Arrakis:
The Operating System is the Control Plane

Simon Peter, Jialin Li, Irene Zhang,
Dan Ports, Doug Woos,
Arvind Krishnamurthy, Tom Anderson

University of Washington

Timothy Roscoe
ETH Zurich

Building an OS for the Data Center

- Server I/O performance matters
  - Key-value stores, web & file servers, lock managers, …
- **Can we deliver performance close to hardware?**

- Example system: Dell PowerEdge R520

Intel X520
10G NIC
2 us / 1KB packet

+

Intel RS3 RAID
1GB flash-backed cache
25 us / 1KB write

+

Sandy Bridge CPU
6 cores, 2.2 GHz

= **$1,200**

Building an OS for the Data Center

- Server I/O performance matters
  - Key-value stores, web & file servers, lock managers, …
- **Can we [illegible]re?**
- Example:

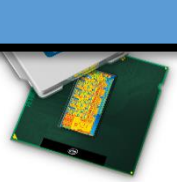
Today's I/O devices are fast

Intel X520
10G NIC
2 us / 1KB packet

+

Intel RS3 RAID
1GB flash-backed cache
25 us / 1KB write

+

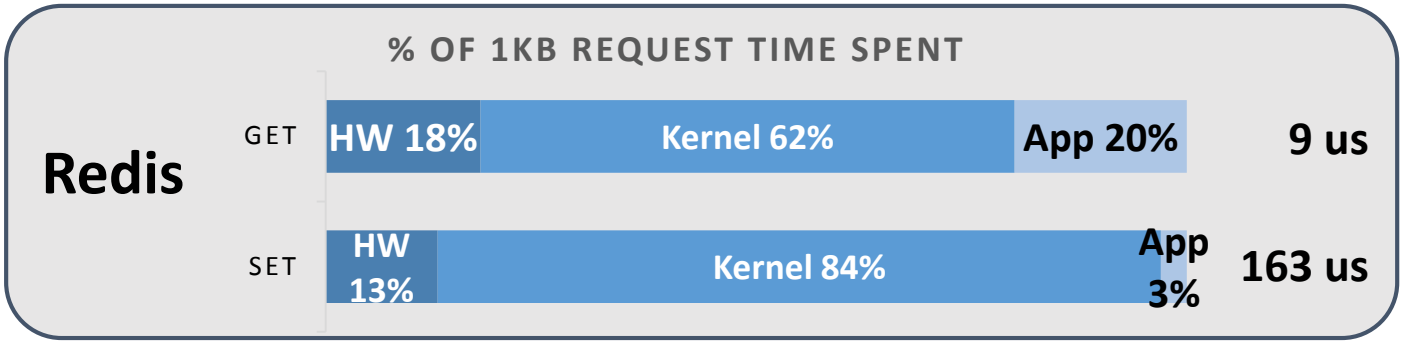


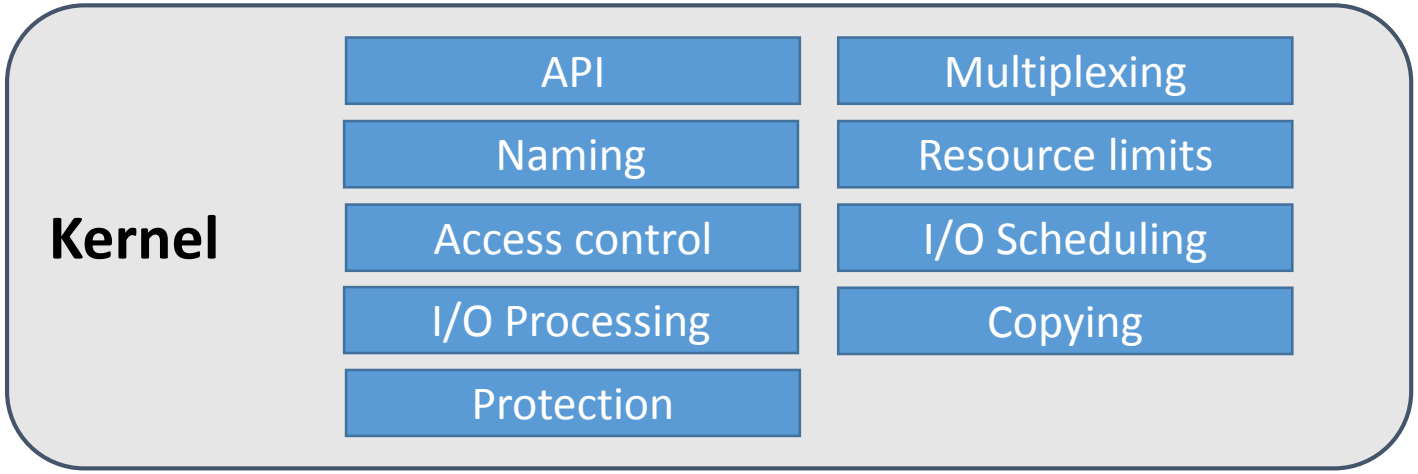
Sandy Bridge CPU
6 cores, 2.2 GHz

= **$1,200**

Can’t we just use Linux?

