Kernel Bypass

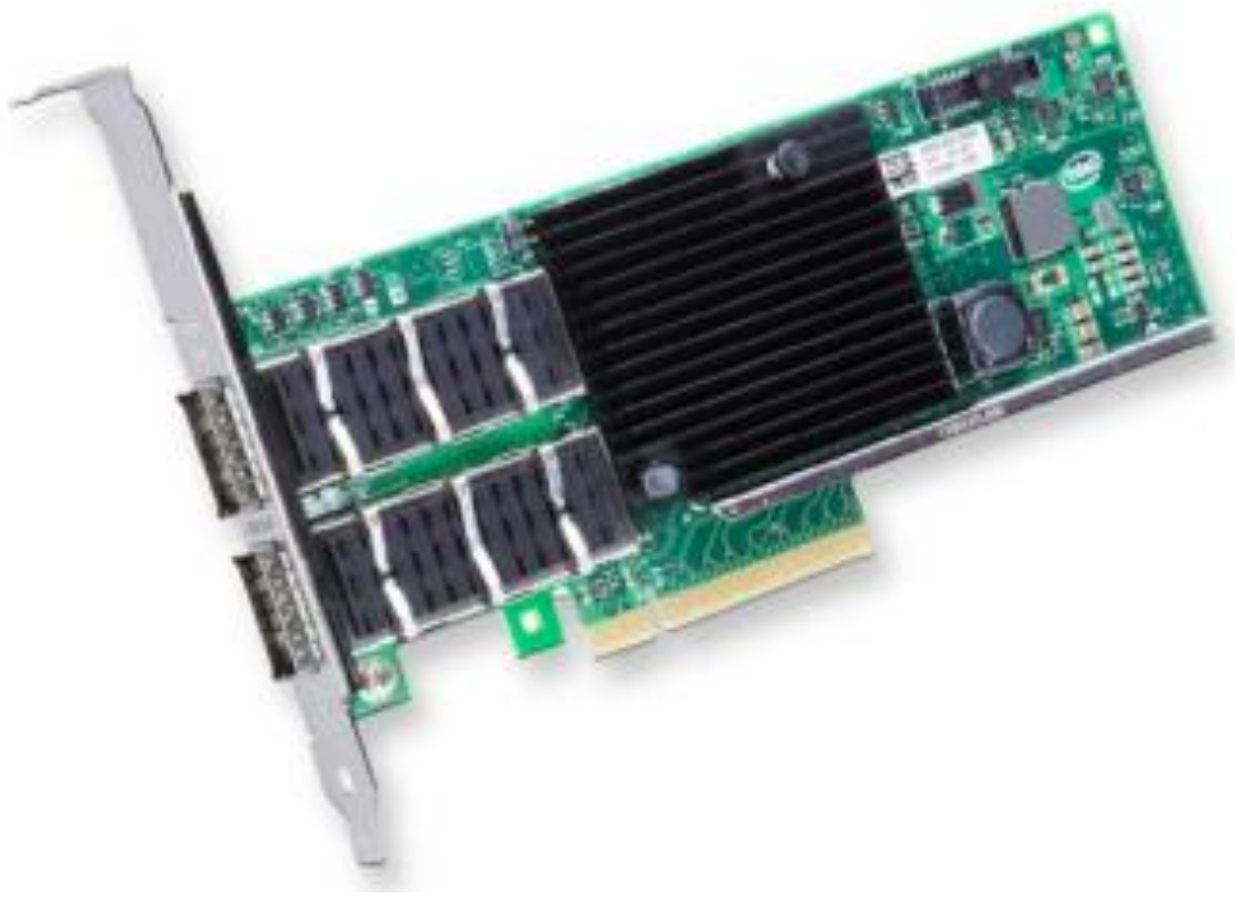
Sujay Jayakar (dsj36)
11/17/2016

Kernel Bypass

- Background
  - Why networking?
  - Status quo: Linux

- Papers
  - Arrakis: The Operating System is the Control Plane. Simon Peter, Jialin Li, Irene Zhang, Dan R. K. Ports, Doug Woos, Arvind Krishnamurthy, and Thomas Anderson, Timothy Roscoe. OSDI 2014.
  - IX: A Protected Dataplane Operating System for High Throughput and Low Latency. Adam Belay, George Prekas, Ana Klimovic, Samuel Grossman, and Christos Kozyrakis, Edouard Bugnion. OSDI 2014.

Why networking?



