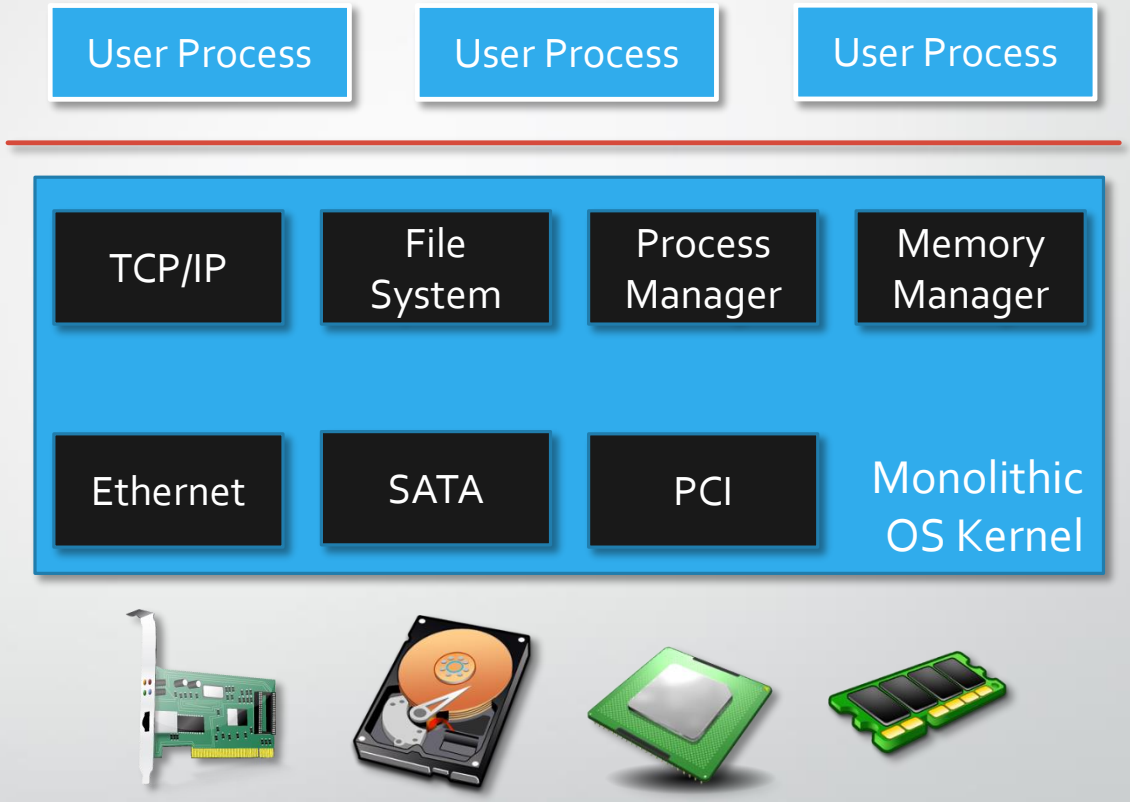
# VirtuOS: an operating system with kernel virtualization

Ruslan Nikolaev, Godmar Back
Virginia Tech

# Motivation

- Component failure may lead to system failure in monolithic OS designs
- Lack of protection for systems code
- Well-studied problem

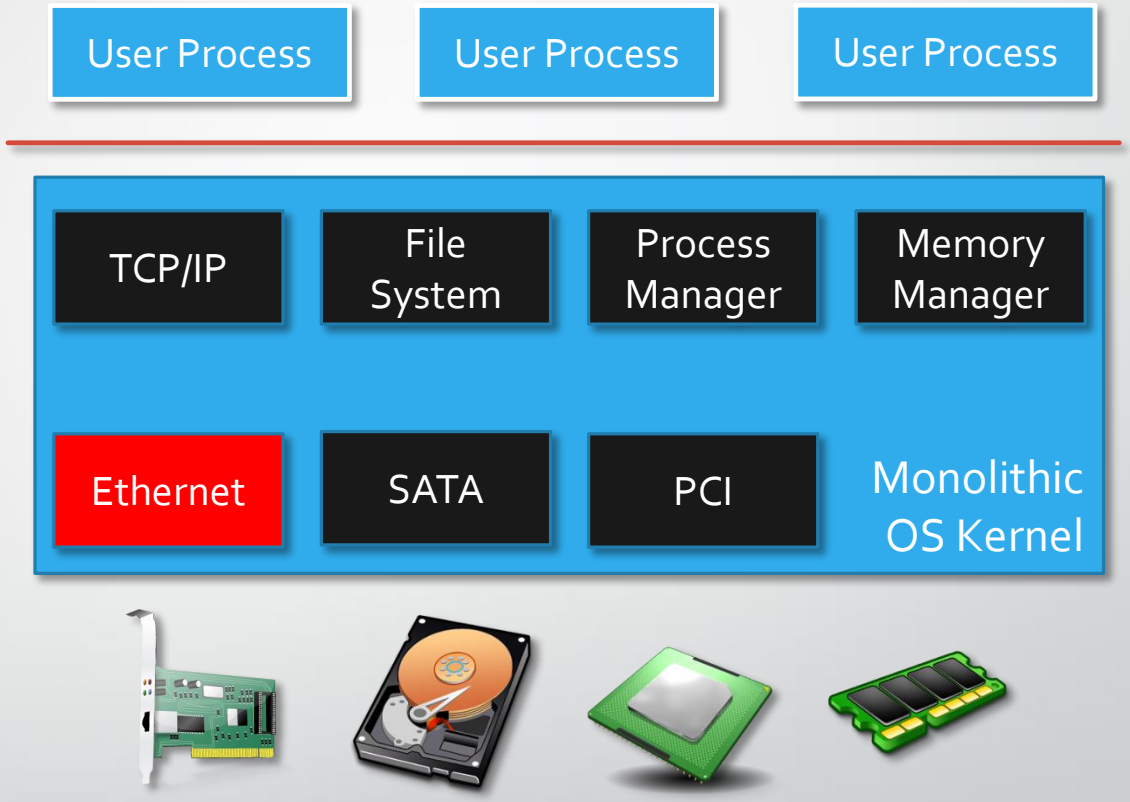
- [Ganapathi LISA'06]
  [Glerum SOSP'09]
  [Herder LADC'09]
  [Murphy Queue'04]



# Motivation

- Component failure may lead to system failure in monolithic OS designs
- Lack of protection for systems code
- Well-studied problem

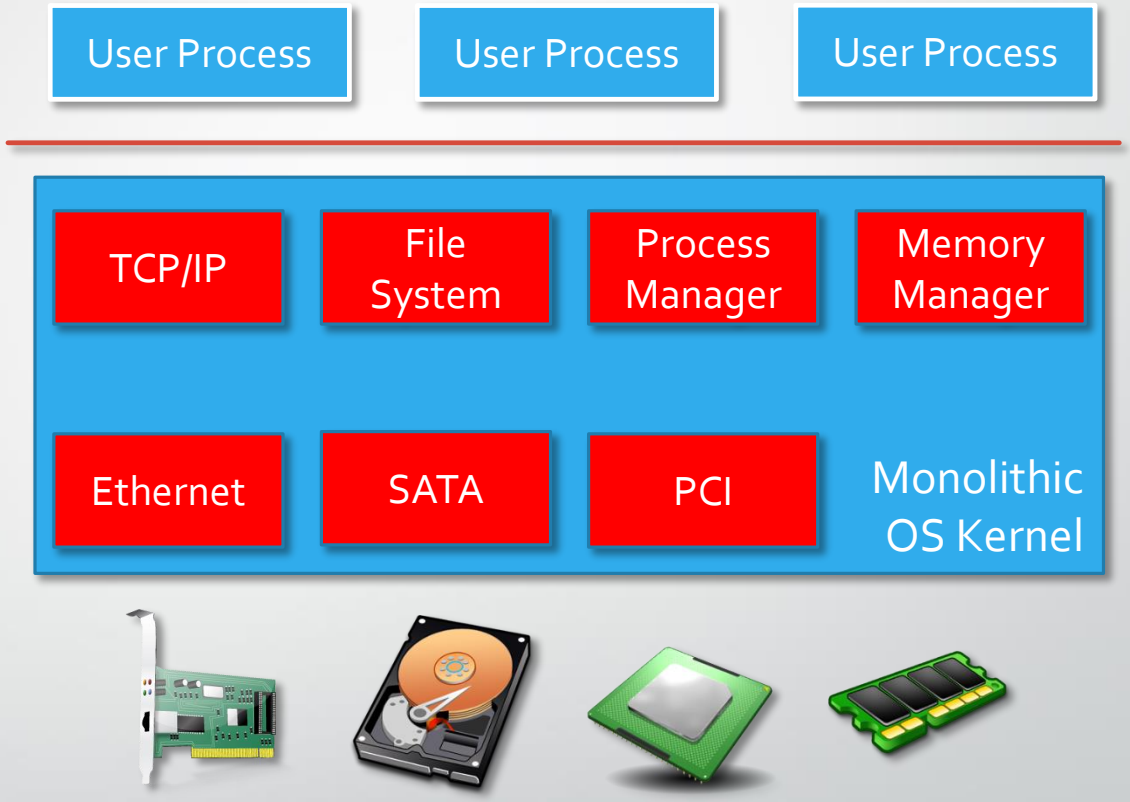
- [Ganapathi LISA'06]
  [Glerum SOSP'09]
  [Herder LADC'09]
  [Murphy Queue'04]



# Motivation

- Component failure may lead to system failure in monolithic OS designs
- Lack of protection for systems code
- Well-studied problem

