adaptation to dynamic I/O patterns: scaling up during peaks and down during troughs to optimize cost-performance trade-offs. Moreover, it alleviates bottlenecks in the DPU's general-purpose processing subsystem by offloading computation to the host's scalable resources.

Running modes: The service VM supports multiple execution models to balance performance and resource efficiency. In performance mode, it uses dedicated CPU cores—achieved via vCPU pinning—to guarantee I/O performance. Alternatively, in zero-cost mode, it harvests idle CPU cycles from co-located workloads, eliminating the need for reserved resources. To ensure predictable I/O behavior, the service VM employs continuously scheduled vCPUs in this mode, thereby eliminating execution gaps and bounding I/O latency. When no idle cycles are available, the system triggers round-robin preemption across busy cores, amortizing interference while preserving continuous execution. Furthermore, ZOC supports a hybrid model, where some vCPUs are pinned to dedicated cores while others operate in zero-cost mode—enabling fine-grained adaptation to diverse I/O workload demands.

4.3 Device Abstraction and MMIO

The device abstraction in ZOC follows the classic trap-and-emulate paradigm, but extend it to a novel cross-domain architecture that spans the host and the service VM. This design uniformly exposes standardized I/O devices—NVMe and virtio-blk for cloud storage, and virtio-net for VPC connectivity—to both bare-metal hosts and provisioned VMs.

Figure 4 illustrates the overall framework and the MMIO workflow for device discovery and configuration. The Virtual Bus, residing in the host, acts as the frontend interface layer, seamlessly integrating with the kernel's device model to expose device interfaces. The I/O Proxy, located in the service VM, serves as the backend emulation engine, responsible

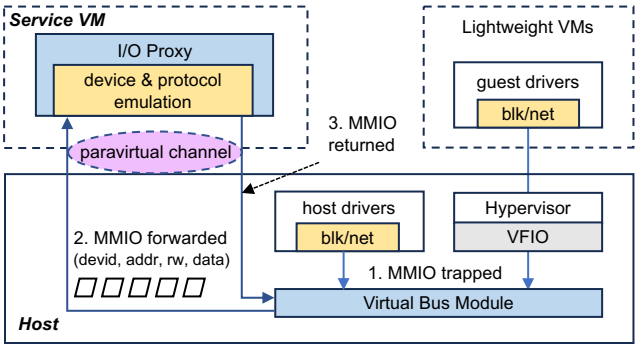


Figure 4: Device framework and MMIO workflow

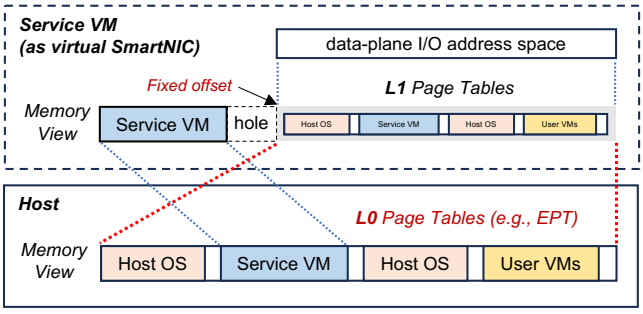


Figure 5: Two-dimensional page tables for DMA

for implementing device logic and protocol semantics (e.g., NVMe or virtio). An intermediate communication channel, enabled by the I/O Link, provides secure and low-latency messaging between these two domains, ensuring efficient forwarding of control-plane MMIO operations.

When ZOC creates a new I/O device, it is first registered on this Virtual Bus. This device can then be directly probed by native host drivers, serving applications running directly on the host or containers. Or, it can be claimed by the mainstream VFIO framework [40] for further passthrough to VMs. MMIO operations issued by host or guest device drivers are intercepted by the Virtual Bus, encapsulated, and forwarded to the backend I/O Proxy via the cross-domain channel. The I/O Proxy in the service VM emulates the target device's register semantics, processes the MMIO request, and returns the result through the same path, preserving full compatibility with standard device protocols.

4.4 DMA Capability

The I/O data plane in the virtual SmartNIC requires DMA capability to efficiently access memory in the target domain, whether the host or a VM, to which the virtual device is finally assigned. To enable this, the I/O Link provides two key mechanisms: (1) a carefully crafted two-dimensional memory mapping, allowing the data plane in the service VM to directly read from and write to the memory of the target domain; and (2) per-domain I/O address space management and isolation.

As illustrated in Figure 5, the I/O Link takes advantage of hardware-assisted L_0 page tables, such as Intel EPT [33], to map the entire host physical address (HPA) space into the service VM's guest physical address (GPA) space. This mapping is performed with a fixed, reserved GPA `offset` (e.g., 1 TB) to achieve a deterministic and fast translation between HPAs and GPAs. With this layout, any memory access to HPA `X` can be transparently issued from the service VM by accessing GPA `X + offset`. For devices assigned to the host domain, the I/O Link further maps the GPA-accessible HPA space into the userspace data plane using the guest's page tables (i.e., L_1 page tables). As a result, the data plane can directly use its I/O virtual addresses to initiate DMA operations for device data transfers. Although a DMA operation involves two levels of page table translation, modern CPUs can perform this two-dimensional address translation efficiently through hardware acceleration.

When an I/O device is detached from the host and assigned to a provisioned VM, the Virtual Bus retrieves the VM's memory layout and forwards it to the service VM. The I/O Link then establishes a dedicated I/O address space for the VM, enabling isolated and secure DMA address translation. Since the VM's memory is ultimately backed by host memory, the existing L_0 page tables that map the entire HPA space can be reused for address resolution. By combining the L_0 mapping with the VM's memory layout, the I/O Link aggregates scattered host pages into a contiguous address range. When multiple virtual devices are assigned to the same VM domain, they can share the I/O address space to reduce metadata overhead.

4.5 Interrupt Delivery

The I/O Link's interrupt controller enables low-latency delivery of device interrupts from ZOC's data plane to their target domains—either the host or a provisioned VM. In traditional setups, the I/O data plane can leverage host-side notification interfaces for direct interrupt injection. In ZOC, however, the data plane resides in the service VM, necessitating a cross-domain interrupt delivery mechanism.

A straightforward method uses VM hypercalls [6] to bridge the semantic gap. During device configuration, the I/O Proxy records the interrupt metadata (e.g., vector, destination, delivery mode) provided by the driver in the target domain. When an interrupt is triggered in the data plane, the I/O Link intercepts the interrupt request and forwards it to the host via a hypercall. The host-side hypercall handler translates the interrupt information and invokes the appropriate notification mechanism completing the delivery path, such as triggering the host's device interrupt handler, or injecting a virtual interrupt into a VM.