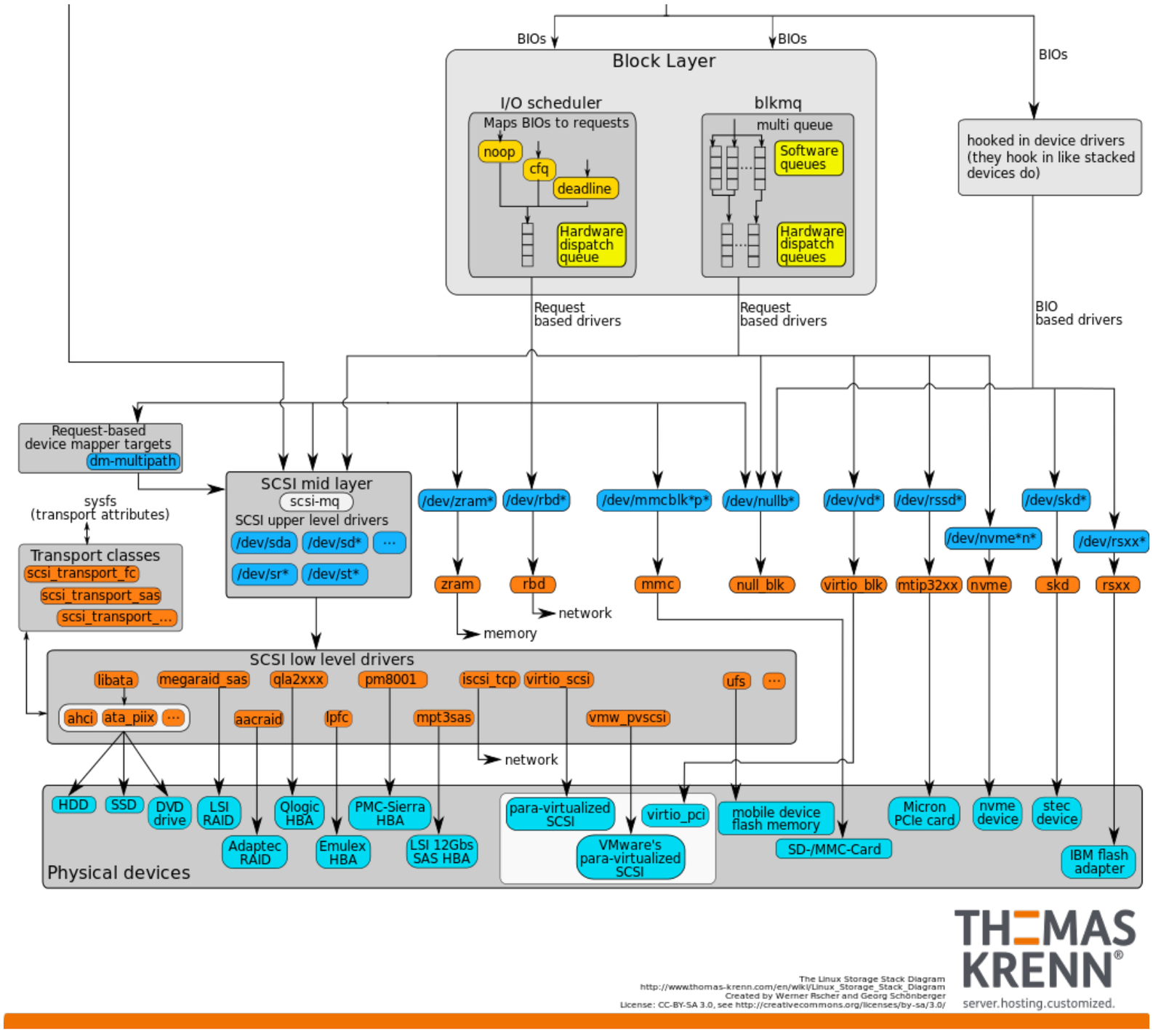
Bigger pipes: Intel XL710 chipset (40 Gbps)

http://www.intel.com/content/www/us/en/ethernet-products/converged-network-adapters/ethernet-xl710.html

Why networking?

Faster storage: Intel P3700 (20 Gbps, 100 µs reads)

Why networking?

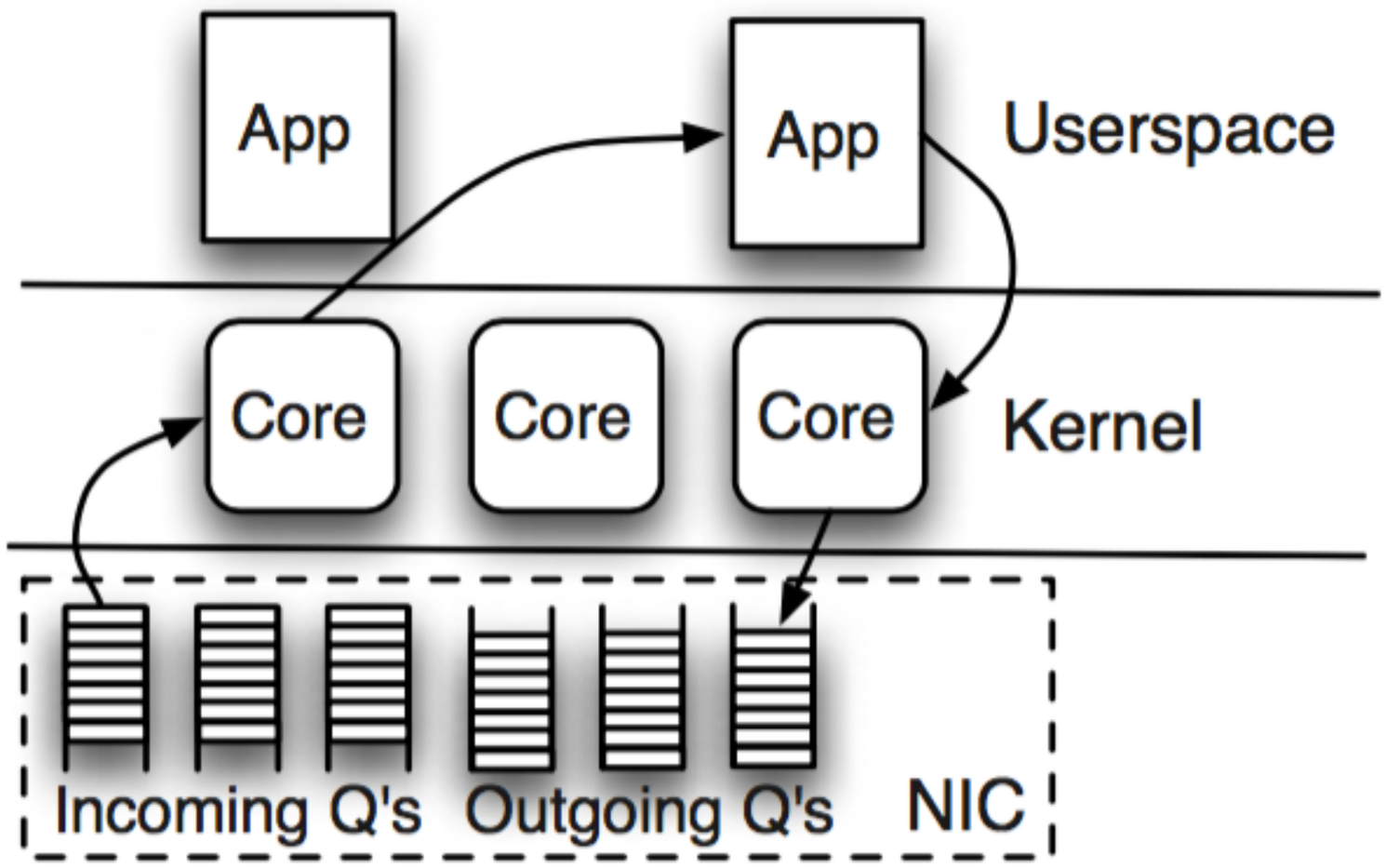


THOMAS KRENN®
server.hosting.customized.