Linux I/O Performance





Data Path

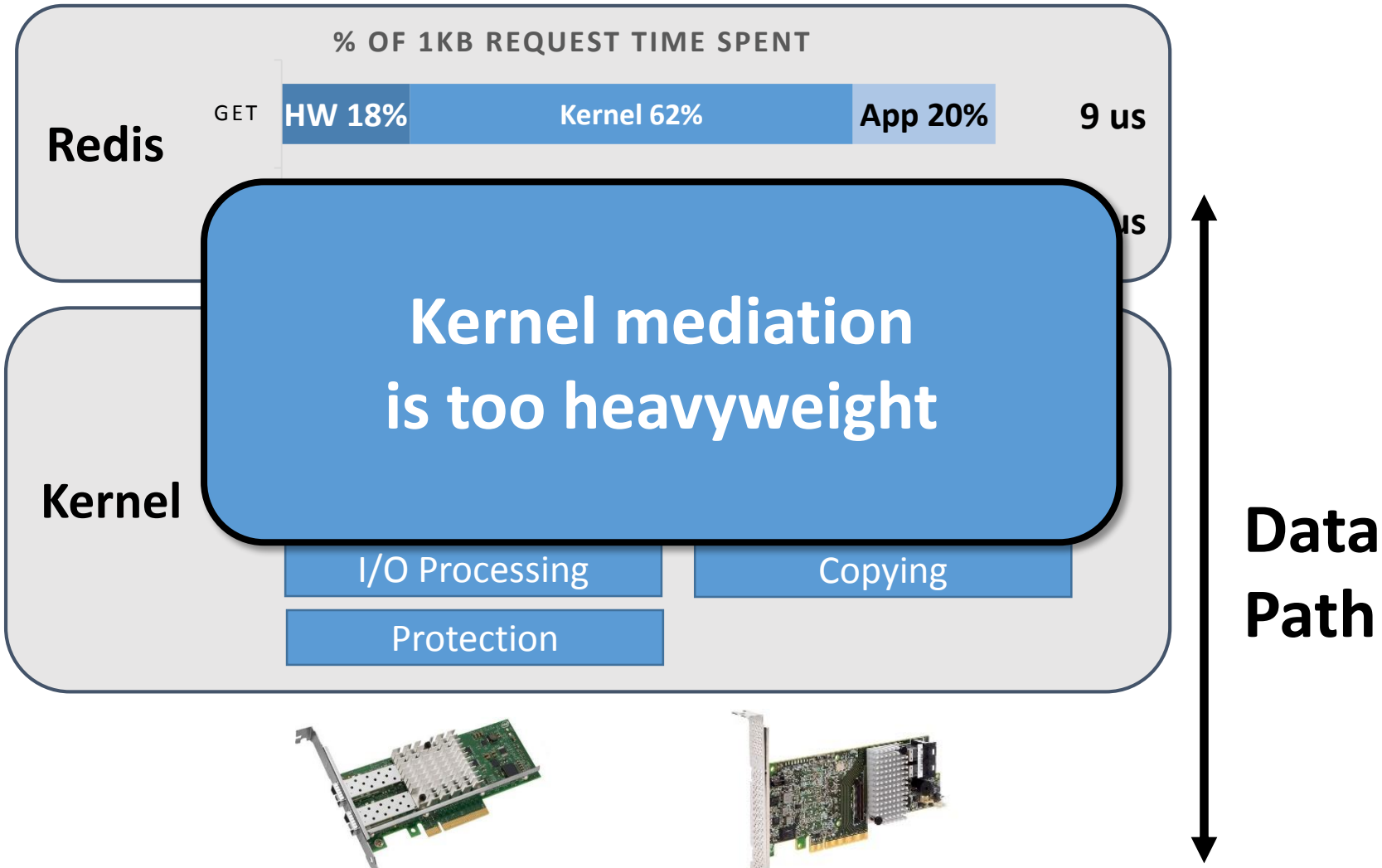
10G NIC

2 us / 1KB packet

RAID Storage

25 us / 1KB write

Linux I/O Performance

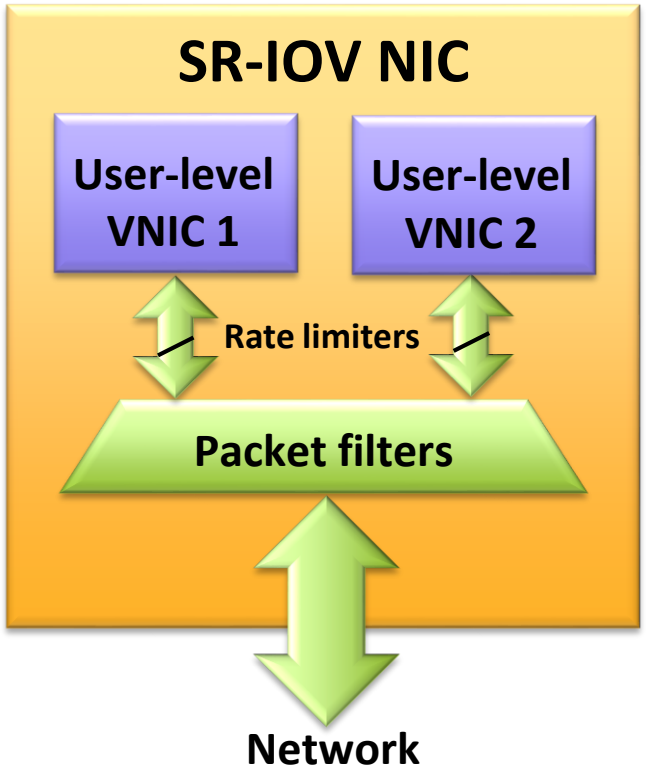


Arrakis Goals

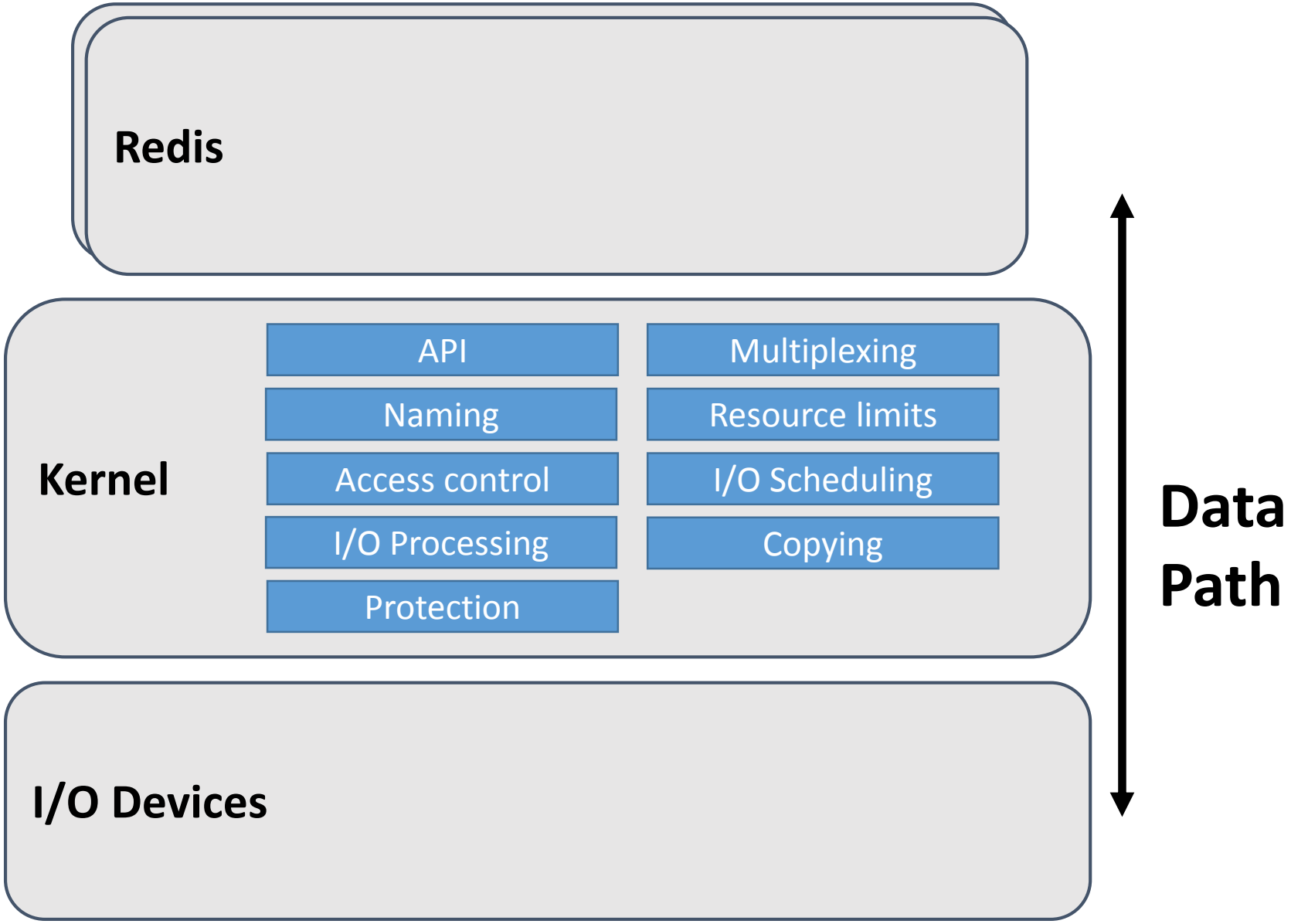
- Skip kernel & deliver I/O directly to applications
  - Reduce OS overhead
- Keep classical server OS features
  - Process protection
  - Resource limits
  - I/O protocol flexibility
  - Global naming
- The hardware can help us…

Hardware I/O Virtualization

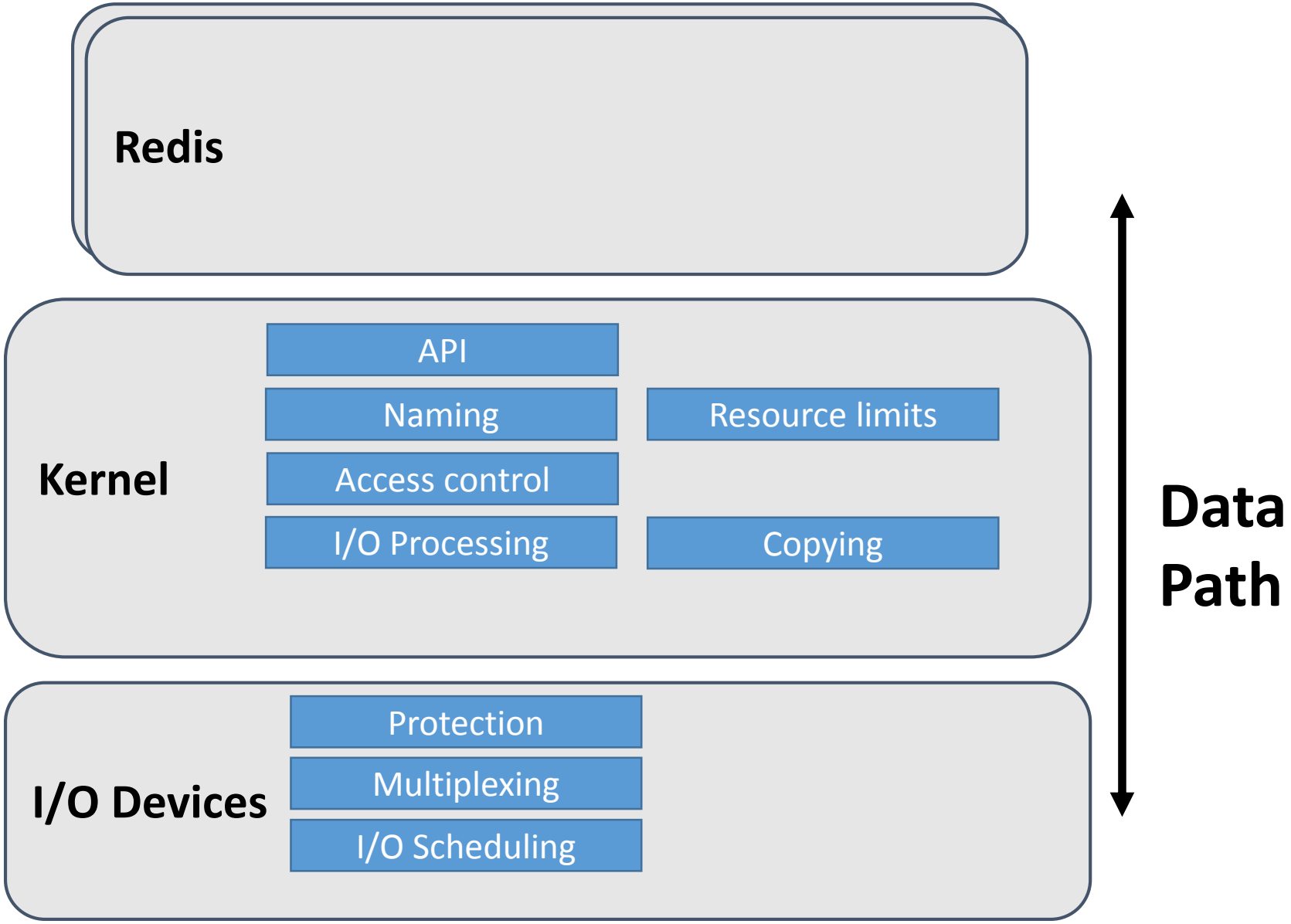
- Standard on NIC, emerging on RAID
- Multiplexing
  - **SR-IOV**: Virtual PCI devices w/ own registers, queues, INTs
- Protection
  - **IOMMU**: Devices use app virtual memory
  - **Packet filters**, **logical disks**: Only allow eligible I/O
- I/O Scheduling
  - NIC **rate limiter**, **packet schedulers**



