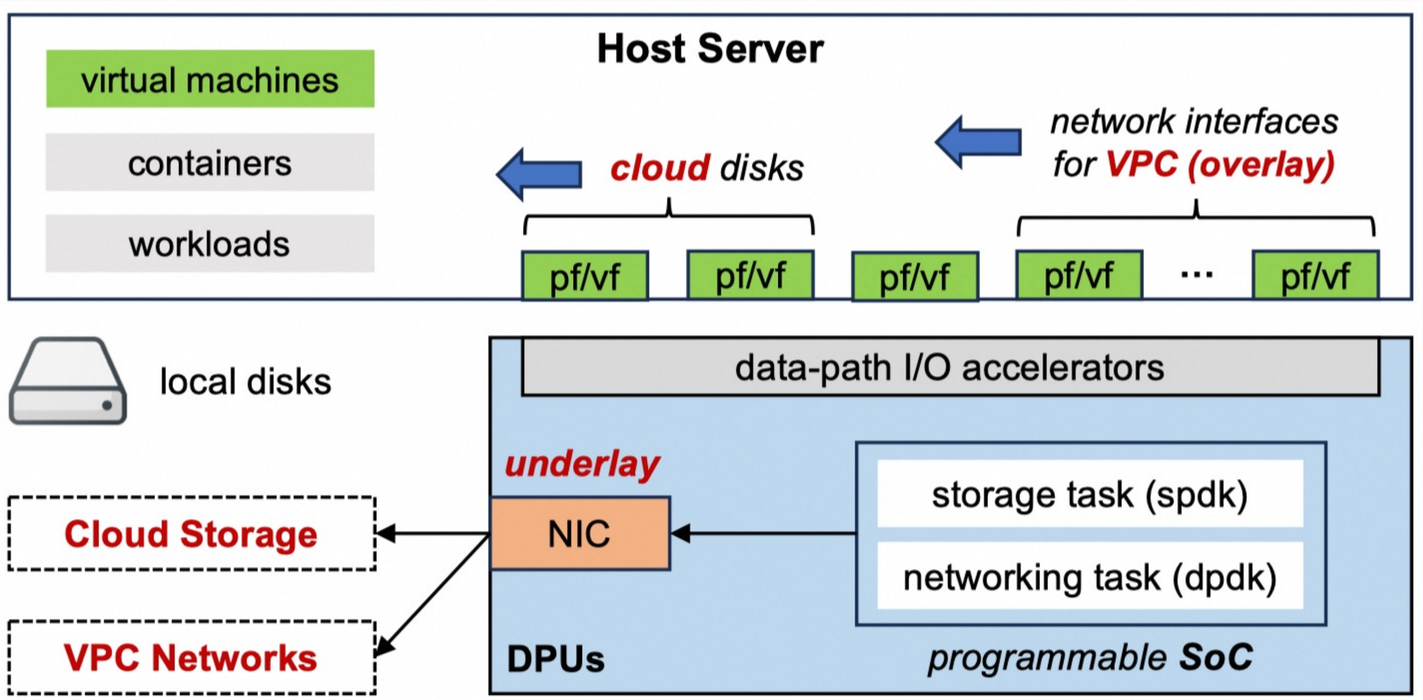
ZOC: Elastic and Cost-Efficient Virtual SmartNIC Architecture for Cloud Physical Machines

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

Background – From basic NIC to DPU

- **SmartNICs** extend basic NICs by integrating specialized or programmable processing units
- Data processing units (**DPUs**)—SmartNICs integrating a System-on-Chip (**SoC**)—have become central to modern cloud infrastructure



DPU-based cloud physical machine

- device model (PF/VF)
- classic VM support
- new model: bare-metal host

Motivation – DPU Challenges in Production

Heterogeneous Fleets (operational inconsistency)

- Legacy servers with only basic NICs; retrofitting with DPUs is costly
- Customer-owned infrastructure in private clouds: **> 50% new nodes lack DPU**

Limited Elasticity

- DPUs have fixed CPU/DRAM/accelerators, while cloud workloads exhibit dynamic I/O demands.
- Provisioned for peak loads to meet SLOs: **< 33% DPU utilization for > 99% of time**

Slow-Path Bottleneck

- Fast-path traffic (on data-plane accelerators) vs. Slow-path operations (e.g., state transitions and RPCs) on general-purpose subsystem (**constrained and precious**)
- Sustained slow-path traffic could saturate the general-purpose CPU/memory capacity.