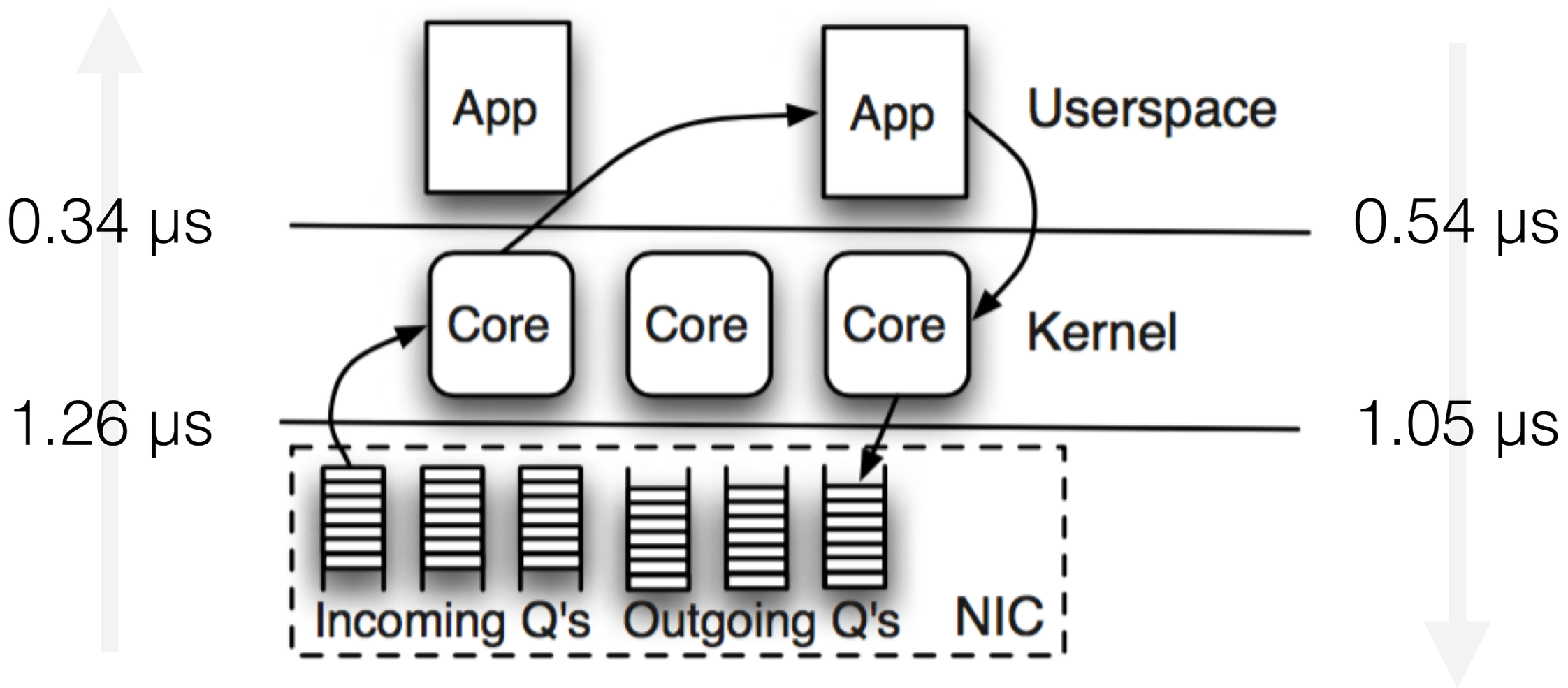
Why networking?

Software requirements

Status Quo: Linux



Status Quo: Linux



= 3.36 µs (+2.23µs to 6.19 µs if off-core)

Status Quo: Linux

- Ethernet frame: 84 bytes to 1538 bytes (1500 MTU)
- 10 Gb/s = 1.25 GB/s = 800K to 15M packets/sec
- Service time is only 67 ns to 1.25 µs per packet!

Status Quo: Linux

- Ethernet frame: 84 bytes to 1538 bytes (1500 MTU)
- 40 Gb/s = 5 GB/s = 3.25M to 60M packets/sec
- Service time is only 17 ns to 307 ns per packet!

Status Quo: Linux

- Ethernet frame: 84 bytes to 1538 bytes (1500 MTU)
- 100 Gb/s = 12.5 GB/s = 8M to 150M packets/sec
- Service time is only 6.7 ns to 125 ns per packet!

Status Quo: Linux

- Network stack: demultiplexing, checks, scheduling synchronization needed for multi-core
- POSIX limitations
  - Copy required for **`send(2)`** and **`recv(2)`**
  - File descriptor migration among processes
- Context switch overhead

Arrakis: The Operating System is the Control Plane.

Simon Peter, Jialin Li, Irene Zhang, Dan R. K. Ports, Doug Woos, Arvind Krishnamurthy, Thomas Anderson, and Timothy Roscoe. OSDI 2014.