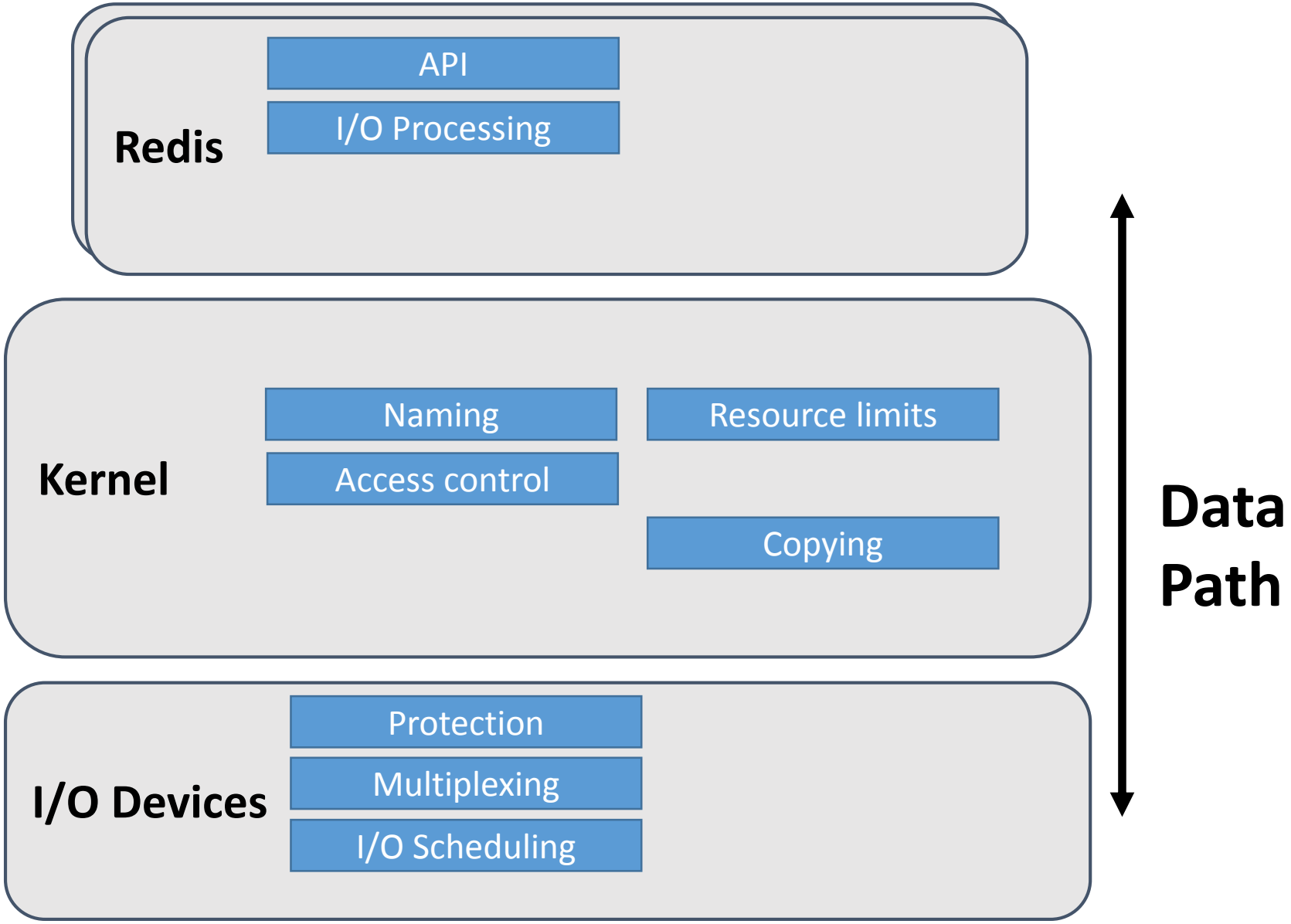
How to skip the kernel?



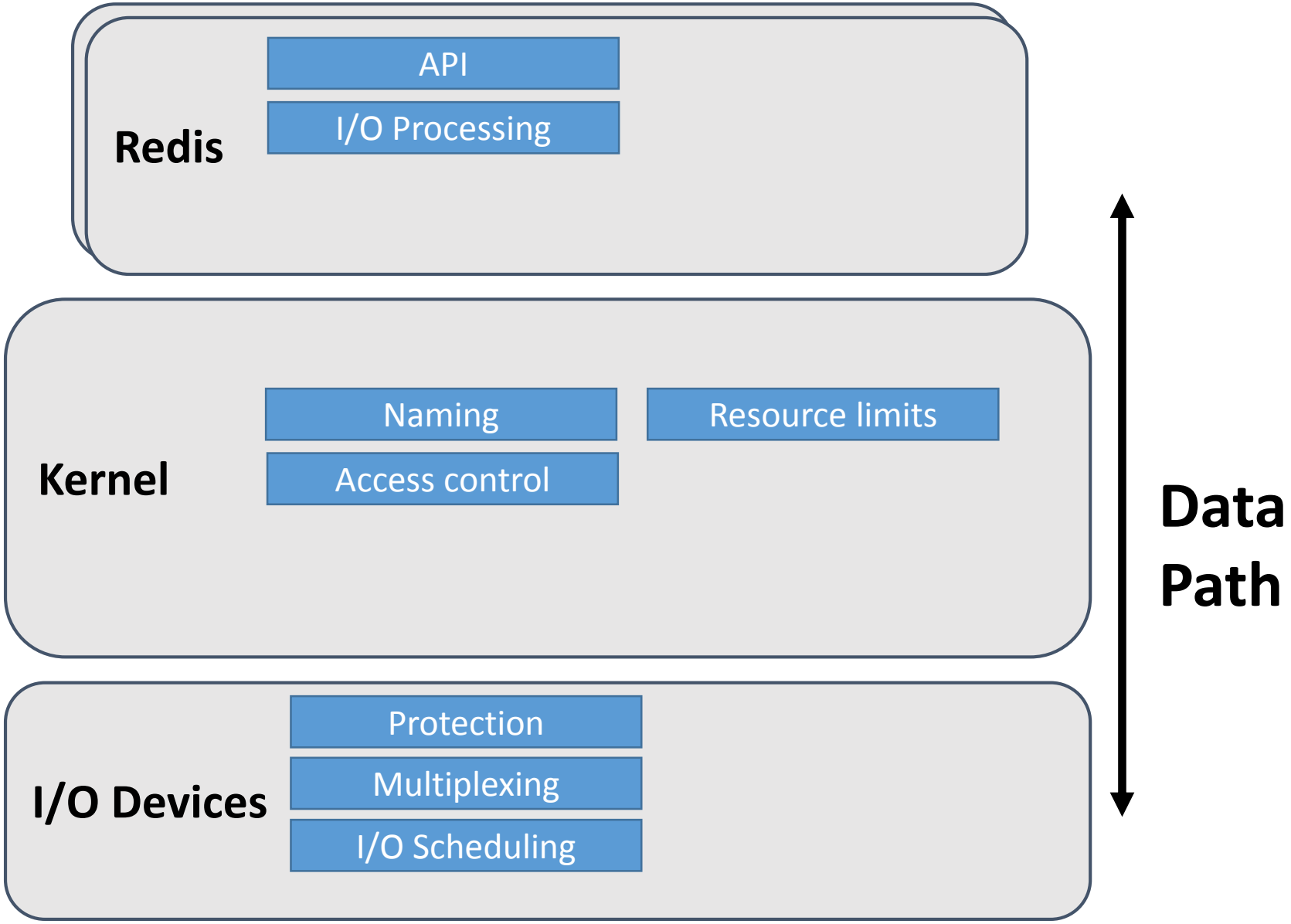
How to skip the kernel?



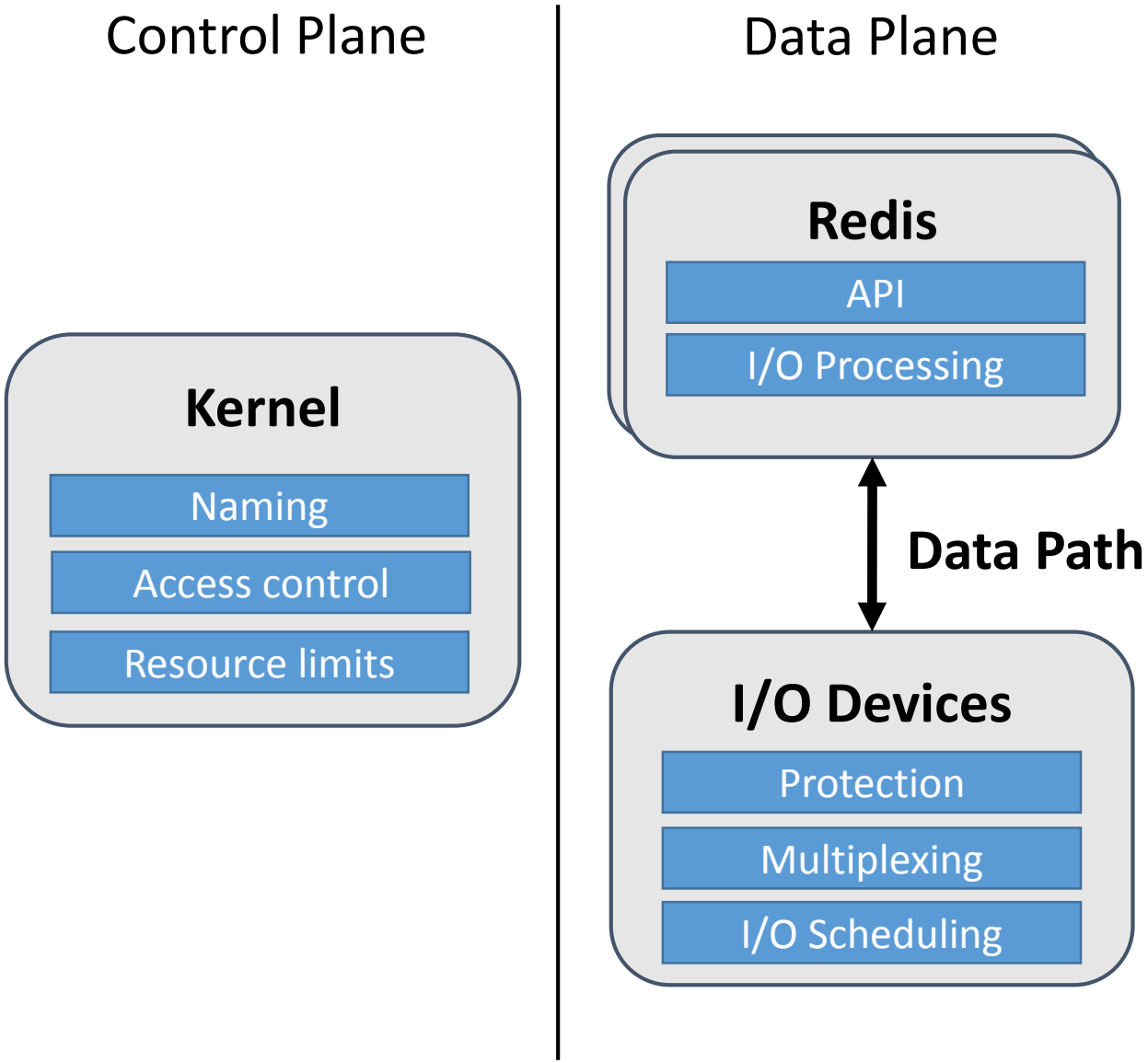
How to skip the kernel?