Opportunities

Limitations of Existing Approaches

Most prior work on SmartNICs

Focuses on what to offload and how to optimize offload efficiency

vDUSE (userspace vDPA device)

Implements a software-defined data path to expose virtio devices to the host server (no NVMe support)

Tight OS coupling: Shared kernel for IaaS & PaaS leads to conflicts on versioning, networking, and maintenance

Key Opportunities

Service VM

Decouple infrastructure from host OS with guest cooperation & para-virtualization

Elastic Host Resources

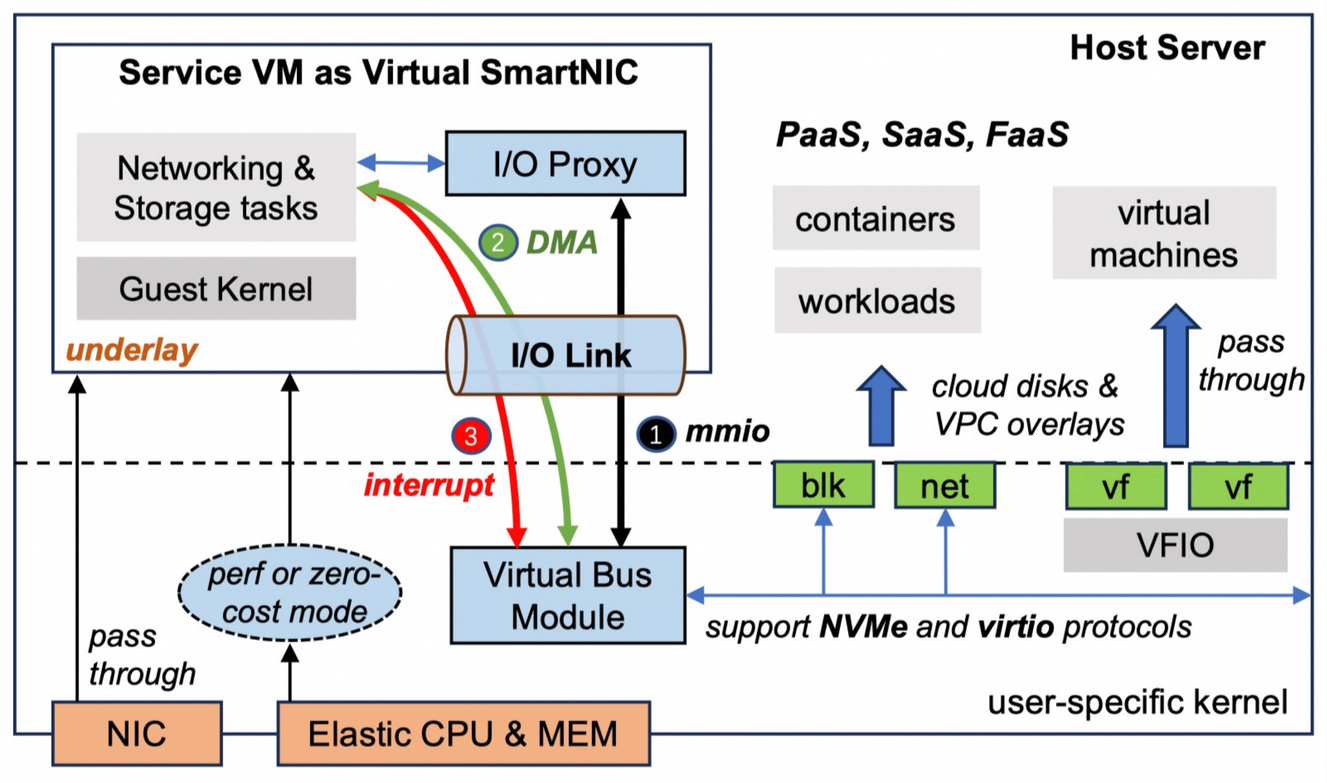
Servers have hundreds of cores & TBs of DRAM; turbo freq far exceeds DPU SoCs

Idle Resource Harvesting

Cloud CPUs are widely underutilized; opportunistic vCPU scheduling to utilize otherwise-idle CPU resources

How to project high-level device abstraction from the service VM to the host

ZOC Architecture – Software-Defined Virtual SmartNIC



A production-ready complement (with only basic NICs) to existing DPU-based architecture

Key Insight

Combine a passthrough NIC with dynamically provisioned host resources in a dedicated **Service VM** to form a virtual SmartNIC.