While functional, the above method incurs latency overhead due to expensive VM Exits [17] on interrupt triggering,

which may become a bottleneck under high I/O load. To eliminate this overhead, ZOC adopts an optimized design: the physical interrupt command register (ICR) is mapped into the service VM. By writing to this register, the I/O Link can directly trigger a special inter-processor interrupt (IPI) on the host, invoking a pre-registered handler without requiring a VM Exit. This handler retrieves the corresponding virtual interrupt context and delivers the event to the target domain. For platforms supporting hardware interrupt virtualization, such as Intel Posted Interrupt [33], this optimization can be further extended to eliminate VM Exits on the receive-side vCPU, enabling minimal interrupt latency for VMs.

5 Implementation

ZOC is designed to run on commodity CPUs and NICs from diverse vendors, relying only on standard virtualization primitives. We have implemented ZOC on both Intel and AMD server platforms, using a range of commodity NICs, including Intel XL710, E810, and Mellanox ConnectX-4/5/6 adapters.

5.1 Core Components

ZOC enables virtual SmartNIC functionality using loadable kernel modules and userspace components. The service VM is a standard virtual machine instantiated by the host hypervisor (QEMU [23] and KVM [39]). A dedicated KVM instance [73] is leveraged for the service VM to isolate its lifecycle and software stack from tenant VMs, enabling independent development and maintenance. On NUMA-enabled servers, the service VM can dynamically allocate CPU and memory resources across different nodes to align with NIC locality and workload affinity. To minimize virtualization overhead, the service VM is optimized with large pages, non-exiting halt mode, and a range of para-virtualized techniques (e.g., direct timer interrupts and IPIs [17]).

The I/O Proxy in the service VM is implemented as a purpose-built QEMU instance that emulates only necessary device models, i.e., NVMe, virtio-blk, and virtio-net. It collaborates with networking and storage stacks via standard `vhost-user` protocols [24] to establish the data plane for device abstraction. On the host side, the Virtual Bus is implemented as a kernel module that registers a custom MMIO bus, fully compliant with Linux Bus-Device-Driver model [64]. The I/O Link is implemented jointly by the host hypervisor and kernel-resident components in the service VM. The para-virtualized communication channel uses a shared-memory ring buffer for efficient bulk transfer of MMIO messages between the host and service VM. Control signals, including request notifications and configuration completions, are delivered asynchronously via VM interrupts and hypercalls.

To support NVMe disks assigned to the host, the Virtual Bus implements the `nvme_ctrl` interface and `blk_mq_ops` callbacks, enabling seamless integration with the kernel's NVMe core subsystem. For virtio devices, it implements the `virtio_config_ops` interface, serving as an MMIO-based transport layer that plugs into the existing virtio framework. For provisioned VMs, ZOC provides full compatibility with the mainstream VFIO framework [40] for device passthrough. The Virtual Bus registers a new device type with the VFIO base driver by implementing the `vfio_device_ops` interface, allowing the host to discover and configure I/O devices via standard VFIO control interfaces.

ZOC supports independent hot updates of all functional modules for security patching or feature rollout. For userspace components (including the service VM), a state-preserving process replacement mechanism [73] is leveraged to save current states in the old process and then load them in the newly forked one with updated codebase. For kernel-space components, ZOC adopts a two-layer design inspired by Read-Copy Update (RCU): a lightweight wrapper layer and a replaceable implementation layer. During an update, a new implementation is loaded and registered, and the wrapper's function pointer is atomically switched to the new version. This in-place update of kernel modules has no impact on I/O data paths.

5.2 DMA & Interrupt

The L_0 page tables, mapping the entire HPA space, are established by KVM using hardware virtualization extensions: Intel EPT [33] or AMD NPT [4]. To minimize page-walk overhead and improve TLB efficiency, large-size mapping are used for most HPAs. Security-critical regions, such as kernel text, data segments, and reserved memory, are explicitly excluded from the mapping to prevent unauthorized access from the service VM, preserving host OS integrity. Based on `vhost-user` negotiation, the I/O Link intercepts `mmap` requests from the data plane and constructs the L_1 page tables. For managing I/O address spaces for VM domains, ZOC implements a virtual IOMMU device [5]. Its frontend resides in the host as part of the Virtual Bus, while the backend runs in the service VM within the I/O Proxy. This implementation enables the host hypervisor to invoke standard VFIO_IOMMU APIs to deliver the VM's memory layout to the service VM.

ZOC provides full support for migrating VMs with assigned virtual I/O devices. The I/O Proxy is responsible for saving and restoring device-specific states during migration. To accelerate the iterative pre-copy phase, the data plane tracks DMA-accessed dirty pages at the granularity of per-domain I/O address space, rather than per-device. The host hypervisor interacts with the Virtual Bus via standard VFIO APIs to control dirty page logging and retrieve accumulated memory change information.

When the data plane performs the write operation to trigger a device interrupt, the write is trapped by the I/O Link to complete the delivery based on ICR passthrough. For devices assigned to the host, interrupt delivery is implemented by

programming the IPI destination to the physical core where the device interrupt is bound. The corresponding IPI handler on that core then directly invokes the registered device interrupt handler. For VM domains, the I/O Link extends this mechanism by mapping the virtual interrupt controllers of the tenant VM into the service VM. On Intel platforms, for example, when an interrupt is triggered, the I/O Link writes the pending interrupt vector to the mapped controller and issues a specialized IPI (namely POSTED_INTR [33]) to trigger hardware-assisted delivery. It eliminates VM Exits on both the sender (service VM) and receiver (tenant VM) sides.

5.3 Zero-Cost Scheduling

In zero-cost mode, the service VM's vCPUs are scheduled across physical cores using a configurable time slice, enforced by hardware mechanisms such as the VMX preemption timer [33]. A smaller time slice reduces latency interference on the co-located primary workload, while a larger one improves I/O processing efficiency. To simplify idleness detection and scheduling decisions, physical cores are grouped into multiple scheduling domains, each mapped to one zero-cost vCPU. After completing a time slice, the vCPU is preferentially rescheduled on its previous core—if still idle—to preserve cache affinity. Otherwise, it is scheduled on a randomly selected idle core from the domain.

Under extreme load when idle cycles are unavailable, the service VM applies round-robin preemption across busy cores to prevent uncontrolled I/O latency. Operators deploying ZOC in zero-cost mode must assess expected I/O load and acceptable performance interference under worst-case conditions—quantified in our evaluation. For instance, on a 128-core host that permits up to 3% CPU overhead for the primary workload, approximately 4 zero-cost vCPUs can be safely allocated to the service VM. The entire scheduling logic is integrated into the KVM hypervisor, enabling fast vCPU control and tight coordination with idle-state management.

6 Evaluation

We evaluate ZOC in production clouds across multiple dimensions, including performance scalability, cost efficiency, slow-path throughput, cloud compatibility, and maintainability.