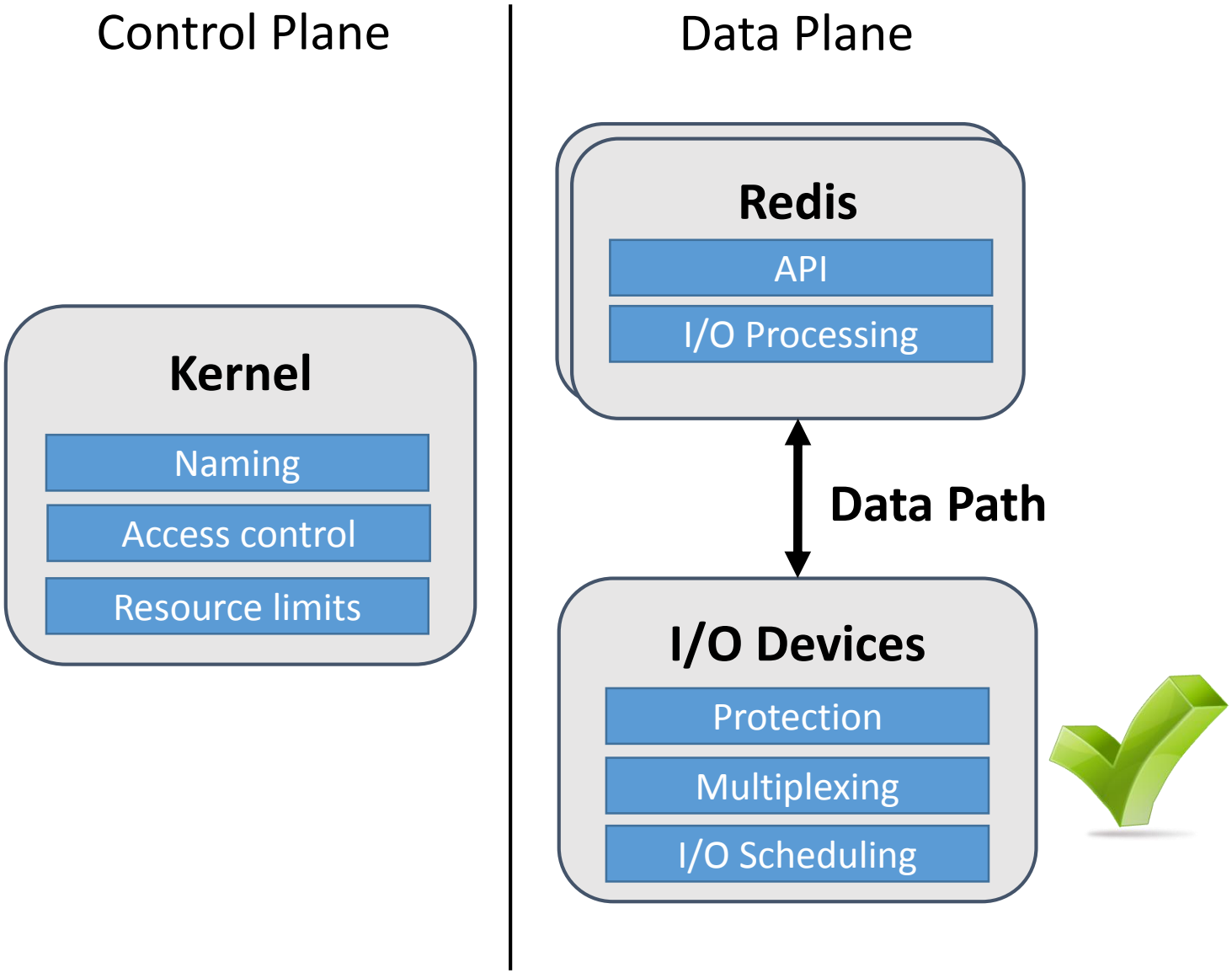
How to skip the kernel?