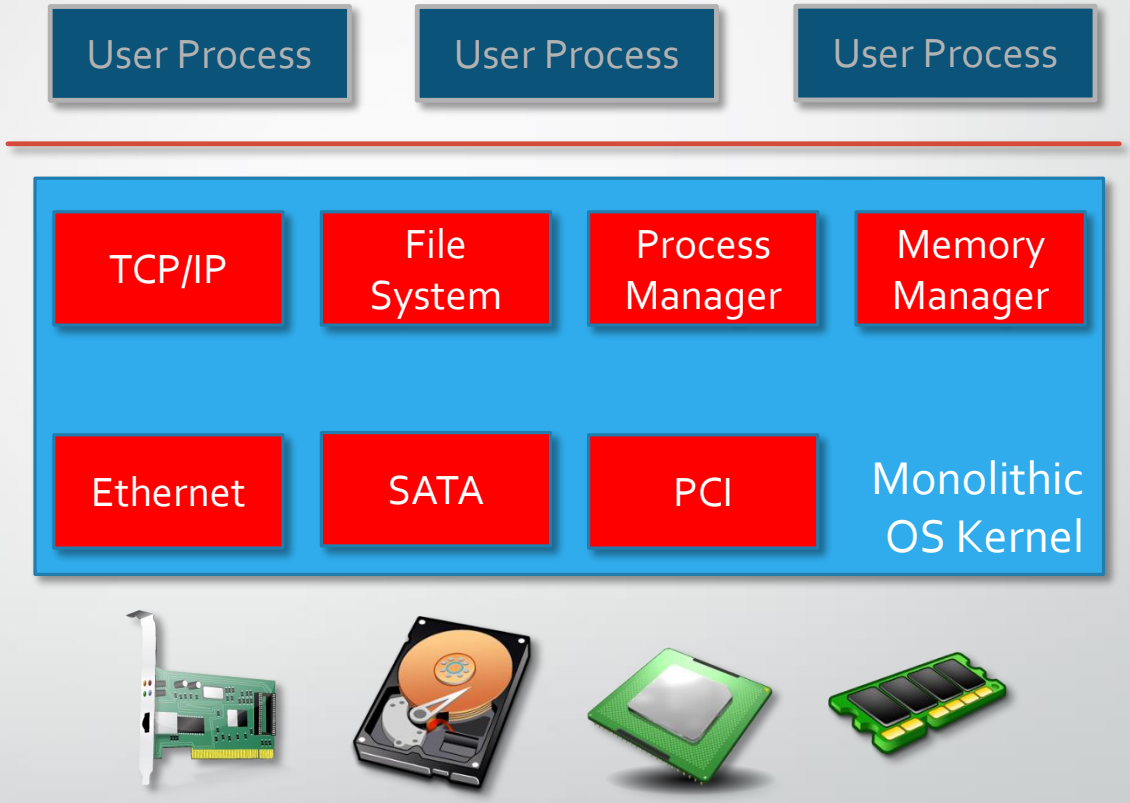
- [Ganapathi LISA'06]
  [Glerum SOSP'09]
  [Herder LADC'09]
  [Murphy Queue'04]



# Motivation

- Component failure may lead to system failure in monolithic OS designs
- Lack of protection for systems code
- Well-studied problem

- [Ganapathi LISA'06]
  [Glerum SOSP'09]
  [Herder LADC'09]
  [Murphy Queue'04]



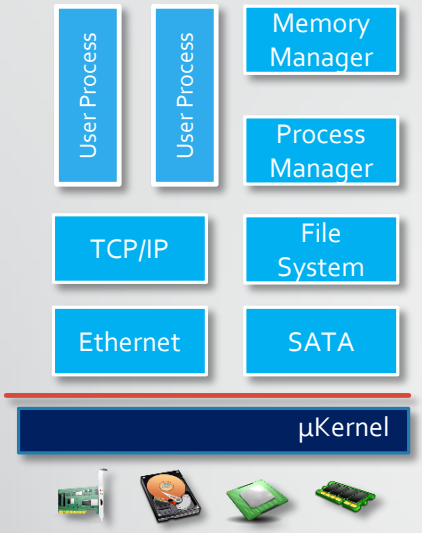
# Decomposition and Isolation

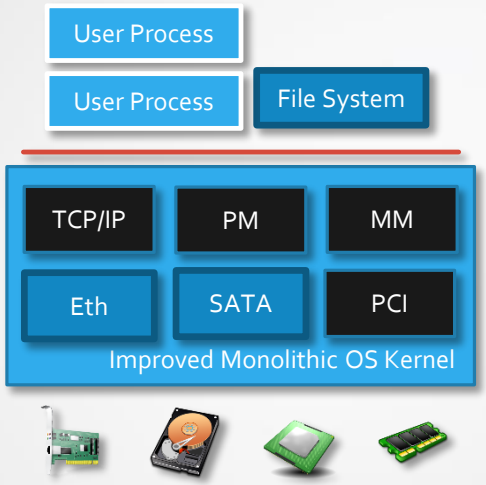
- Split monolithic system into separate and isolated components
  - Improves modularity
  - Contains faults
  - Improves reliability

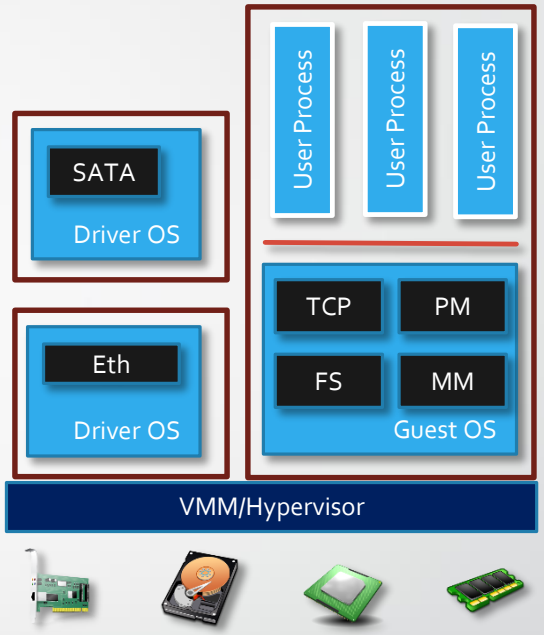
# Existing Approaches

- Decomposition is not a new idea – to understand VirtuOS, consider existing design space
  - Microkernel-based systems (L3/L4, Minix-3, etc.)
  - Retrofit driver protection schemes (Nooks, SUD, MicroDrivers, User-level Drivers & File Systems)
  - VM-based systems (DD/OS, Xen Driver Domains)
  - **VirtuOS** is a **novel design** that occupies a new point in the **microkernel-VM** design space
- Challenge:
  - How to provide **flexible decomposition** & **strong isolation** while retaining **good performance, transparency & compatibility?**

# 1. Flexibility



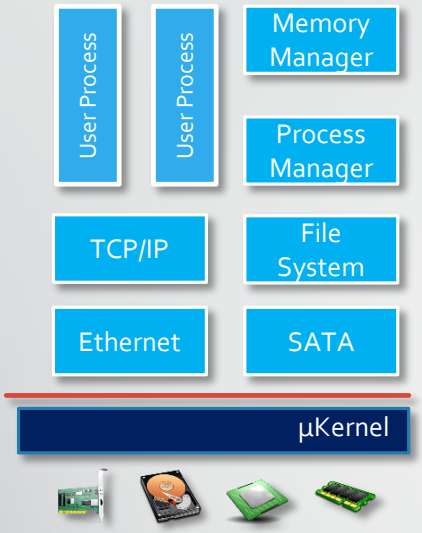



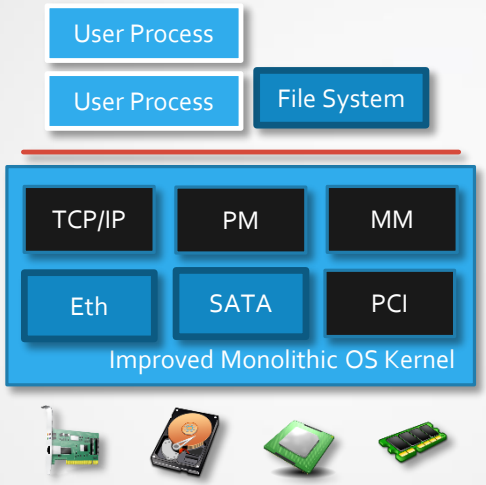



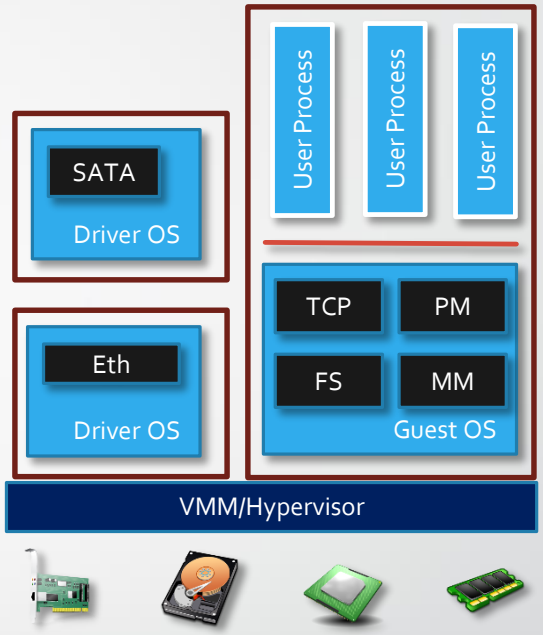

- Provide flexibility
  - Ability to protect low- and high-level components

**VirtuOS:**
Protects low- and high-level components

# 2. Strong Isolation




- Provide strong isolation
  - Protection Domains
  - Privilege Separation
  - Device Protection