Experimental Setup. ZOC was deployed on a dual-socket server equipped with two Intel Xeon Platinum 8369B processors, each with 64 logical cores (128 cores total), operating at 3.5 GHz turbo mode, and 1 TB of DRAM. The service VM was configured with varying host resources and a passthrough Mellanox ConnectX-5 NIC (dual-port 25 GbE), exposing cloud disks (virtio-blk or NVMe) and virtio-net interfaces to the host. By default, networking and storage tasks were allocated roughly equal shares—about half each—of the service VM's computing resources. Both the service VM and the host ran a standard Linux kernel (version 5.10).

For comparison, we evaluated ZOC against the BlueField-3 P-Series DPU (BF3 for short) [54], a modern SoC-based SmartNIC integrating 16 Arm Cortex-A78AE cores, 32 GB of on-board DRAM, and ConnectX-7 subsystem (dual-port 200 GbE). The networking and storage tasks were offloaded to the BF3 DPU, using the official DOCA SDK [53] to make the most of its hardware capabilities.

6.1 Elasticity and Cost Efficiency

We evaluated cost efficiency by comparing the incremental hardware cost of deploying ZOC (performance mode) versus BF3, under equivalent I/O performance demands.

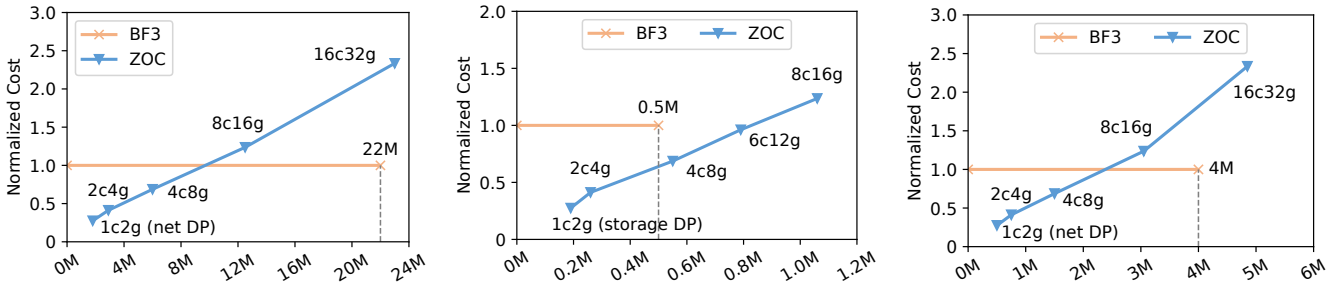
6.1.1 Estimation of Hardware Cost

Based on publicly available market data [61]—without uniformly accounting for volume procurement discounts—the high-end BF3 P-Series costs approximately $2,500 more than the basic NIC used in ZOC. In our deployment, the offloaded infrastructure workloads on BF3 primarily utilize the general-purpose ARM cores, the DMA engine (which enables efficient data movement between the host and the DPU) [38], and the ConnectX-7 NIC subsystem, especially the embedded switch (eSwitch). However, certain hardware capabilities—such as the RISC-V processors [15] and look-aside accelerators (e.g., for RegEx, encryption, and compression)—are currently underutilized by the offloaded workloads. To ensure a fair cost comparison, we apply a conservative cost-adjustment factor of 0.6 to account for this underutilization, yielding an effective incremental cost of $1,500 for BF3 in our evaluation.

The incremental cost of ZOC is determined by the CPU and memory resources allocated to the service VM. The tested Intel Ice Lake Xeon processor costs approximately $95 per logical core according to recommended customer price [36], and newer generations exhibit a downward cost-per-core trend [35]. For memory, the tested DDR4 ECC server DIMM costs about $4 per gigabyte. In performance mode, the service VM was configured with dedicated cores through vCPU pinning. The control plane (including essential OS services and the I/O Proxy) required a fixed allocation of 2 vCPUs and 4 GB RAM (denoted as *2c4g*), which incurred a total hardware cost of $206, amortized by the networking and storage functions (i.e., $103 for each). The cost of the data plane scales linearly with the amount of CPU and memory resources allocated to software stacks.

6.1.2 Key I/O Metrics

We first evaluated packet processing throughput using Netperf udp_stream [55] from remote clients to load the host server equipped with virtio-net interfaces, as shown in Figure 6a. The



(a) Network PPS (b) Storage IOPS (c) Nginx keepalive QPS

Figure 6: Cost efficiency comparison between BF3 and ZOC (performance mode) across key I/O metrics. BF3 has fixed normalized cost; ZOC delivers scalable performance and incurs lower cost for low-to-moderate I/O demands.

networking offload in BF3 achieves an upper limit of approximately 22 million packets per second (PPS) by offloading traffic processing to the eSwitch using pre-defined rules and accelerated pipelines—a fast path that can bypass the general-purpose ARM cores [38]. Its cost, priced at $750 for half of the DPU resources, is normalized to 1.0× baseline. ZOC's networking data plane (labeled "net DP" in the figure) scaled with allocated CPU and memory resources, ranging from *1c2g* to *16c32g*. At *4c8g*, ZOC provided 6 million PPS, sufficient for a wide range of cloud services, while incurring a total cost of $412 (networking data plane) + $206/2 (amortized control plane) = $515, or roughly 69% of the normalized BF3 cost. ZOC outperformed BF3 in cost efficiency across moderate PPS requirements until the target throughput approached 9.5 million PPS, beyond which the DPU acceleration became more economical.

The performance of cloud disk was evaluated with the FIO benchmark [11] (4KB random read/write) in the host, measuring I/O operations per second (IOPS) as a key metric for storage-intensive workloads, as shown in Figure 6b. BF3 delivered an upper bound of approximately 0.5 million IOPS, primarily constrained by the compute capacity of its general-purpose ARM subsystem. Despite leveraging hardware acceleration, block I/O requests must still be processed by the ARM cores to handle service headers and RPC metadata—fields that are privately defined by CSP [50]. In contrast, ZOC utilizes the host's high-performance cores for storage processing. With *4c8g* dedicated to the storage data plane (labeled "storage DP"), ZOC achieved comparable 0.55 million IOPS at less than 70% of the normalized BF3 cost. Moreover, scaling to *8c16g* enabled ZOC to exceed 1.06 million IOPS, more than doubling BF3's peak performance.

In terms of I/O bandwidth, the BF3 data plane achieved full line-rate utilization of its dual-port 200 GbE interfaces by evaluating multi-flow VPC networking and RDMA-accelerated cloud storage. In contrast, with a *4c8g* data plane configuration, ZOC provided either 38 Gbps of network bandwidth or 4 GB/s of disk bandwidth. ZOC still provided better cost efficiency for workloads with moderate bandwidth requirements.