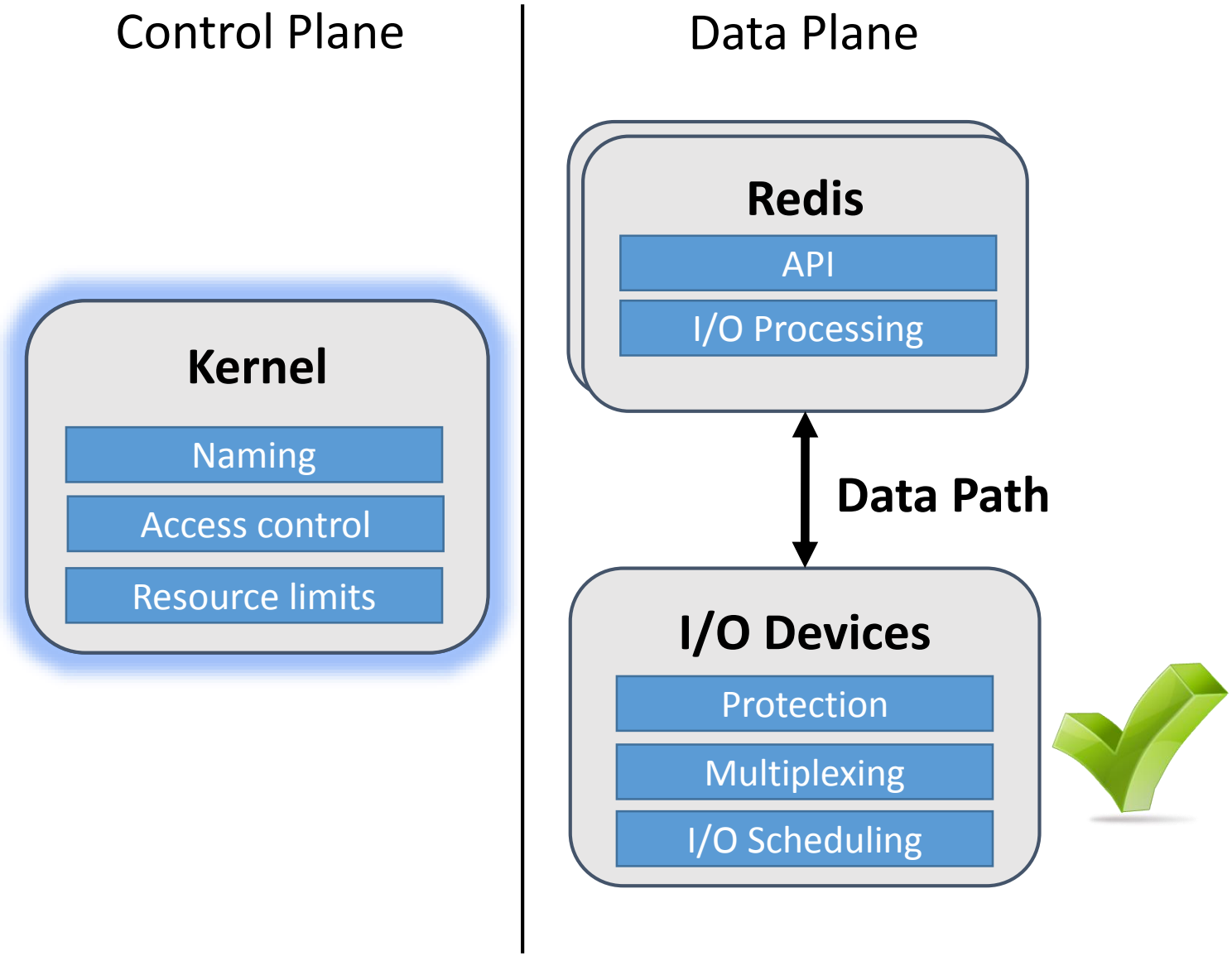
Arrakis I/O Architecture



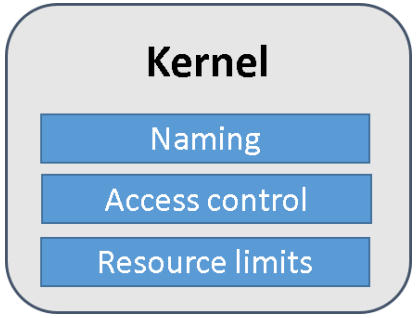
Arrakis I/O Architecture



Arrakis I/O Architecture

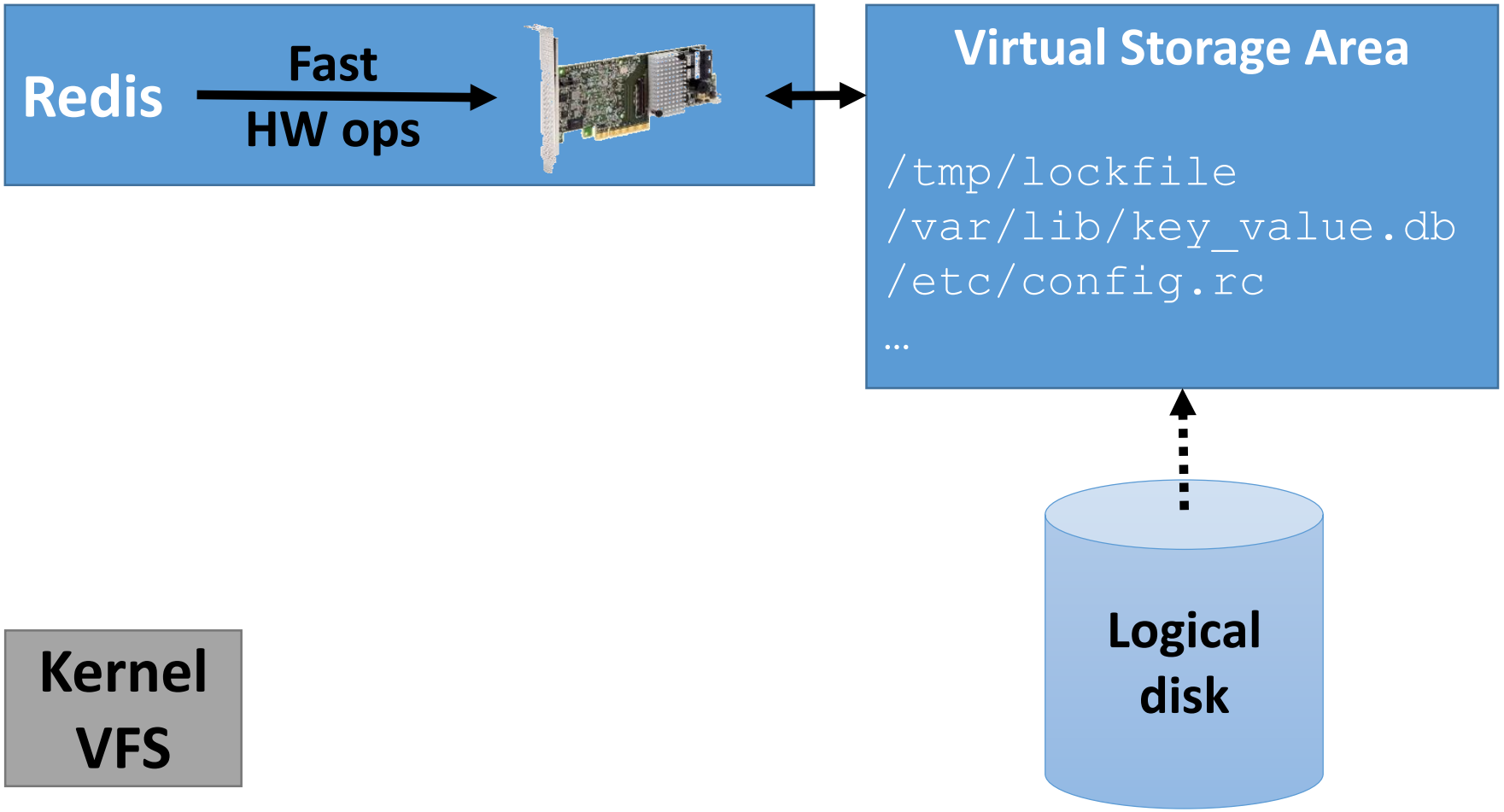


Arrakis Control Plane

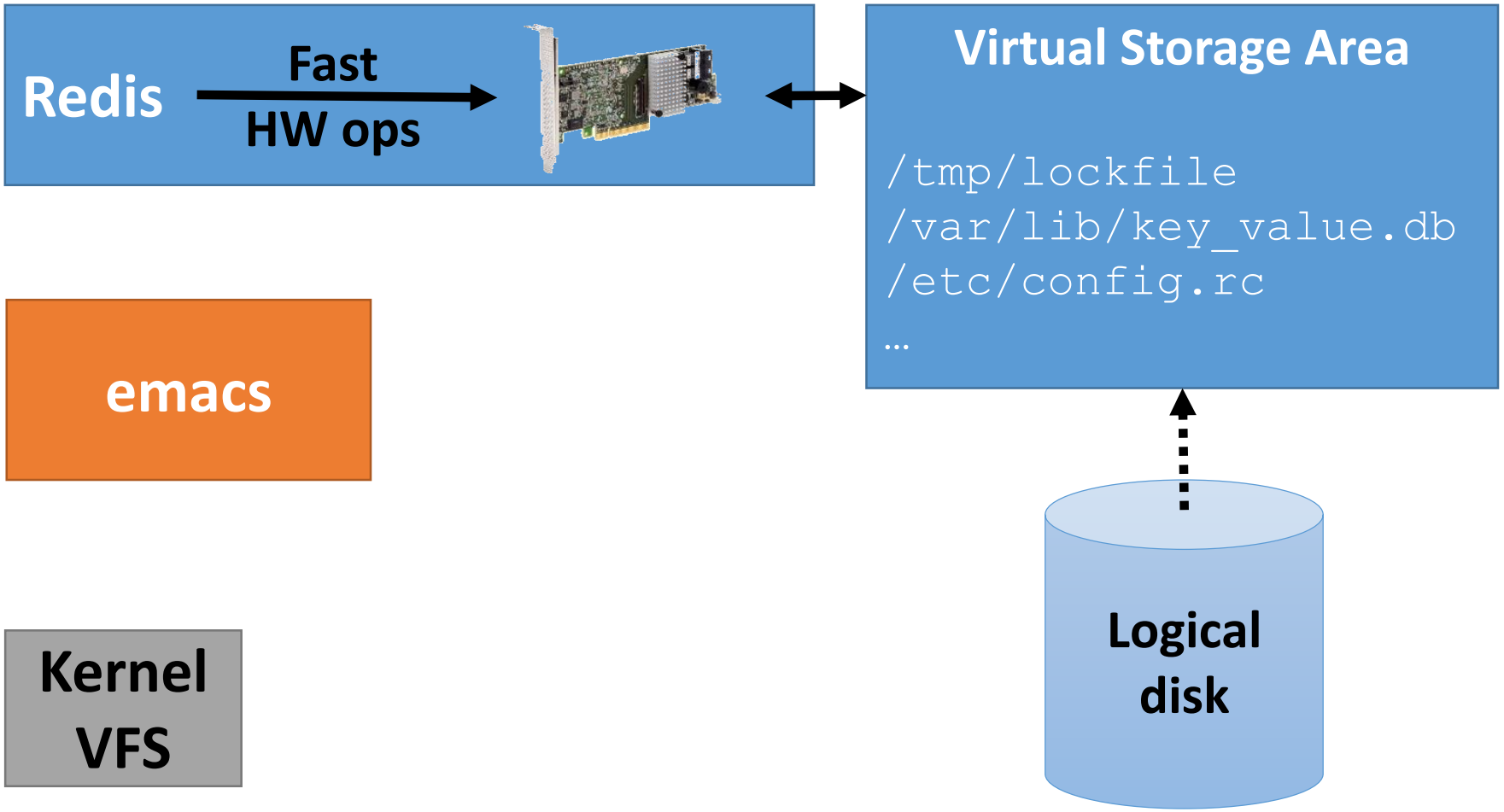


- Access control
  - Do once when configuring data plane
  - Enforced via NIC filters, logical disks
- Resource limits
  - Program hardware I/O schedulers
- Global naming
  - Virtual file system still in kernel
  - Storage implementation in applications