Three Components

I/O Proxy (Service VM)
Device & protocol emulation (virtio / NVMe)

Virtual Bus (Host)
Exposes standardized I/O device interfaces

I/O Link (Bridge)
MMIO, DMA transfer, interrupt delivery

Service VM Design

Operational Consistency

- Dedicated OS, fully decoupled from host; the passthrough NIC enables transparency.
- Reuse existing DPU infrastructure components; Consistent user experience for cloud workloads

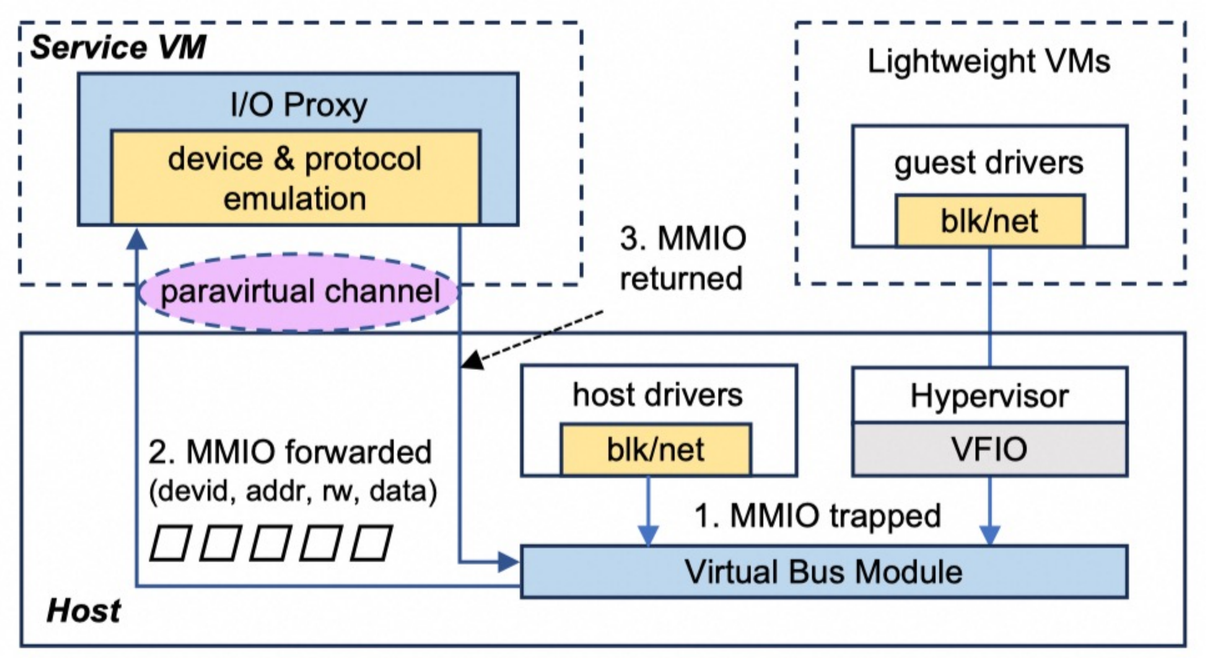
Resource Elasticity

- Elastically configured—from lightweight instances (e.g., 2c/8g) to scaled-up setups (e.g., 16c/64g)
- Runtime hot-plug of CPU or memory for dynamic I/O patterns

Two running modes

- Performance: dedicated CPU cores via vCPU pinning
- Zero-cost: harvest idle cycles from co-located workloads (controlled interference)

Cross-Domain MMIO for Device Abstraction



MMIO Workflow (Trap-and-Emulate)