Figure 6c illustrates the performance of Nginx evaluated using the wrk benchmark from remote clients, representing a comprehensive real-world workload that stresses the networking data plane. BF3 achieved a peak throughput of approximately 4 million queries per second (QPS) while utilizing half of the DPU resources at a cost of $750. In contrast, ZOC delivered nearly 75% of BF3's peak QPS with an *8c16g* configuration (as "net DP"). It can either reduce costs for workloads requiring below 2 million QPS (e.g., *4c8g*) or scale to larger configurations (e.g., *16c32g*) to surpass BF3's throughput ceiling.

In summary, ZOC's elastic architecture offers CSPs a compelling complement to fixed-cost DPU solutions, delivering superior cost efficiency, particularly under low to moderate I/O demands.

6.2 Zero-Cost Mode

The zero-cost mode leverages otherwise-idle CPU cycles from co-located cloud workloads to execute I/O tasks, thereby minimizing incremental hardware costs. When idle cores are available, vCPUs in this mode deliver performance comparable to the dedicated performance mode, and can even exceed it by avoiding hyper-threading contention. However, a critical question remains: under worst-case scenarios where no idle CPU cycles exist, how does this shared-resource model impact ZOC's I/O throughput, and what is the resulting interference with the primary workload?

To answer this, we configured four vCPUs in zero-cost mode to handle either control-plane or data-plane tasks. These vCPUs were scheduled across a pool of 96 physical cores occupied by a synthetic primary workload featuring controllable CPU utilization and instrumented to detect millisecond-scale jitter (>1 ms). As shown in Figure 7, using a short vCPU time slice of 150 μs as the preemption interval for control-plane tasks achieved approximately 82% of baseline performance (i.e., relative to dedicated cores in performance mode). In contrast, the storage data plane suffered more pronounced

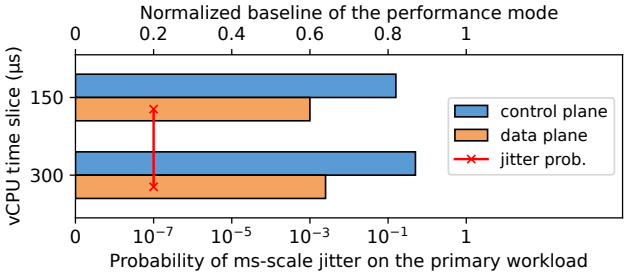
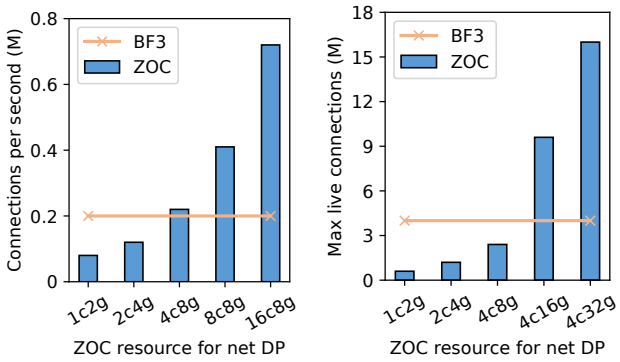


Figure 7: ZOC's performance in zero-cost mode under worst-case conditions (no idle cycles) and the induced latency interference on the primary workload.



(a) Network CPS (b) Max live connections

Figure 8: I/O performance for slow-path processing

performance degradation due to core migration and cache thrashing, achieving only about 60% of the baseline IOPS. Extending the vCPU time slice to 300 μs marginally improved efficiency in zero-cost mode. Crucially, the round-robin preemption strategy across busy cores ensures that the probability of millisecond-scale jitter in the primary workload remains below 10^{-7} (measured over ten million gateway ping samples), thereby preserving strong tail latency guarantees.

In summary, the zero-cost mode reduces incremental hardware overhead to near-zero (limited only to modest memory provisioning), without compromising I/O predictability or introducing significant latency interference on co-located cloud workloads.

6.3 Slow-path Processing

The performance of slow-path processing for storage I/O has been presented in Figure 6b. Since each block I/O request incurred general-purpose CPU cycles, the maximum IOPS on BF3 was capped at 0.5 M. In contrast, ZOC can effectively eliminate this bottleneck by leveraging the host's abundant and high-performance computing resources. With *8c16g* for the storage data plane, ZOC doubled the IOPS compared to BF3, demonstrating its ability to scale I/O performance.

Networking slow-path operations—such as connection establishment and state management for concurrent sessions—must be processed by the general-purpose ARM subsystem, making them susceptible to CPU and memory bottlenecks. We evaluated ZOC's advantage in this regime using two key metrics: connections per second (CPS), which reflects CPU-bound slow-path throughput, and maximum concurrent live connections, which is constrained by available memory. As shown in Figure 8a, BF3 achieved an upper bound of 0.2 M CPS, limited by the fixed CPU capacity. In contrast, ZOC scaled nearly linearly with allocated host resources; with *16c8g* for the networking data plane, it sustained 0.72 M CPS—a 3.6× improvement. For memory-bound workloads (Figure 8b), BF3 supported only 4 M concurrent connections due to the limited on-chip memory. ZOC, by provisioning up to 32 GB of host memory to its service VM, scaled to 16 M concurrent connections.

In summary, ZOC demonstrates significant advantages in slow-path processing by leveraging elastic host resources. Furthermore, it provides fine-grained flexibility in tailoring the CPU-to-memory ratio to match service requirements economically.

6.4 Compatibility

ZOC is designed to provide high compatibility with existing infrastructure, enabling unified deployment and operational consistency across both IaaS components running on the SmartNIC and cloud services deployed on the host. We evaluated this compatibility in a real-world production cloud.

First, the majority of existing IaaS components, developed for the CSP's DPUs, can be directly reused in ZOC's service VM without modification. These include OS images, management agents (e.g., cluster controllers, telemetry daemons), and device control planes. Moreover, the networking and storage tasks require only minor adjustments as the vhost-user protocol abstracts away hardware-specific differences in I/O address mapping and interrupt delivery.

Second, cloud services keep nearly identical runtime behavior on ZOC-based nodes compared to DPU-based ones. They can deploy the same OS versions, run unmodified applications or containers, and access cloud disks or VPC networks using existing toolchains. Launching VMs with VFIO devices backed by the Virtual Bus requires a minimal QEMU patch to ensure robust operation.

The benefit of above compatibility comes with a small virtualization overhead introduced by the service VM. As shown in Table 1, even under extreme load, ZOC incurs only a modest performance penalty on the I/O data plane compared to an all-in-one testing configuration without a service VM. Concretely, network packet processing (measured in PPS) experienced a 2% degradation for peak performance, and

Table 1: ZOC's virtualization overhead under extreme load

	Physical CPU	*ZOC vCPU*
Peak Network PPS	100%	98%
Peak Storage IOPS	100%	96.5%

Table 2: Hot updates of ZOC components

Components	*Service downtime*	*Scale with device counts*	*Influences*
Kernel parts	0	/	None
I/O Proxy	10+ ms	Yes	Control plane
Service VM	50 ms	No	Control & Data

storage I/O (measured in IOPS) suffered a 3.5% reduction.