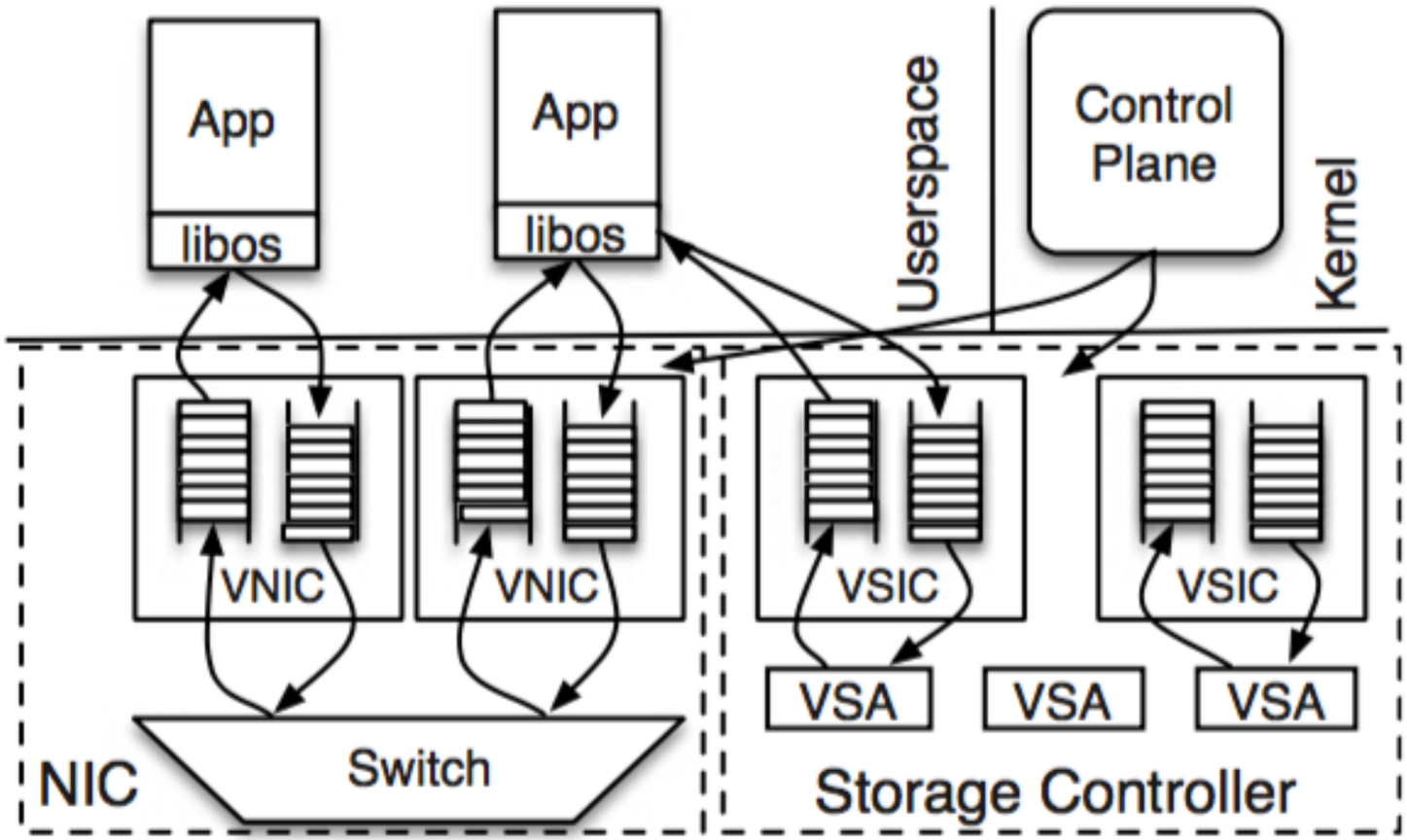
Network virtualization

- VNICs managed by “control plane” in kernel
- NIC responsible for bandwidth allocation & demultiplexing
  - Intel 82599 chipset
    - Only supports filtering by MAC address, want arbitrary header fields
    - Weighted round-robin bandwidth allocation
    - At most 64 VNICs

Storage virtualization

- VSIC: SCSI/ATA queue with rate limiter
- Virtual Storage Area (VSA): Persistent disk segment
- Each VSIC has many VSAs, and each VSA can be mapped into many VSICs (for interprocess sharing)
- Don’t have a device like this, emulated in software with a dedicated core

Arrakis


App
libos
App
libos
Userspace
Control Plane
Kernel
VNIC
VNIC
VSIC
VSIC
VSA
VSA
VSA
NIC
Switch
Storage Controller

Arrakis: Control Plane

- VIC management: queue pair creation & deletion
- Doorbells: IPC endpoint for VIC notifications
- Hardware control
  - Demultiplexing filters (just layer 2 for now)
  - Rate specifiers for rate limiting

Arrakis: Doorbells

- SR-IOV virtualizes the MSI-x interrupt registers
- libOS driver handles interrupts in user space if on-core, otherwise goes through control plane
- API is just a regular POSIX file descriptor

Arrakis: User APIs

Extaris	POSIX sockets	Arrakis/N Zero-copy rings
Ethernet	**`send_packet(queue, array)`** **`receive_packet(queue)`** → **`packet`** **`packet_done(packet)`** doorbells on completion	

IX: A Protected Dataplane Operating System for High Throughput and Low Latency. Adam Belay, George Prekas, Ana Klimovic, Samuel Grossman, Christos Kozyrakis, and Edouard Bugnion. OSDI 2014.

IX: More protection

- Network stack in user-space unacceptable
  - Firewall, ACL management in data plane
  - Sending raw packets requires root
  - Does zero-copy send require memory protection?
- Approach: use CPU virtualization to get three-way protection between kernel, networking, and user

IX: Protection can be cheap

- FlexSC (OSDI ’10): Why are syscalls expensive?
- Direct effects: User-mode pipeline flush, mode switch, stashing registers. Only ~150 cycles!