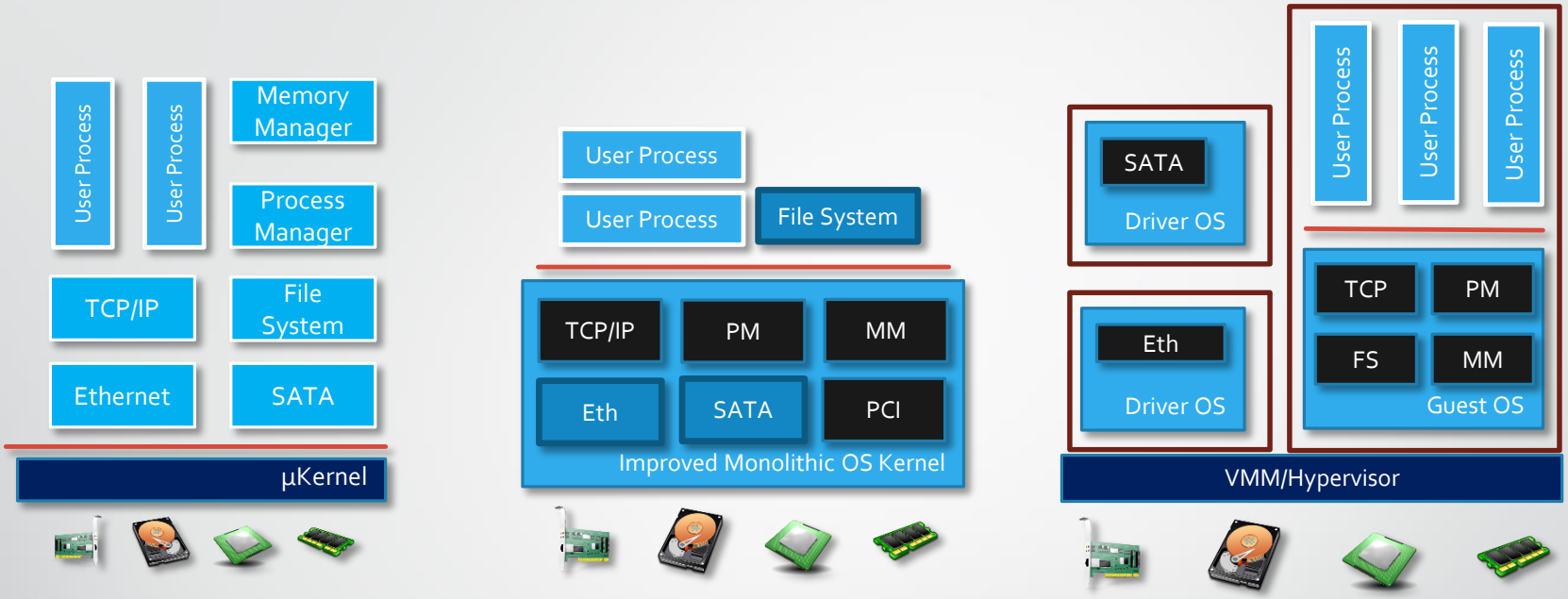
**VirtuOS:**

✓

✓

✓

# 3. Performance

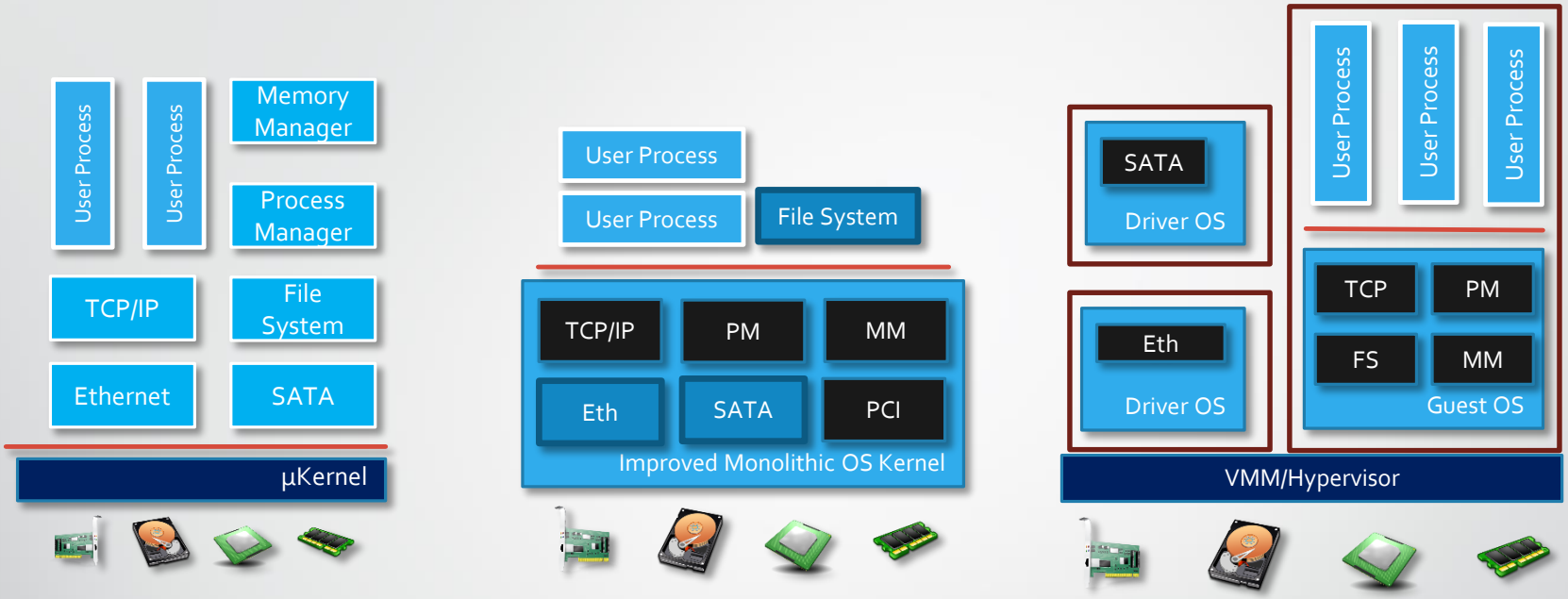



- Retain Good Performance

**VirtuOS:**

Comparable to performance of monolithic system for server workloads

# 4. Compatibility



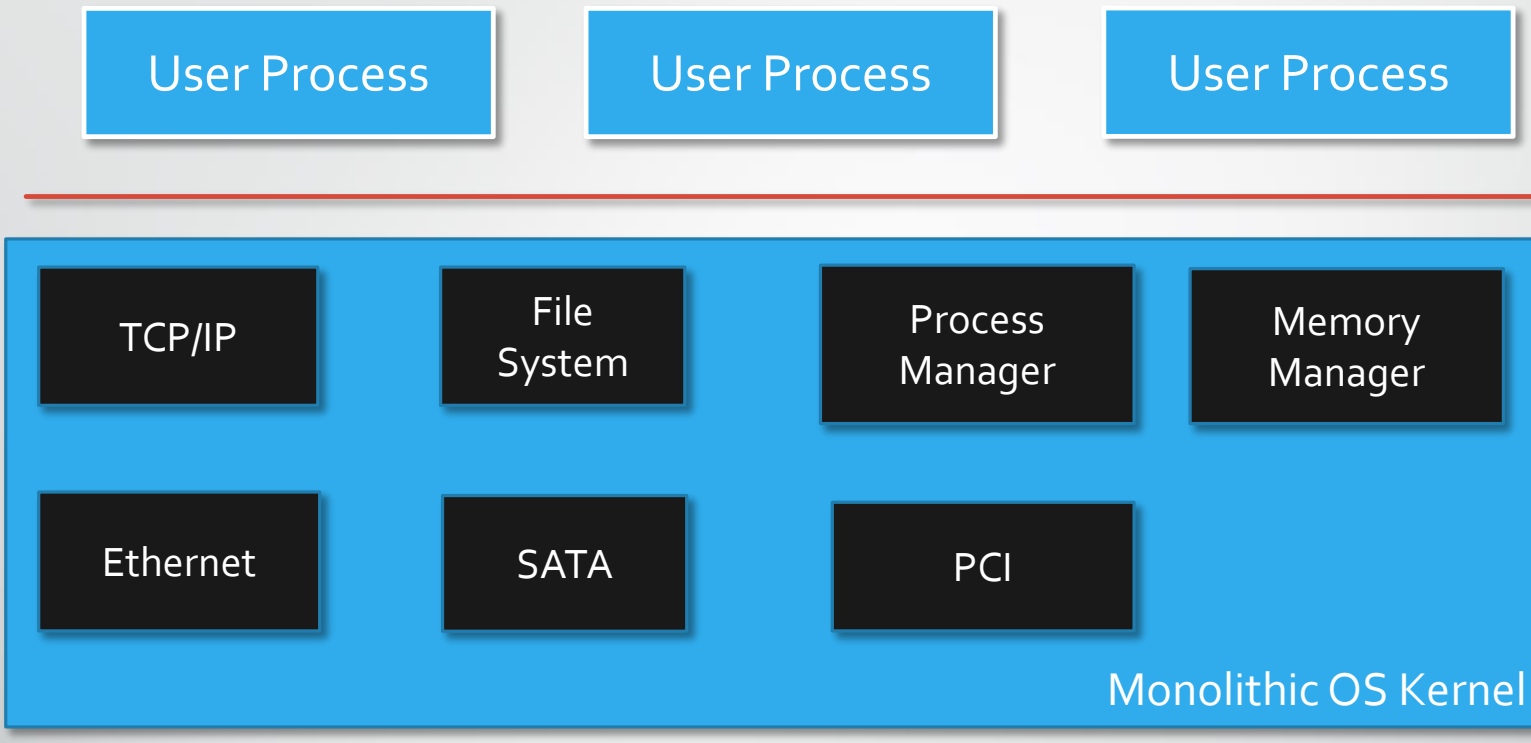

- Compatibility
  - Applications
  - Systems Code

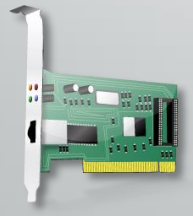
VirtuOS:

