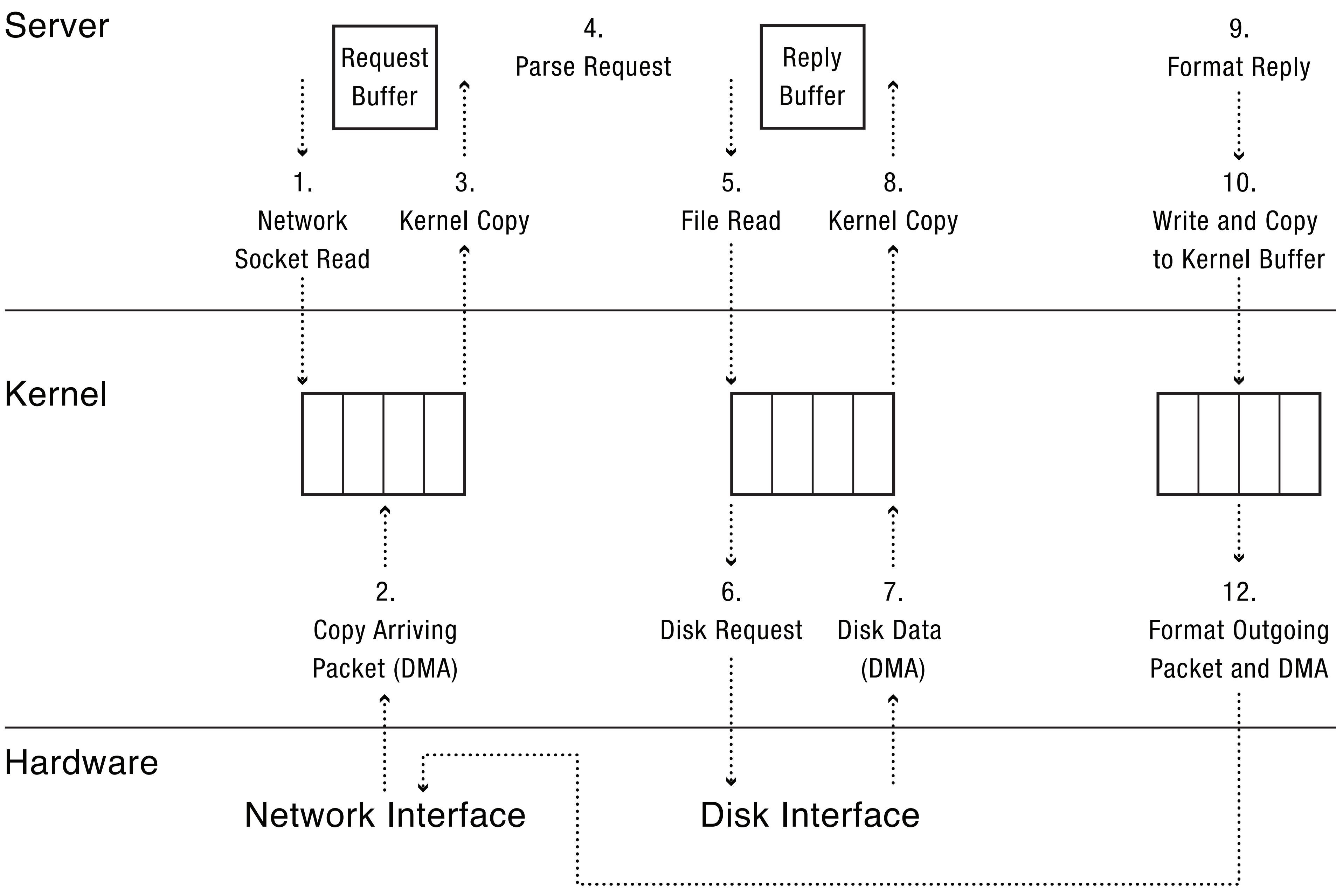
Main Points

- How virtual machines work
- Why network and disk I/O is slow
- What we can do about it

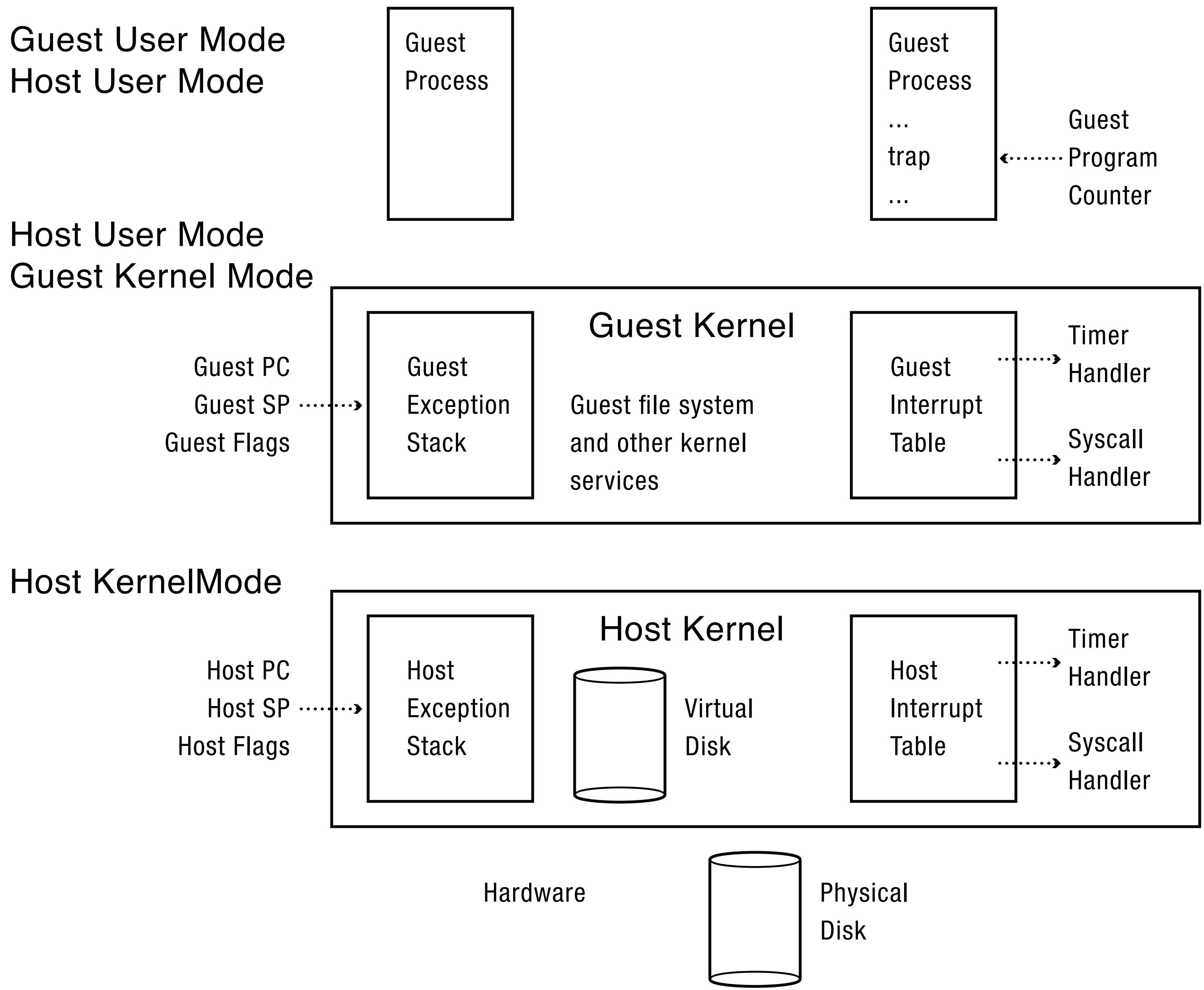

Server
Request Buffer
4. Parse Request
Reply Buffer
9. Format Reply
1. Network Socket Read
3. Kernel Copy
5. File Read
8. Kernel Copy
10. Write and Copy to Kernel Buffer
Kernel
2. Copy Arriving Packet (DMA)
6. Disk Request
7. Disk Data (DMA)
12. Format Outgoing Packet and DMA
Hardware
Network Interface
Disk Interface

Virtual Machines

- Most data centers insert an extra “virtual machine” layer
- Modify host operating system so that it can run a “guest” operating system as a (user-level) application
- Guest operating system thinks it is running on raw hardware
  - Runs at user-level
- Application (guest user-level) thinks it is running on guest OS running on raw hardware
  - Has guest OS to itself