IX: Protection can be cheap

- FlexSC (OSDI ’10): Why are syscalls expensive?
- Direct effects: User-mode pipeline flush, mode switch, stashing registers. Only ~150 cycles!
- Indirect effects: TLB flush (if no tagged TLB), cache pollution with kernel instructions/data

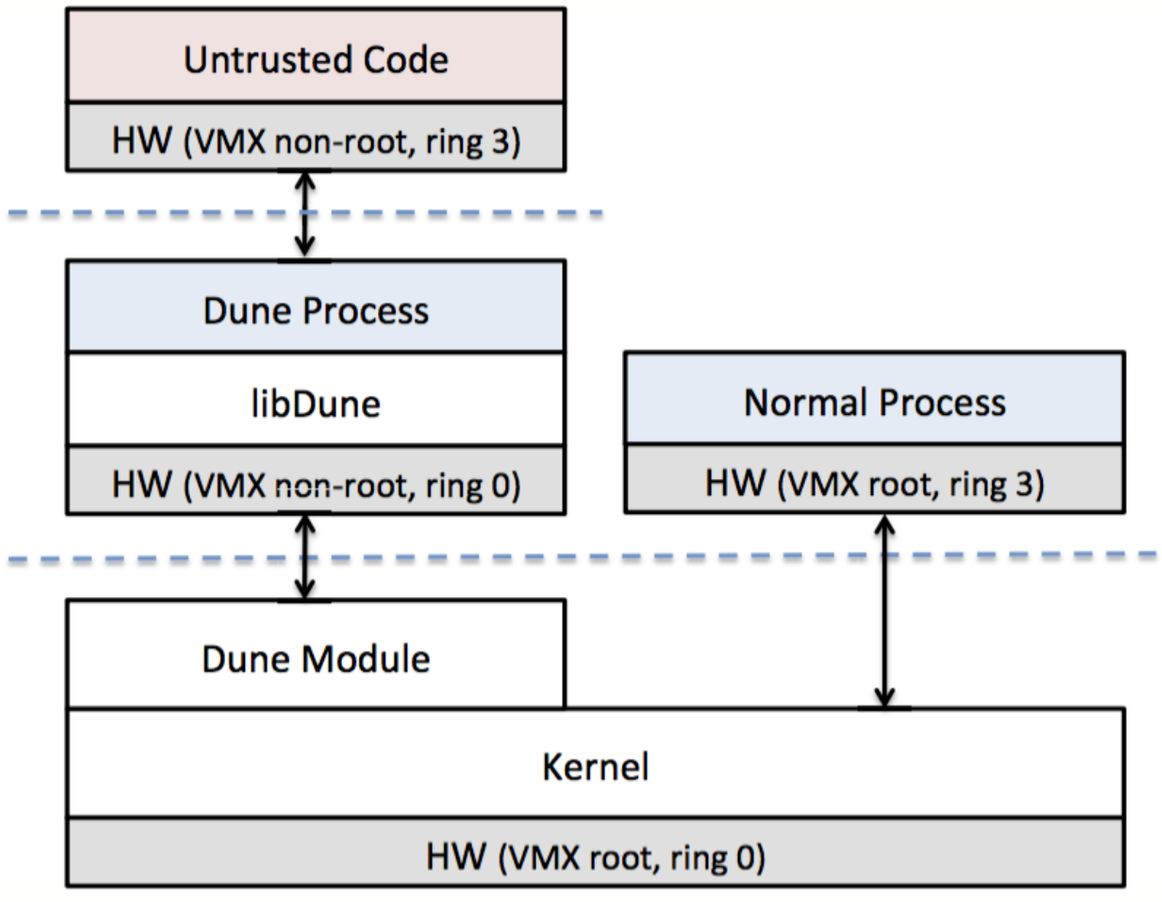
	IPC	i-cache	d-cache	L2	L3	d-TLB
pwrite(2)	0.18	50	373	985	3160	44

IX: Protection can be cheap

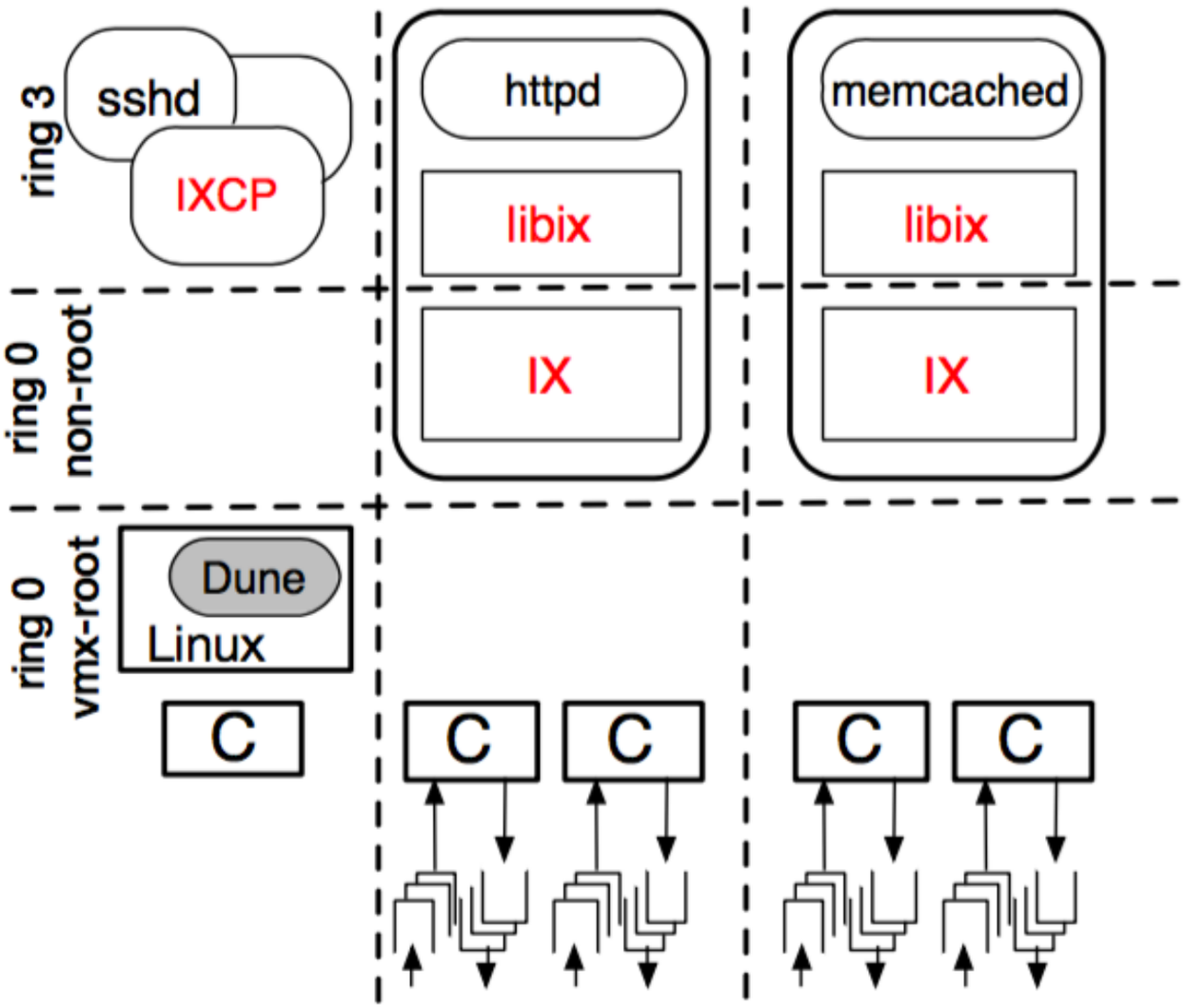
- Dune (OSDI ’12): Run Linux processes as VMX non-root ring0 with syscalls replaced with hypercalls