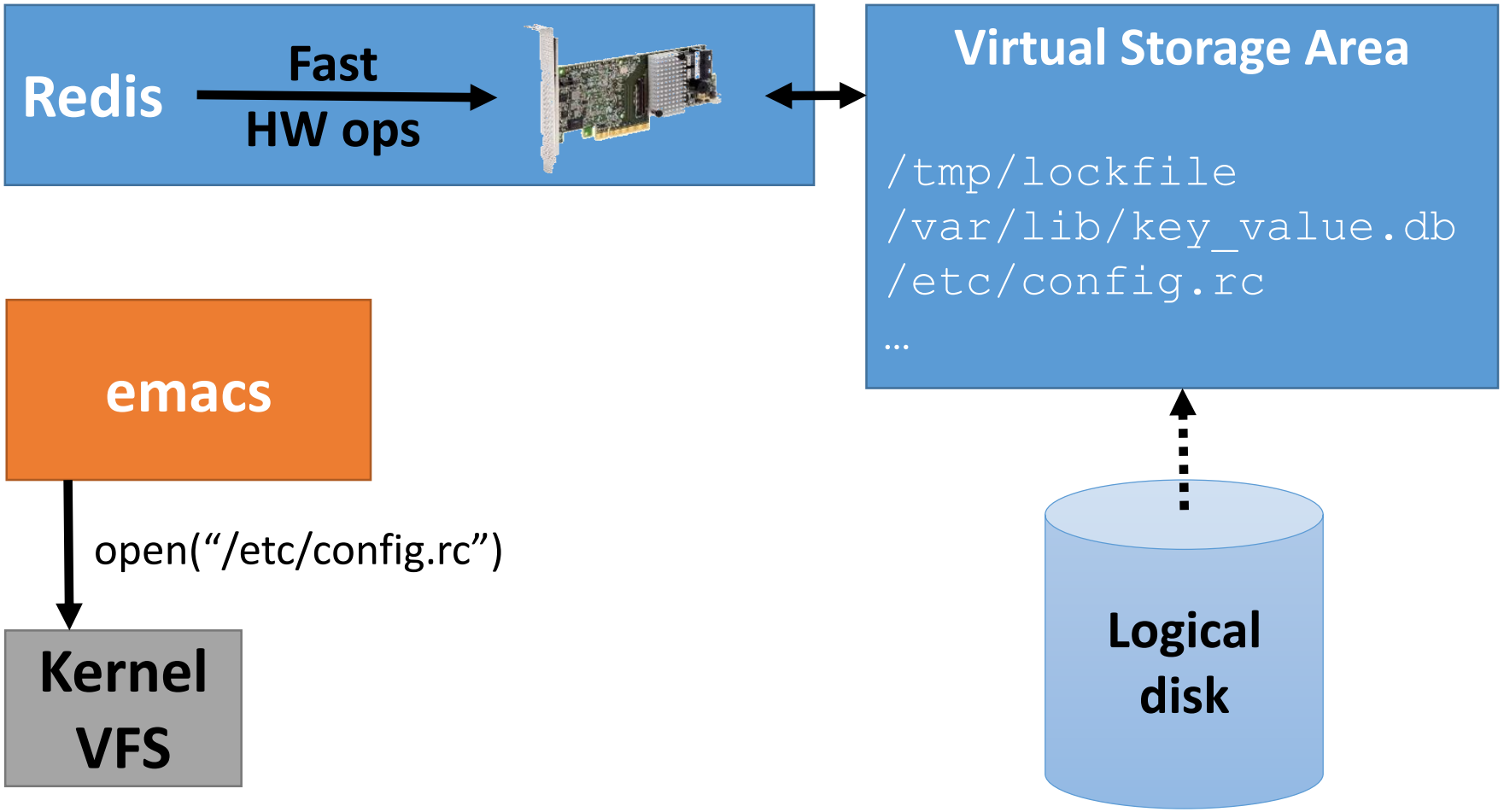
Global Naming



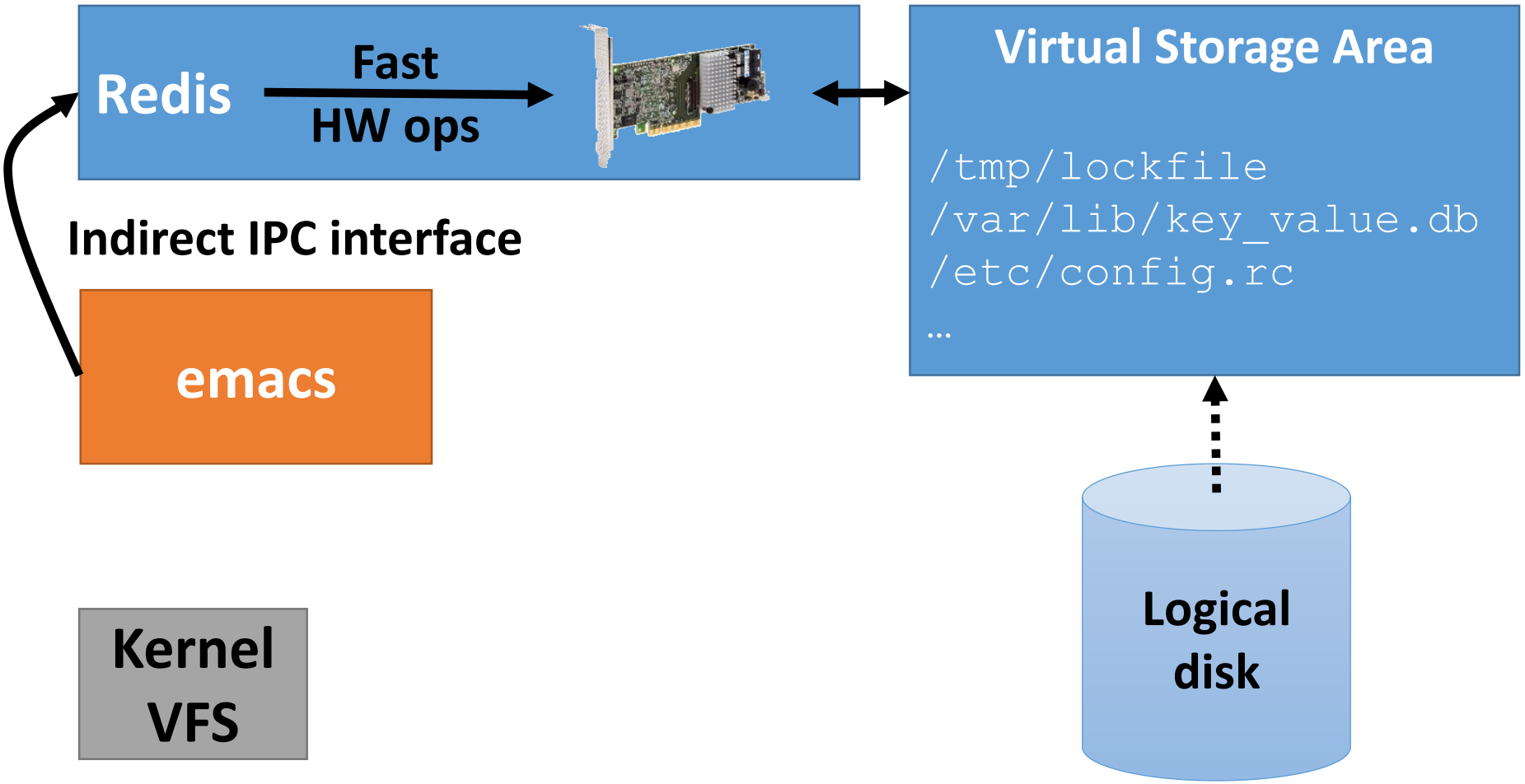
Global Naming



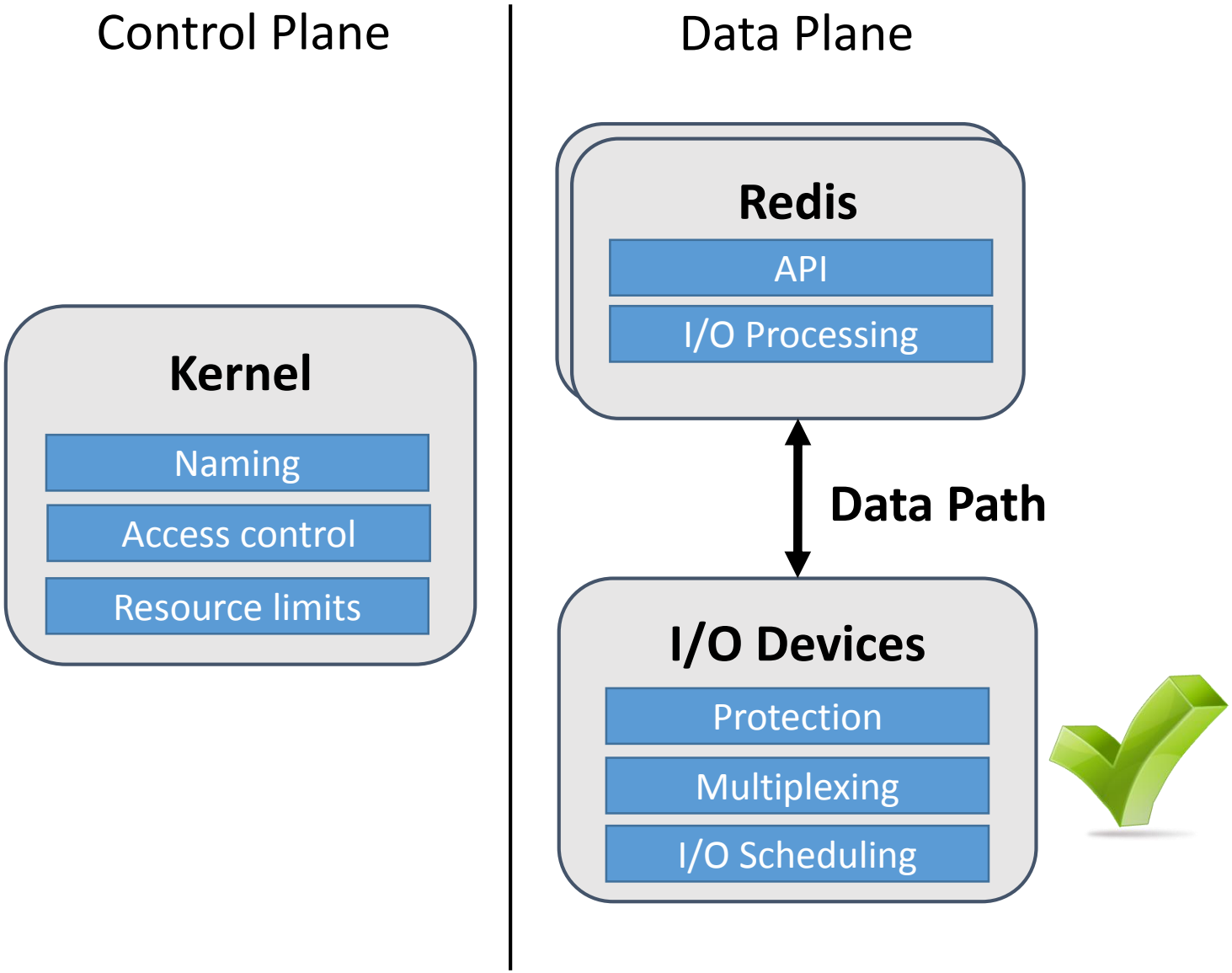
Global Naming



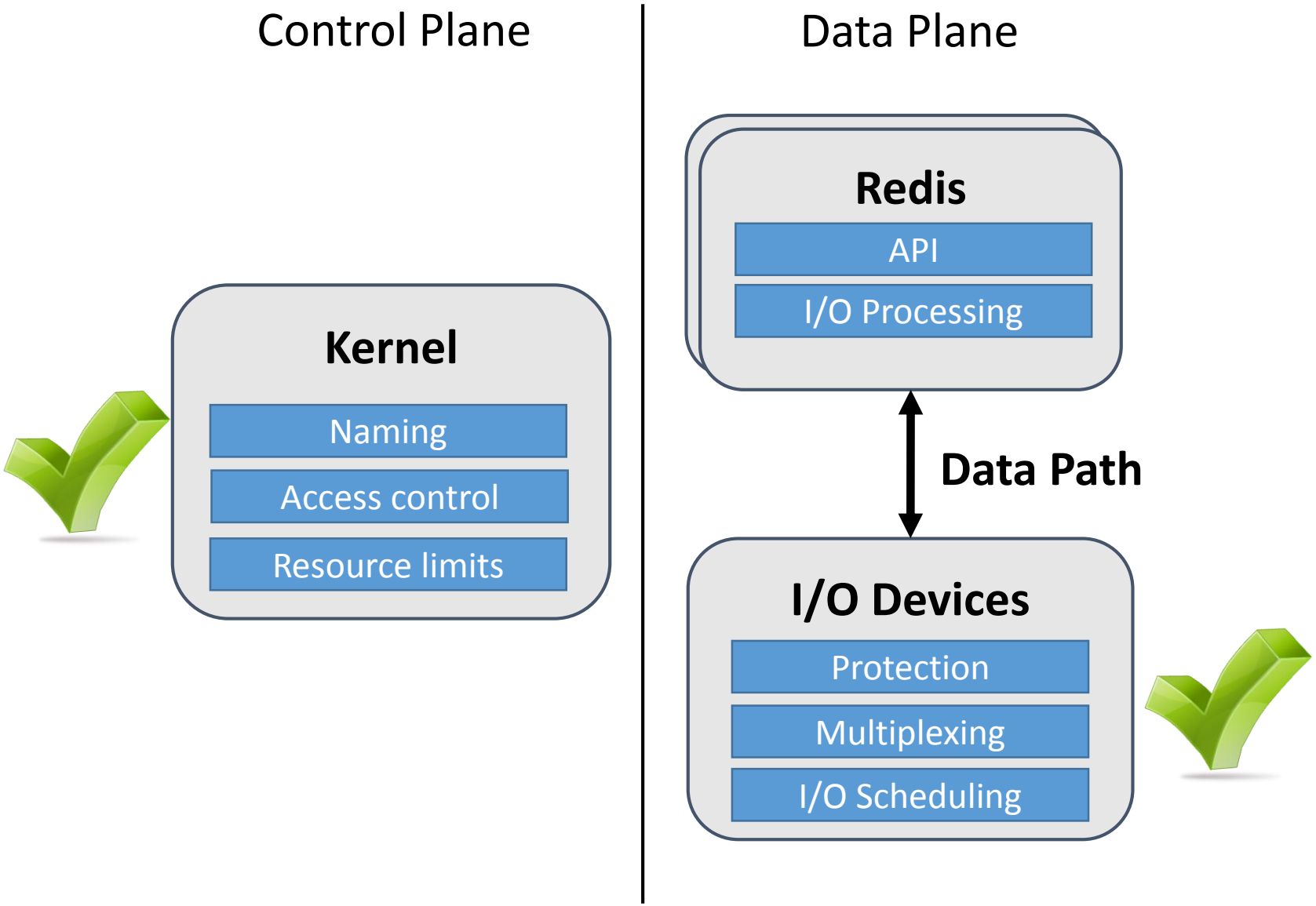
Global Naming



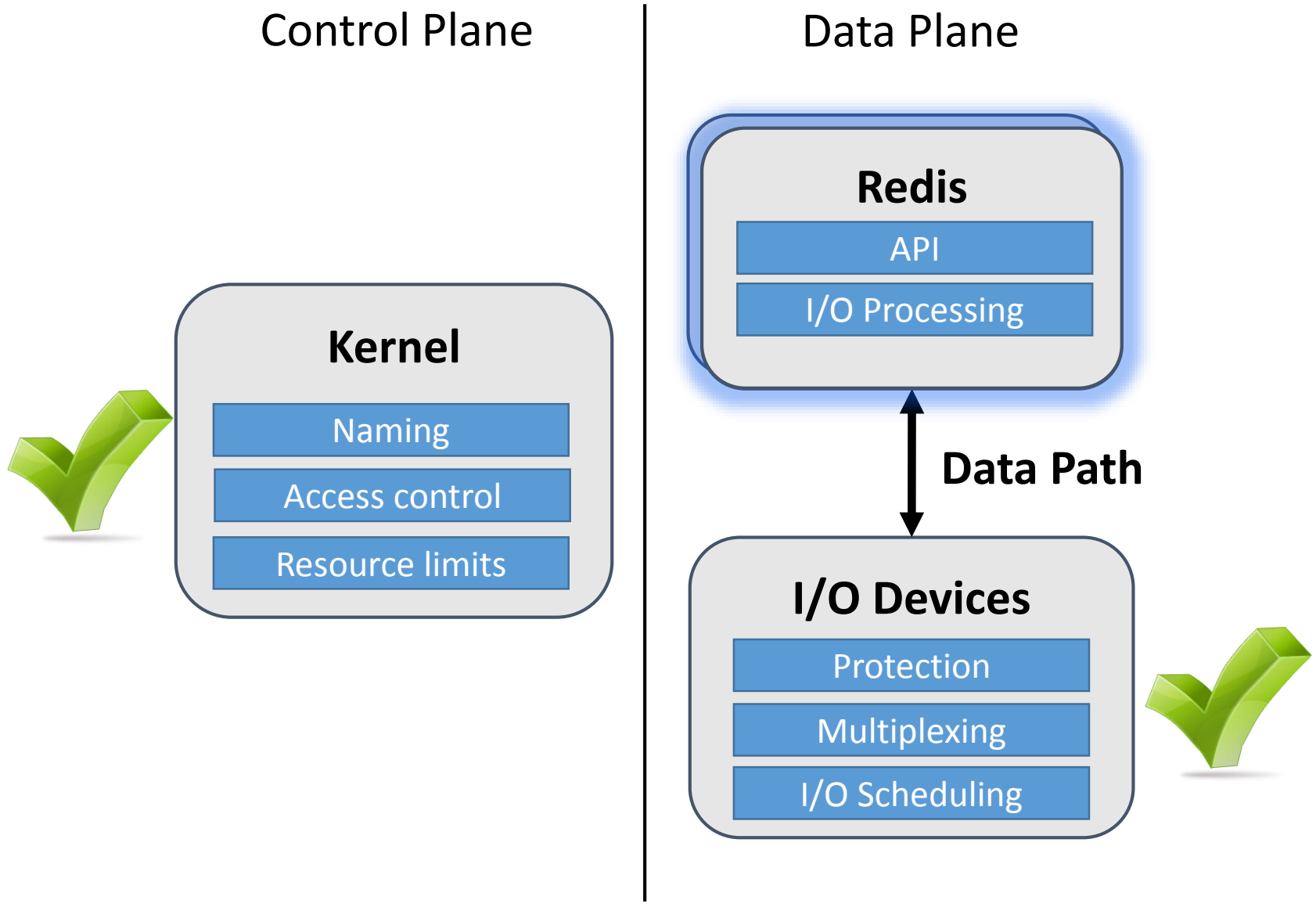
Arrakis I/O Architecture



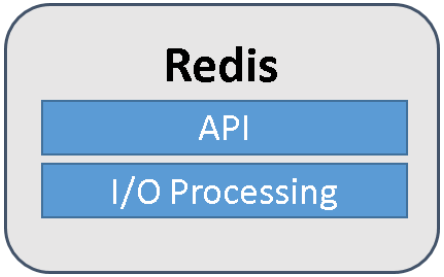
Arrakis I/O Architecture