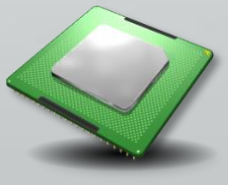
✓

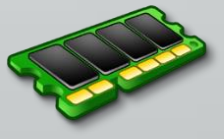
✓

# VirtuOS Architecture

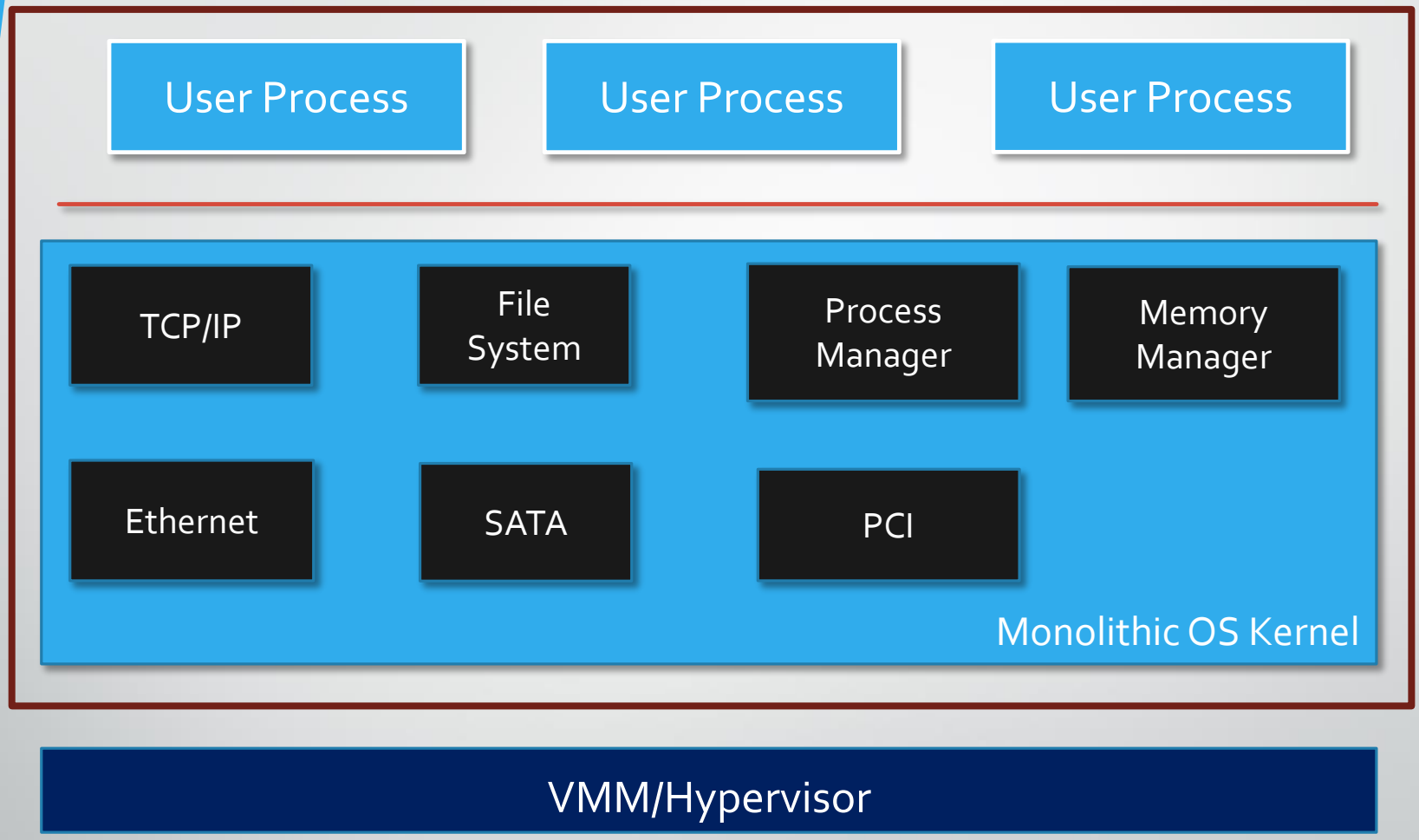


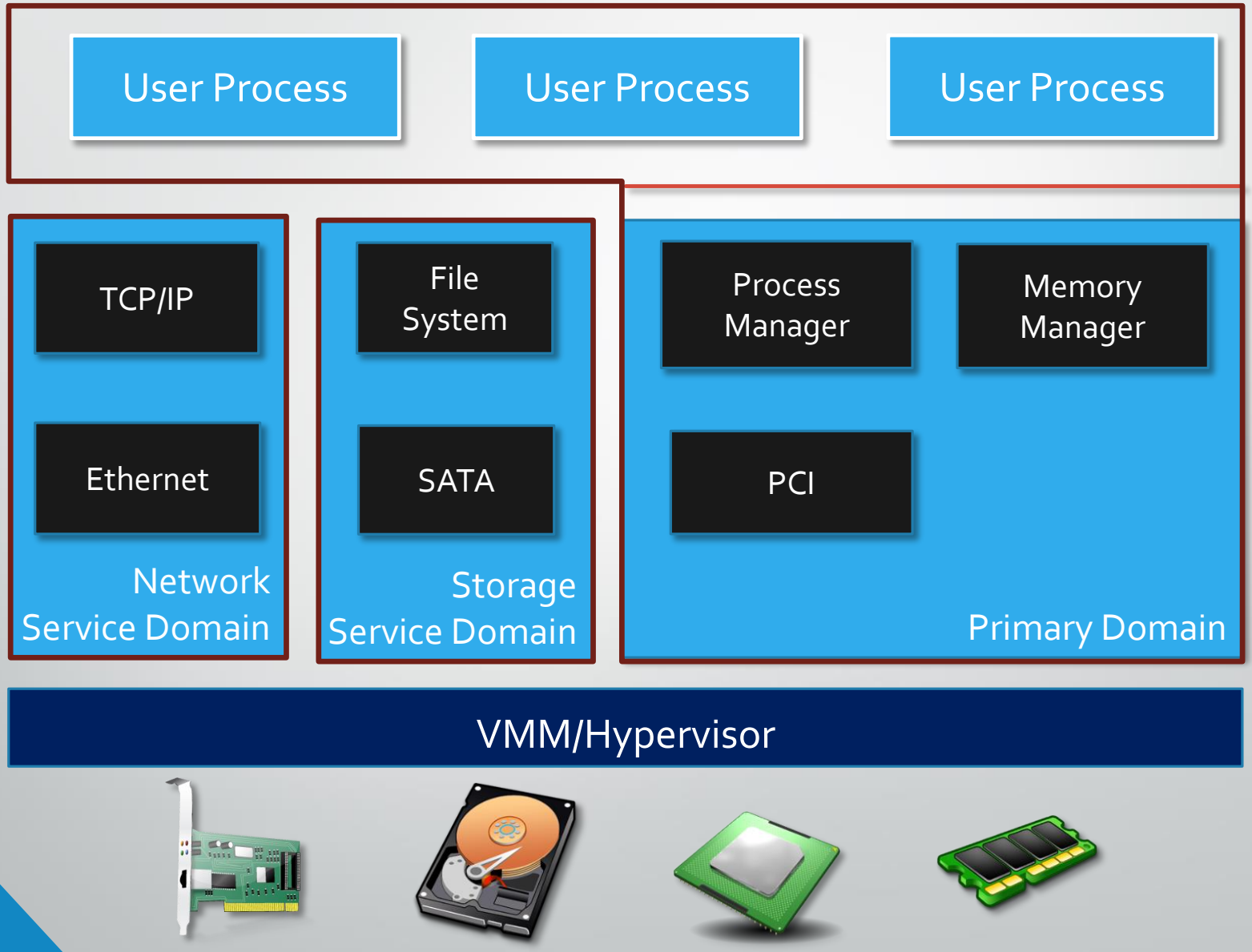




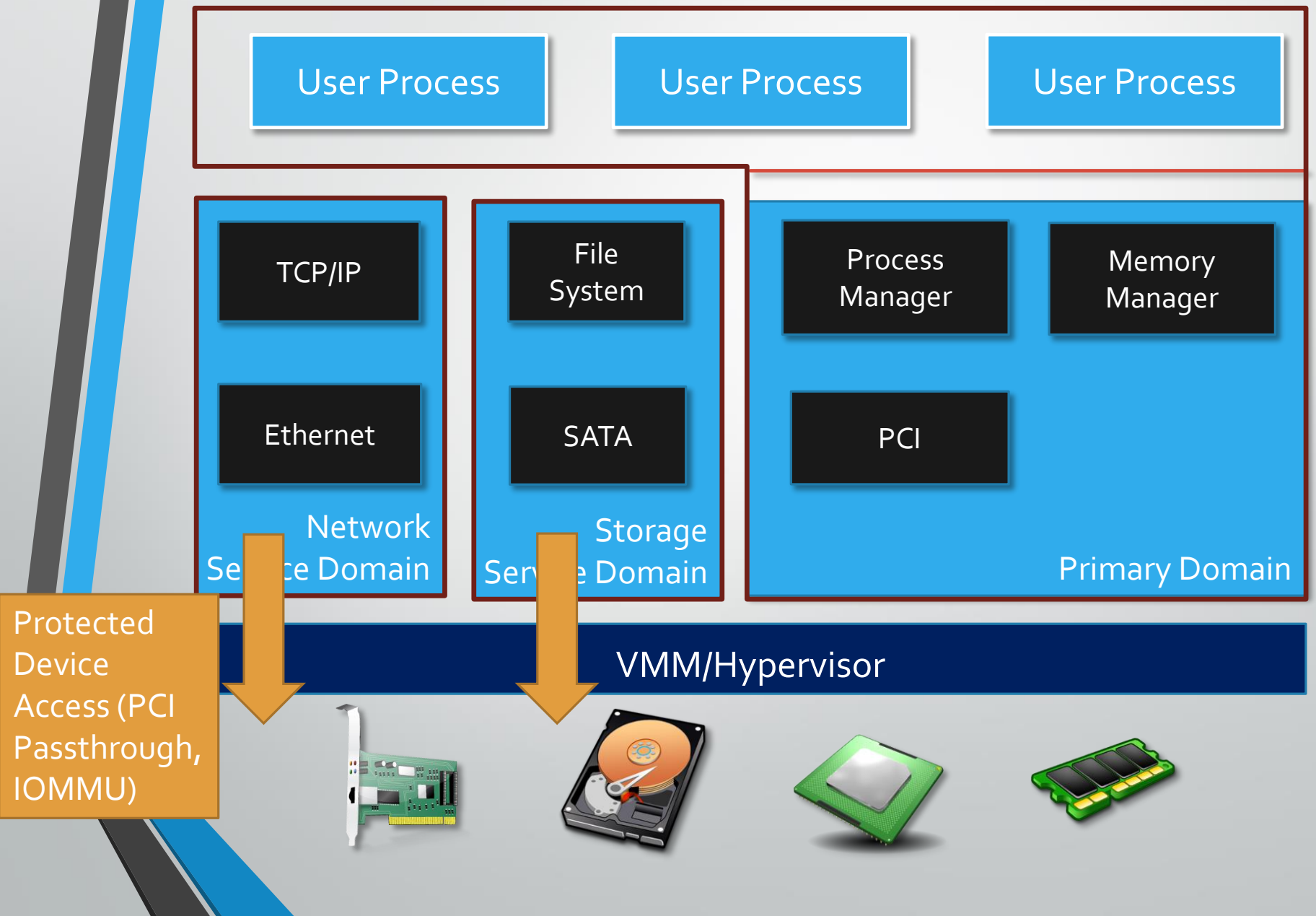