Virtual Machine Pros/Cons

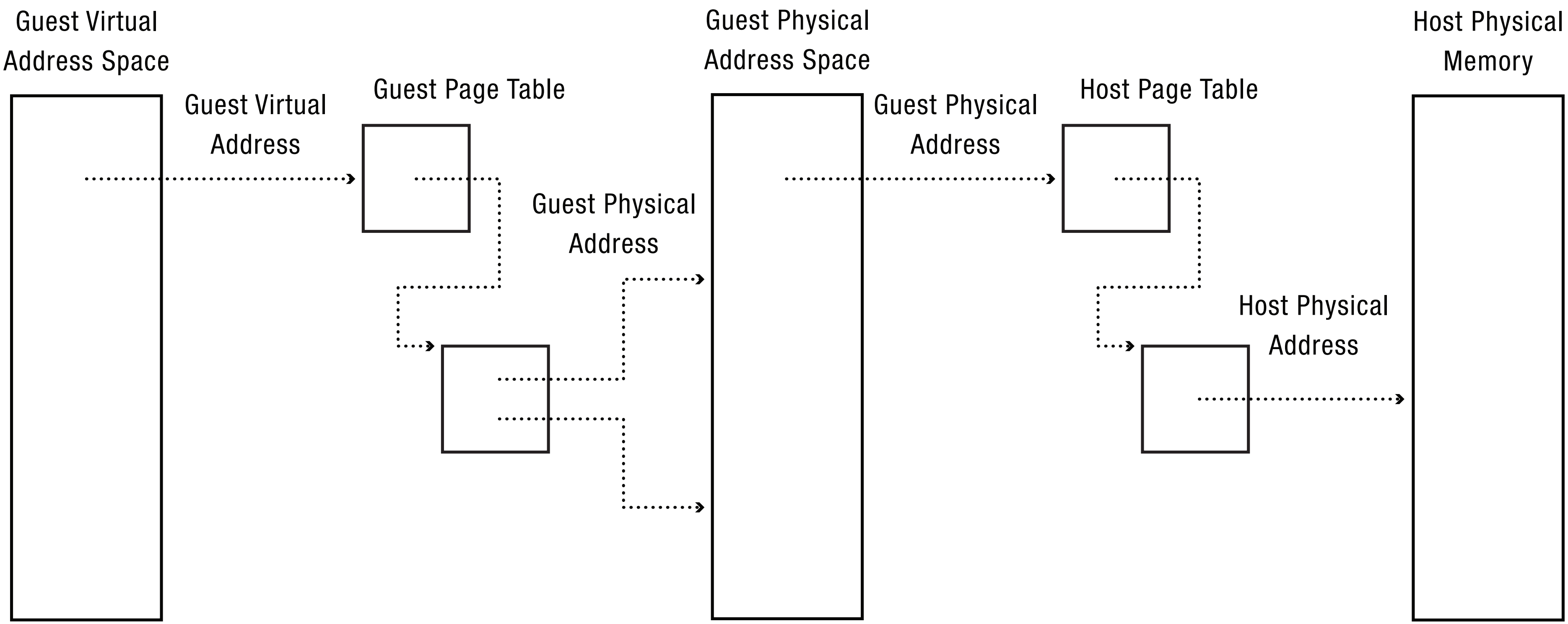
- Separation of data center management from application's choice of operating system
  - Multiple web servers per physical machine
  - Each with a different OS
  - Easy to migrate virtual machine
  - Easy to limit access by guest OS to other nodes
- Cost of redirection
  - For virtual memory mapping and I/O
  - Emerging hardware support to reduce cost


Guest User Mode
Host User Mode
Guest
Process
Guest
Process
...
trap
...
Guest
Program
Counter
Host User Mode
Guest Kernel Mode
Guest Kernel
Guest PC
Guest SP
Guest Flags
Guest
Exception
Stack
Guest file system
and other kernel
services
Guest
Interrupt
Table
Timer
Handler
Syscall
Handler
Host KernelMode
Host Kernel
Host PC
Host SP
Host Flags
Host
Exception
Stack
Virtual
Disk
Host
Interrupt
Table
Timer
Handler
Syscall
Handler
Hardware
Physical
Disk

Question

- How many crossings are needed to handle a web request on a server running on a guest OS running on a virtual machine?
  - Network I/O interrupt delivered to host kernel
  - Transfers control to guest OS to handle interrupt
  - Return from interrupt back to host kernel
  - Resumes application
  - System call trap to read from network, to host kernel
  - Transfers control to guest OS to handle system call
  - Return from system call back to host kernel
  - Resumes application

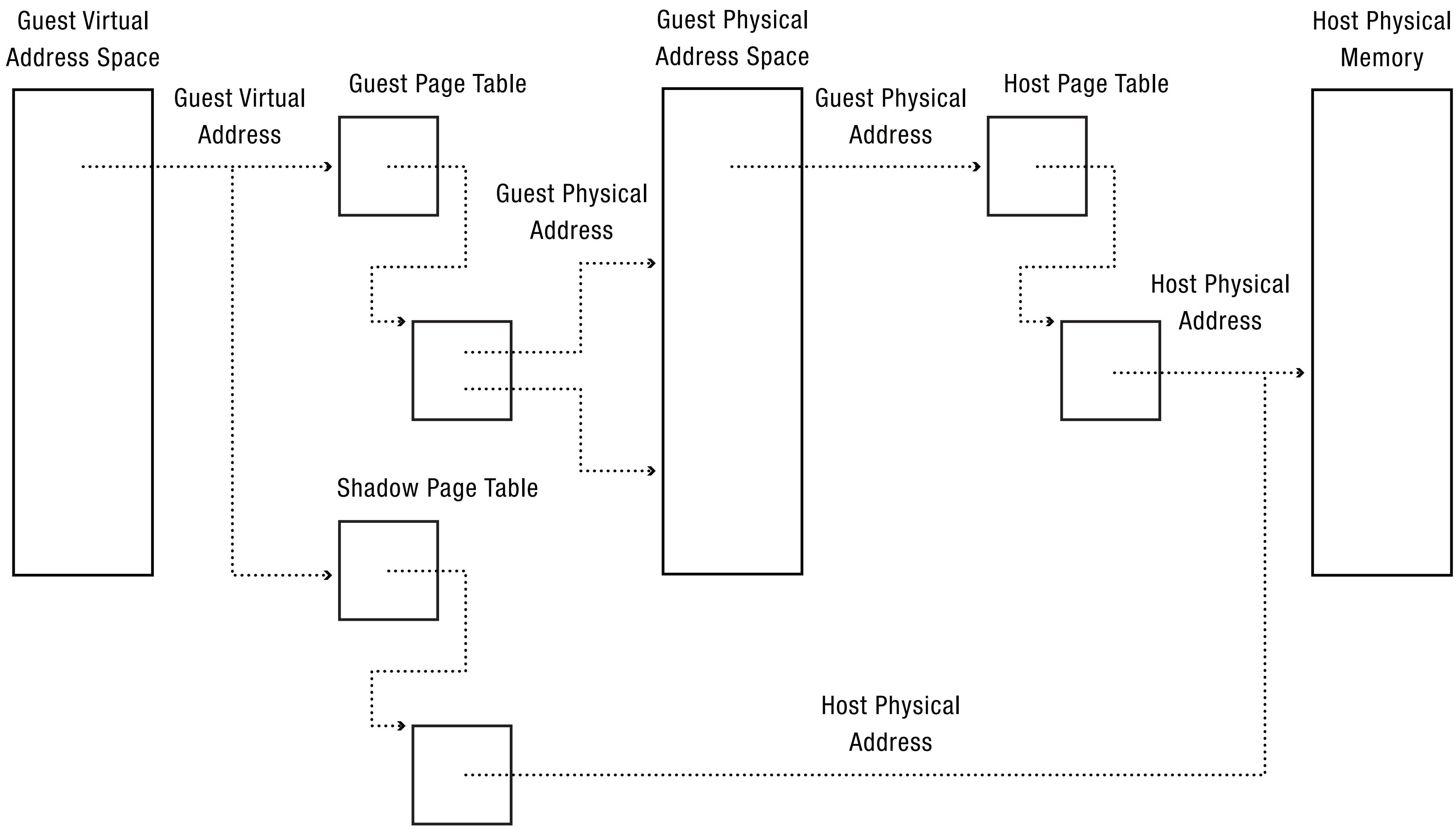
Virtual Machines and Virtual Memory



Segment Table		Page Table A		Page Table B	
0	Page Table A	0	0002	0	0001
1	Page Table B	1	0006	1	0004
x	(rest invalid)	2	0000	2	0003
		3	0005	x	(rest invalid)
		x	(rest invalid)		

Segment Table		Page Table K	
0	Page Table K	0	BEEF
x	(rest invalid)	1	F000
		2	CAFE
		3	3333
		4	(invalid)
		5	BA11
		6	DEAD
		7	5555
		x	(rest invalid)

Shadow Page Tables



Hardware Support for Virtual Machine Translation

- x86 recently added hardware support for running virtual machines at user level
- Operating system kernel initializes two sets of translation tables
  - One for the guest OS
  - One for the host OS
- Hardware translates address in two steps
  - First using guest OS tables, then host OS tables
  - TLB holds composition

Containers

- Provide applications the illusion of their own virtual machine
  - Own process ID table
  - Own network socket addresses
  - Own file descriptor table
- Running directly on Linux or other OS
  - By modifying system call handling
  - No system call redirection
  - No virtual machine redirection

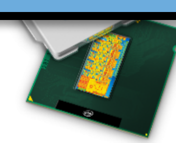
Arrakis: High I/O Performance OS

- Server I/O performance matters
  - Key-value stores, web & file servers, lock managers, …
- **Can OSe**
- Example