6.5 Maintainability

A standing issue in deploying commercial DPUs (e.g., BF3) is poor maintainability: FPGA- or ASIC-based firmware typically requires a cold reboot for patches. Although CSPs may implement custom hot-update mechanisms for their in-house DPUs, such updates often demand redeploying the entire firmware image, even for changes to a single functional module, resulting in prolonged service downtime.

As a virtual SmartNIC, ZOC is designed from the ground up to support fine-grained updates across all components, both in kernel space and userspace, as summarized in Table 2. Kernel-resident components, including the DMA part, interrupt part, and Virtual Bus, are updated using an RCU-like mechanism that ensures zero service downtime. Ongoing DMA transfers and interrupt deliveries proceed uninterrupted.

In userspace, the service downtime of updating I/O proxy scales with the number of managed devices (e.g., 10 ms for four devices), but the impact is confined to the control plane. The hot update of the whole service VM (actually a process in the host) took about 50 ms, independent of device counts. Although it leads to the suspending of both control and data planes, the short duration enables operators to schedule maintenance during low-load periods with negligible user impact.

7 Deployment Experience

ZOC has been deployed at a major global CSP to serve both public and private cloud services. Below, we present key operational insights gained from real-world deployment.

OS and network isolation are critical for multi-team collaboration. The IaaS team and higher-layer service teams (e.g., PaaS) require isolated execution environments to enable independent development, versioning, and maintenance of their software stacks. Without such isolation, conflicting dependencies and configuration drift can lead to instability and deployment delays. This operational need is a primary driver for adopting the service VM model.

Key operational challenges: cooperative deployment, isolation, and high availability. First, the host OS image must include ZOC packages and configuration prior to initialization, requiring coordinated deployment and maintenance. Second, while bare-metal allocation eliminates noisy neighbors, performance interference from the ZOC's service VM can still arise due to shared resources like the last level cache; this can be mitigated via cache partitioning [32]. Third, the service VM introduces new failure points due to its independent OS—but also enables key reliability improvements, such as hot upgrades of all components, allowing timely patching of software vulnerabilities.

ZOC revitalizes legacy infrastructure. Cloud service teams often prioritize deploying workloads on aging servers, as these assets represent largely sunk costs, thereby minimizing amortization expenses. ZOC amplifies this economic advantage by transforming such legacy hardware into full-featured cloud physical machines, eliminating the need for costly retrofitting.

ZOC seamlessly supports VM-centric service models. DPU-based nodes can support not only bare-metal allocation, but also VM provisioning for tenants. ZOC replicates this capability by exposing standardized virtual devices through the VFIO framework, fully compatible with existing VM-centric service model.

Unified software stacks significantly reduce development and operational costs. By emulating DPU-equivalent I/O abstractions, ZOC enables general-purpose servers to run the same software stack, across IaaS components and higher-layer cloud services, as DPU-based nodes. This uniformity drastically reduces development effort and testing complexity. The benefit is especially pronounced when extending cloud platforms to private or hybrid environments, where customer-owned DPU-less infrastructure is dominant.

ZOC augments, rather than replaces, existing DPUs. ZOC is not mutually exclusive with physical DPUs; instead, they have the potential to cooperate synergistically:

- Elastic I/O expansion: ZOC can offload control-heavy I/O tasks while DPUs handle high-throughput data paths. This extends the effective capacity of early-generation or resource-constrained DPUs, and dynamically scales beyond their limits under overload.

- Pre-deployment simulation: As DPU development cycles span years, ZOC provides a software-equivalent environment aligned with target hardware. Critical functions, such as I/O offload, security enforcement, and live migration, can be validated in production-like settings before tape-out, reducing integration risk and prototyping cost.

8 Discussion

Use Cases and Limitations. ZOC is not a universal replacement for DPUs but a complementary solution that excels in specific scenarios—particularly legacy infrastructure modernization, workloads with moderate or dynamic I/O demands, and services with high slow-path demands. Conversely, DPUs remain superior for latency- or bandwidth-critical workloads that benefit from hardware-accelerated offloads (e.g., fast-path traffic processing, inline encryption). In these cases, fixed-function accelerators on modern DPUs deliver performance and power efficiency that software emulation cannot match.

ZOC's limitations stem from its software-defined nature. Although the passthrough NIC preserves native hardware offloads within the service VM—enabling efficient data-plane stacks such as DPDK and SPDK—these optimizations do not extend to the host or tenant VMs. Consequently, user applications interact with standard cloud-oriented I/O devices that lack hardware acceleration. Features requiring fixed-function logic (e.g., IPsec or TLS offload) are either unsupported or must be implemented in software, potentially degrading performance.

Security. Like the DPU architecture, ZOC's service VM is treated as part of the root of trust for the cloud infrastructure. It is protected from unauthorized access through hypervisor-enforced memory isolation, which can be further strengthened using confidential computing technologies such as Intel TDX [18]. To mitigate potential attacks from malicious or buggy drivers, ZOC enforces rate limiting on MMIO accesses over the Virtual Bus. Any read or write operation that exceeds a predefined frequency threshold is monitored and automatically suppressed, preventing resource exhaustion or denial-of-service via high-frequency MMIO traffic.

9 Related Work

SmartNIC Offloading. Offloading computation to SmartNICs has received extensive attention in both industry and academia. A wide range of functions have been successfully accelerated using programmable SmartNICs, including network virtualization [26, 58], storage systems [50, 72], microservices [19, 47], distributed transactions [60, 66], and others [37, 45, 57, 62]. These studies mainly focused on what to offload and how to optimize offload efficiency, while ZOC aims to address practical DPU limitations in production.

Service VM. The idea of using a dedicated VM for system-level services dates back to the XEN project [13], which introduced Domain 0 as a privileged VM to manage device emulation and I/O forwarding. However, Domain 0 is not designed to coexist with a bare-metal host domain that can be allocated to users. More recent approaches employed isolated service VMs to deploy virtual network functions [2, 27] or intrusion detection systems [51]. ZOC builds upon this principle by introducing a low-overhead service VM that runs alongside the host, providing an isolated execution environment for infrastructure tasks.

Virtualization. Prior work has long explored the use of dedicated CPU cores or even full servers to accelerate I/O virtualization for VM-centric models [31, 41, 69]. While ZOC similarly utilizes host CPU resources for I/O acceleration, it enables cross-domain device abstraction to serve the bare-metal host. Some studies [17, 29] introduced para-virtualized techniques (involving guest cooperation) to achieve near-bare-metal VM performance; ZOC leverages these to optimize the service VM. The state-of-the-art vDUSE [68] can provide virtio devices to the host server, yet as analyzed in §3.2, several limitations hinder its practicality in production cloud environments.