# VirtuOS Architecture



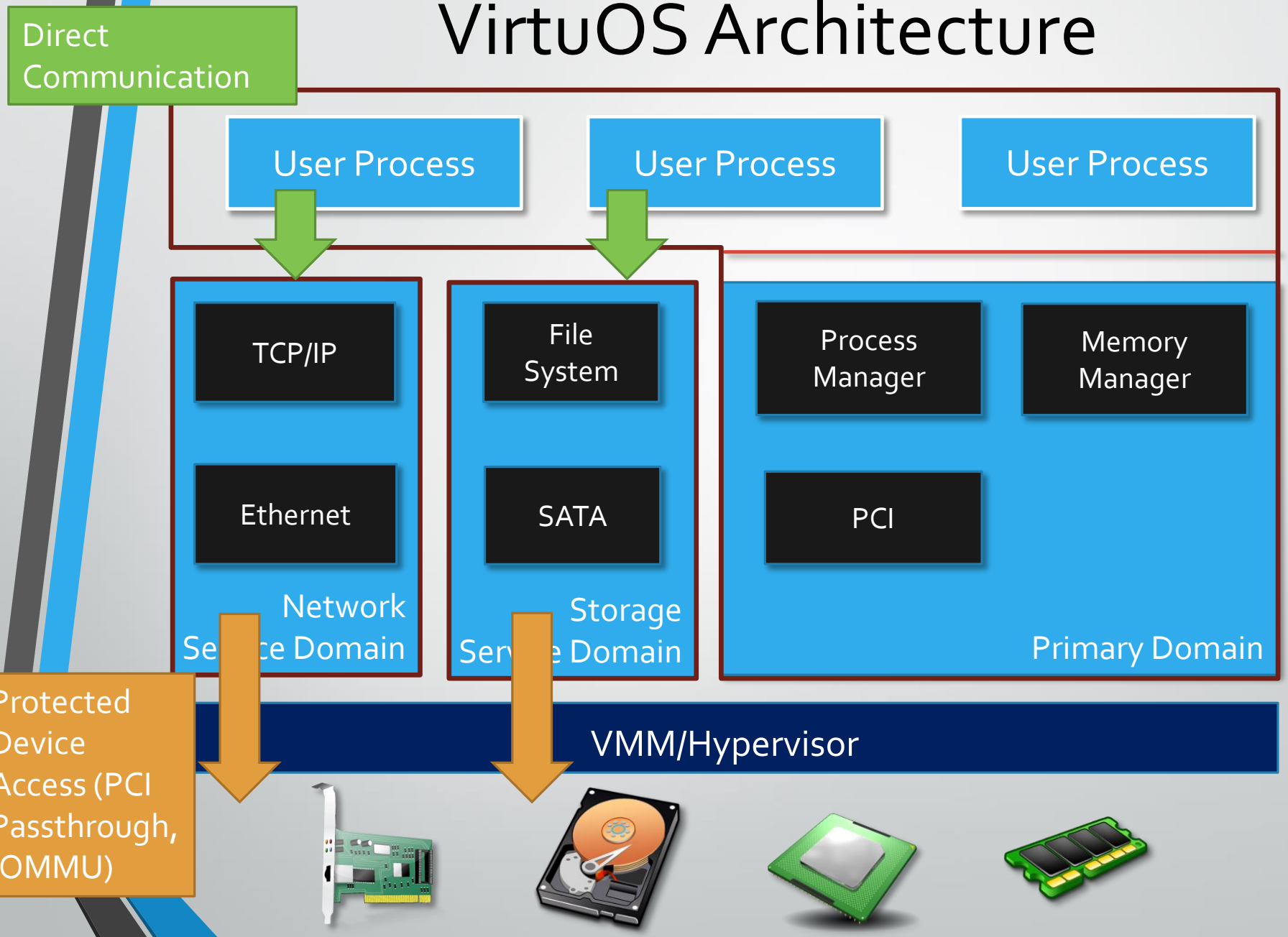
# VirtuOS Architecture



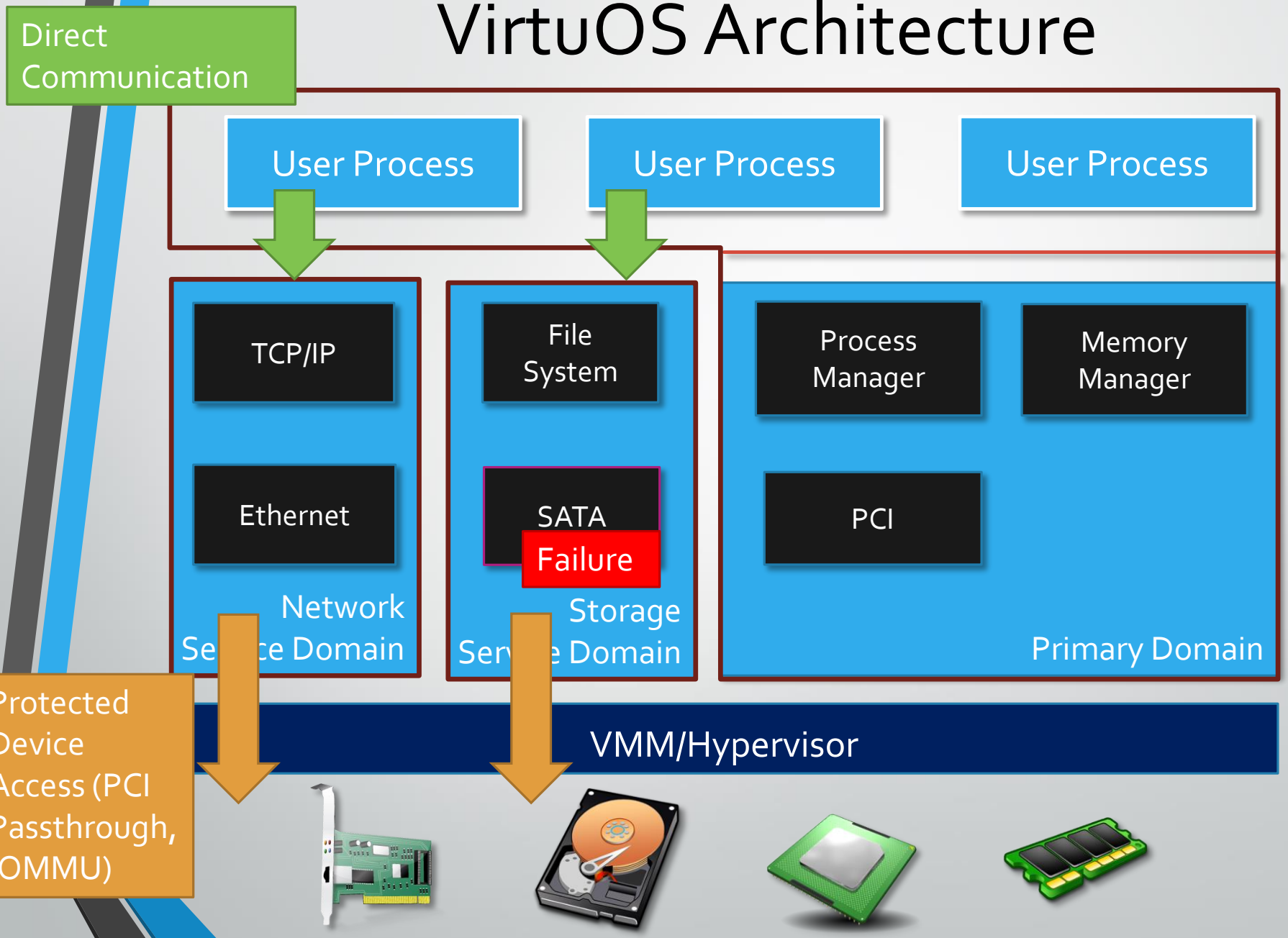
# VirtuOS Architecture



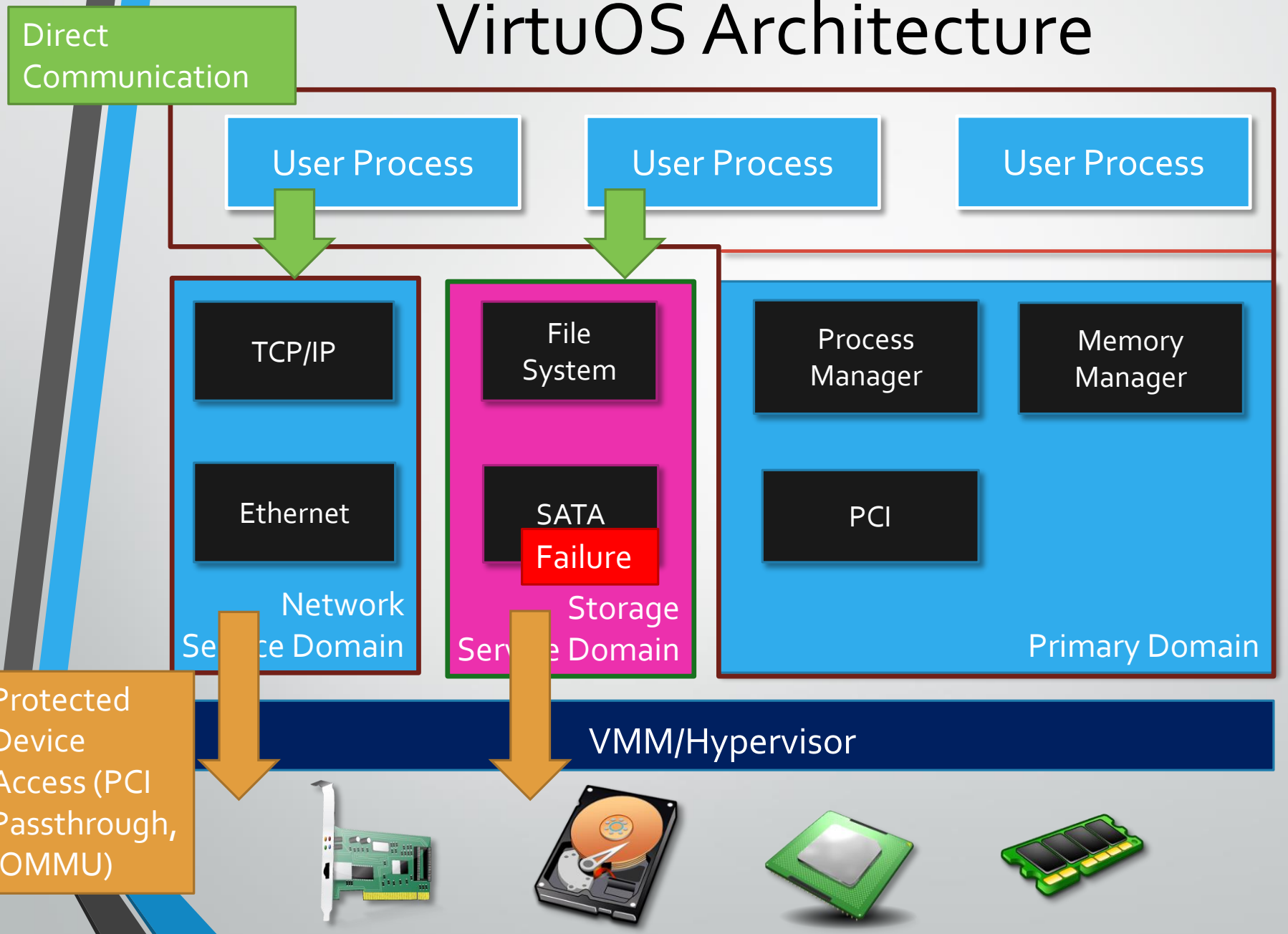
# VirtuOS Architecture



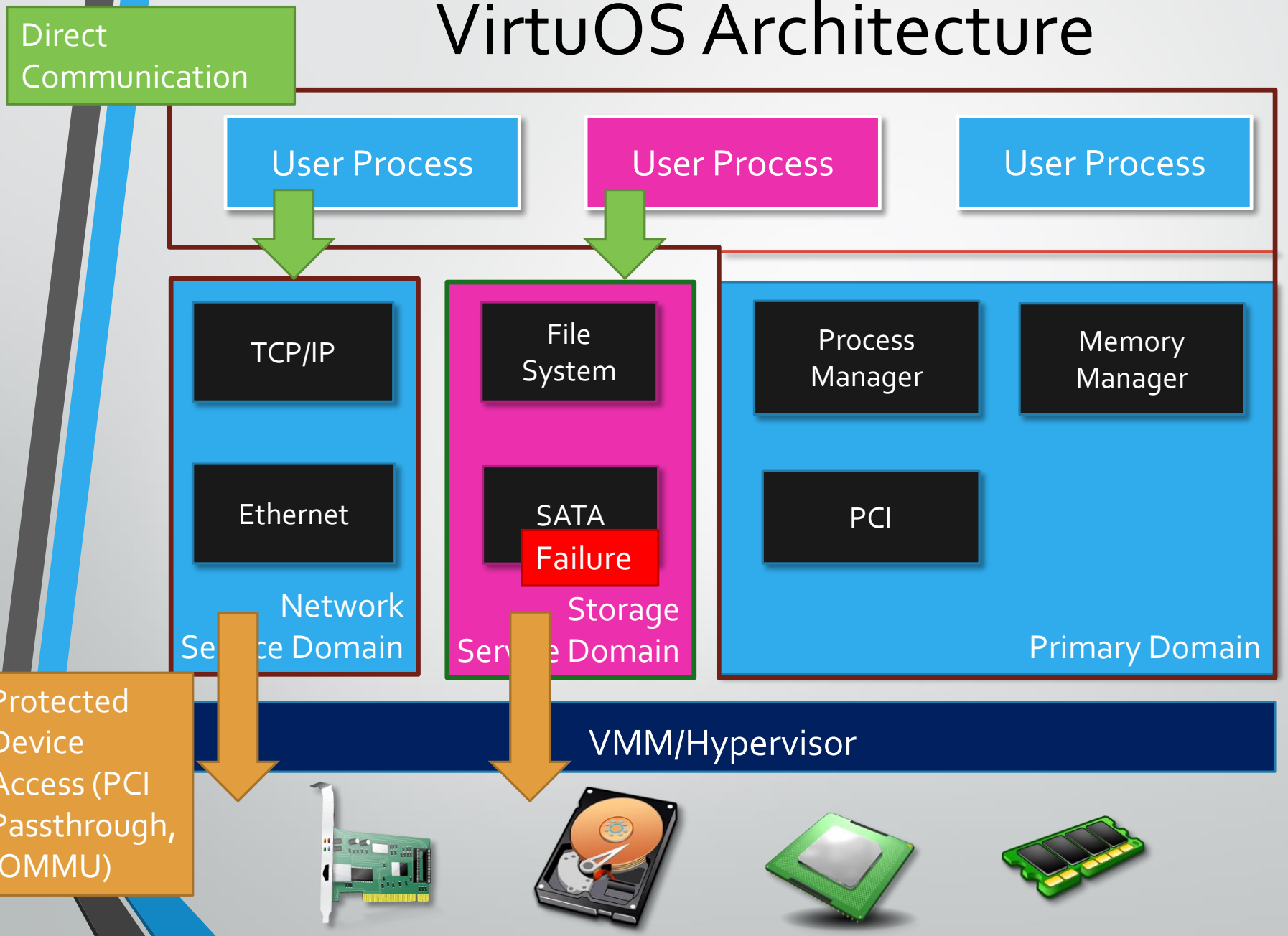