Today's I/O devices are fast

\+

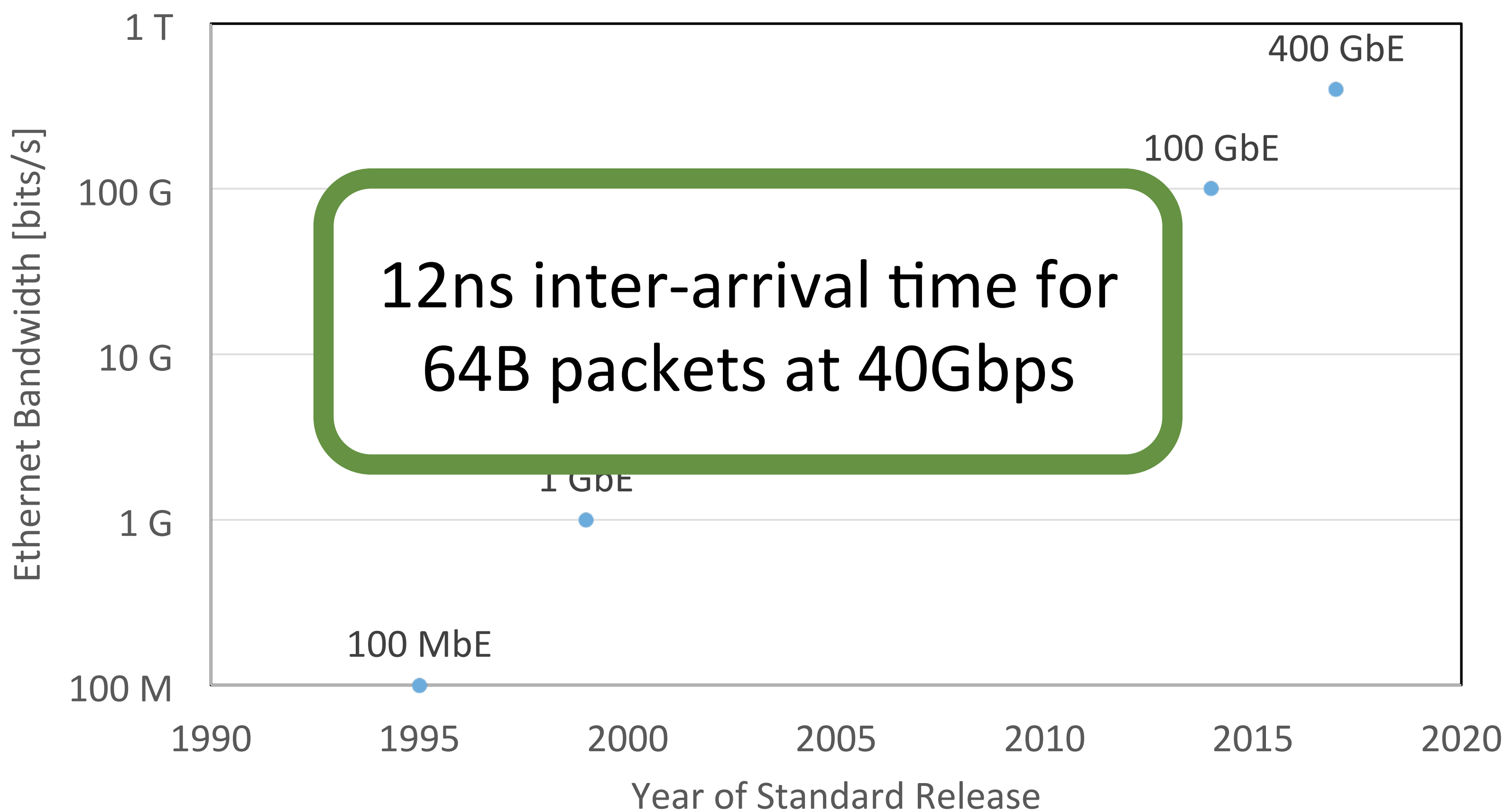
\+



= **$1,200**

Intel X520	Intel RS3 RAID	Sandy Bridge CPU
10G NIC	**1GB flash-backed cache**	**6 cores, 2.2 GHz**
50ns / 64B pkt	**25 us / 1KB write**	

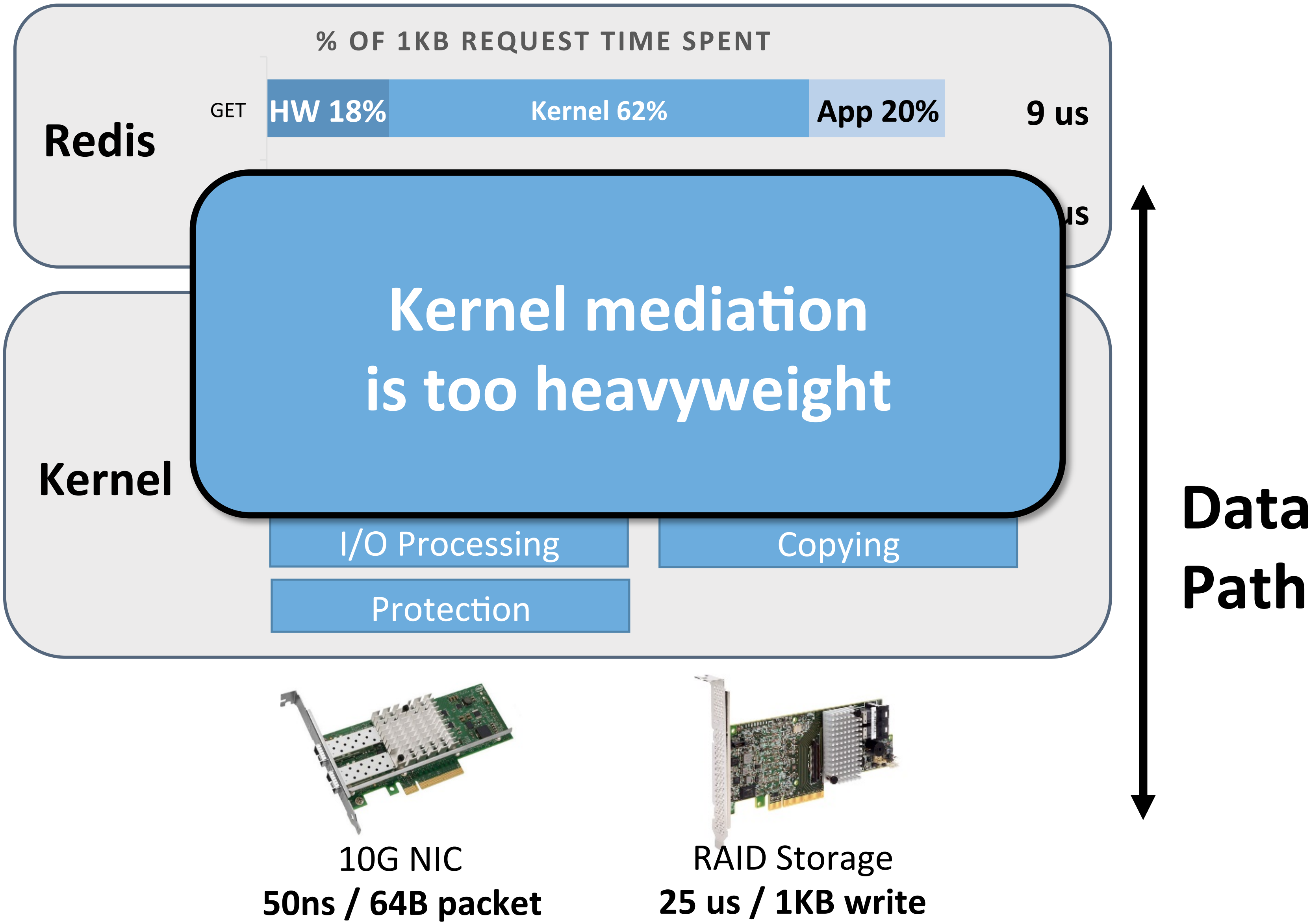
Networks: Fast and Growing Faster



12ns inter-arrival time for 64B packets at 40Gbps

Can’t we just use Linux?

Linux I/O Performance



Arrakis Goals

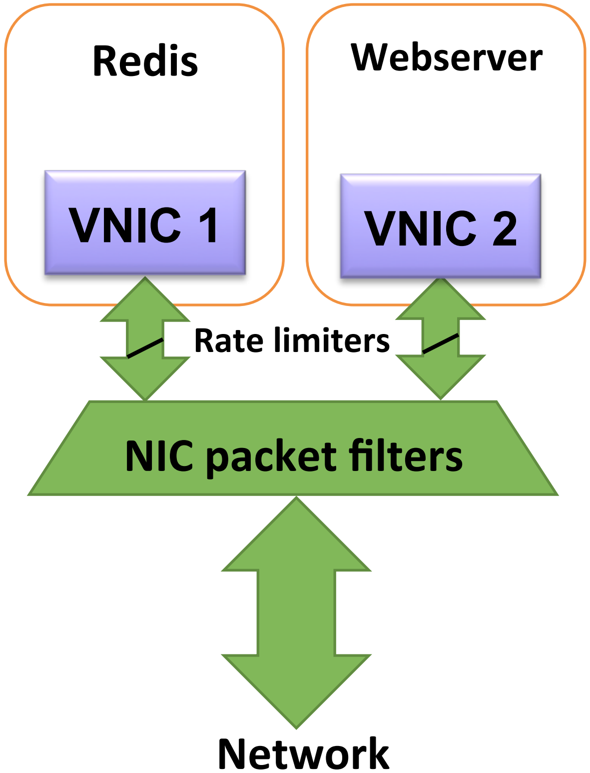
- **Skip kernel** & deliver I/O **directly** to applications
  - Reduce OS overhead
- **Keep** classical server OS kernel features
  - Process protection
  - Resource limits
  - I/O protocol flexibility
  - Global naming
- The hardware can help us…

Hardware I/O Virtualization

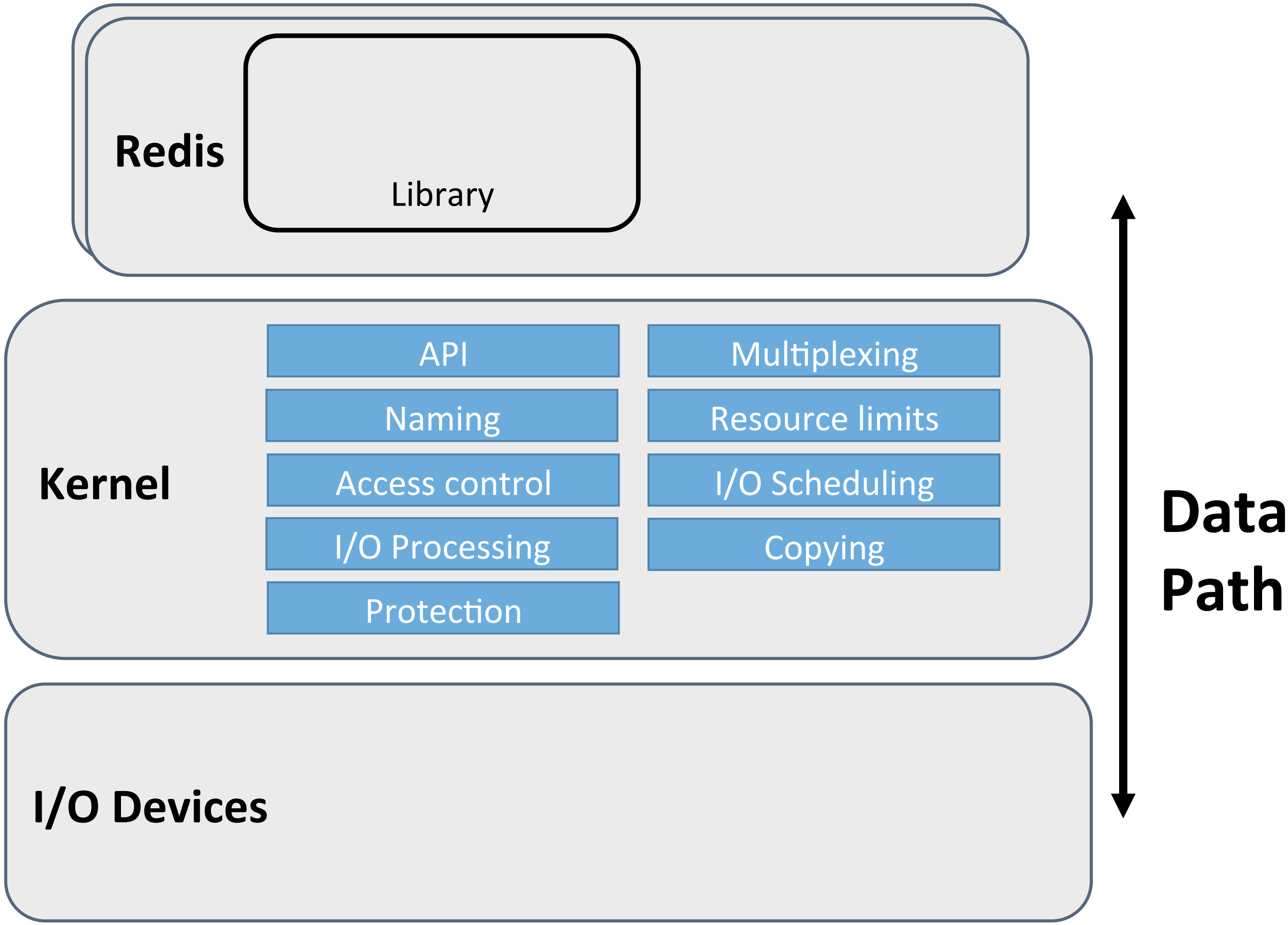
- **Standard** on network, emerging on storage