1. **MMIO Trapped**
   Host/guest driver issues MMIO to Virtual Bus

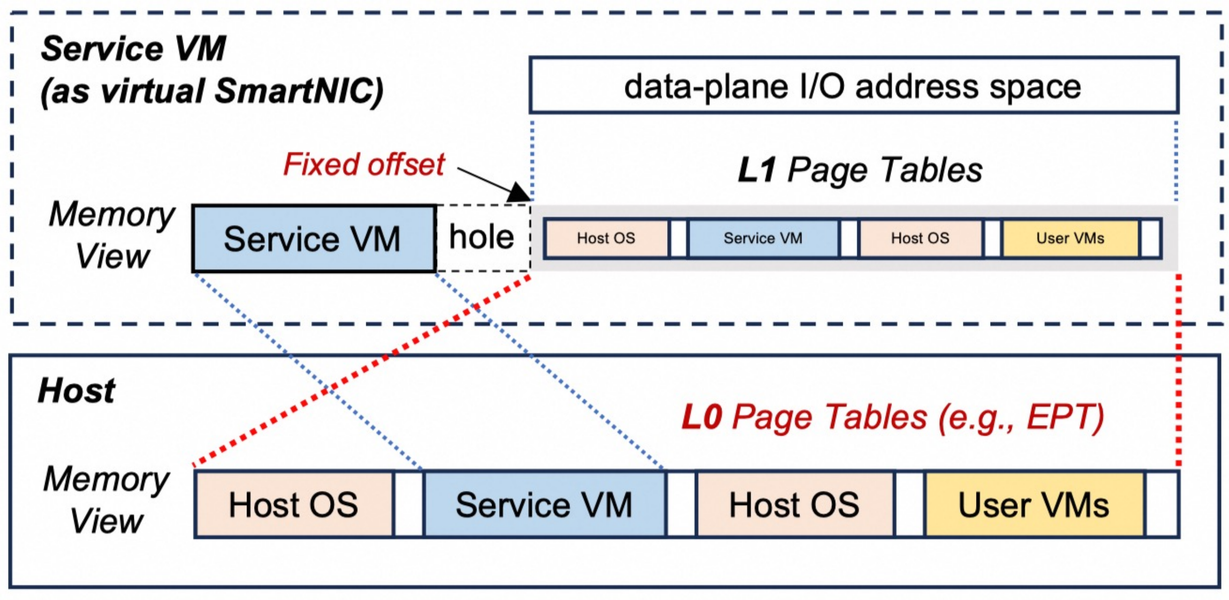
2. **MMIO Forwarded**
   Encapsulated & sent to I/O Proxy via paravirtual channel (devid, addr, rw, data)

3. **MMIO Returned**
   I/O Proxy emulates device register semantics, returns result through same path

Devices can be either probed by **native host drivers** or claimed via **VFIO** for guest passthrough

DMA Transfers & Interrupt Delivery

DMA via 2D Page Tables (Hardware-accelerated)



- L0 page tables map entire **HPA** into Service VM's **GPA** with fixed offset (fast calculation)
- L1 page tables map **GPA** to userpace **IOVA**
- Address space management: shared by devices for the same domain, isolated across domains (host, user VMs)

Efficient Interrupt Delivery

Basic
VM hypercall to forward interrupts (VM Exit overhead)

Optimized
Map **physical ICR** into Service VM; cross-domain IPI to forward interrupts (without VM Exit)

For user VMs
Map virtual interrupt controller; use Posted Interrupt for zero-exit delivery

Experimental Setup & Cost Model

ZOC Testbed

- 2x Intel Xeon 8369B (128 cores, 3.5GHz turbo), 1TB DRAM
- Mellanox ConnectX-5 as Basic NIC
- Linux kernel 5.10, varied Service VM configs

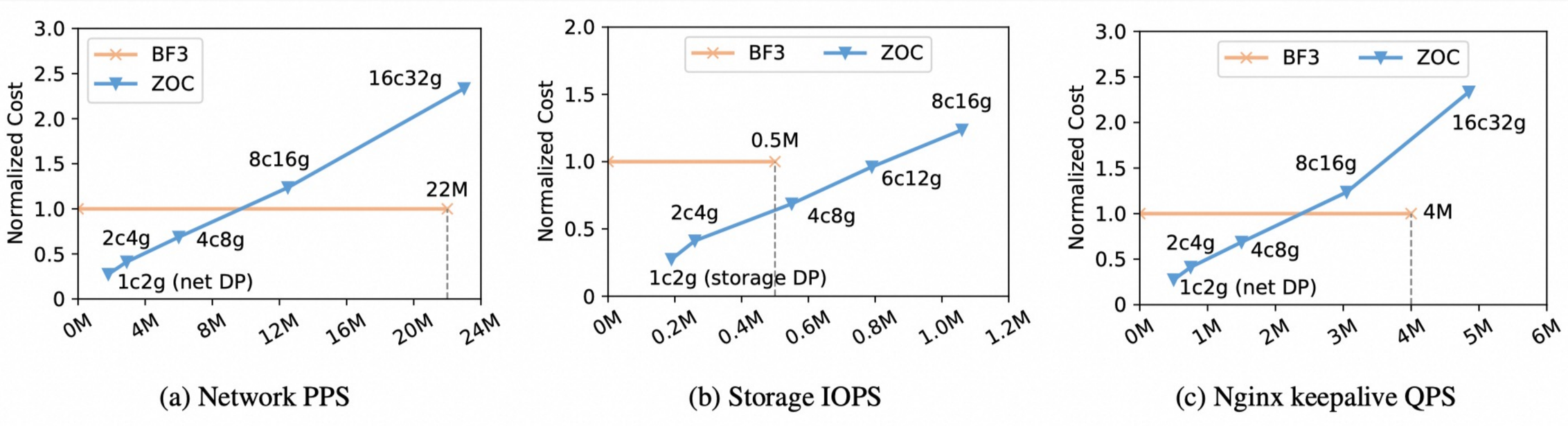
Comparison: BlueField-3 P-Series DPU (BF3)