- Untrusted code can run in ring3, can cheaply mode switch back into ring0

- Processes can then manipulate page tables, interrupts, hardware, privilege, …



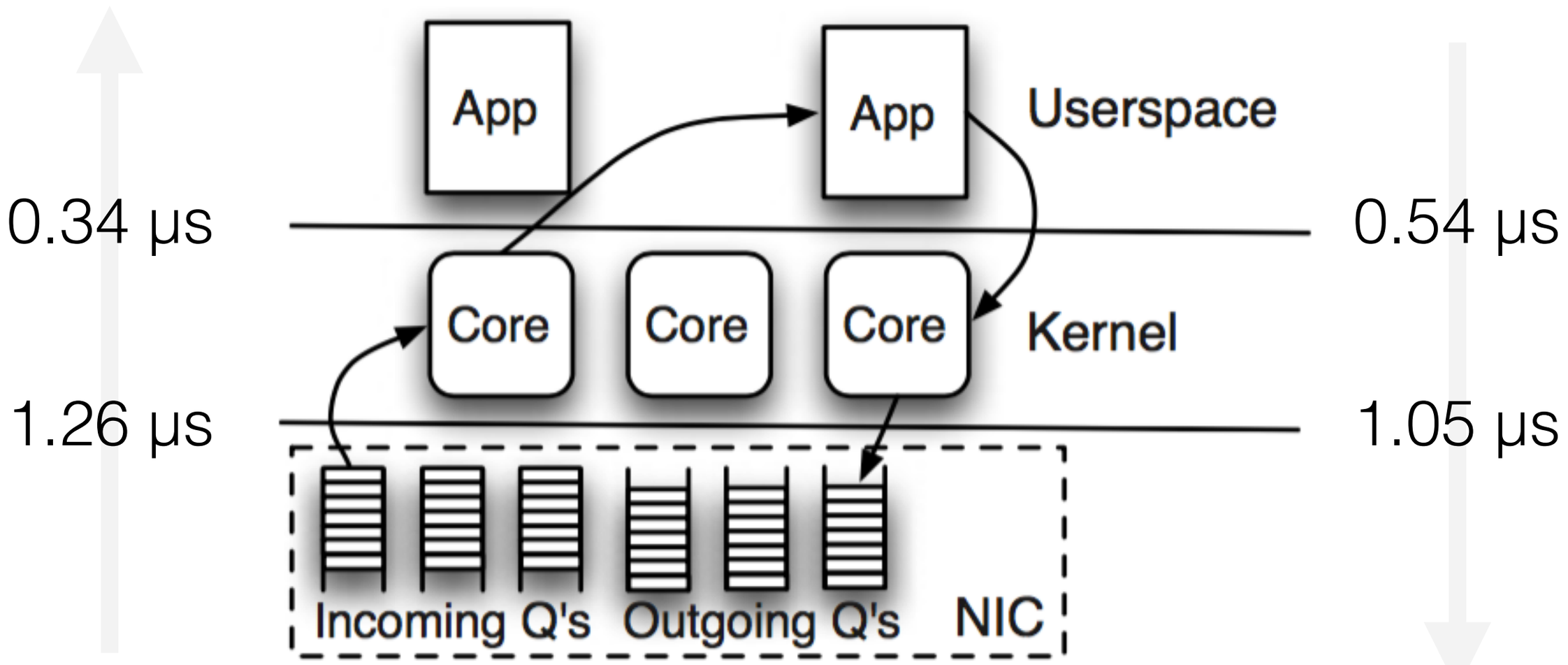
IX: More protection



Why not both?

- IX's use of VMX non-root ring0 is clever.
- Its embedding in Linux (with Dune) is nice.
- But if they used SR-IOV, they wouldn't have needed the Toeplitz hash hack and monopolizing a queue.
- And then they would have gotten (fast!) virtualized interrupts to not have to poll.