Provided in hardware:

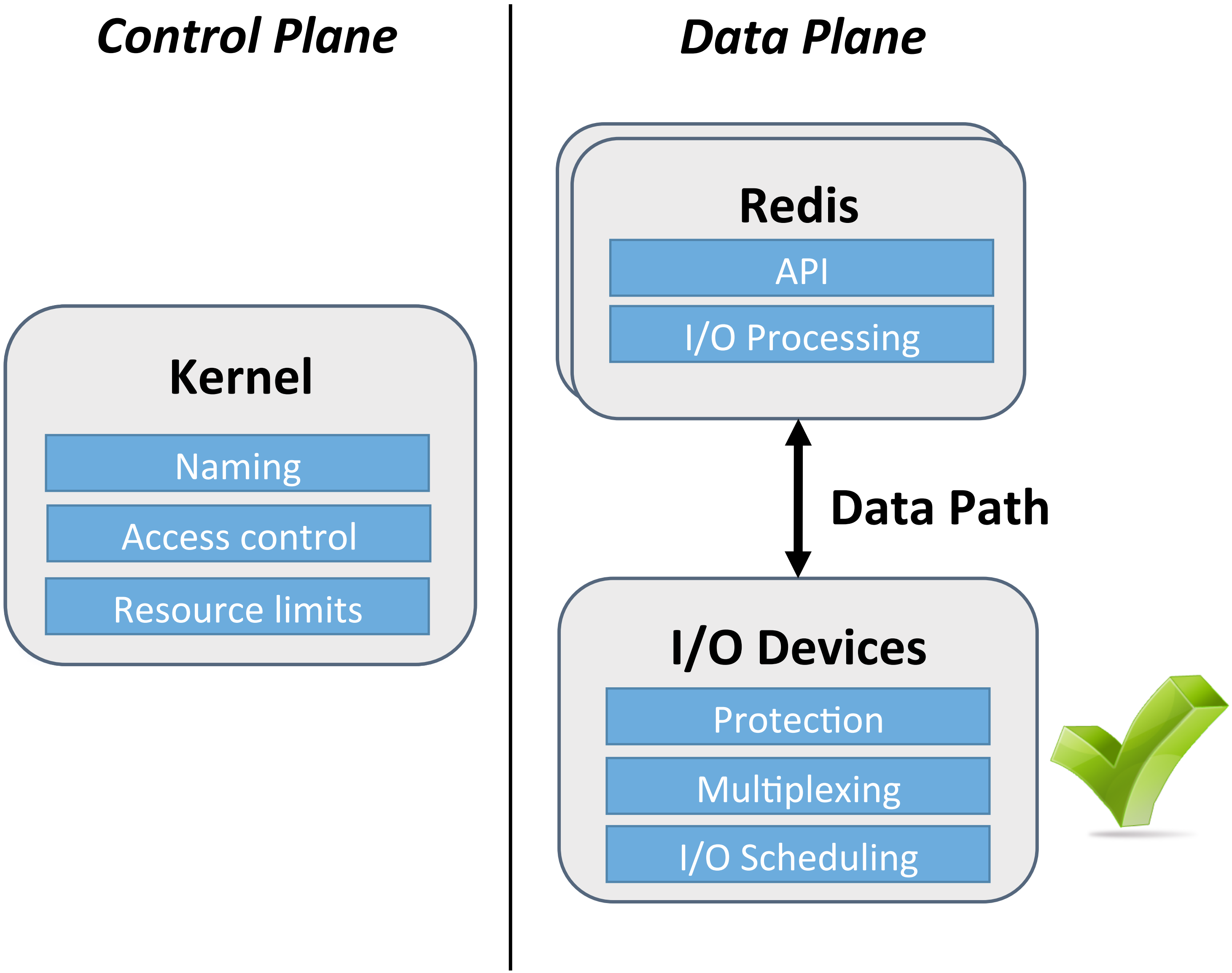
- Multiplexing
  - E.g., Virtual network cards (VNICs)
- Protection
  - Attach VNICs to application memory
  - **Packet filters**, **logical disks**:
    Allow only eligible I/O from apps
- I/O Scheduling
  - **Rate limiters**, **packet schedulers**



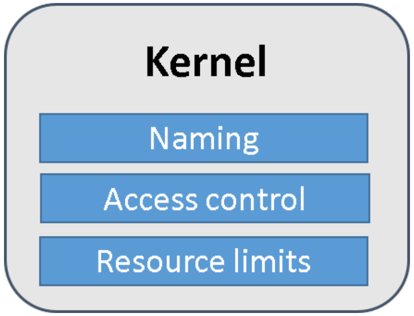
How to skip the kernel?