10 Conclusion and Future Work

This paper presents ZOC, an elastic and cost-efficient virtual SmartNIC architecture built entirely on commodity hardware. ZOC addresses key limitations of DPUs in production environments while matching the user experience of DPU-based nodes for cloud services. By dynamically integrating host-side computing resources into a dedicated service VM, ZOC achieves fine-grained elasticity, enabling efficient adaptation to diverse workloads. ZOC's cross-domain device abstraction exposes standardized I/O devices to both the host and provisioned VMs, ensuring compatibility with existing cloud stacks. Extensive evaluation demonstrates that ZOC outperforms modern DPU-based solutions in cost efficiency, slow-path throughput, and maintainability.

We envision several promising directions for the evolution of ZOC. First, ZOC can collaborate with existing DPUs to expand the overall I/O capabilities, as discussed above. Second, resource efficiency within the service VM itself can be improved through dynamic reclamation of idle CPU cycles and cold memory pages. Finally, in CXL-connected server racks, ZOC can provide remote I/O services by decoupling the service VM from its local host, enabling rack-scale elasticity and shared acceleration infrastructure.

Acknowledgments

We thank our shepherd, Eric Keller, and the anonymous reviewers for their insightful comments that improved the quality of this paper.

References

[1] Alexandru Agache, Marc Brooker, Alexandra Iordache, Anthony Liguori, Rolf Neugebauer, Phil Piwonka, and Diana-Maria Popa. Firecracker: Lightweight virtualization for serverless applications. In *17th USENIX*

symposium on networked systems design and implementation (NSDI), pages 419–434, 2020.

[2] Abdulrahman K Alnaim, Ahmed M Alwakeel, and Eduardo B Fernandez. A pattern for an nfv virtual machine environment. In *2019 IEEE International Systems Conference (SysCon)*, pages 1–6, 2019.

[3] AMD. Amd pensando dsc3-400 distributed services card, 2024. https://www.amd.com/content/dam/amd/en/documents/pensando-technical-docs/product-briefs/pensando-dsc3-product-brief.pdf, accessed August 2025.

[4] AMD. Amd64 architecture programmer's manual, 2025. https://docs.amd.com/v/u/en-US/40332-PUB_4.08, accessed August 2025.

[5] Nadav Amit, Muli Ben-Yehuda, Dan Tsafrir, Assaf Schuster, et al. {vIOMMU}: Efficient {IOMMU} emulation. In *2011 USENIX Annual Technical Conference (USENIX ATC)*, 2011.

[6] Nadav Amit and Michael Wei. The design and implementation of hyperupcalls. In *2018 USENIX Annual Technical Conference (USENIX ATC)*, pages 97–112, 2018.

[7] AWS. Amazon elastic block store, 2025. https://aws.amazon.com/ebs/, accessed August 2025.

[8] AWS. The security design of the aws nitro system. Technical report, Amazon Web Services, 2025. https://docs.aws.amazon.com/whitepapers/latest/security-design-of-aws-nitro-system/the-components-of-the-nitro-system.html, accessed August 2025.

[9] AWS. Use nvme reservations with multi-attach enabled amazon ebs volumes, 2025. https://docs.aws.amazon.com/ebs/latest/userguide/nvme-reservations.html, accessed August 2025.

[10] AWS. What is amazon vpc?, 2025. https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html, accessed August 2025.

[11] Jens Axboe. Flexible i/o tester, 2017. https://fio.readthedocs.io/en/latest/, accessed August 2025.

[12] Azure. Microsoft azure boost. Technical report, Microsoft, 2025. https://learn.microsoft.com/en-us/azure/azure-boost/overview, accessed August 2025.

[13] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *ACM SIGOPS operating systems review*, 37(5):164–177, 2003.

[14] Idan Burstein. Nvidia data center processing unit (dpu) architecture. In *Proceedings of the 2021 IEEE Hot Chips Symposium (HCS)*, pages 1–20, 2021.

[15] Xuzheng Chen, Jie Zhang, Ting Fu, Yifan Shen, Shu Ma, Kun Qian, Lingjun Zhu, Chao Shi, Yin Zhang, Ming Liu, et al. Demystifying datapath accelerator enhanced off-path smartnic. In *2024 IEEE 32nd International Conference on Network Protocols (ICNP)*, pages 1–12, 2024.

[16] Zongzhi Chen, Xinjun Yang, Feifei Li, Xuntao Cheng, Qingda Hu, Zheyu Miao, Rongbiao Xie, Xiaofei Wu, Kang Wang, Zhao Song, et al. Cloudjump: optimizing cloud databases for cloud storages. *Proceedings of the VLDB Endowment*, 15(12):3432–3444, 2022.

[17] Kevin Cheng, Spoorti Doddamani, Tzi-Cker Chiueh, Yongheng Li, and Kartik Gopalan. Directvisor: virtualization for bare-metal cloud. In *Proceedings of the 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE)*, pages 45–58, 2020.

[18] Pau-Chen Cheng, Wojciech Ozga, Enriquillo Valdez, Salman Ahmed, Zhongshu Gu, Hani Jamjoom, Hubertus Franke, and James Bottomley. Intel tdx demystified: A top-down approach. *ACM Computing Surveys*, 56(9):1–33, 2024.

[19] Sean Choi, Muhammad Shahbaz, Balaji Prabhakar, and Mendel Rosenblum. λ-nic: Interactive serverless compute on programmable smartnics. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 67–77, 2020.

[20] Alibaba Cloud. What is vpc?, 2025. https://www.alibabacloud.com/help/en/vpc/product-overview/what-is-vpc, accessed August 2025.

[21] Google Cloud. Data availability and durability, 2025. https://cloud.google.com/storage/docs/availability-durability, accessed August 2025.

[22] Alibaba Cloud Community. Enterprise-level tool: Alibaba cloud nvme disk and shared storage. Technical report, Alibaba Cloud, 2022. https://www.alibabacloud.com/blog/enterprise-level-tool-alibaba-cloud-nvme-disk-and-shared-storage_598857, accessed August 2025.

[23] The QEMU Project Developers. About qemu, 2025. https://www.qemu.org/docs/master/about/index.html, accessed August 2025.

[24] The QEMU Project Developers. Vhost-user protocol, 2025. https://qemu-project.gitlab.io/qemu/interop/vhost-user.html, accessed August 2025.

[25] Bang Di, Yun Xu, Kaijie Guo, Yibin Shen, Yu Li, Sanchuan Cheng, Hao Zheng, Fudong Qiu, Xiaokang Hu, Naixuan Guan, et al. Tai chi: A general high-efficiency scheduling framework for smartnics in hyperscale clouds. In *Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles (SOSP)*, pages 892–906, 2025.

[26] Daniel Firestone, Andrew Putnam, Sambhrama Mundkur, Derek Chiou, Alireza Dabagh, Mike Andrewartha, Hari Angepat, Vivek Bhanu, Adrian Caulfield, Eric Chung, et al. Azure accelerated networking: Smartnics in the public cloud. In *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 51–66, 2018.