Arrakis I/O Architecture



Storage Data Plane: Persistent Data Structures



- Examples: **log, queue**
- Operations immediately persistent on disk

Benefits:

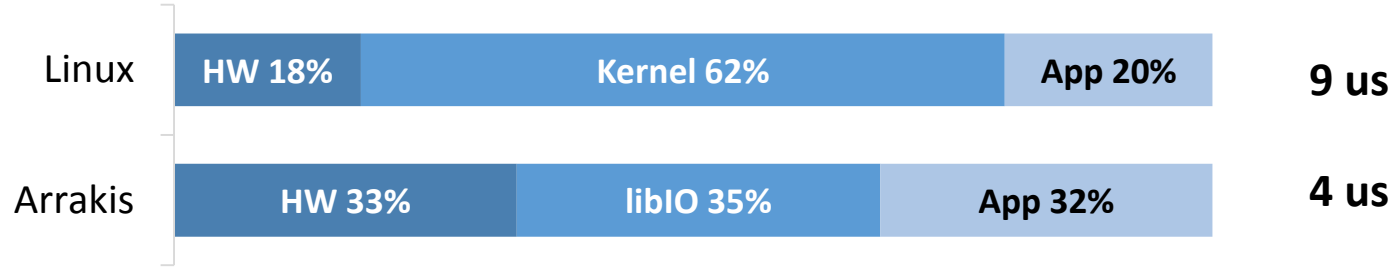
- In-memory = on-disk layout
  - Eliminates marshaling
- Metadata in data structure
  - Early allocation
  - Spatial locality
- Data structure specific caching/prefetching