Arrakis I/O Architecture

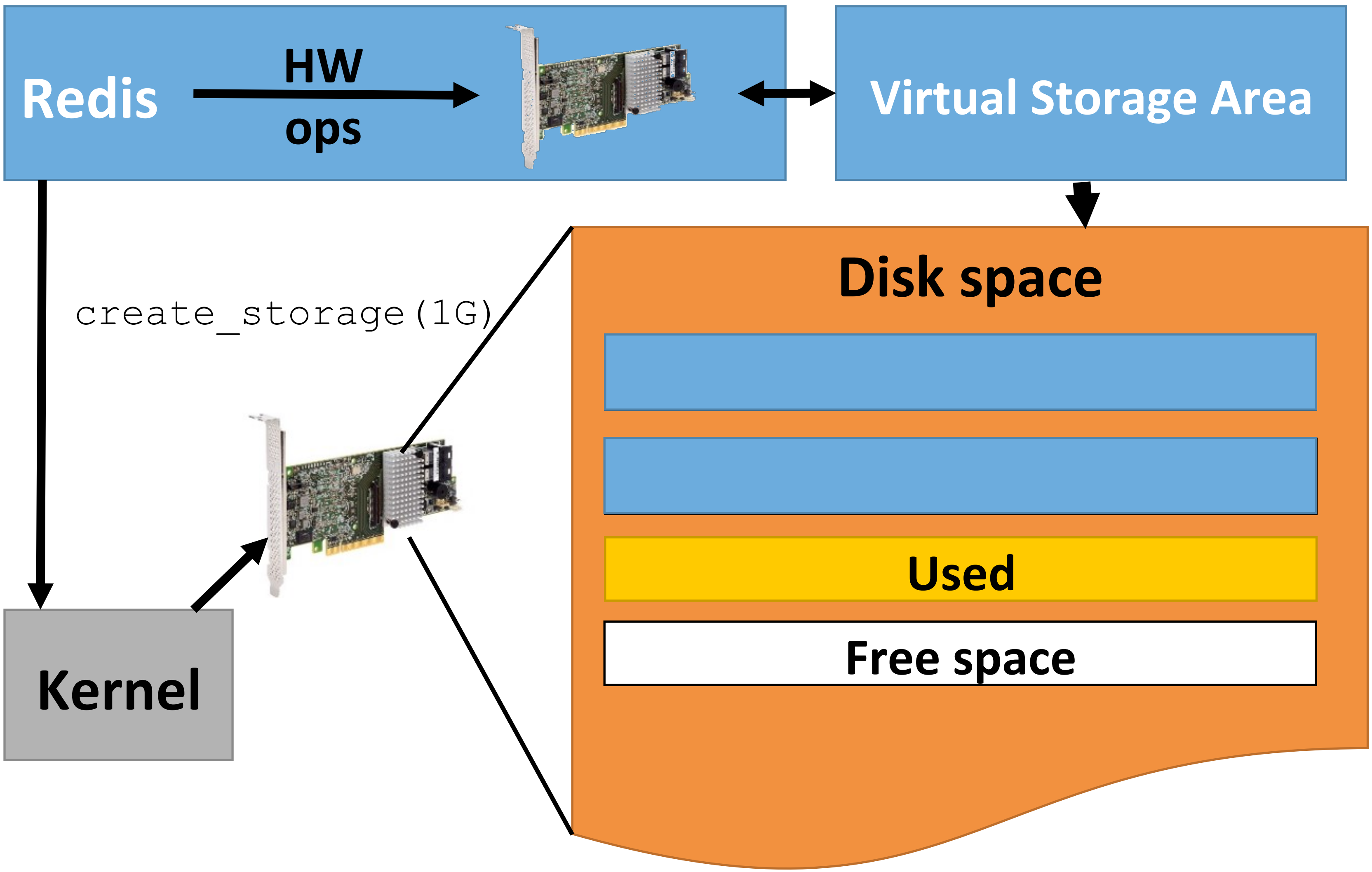


Arrakis Control Plane

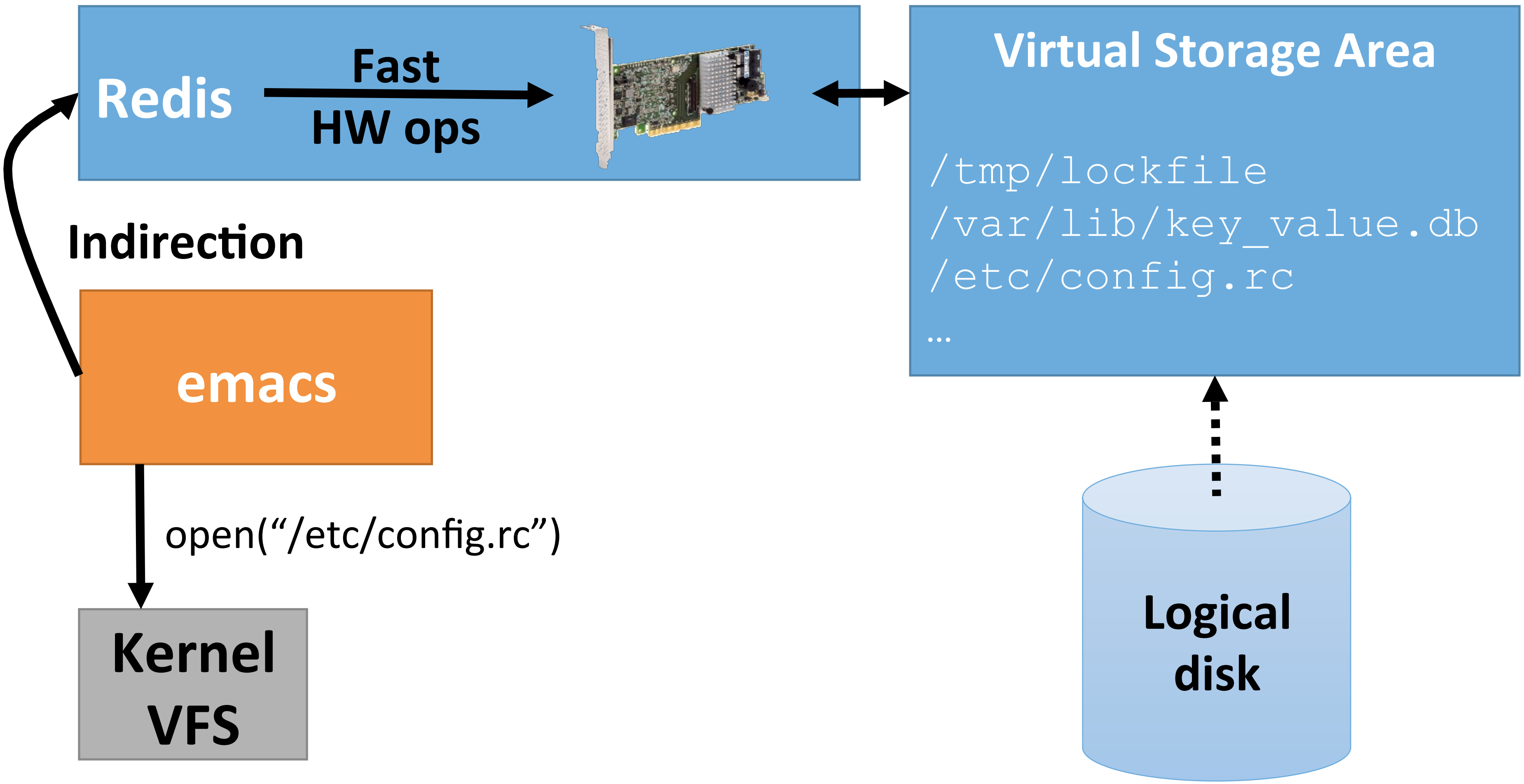


- Access control
  - Do **once** when configuring data plane
  - Enforced via NIC filters, logical disks
- Resource limits
  - Program hardware I/O schedulers
- Global naming
  - Virtual file system still **in kernel**
  - **Storage implementation in applications**