- 16 ARM Cortex-A78AE cores, 32GB DRAM
- ConnectX-7 (dual-port 200GbE) NIC inside
- DOCA SDK utilized to maximize hardware capabilities

Incremental Hardware Cost

- BF3: ~$1,500 (estimated based on incremental retail price plus an adjustment factor)
- ZOC: $95 per logical core and $4 per gigabyte, **scales linearly** with the amount of CPU and memory resources allocated to the service VM

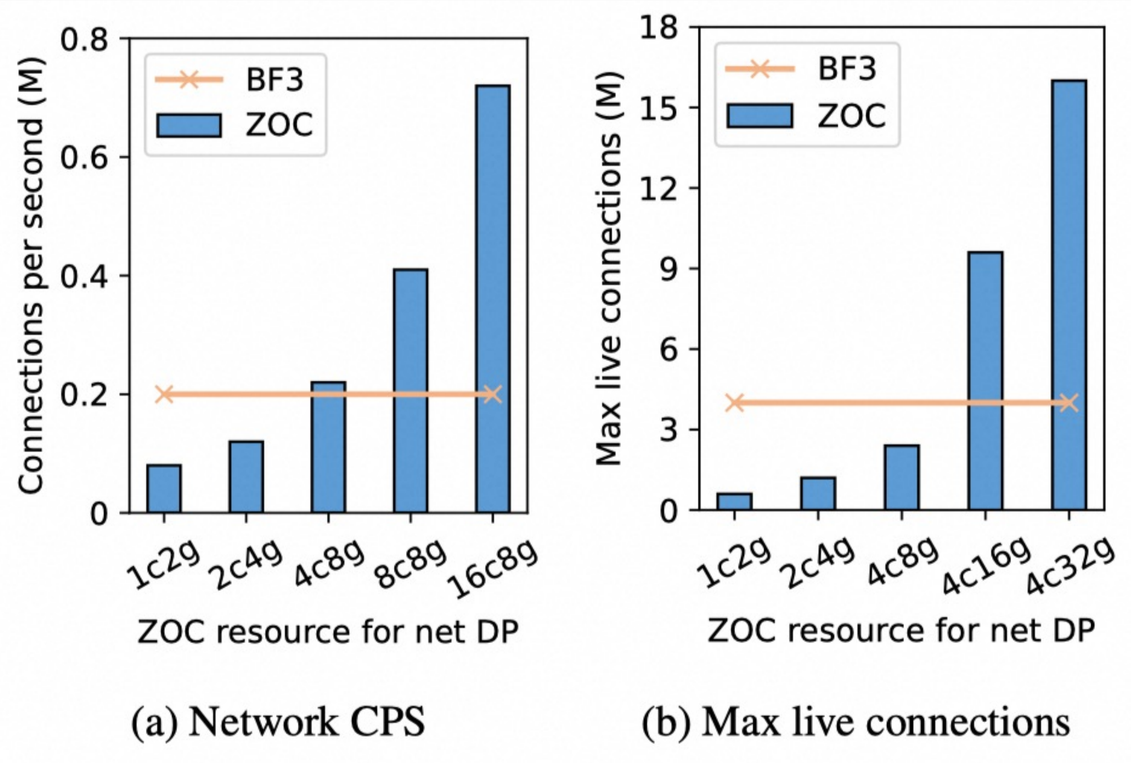
Cost Efficiency and Performance Scalability



BF3 has a fixed cost; ZOC has scalable performance and lower cost for low-to-moderate I/O demands

- Networking: 4c8g provides 6M PPS at 69% of BF3 cost, sufficient for diverse cloud services
- Storage: 8c16g delivers over 1M IOPS, more than doubling BF3’s peak performance

Slow-Path Processing



Connections per second (cpu-intensive)