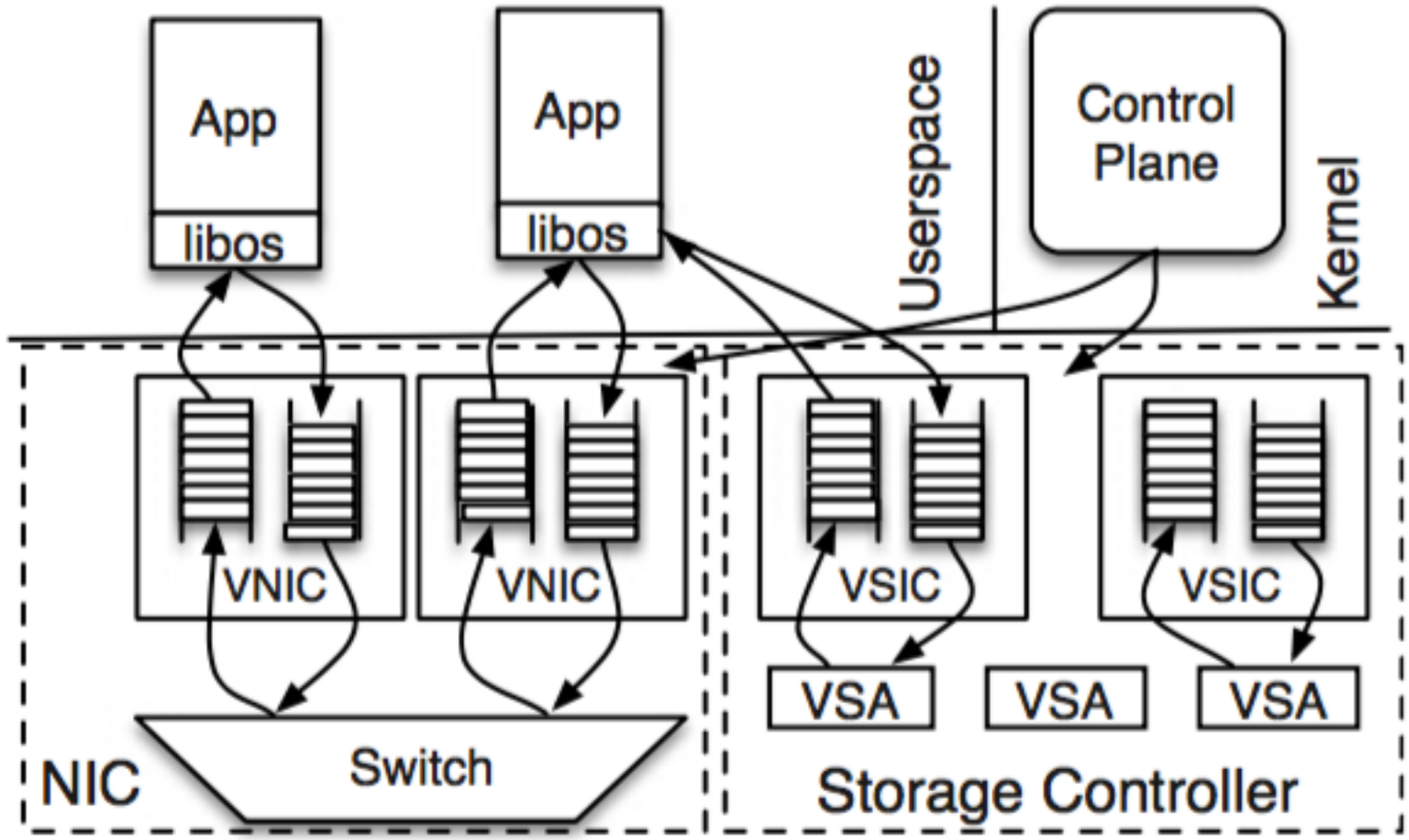
Arrakis: Evaluation



= 3.36 µs (+2.23µs to 6.19 µs if off-core)