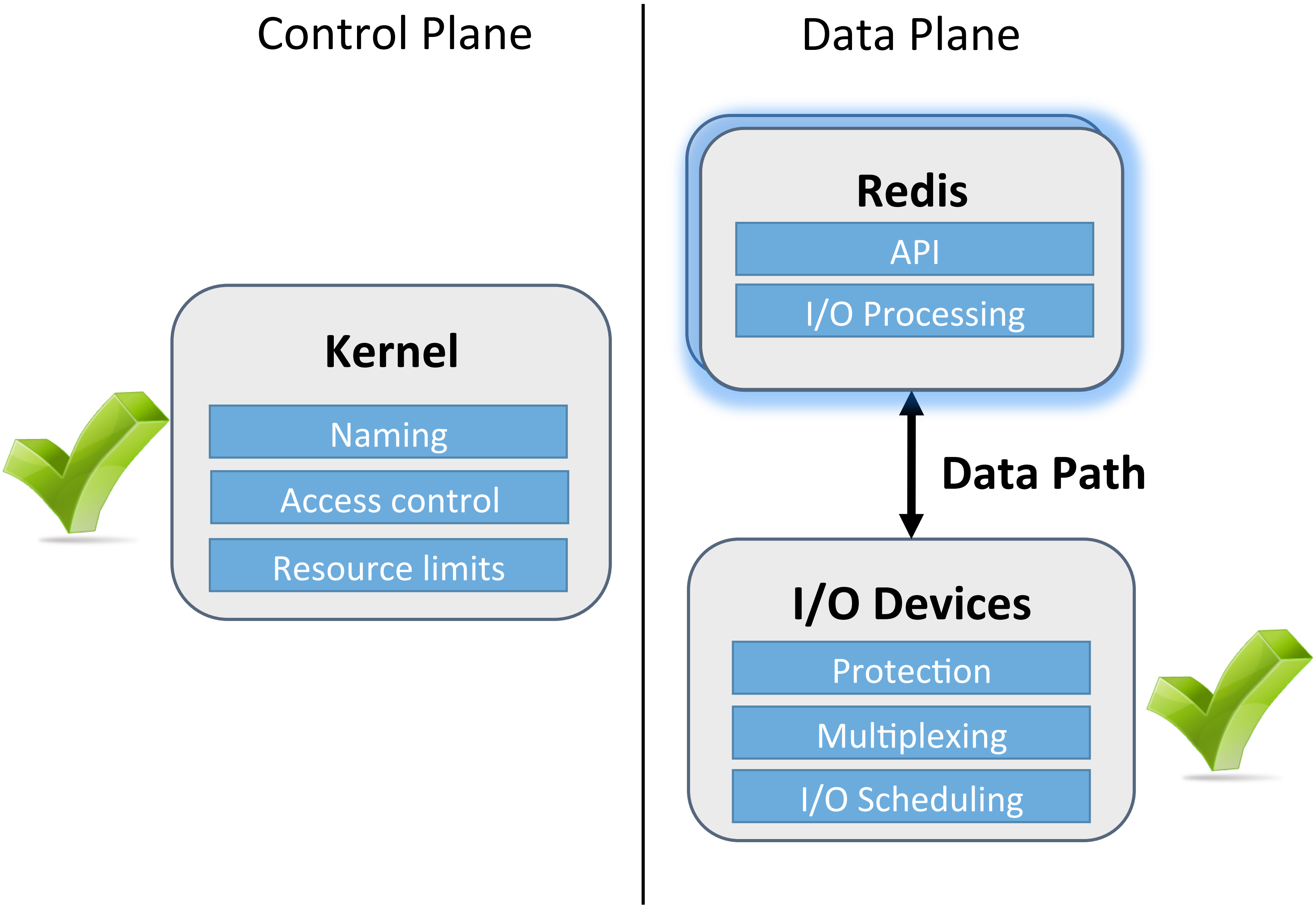
Storage Space Allocation



Global Naming

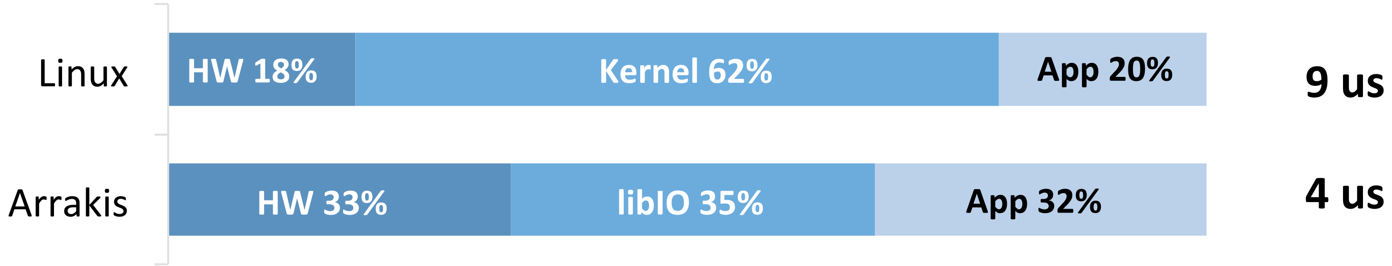


Arrakis I/O Architecture

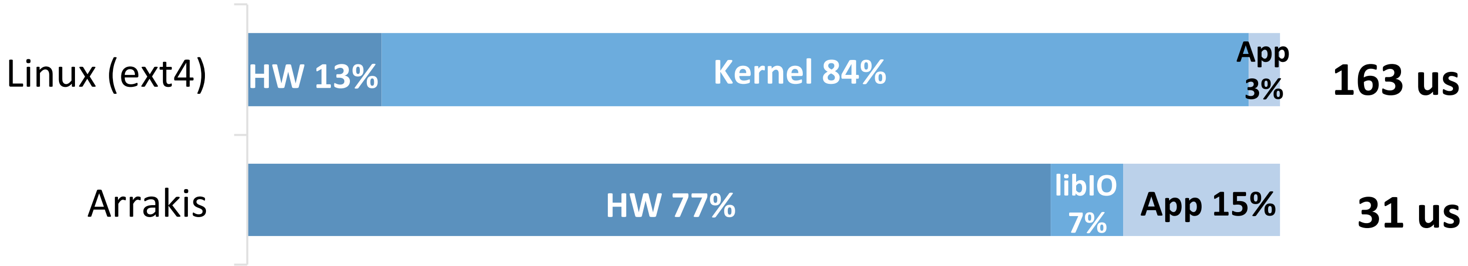


Redis Latency

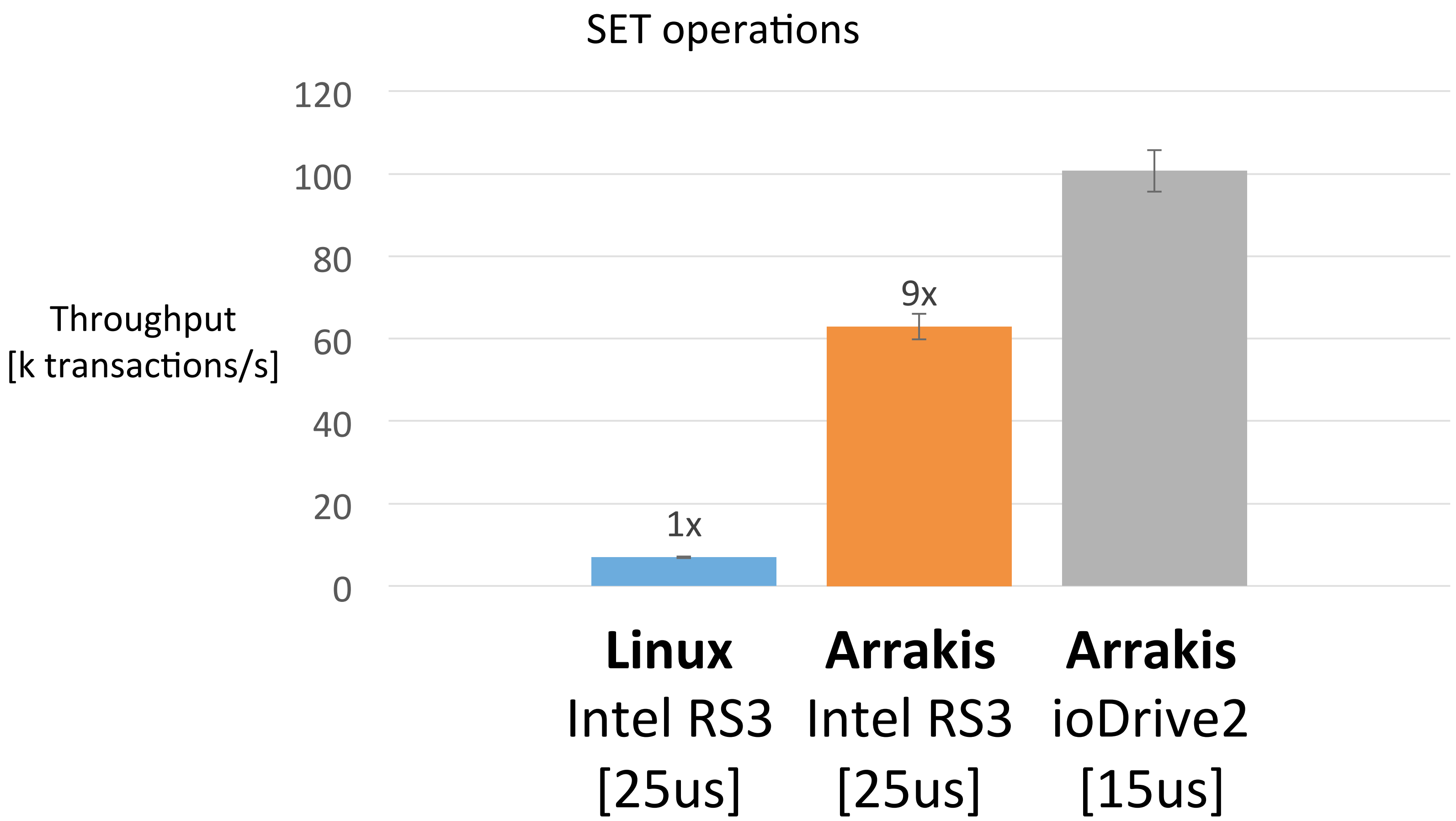
- Reduced (in-memory) GET latency by **65%**



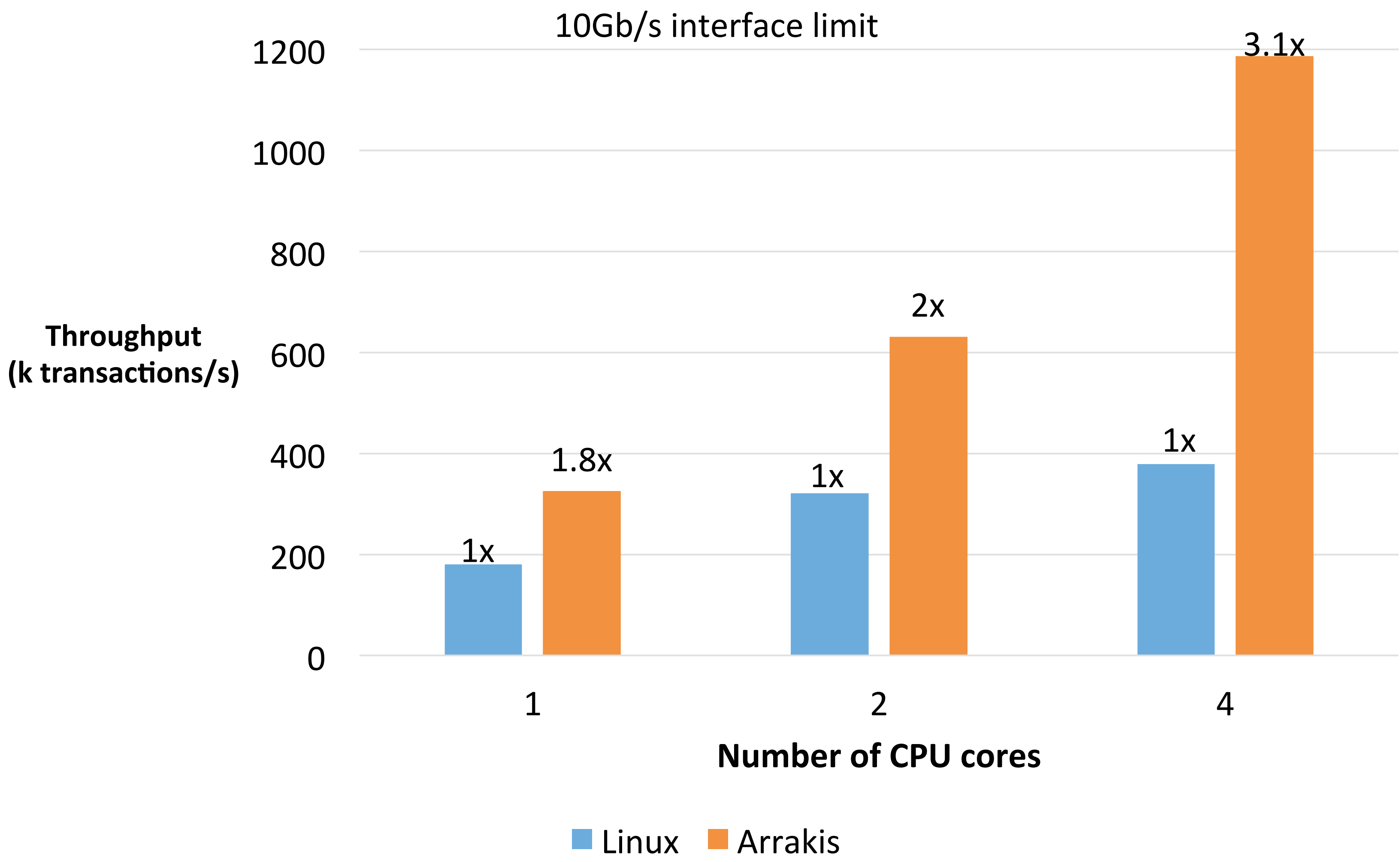
- Reduced (persistent) SET latency by **81%**



Redis Throughput

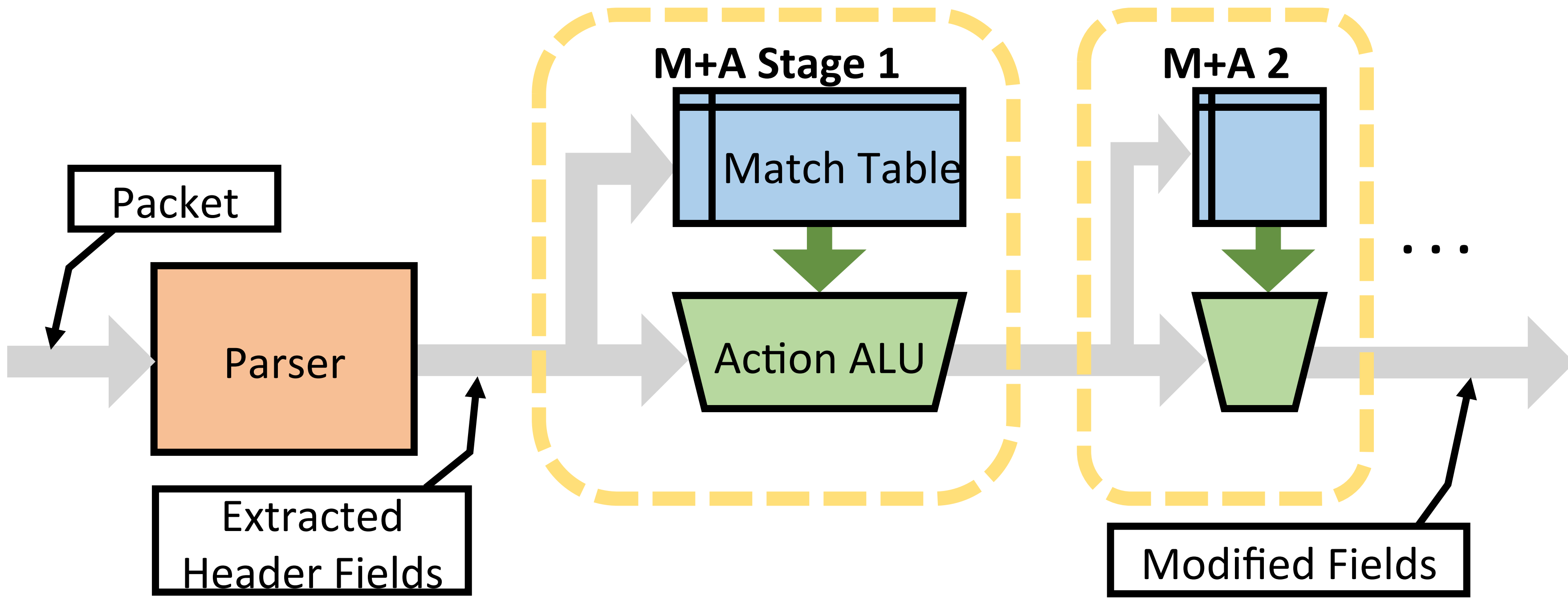


memcached Scalability



FlexNIC:
A Model for Integrated NIC/SW Processing

- Must be implementable at line rate with low cost
- Match+action pipeline:



Match+Action Programs: Actions

Match:
IF udp.port == kvs
Action:
core = HASH(kvs.key) % 2
DMA hash, kvs **TO** Cores[core]

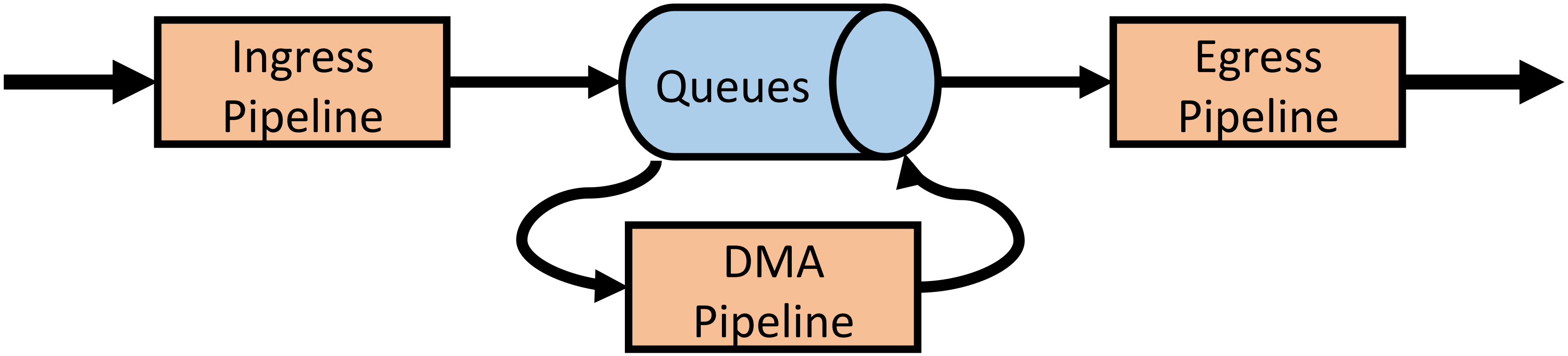
Supports:

- Steer packet
- Calculate hash/Xsum
- Initiate DMA operations
- Trigger reply packet
- Modify packets

Does not support:

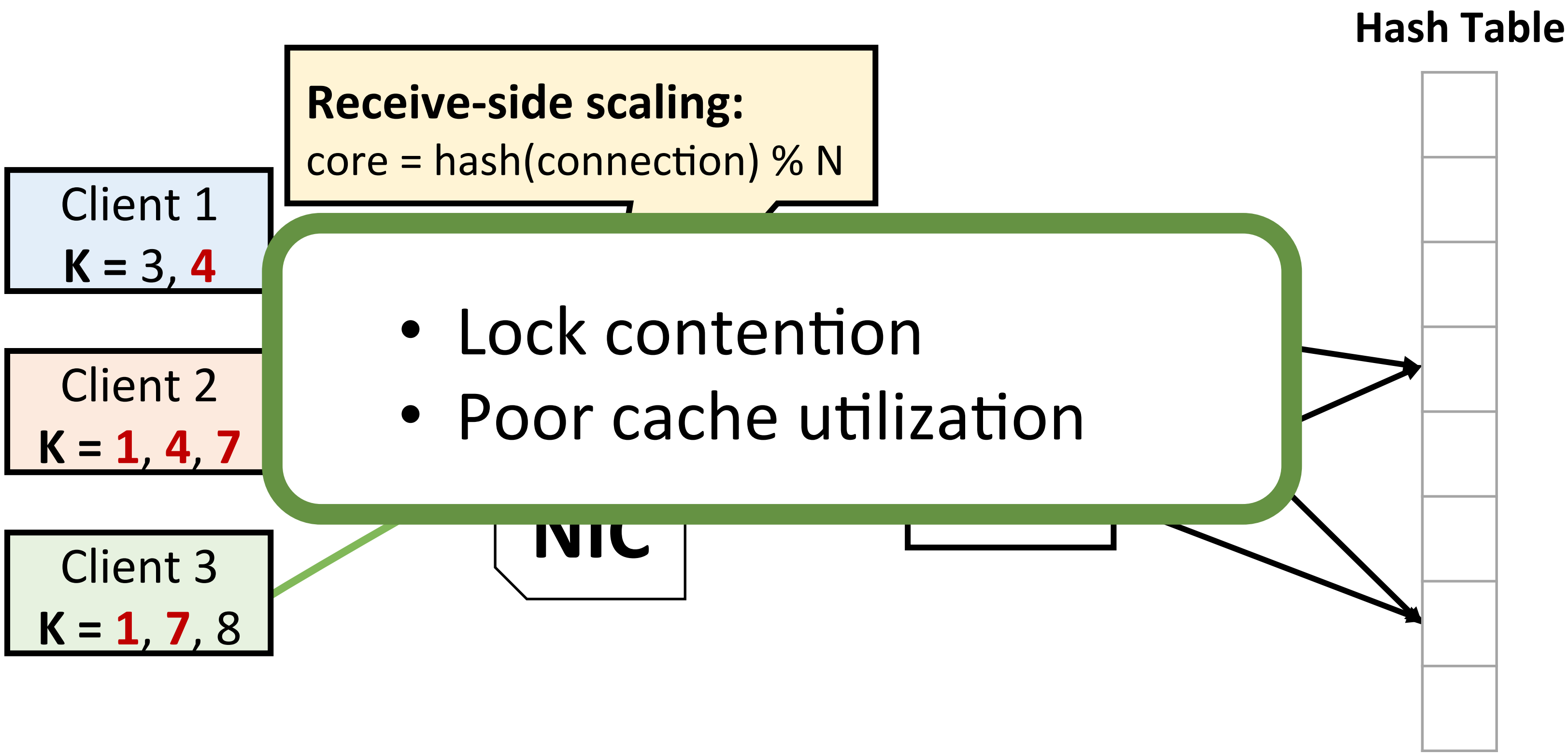
- Loops
- Complex calculations
- Keeping large state

FlexNIC: M+A for NICs

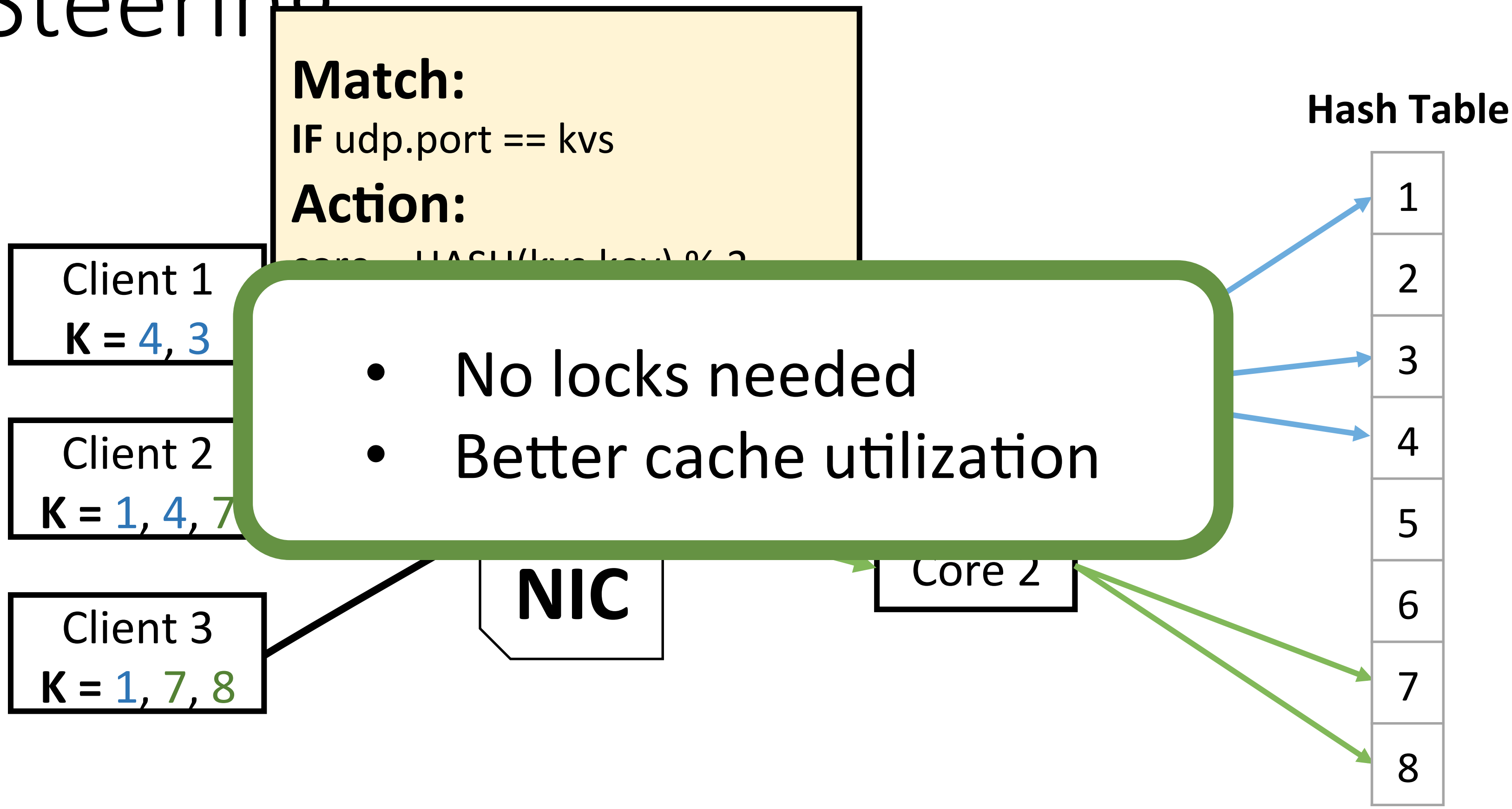


- Efficient application level processing in the NIC
  - Improve locality by steering to cores based on app criteria
  - Transform packets for efficient processing in SW
  - DMA directly into and out of application data structures
  - Send acknowledgements on NIC