# VirtuOS Architecture



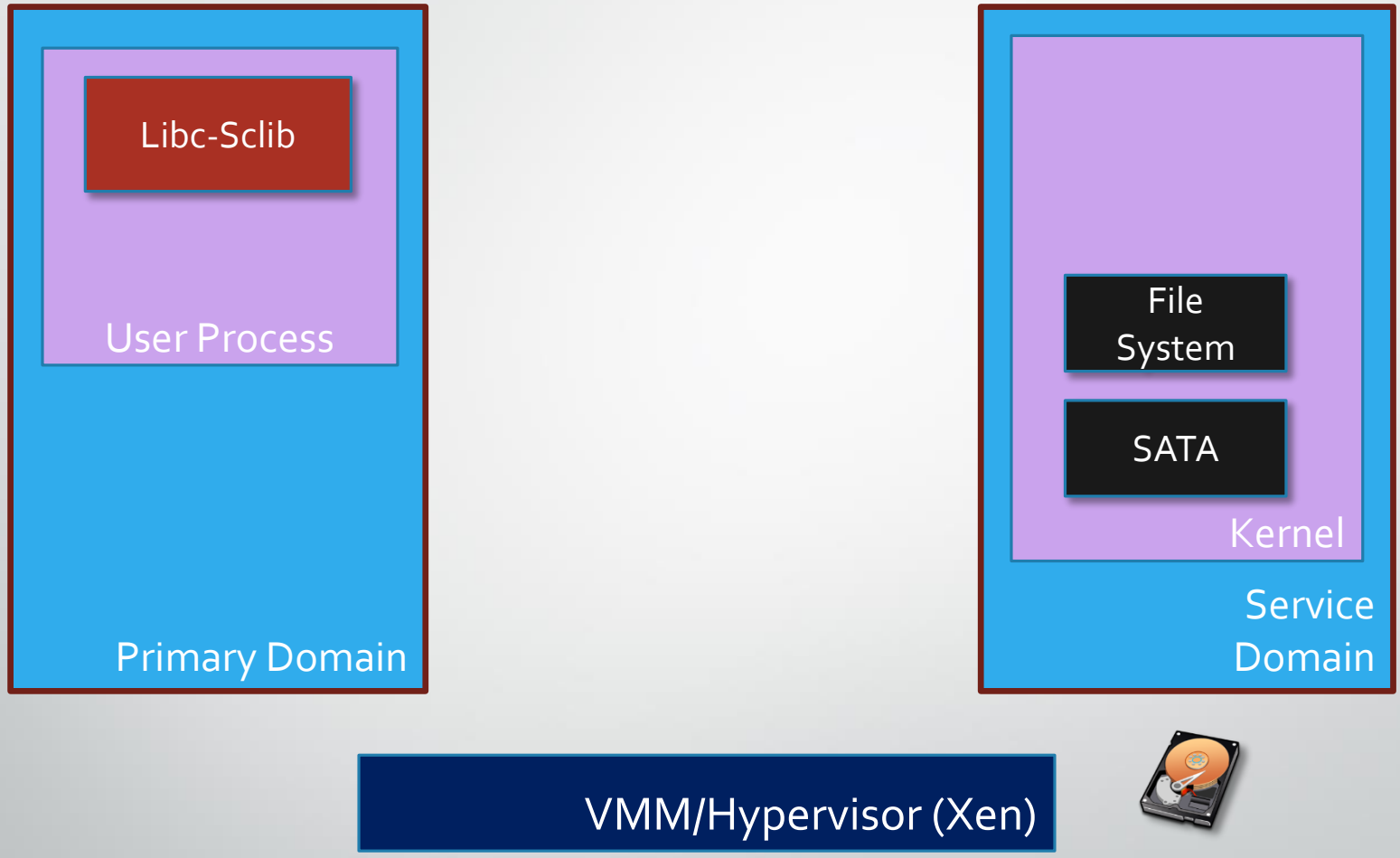
# VirtuOS Architecture



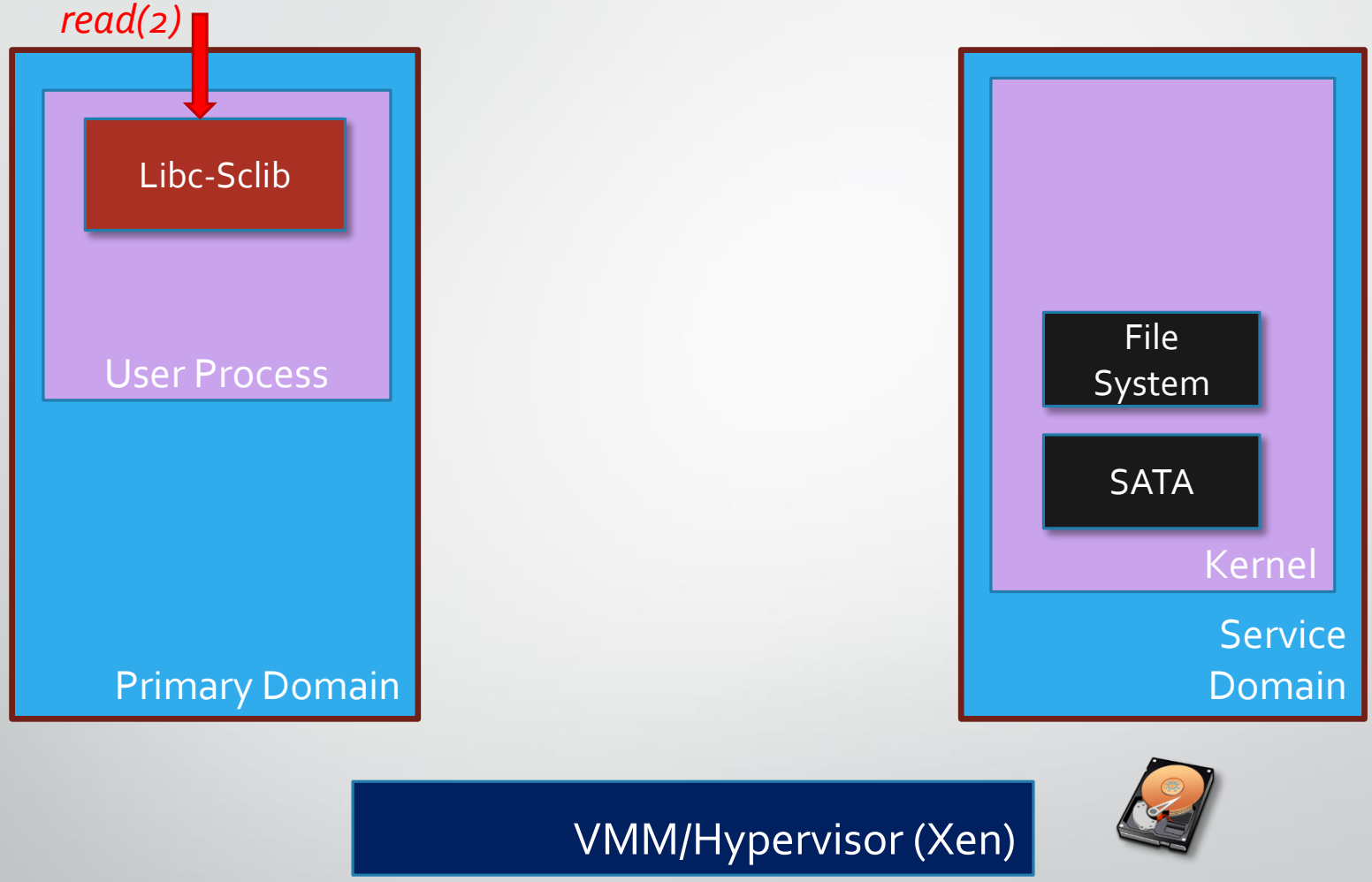
# VirtuOS Architecture



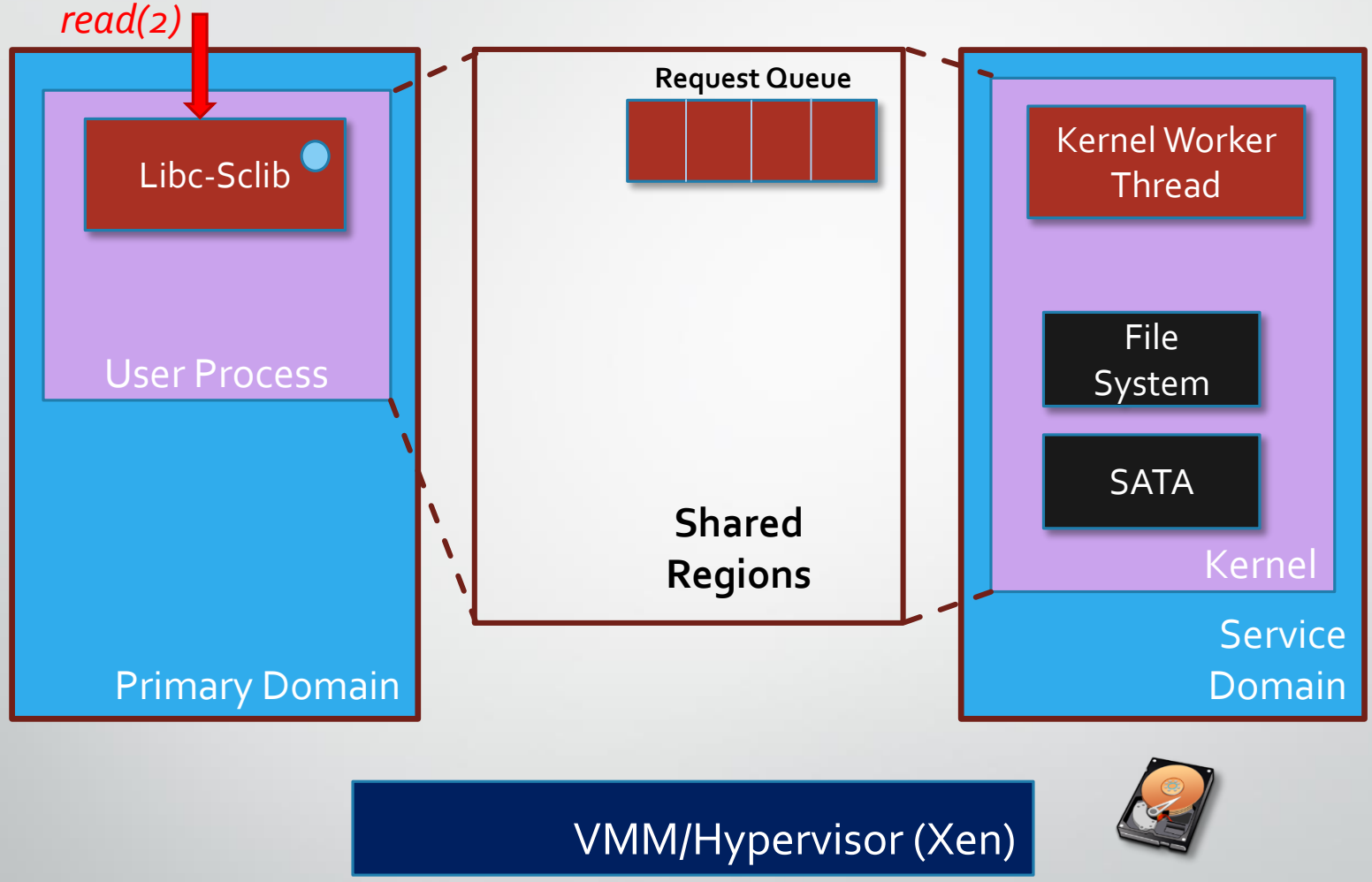
# Implementation: Components



# Implementation: Direct Dispatch

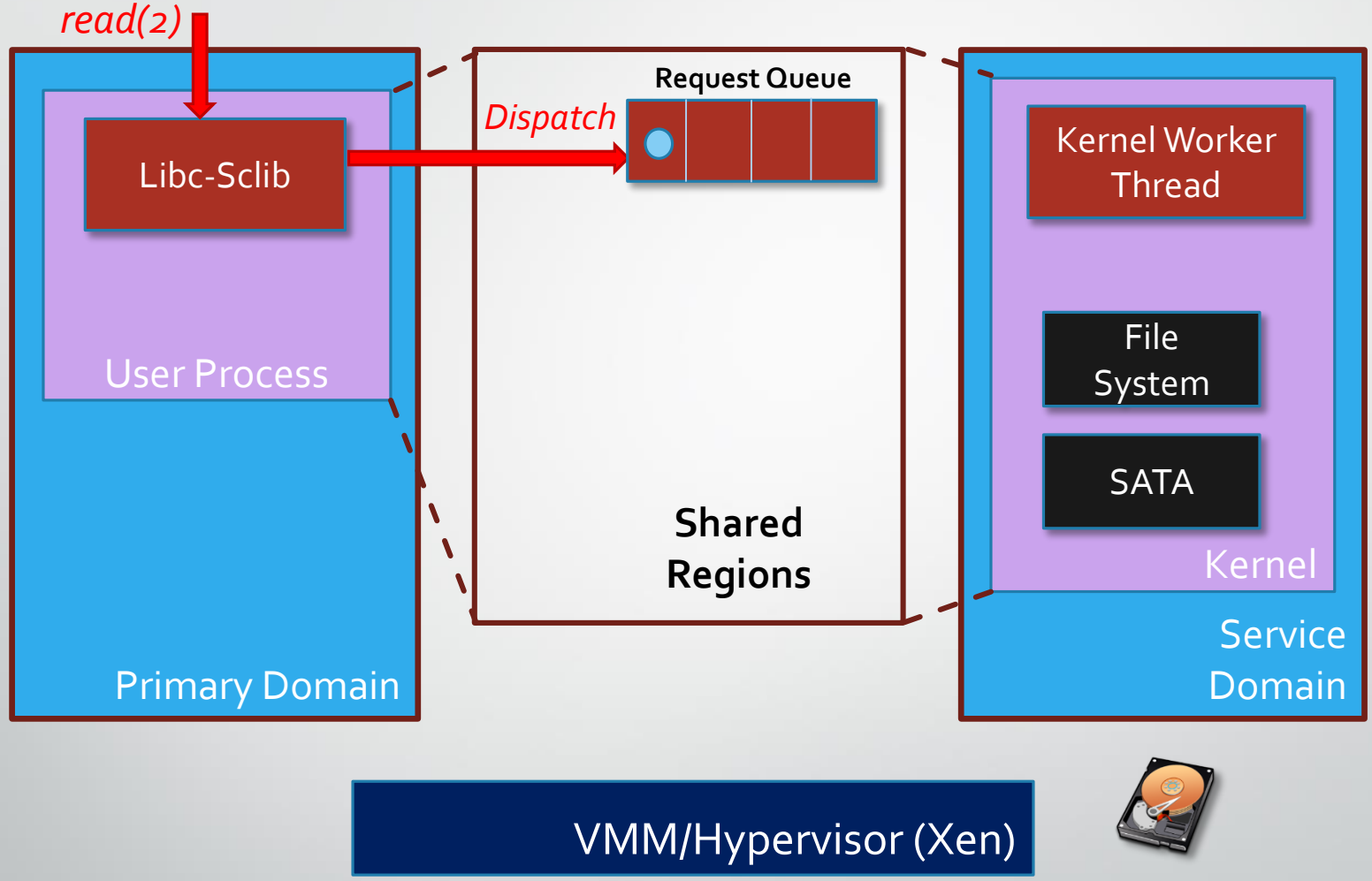