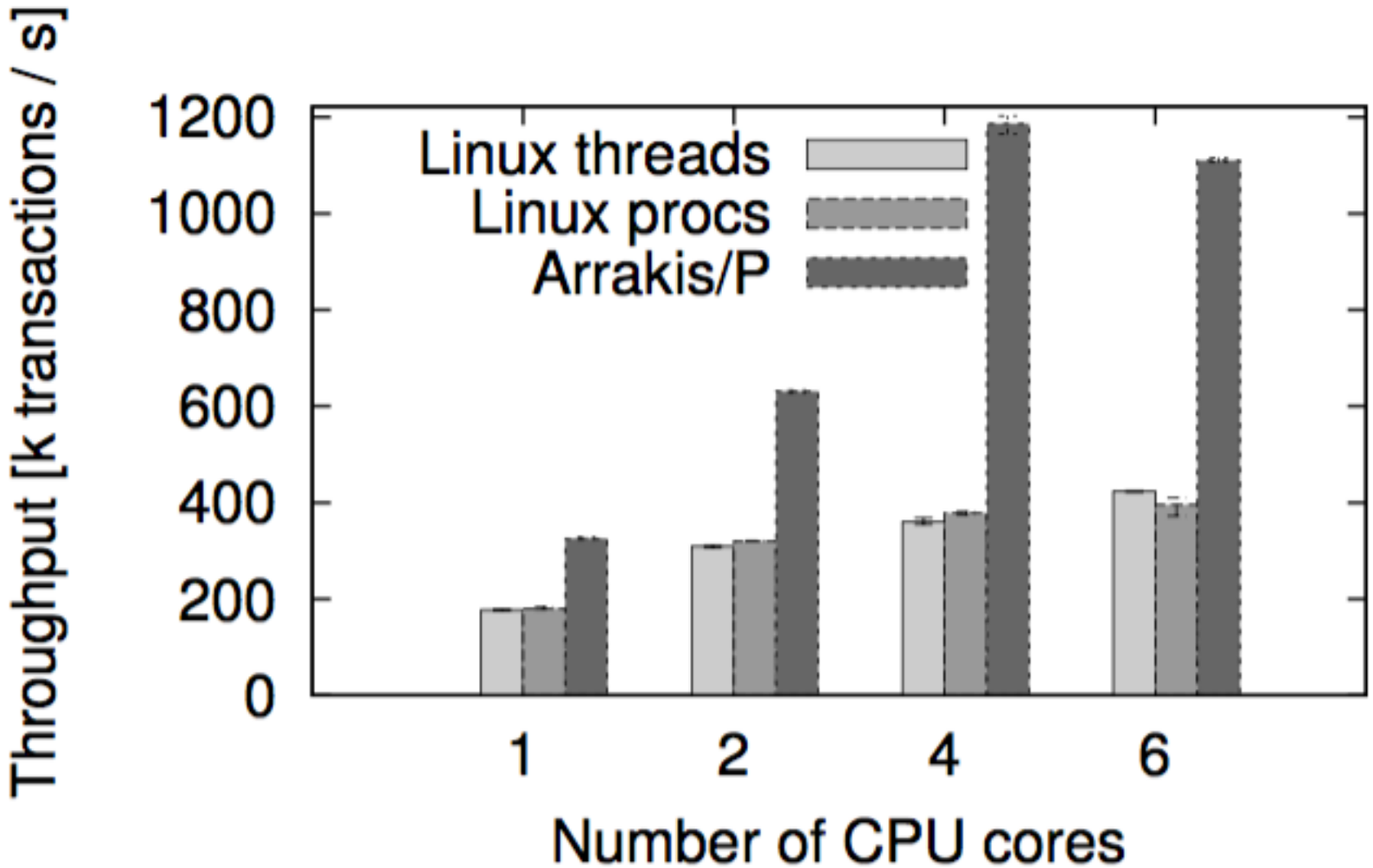
Arrakis: Evaluation

0.21 µs 0.17 µs

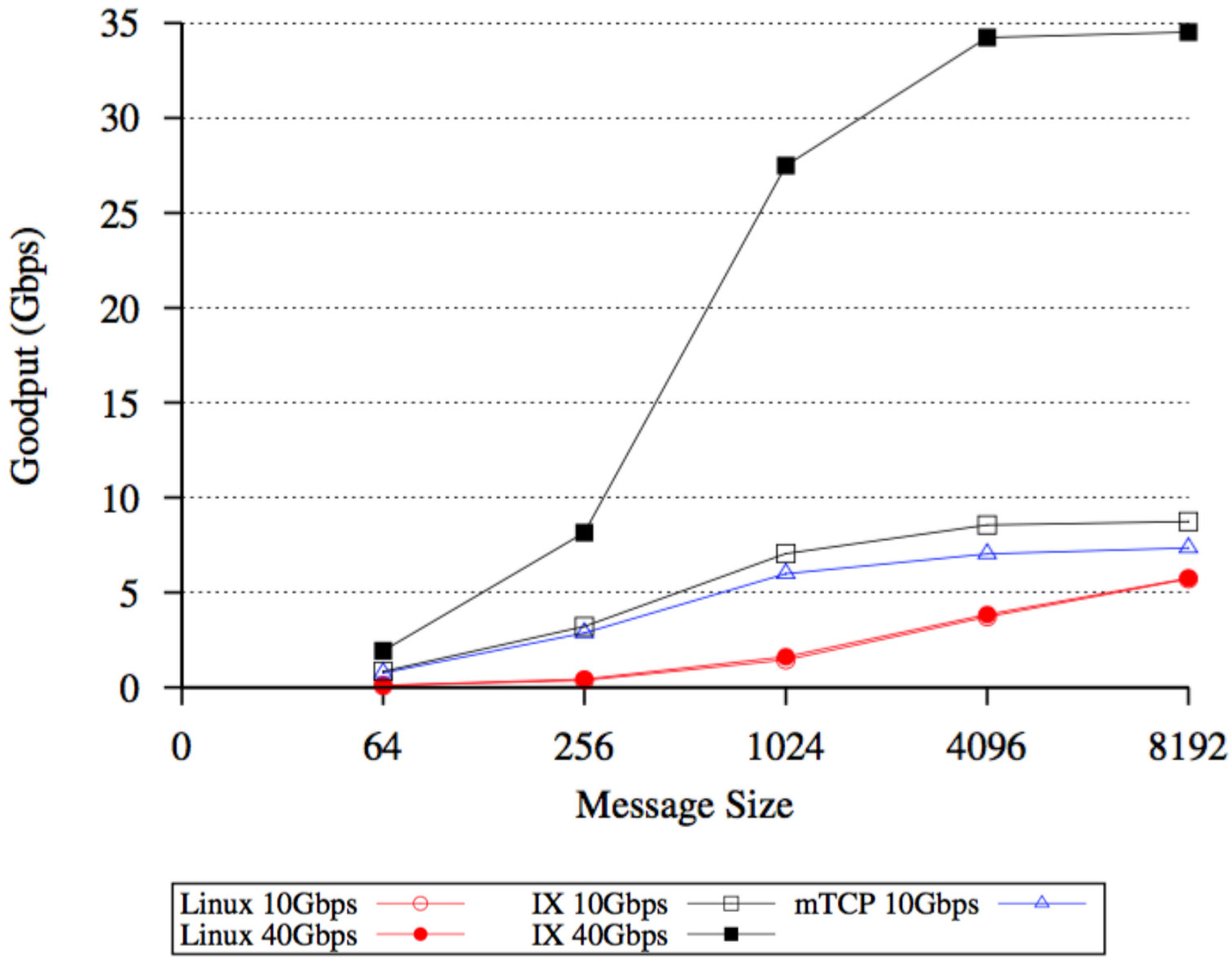


= 0.38 µs

Arrakis: memcached



IX: Evaluation



IX: Give up on POSIX

- No to POSIX: Non-blocking API with batching
- Packets processed to completion
- “Elastic” vs. “background” threads
- No internal buffering
- Slow consumers lead to delayed ACKs

IX: Packet processing