Example: Key-Value Store

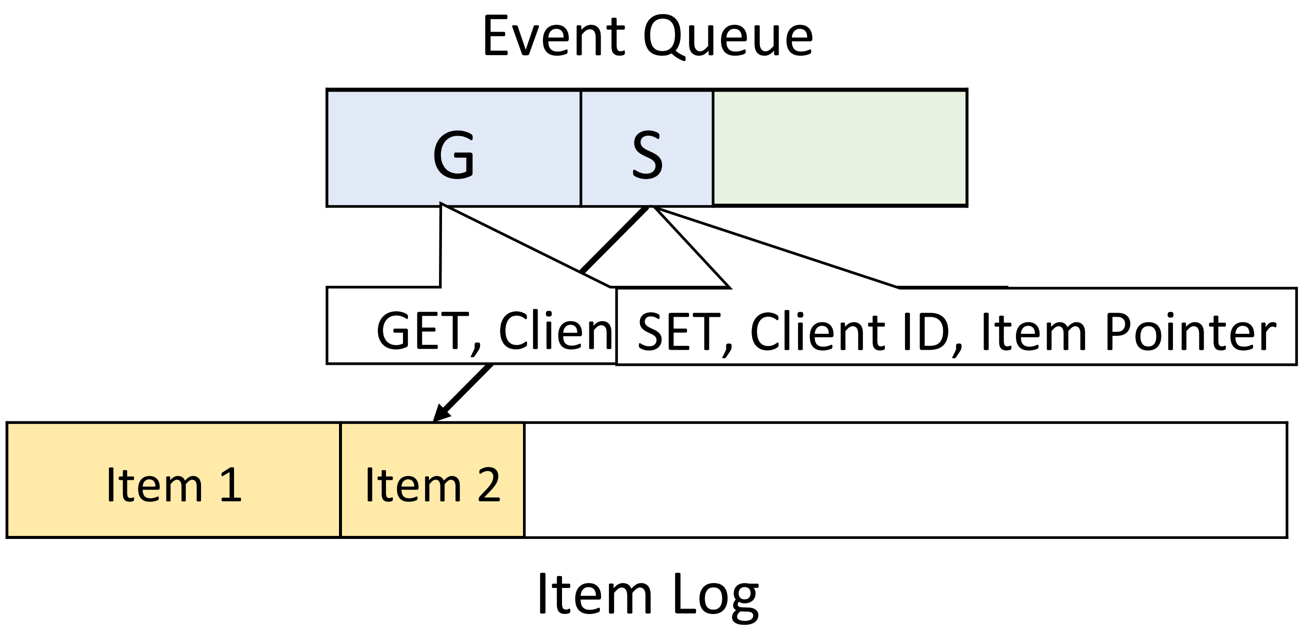


Optimizing Reads: Key-based Steering



- No locks needed
- Better cache utilization

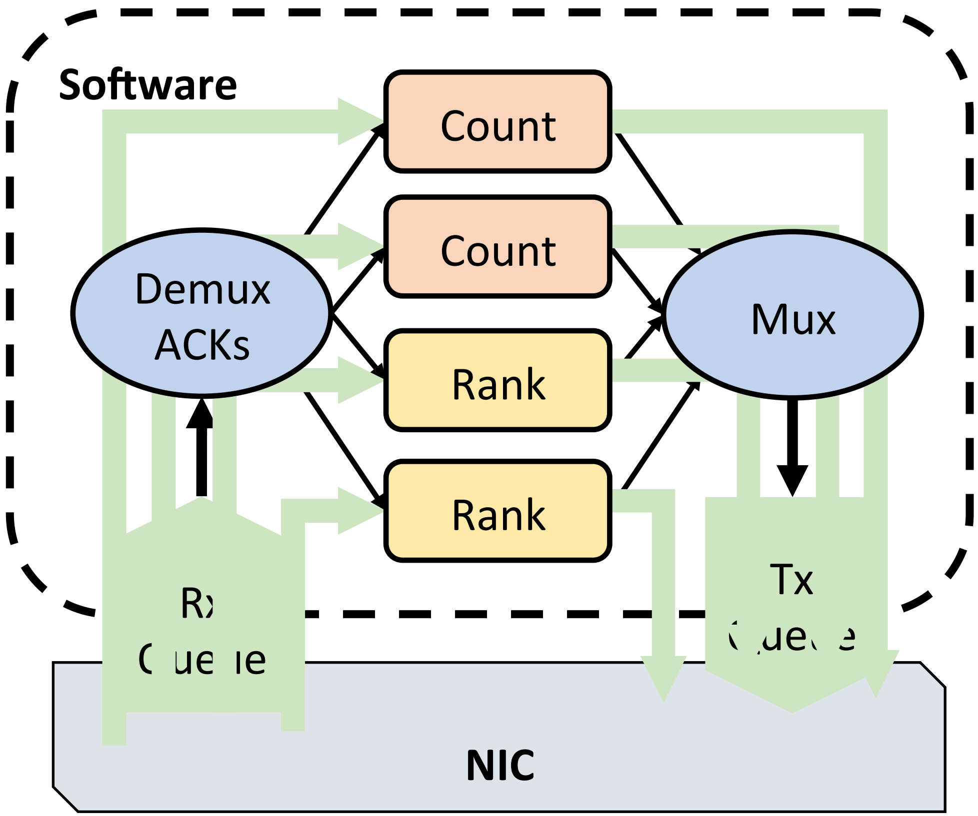
Optimizing Writes: Custom DMA



- DMA to application-level data structures
- Requires packet validation and transformation

Real-time Analytics System

- Offload (de)multiplexing and ACK generation to FlexNIC



Evaluation of the Model

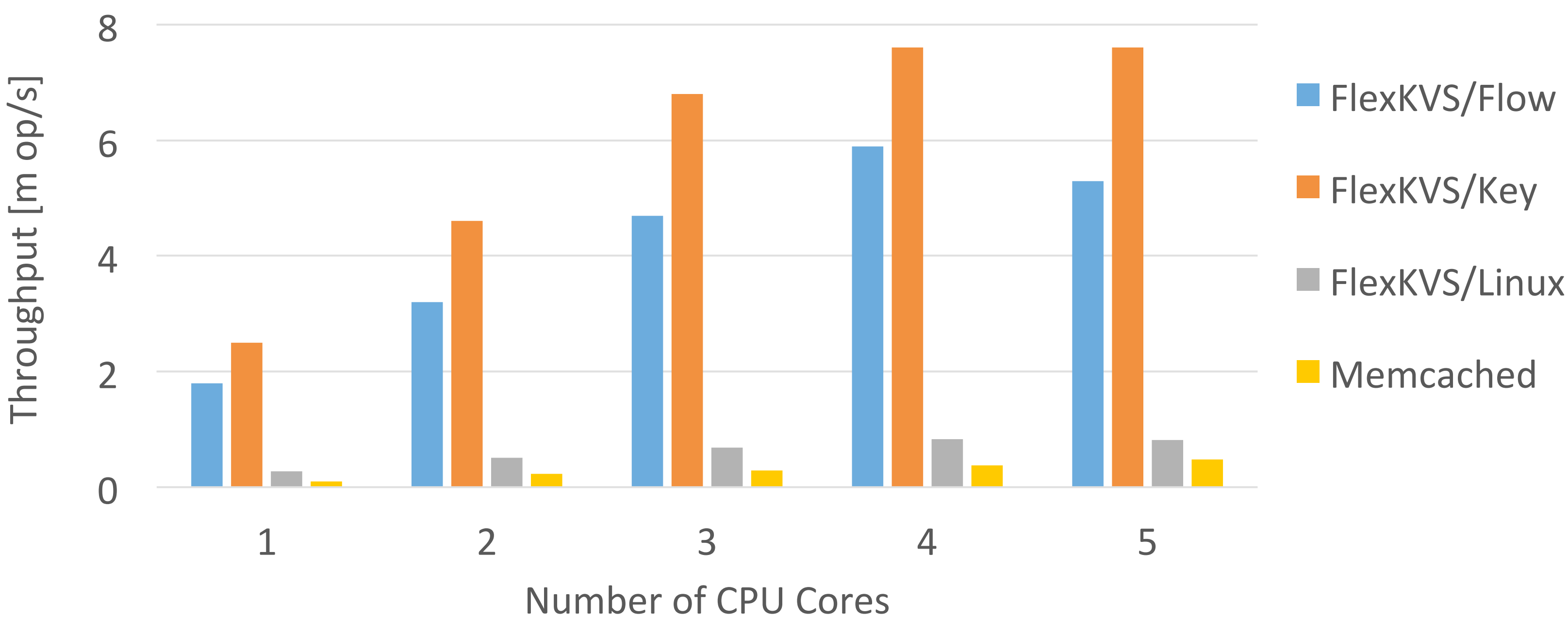
- Measure impact on application performance
  - Without waiting for hardware implementation

- Re-use existing NIC functionality
  - Hash on certain fields
- Software emulation of M+A pipeline

Key-value store:

- Workload: 100k 32B keys, 64B values, 90% GET
- 6 Core Sandy Bridge Xeon 2.2GHz, 2x 10G links

Key-based steering



- Better
  - PCIe
- 30-45%
- Process… ns

Steering and custom DMA reduces time from 510ns to 200ns

Arrakis and FlexNIC

- Data center applications need high performance I/O
- Rethink operating system and I/O hardware
  - Deliver packets, storage directly to applications
  - Hardware support for flexible I/O pipeline processing
- Source code, papers:
  `http://arrakis.cs.washington.edu/`