# Implementation: Shared Regions



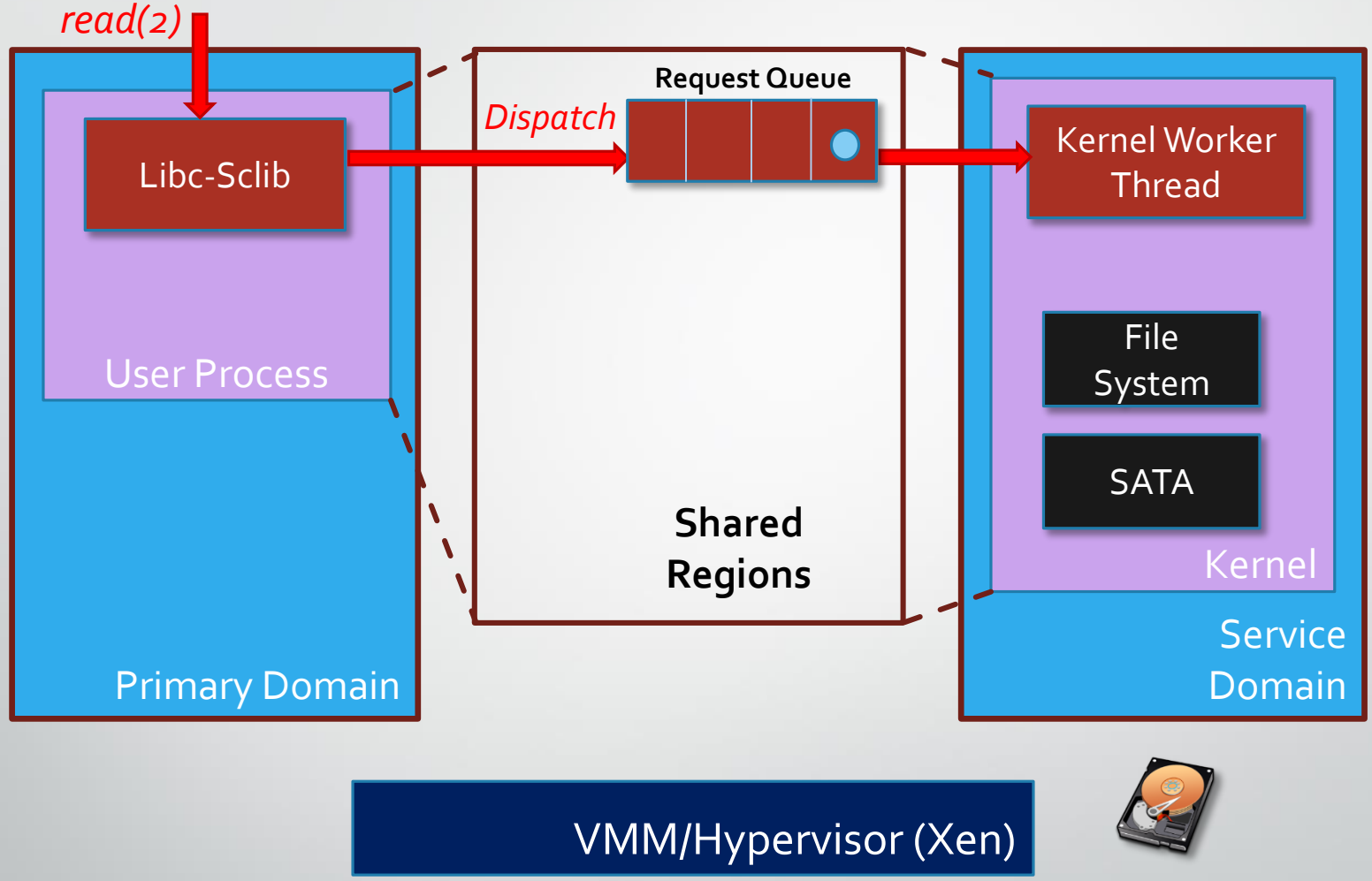

Exceptionless protocol. Shared pages for request data, lock-free request queues.

# Implementation: Shared Regions



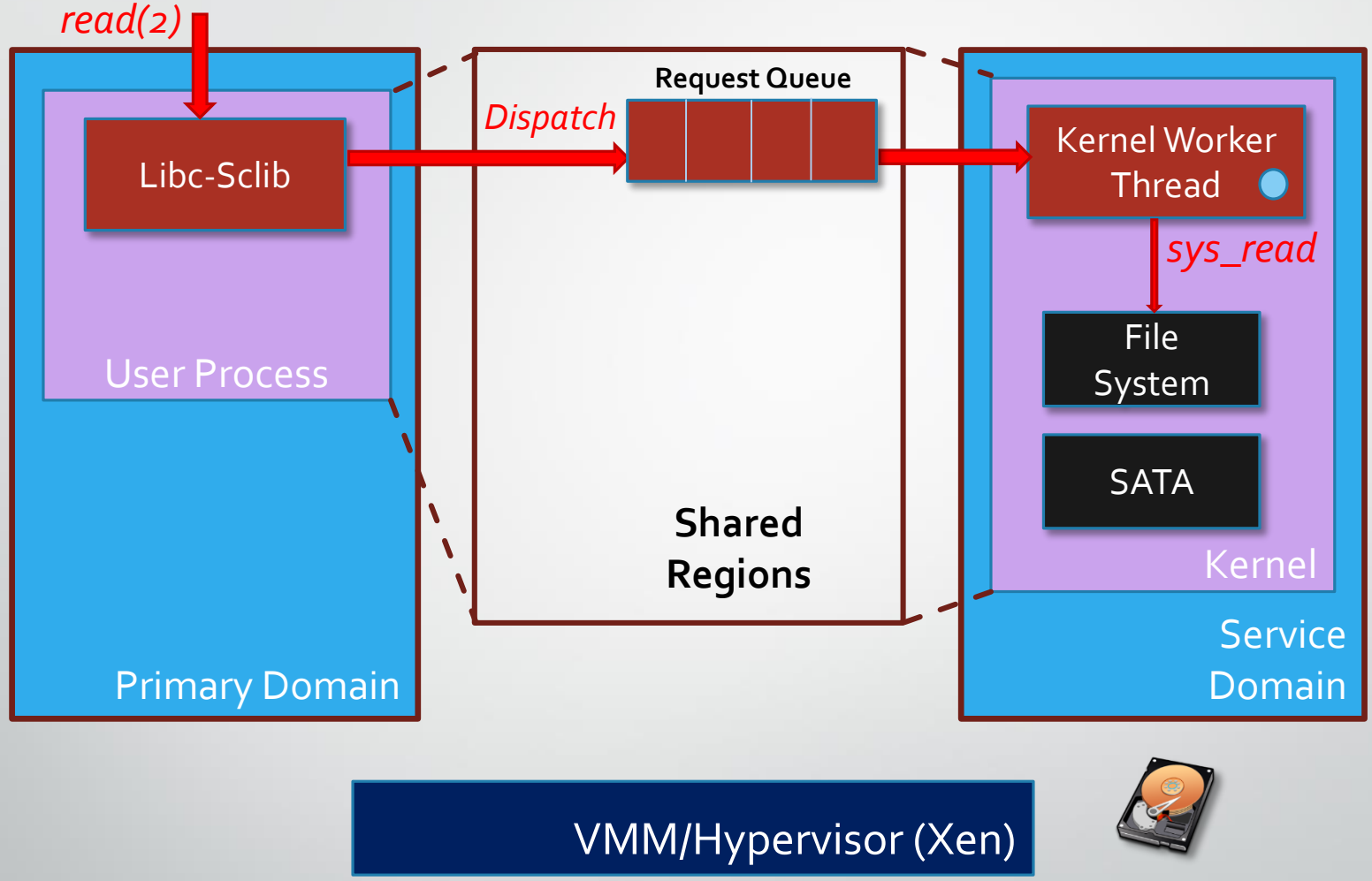