[27] Tao Gao, Xin Li, Yu Wu, Weixia Zou, Shanguo Huang, Massimo Tornatore, and Biswanath Mukherjee. Cost-efficient vnf placement and scheduling in public cloud networks. *IEEE Transactions on Communications*, 68(8):4946–4959, 2020.

[28] Yixiao Gao, Qiang Li, Lingbo Tang, Yongqing Xi, Pengcheng Zhang, Wenwen Peng, Bo Li, Yaohui Wu, Shaozong Liu, Lei Yan, et al. When cloud storage meets rdma. In *Proceedings of the 18th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 519–533, 2021.

[29] Abel Gordon, Nadav Amit, Nadav Har'El, Muli Ben-Yehuda, Alex Landau, Assaf Schuster, and Dan Tsafrir. Eli: Bare-metal performance for i/o virtualization. *ACM SIGPLAN Notices*, 47(4):411–422, 2012.

[30] Yang Hang. A detailed explanation about alibaba cloud cipu. Technical report, Alibaba Cloud, 2022. https://www.alibabacloud.com/blog/a-detailed-explanation-about-alibaba-cloud-cipu_599183, accessed August 2025.

[31] Nadav Har'El, Abel Gordon, Alex Landau, Muli Ben-Yehuda, Avishay Traeger, and Razya Ladelsky. Efficient and scalable paravirtual i/o system. In *2013 USENIX Annual Technical Conference (USENIX ATC)*, pages 231–242, 2013.

[32] Andrew Herdrich, Edwin Verplanke, Priya Autee, Ramesh Illikkal, Chris Gianos, Ronak Singhal, and Ravi Iyer. Cache qos: From concept to reality in the intel® xeon® processor e5-2600 v3 product family. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 657–668, 2016.

[33] Intel. Intel® 64 and ia-32 architectures software developer manuals, 2025. https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html, accessed August 2025.

[34] Intel. Intel® ethernet network adapters product guide, 2025. https://www.intel.com/content/www/us/en/content-details/252454/intel-ethernet-network-adapters-product-guide.html, accessed August 2025.

[35] Intel. Intel® xeon® 6980p processor, 2025. https://www.intel.com/content/www/us/en/products/sku/240777/intel-xeon-6980p-processor-504m-cache-2-00-ghz/specifications.html, accessed August 2025.

[36] Intel. Intel® xeon® platinum 8368 processor, 2025. https://www.intel.com/content/www/us/en/products/sku/212455/intel-xeon-platinum-8368-processor-57m-cache-2-40-ghz/specifications.html, accessed August 2025.

[37] Arpan Jain, Nawras Alnaasan, Aamir Shafi, Hari Subramoni, and Dhabaleswar K Panda. Accelerating cpu-based distributed dnn training on modern hpc clusters using bluefield-2 dpus. In *2021 IEEE Symposium on High-Performance Interconnects (HOTI)*, pages 17–24. IEEE, 2021.

[38] Arjun Kashyap, Yuke Li, Darren Ng, and Xiaoyi Lu. Understanding the idiosyncrasies of emerging bluefield dpus. In *Proceedings of the 39th ACM International Conference on Supercomputing (ICS)*, pages 807–821, 2025.

[39] The kernel development community. Kvm – the linux kernel documentation, 2025. https://docs.kernel.org/virt/kvm/index.html, accessed August 2025.

[40] The kernel development community. Vfio – "virtual function i/o", 2025. https://docs.kernel.org/driver-api/vfio.html, accessed August 2025.

[41] Yossi Kuperman, Eyal Moscovici, Joel Nider, Razya Ladelsky, Abel Gordon, and Dan Tsafrir. Paravirtual remote i/o. In *Proceedings of the 21st International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. ACM, 2016.

[42] Qiang Li, Lulu Chen, Xiaoliang Wang, Shuo Huang, Qiao Xiang, Yuanyuan Dong, Wenhui Yao, Minfei Huang, Puyuan Yang, Shanyang Liu, et al. Fisc: a large-scale cloud-native-oriented file system. In *21st USENIX Conference on File and Storage Technologies (FAST)*, pages 231–246, 2023.

[43] Qiang Li, Qiao Xiang, Yuxin Wang, Haohao Song, Ridi Wen, Wenhui Yao, Yuanyuan Dong, Shuqi Zhao, Shuo Huang, Zhaosheng Zhu, et al. More than capacity: Performance-oriented evolution of pangu in alibaba. In *Proceedings of the 21st USENIX Conference on File and Storage Technologies (FAST)*, pages 331–346, 2023.

[44] Xing Li, Xiaochong Jiang, Ye Yang, Lilong Chen, Yi Wang, Chao Wang, Chao Xu, Yilong Lv, Bowen Yang, Taotao Wu, et al. Triton: A flexible hardware offloading architecture for accelerating apsara vswitch in alibaba cloud. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 750–763, 2024.

[45] Yuke Li, Arjun Kashyap, Weicong Chen, Yanfei Guo, and Xiaoyi Lu. Accelerating lossy and lossless compression on emerging bluefield dpu architectures. In *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 373–385. IEEE, 2024.

[46] Zijun Li, Jiagan Cheng, Quan Chen, Eryu Guan, Zizheng Bian, Yi Tao, Bin Zha, Qiang Wang, Weidong Han, and Minyi Guo. Rund: A lightweight secure container runtime for high-density deployment and high-concurrency startup in serverless computing. In *2022 USENIX Annual Technical Conference (USENIX ATC)*, pages 53–68, 2022.

[47] Ming Liu, Simon Peter, Arvind Krishnamurthy, and Phitchaya Mangpo Phothilimthana. E3:energy-efficient microservices on smartnic-accelerated servers. In *2019 USENIX Annual Technical Conference (USENIX ATC)*, pages 363–378, 2019.

[48] Chengzhi Lu, Kejiang Ye, Guoyao Xu, Cheng-Zhong Xu, and Tongxin Bai. Imbalance in the cloud: An analysis on alibaba cluster trace. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2884–2892, 2017.

[49] Mellanox. Connectx®-5 en card up to 100gb/s ethernet adapter cards. Technical report, Mellanox Technologies, 2020. https://network.nvidia.com/files/doc-2020/pb-connectx-5-en-card.pdf, accessed August 2025.

[50] Rui Miao, Lingjun Zhu, Shu Ma, Kun Qian, Shujun Zhuang, Bo Li, Shuguang Cheng, Jiaqi Gao, Yan Zhuang, Pengcheng Zhang, et al. From luna to solar: the evolutions of the compute-to-storage networks in alibaba cloud. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 753–766, 2022.