- Modified Redis to use **persistent log**: **109 LOC** changed

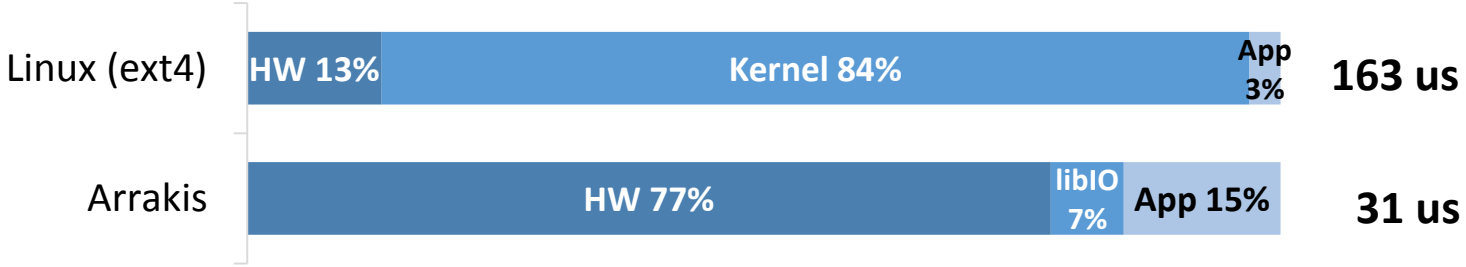
Evaluation

Redis Latency

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



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

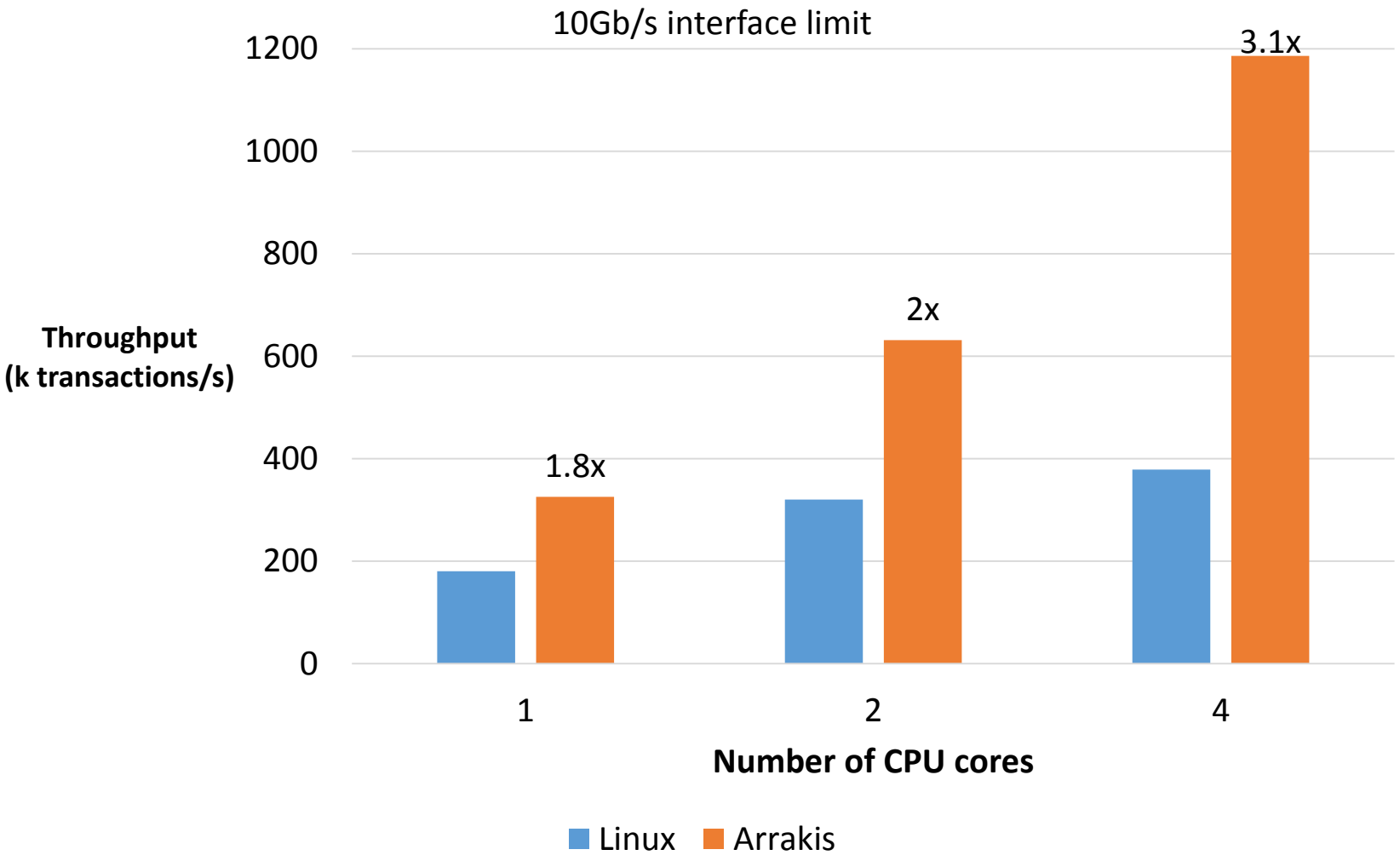


Redis Throughput

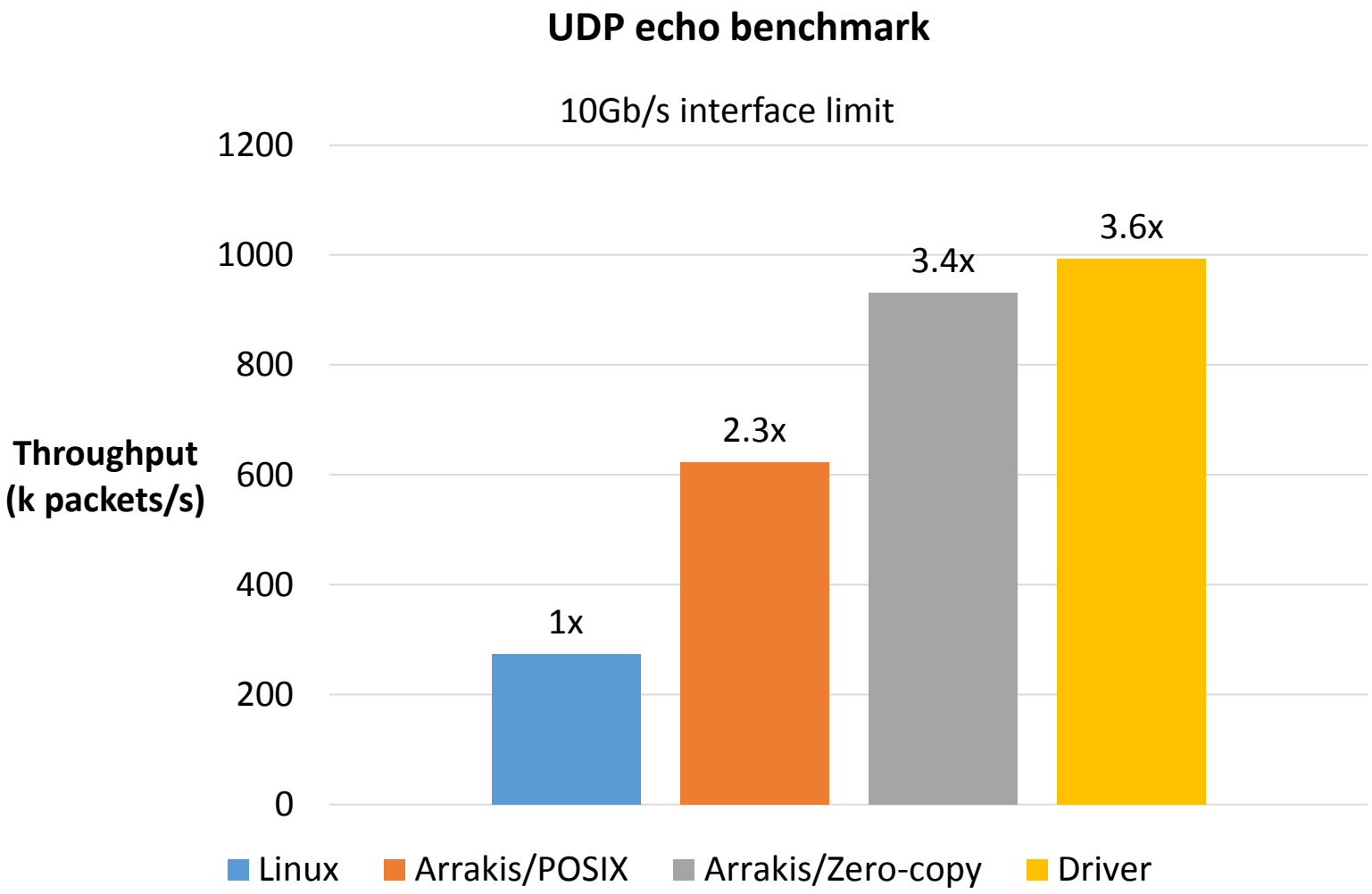
- Improved GET throughput by **1.75x**
  - Linux: **143k** transactions/s
  - Arrakis: **250k** transactions/s

- Improved SET throughput by **9x**
  - Linux: **7k** transactions/s
  - Arrakis: **63k** transactions/s

memcached Scalability



Single-core Performance



Summary

- OS is becoming an I/O bottleneck
  - Globally shared I/O stacks are slow on data path
- **Arrakis:** Split OS into control/data plane
  - Direct application I/O on data path
  - Specialized I/O libaries
- Application-level I/O stacks deliver great performance
  - **Redis:** up to **9x** throughput, **81%** speedup
  - Memcached **scales linearly** to **3x** throughput

Source code: ***http://arrakis.cs.washington.edu***