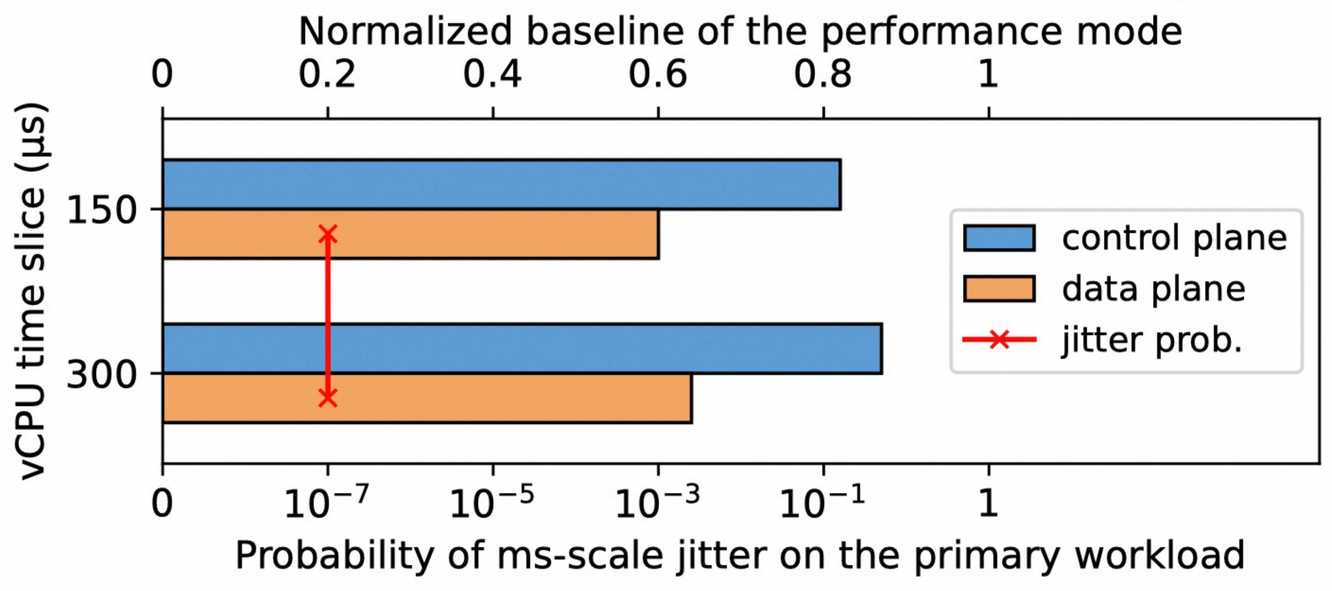
- 16c8g to deliver 0.72M CPS - 3.6x improvement over BF3 (0.2M)
- Near-linear scaling with host resources

Max Sessions (memory-intensive)

- 4c32g to deliver 16M sessions - 4x BF3 (4M)
- Fine-grained CPU-to-memory ratio tuning

Zero-Cost Mode

Question: In worst-case scenarios (no idle CPU cycles), what is ZOC's I/O throughput and the resulting interference?



Key Design

- Fine-grained vCPU time slice (150µs)
- Round-robin preemption across busy cores to amortize interference

- ~82% of baseline for control plane; ~60% of baseline for data plane (cache thrashing)
- Strong tail latency guarantee for primary workload: < 10^-7 probability of ms-scale jitter

Compatibility & Maintainability

High Compatibility in the Real Cloud

- Reuse most existing DPU IaaS components inside the service VM (OS, agents, control planes)
- Cloud services maintain nearly identical runtime behavior on ZOC-based nodes
- Only a small virtualization overhead

High Maintainability

- DPU: FPGA/ASIC firmware may require cold reboot even for minor changes
- ZOC: designed from the ground up to support fine-grained updates across all components

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

Conclusion & Future Work

Key Contributions

1. Transform legacy servers at **no additional hardware cost**
2. Enable **unified software stacks** across DPU-based and DPU-less environments
3. Fine-grained **resource elasticity** + zero-cost mode harvesting idle cycles
4. Performance **scalability** to eliminate DPU slow-path bottlenecks

Future Directions

- ZOC + DPU synergistic cooperation for expanded I/O capacity
- Reclamation of idle CPU cycles & cold memory pages within Service VM
- CXL-connected rack-scale elasticity: remote I/O services & shared acceleration

Thank You!

Email: xiaokang.hxk@alibaba-inc.com