[51] Preeti Mishra, Vijay Varadharajan, Emmanuel S Pilli, and Uday Tupakula. Vmguard: A vmi-based security architecture for intrusion detection in cloud environment. *IEEE Transactions on Cloud computing*, 8(3):957–971, 2018.

[52] Andrew Moore and Jim Henrys. Ipu-based cloud infrastructure: The fulcrum for digital business. Technical report, Intel Corporation, 2021. https://www.intel.com/content/dam/www/central-libraries/us/en/documents/ipu-based-cloud-infrastructure-white-paper.pdf, accessed August 2025.

[53] NVIDIA. Doca overview, 2025. https://docs.nvidia.com/doca/sdk/doca-overview/index.html, accessed August 2025.

[54] NVIDIA. Nvidia bluefield-3 networking platform user guide, 2025. https://docs.nvidia.com/networking/display/bf3dpu/specifications, accessed August 2025.

[55] Hewlett Packard. Netperf, 2025. https://github.com/HewlettPackard/netperf, accessed August 2025.

[56] Tian Pan, Kun Liu, Xionglie Wei, Yisong Qiao, Jun Hu, Zhiguo Li, Jun Liang, Tiesheng Cheng, Wenqiang Su, Jie Lu, et al. Luoshen: A hyper-converged programmable gateway for multi-tenant multi-service edge clouds. In *Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 877–892, 2024.

[57] Noah Perryman, Sebastian Sabogal, Christopher Wilson, and Alan George. Dependable dpu architectures on amd-xilinx versal adaptive socs for space applications. *IEEE Transactions on Aerospace and Electronic Systems*, 2025.

[58] Yiming Qiu, Jiarong Xing, Kuo-Feng Hsu, Qiao Kang, Ming Liu, Srinivas Narayana, and Ang Chen. Automated smartnic offloading insights for network functions. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (SOSP)*, pages 772–787, 2021.

[59] Adam Ruprecht, Danny Jones, Dmitry Shiraev, Greg Harmon, Maya Spivak, Michael Krebs, Miche Baker-Harvey, and Tyler Sanderson. Vm live migration at scale. *ACM SIGPLAN Notices*, 53(3):45–56, 2018.

[60] Henry N Schuh, Weihao Liang, Ming Liu, Jacob Nelson, and Arvind Krishnamurthy. Xenic: Smartnic-accelerated distributed transactions. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (SOSP)*, pages 740–755, 2021.

[61] SHI. Computer hardware for your business workplace, 2025. https://www.shi.com/category/hardware, accessed August 2025.

[62] Kaushik Kandadi Suresh, Benjamin Michalowicz, Bharath Ramesh, Nick Contini, Jinghan Yao, Shulei Xu, Aamir Shafi, Hari Subramoni, and Dhabaleswar Panda. A novel framework for efficient offloading of communication operations to bluefield smartnics. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 123–133. IEEE, 2023.

[63] Michael S Tsirkin, Cornelia Huck, and P Moll. Virtual i/o device (virtio) version 1.2. *OASIS Committee*, 2022.

[64] Sreekrishnan Venkateswaran. *Essential Linux device drivers*. Prentice Hall Press, 2008.

[65] Yawen Wang, Kapil Arya, Marios Kogias, Manohar Vanga, Aditya Bhandari, Neeraja J Yadwadkar, Siddhartha Sen, Sameh Elnikety, Christos Kozyrakis, and Ricardo Bianchini. Smartharvest: Harvesting idle cpus safely and efficiently in the cloud. In *Proceedings of the 16th European Conference on Computer Systems (Eurosys)*, pages 1–16, 2021.

[66] Xingda Wei, Rongxin Cheng, Yuhan Yang, Rong Chen, and Haibo Chen. Characterizing off-path smartnic for accelerating distributed systems. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 987–1004, 2023.

[67] NVM Express Workgroup. Nvm express® base specification revision 2.3. *NVM Express Workgroup*, 2025.

[68] YongJi Xie and Jason Wang. Introducing vduse: a software-defined datapath for virtio. Technical report, 2022. https://www.redhat.com/en/blog/introducing-vduse-software-defined-datapath-virtio, accessed August 2025.

[69] Cong Xu, Sahan Gamage, Hui Lu, Ramana Kompella, and Dongyan Xu. vturbo: Accelerating virtual machine i/o processing using designated turbo-sliced core. In *2013 USENIX Annual Technical Conference (USENIX ATC)*, pages 243–254, 2013.

[70] Tingting Xu, Bengbeng Xue, Yang Song, Xiaomin Wu, Xiaoxin Peng, Yilong Lyu, Xiaoliang Wang, Chen Tian, Baoliu Ye, Camtu Nguyen, et al. Cyberstar: Simple, elastic and cost-effective network functions management in cloud network at scale. In *2024 USENIX Annual Technical Conference (USENIX ATC)*, pages 227–246, 2024.

[71] Chaoqun Zhan, Maomeng Su, Chuangxian Wei, Xiaoqiang Peng, Liang Lin, Sheng Wang, Zhe Chen, Feifei Li, Yue Pan, Fang Zheng, et al. Analyticdb: real-time olap database system at alibaba cloud. *Proceedings of the VLDB Endowment*, 12(12):2059–2070, 2019.

[72] Weidong Zhang, Erci Xu, Qiuping Wang, Xiaolu Zhang, Yuesheng Gu, Zhenwei Lu, Tao Ouyang, Guanqun Dai, Wenwen Peng, Zhe Xu, et al. What's the story in ebs glory: Evolutions and lessons in building cloud block store. In *Proceedings of the 22nd USENIX Conference on File and Storage Technologies (FAST)*, pages 277–291, 2024.

[73] Xiantao Zhang, Xiao Zheng, Zhi Wang, Qi Li, Junkang Fu, Yang Zhang, and Yibin Shen. Fast and scalable vmm live upgrade in large cloud infrastructure. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 93–105, 2019.

[74] Xiantao Zhang, Xiao Zheng, Zhi Wang, Hang Yang, Yibin Shen, and Xin Long. High-density multi-tenant bare-metal cloud. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 483–495, 2020.

[75] Zongpu Zhang, Jiangtao Chen, Banghao Ying, Yahui Cao, Lingyu Liu, Jian Li, Xin Zeng, Junyuan Wang, Weigang Li, and Haibing Guan. Hd-iov: Sw-hw co-designed i/o virtualization with scalability and flexibility for hyper-density cloud. In *Proceedings of the 19th European Conference on Computer Systems (Eurosys)*, pages 834–850, 2024.

[76] Zongpu Zhang, Chenbo Xia, Cunming Liang, Jian Li, Chen Yu, Tiwei Bie, Roberts Martin, Daly Dan, Xiao Wang, Yong Liu, et al. Un-iov: Achieving bare-metal level i/o virtualization performance for cloud usage with migratability, scalability and transparency. *IEEE Transactions on Computers*, 73(7):1655–1668, 2024.