Exceptionless protocol. Shared pages for request data, lock-free request queues.

# Implementation: Shared Regions



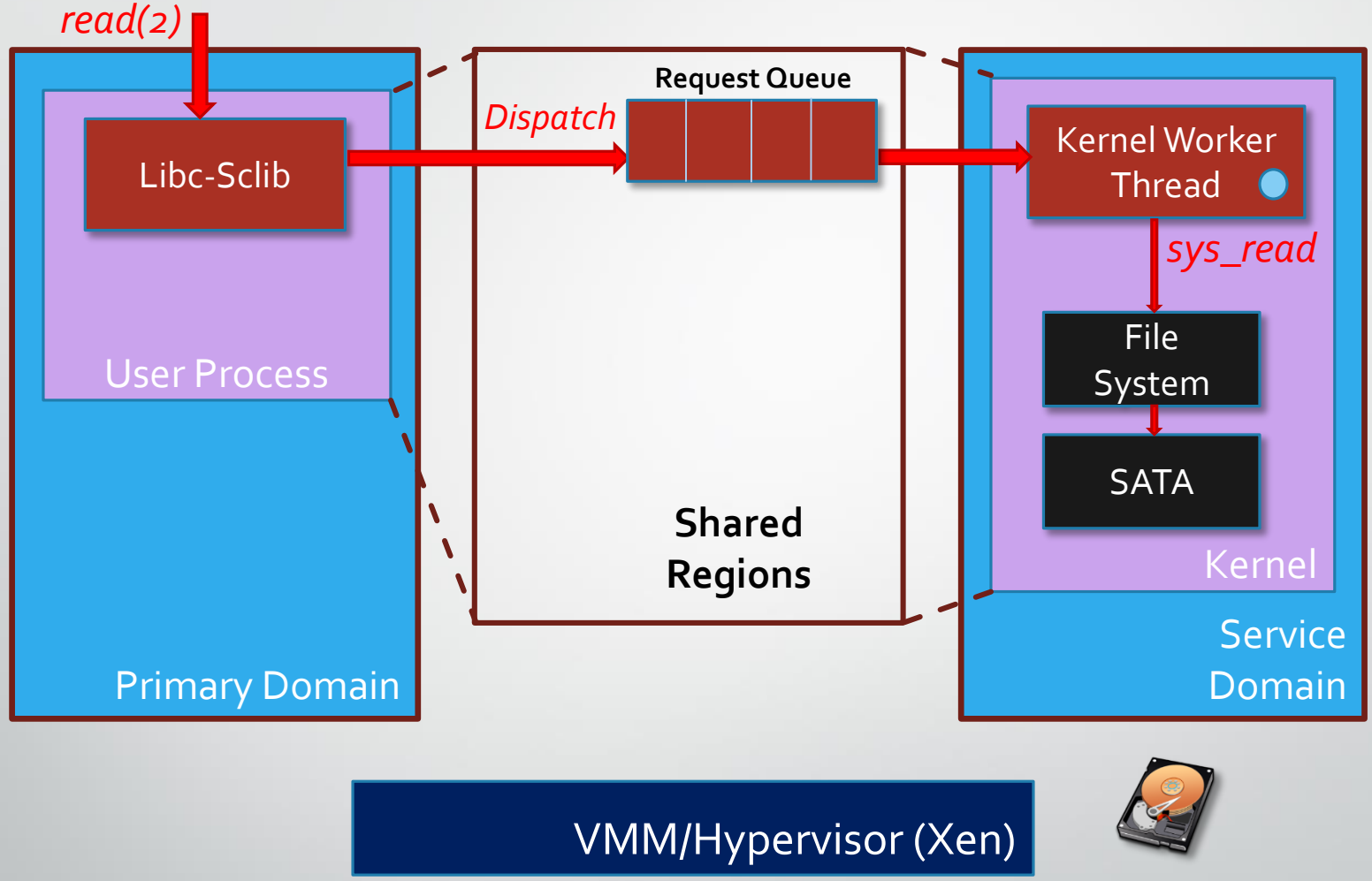

Exceptionless protocol. Shared pages for request data, lock-free request queues.

# Implementation: Shared Regions



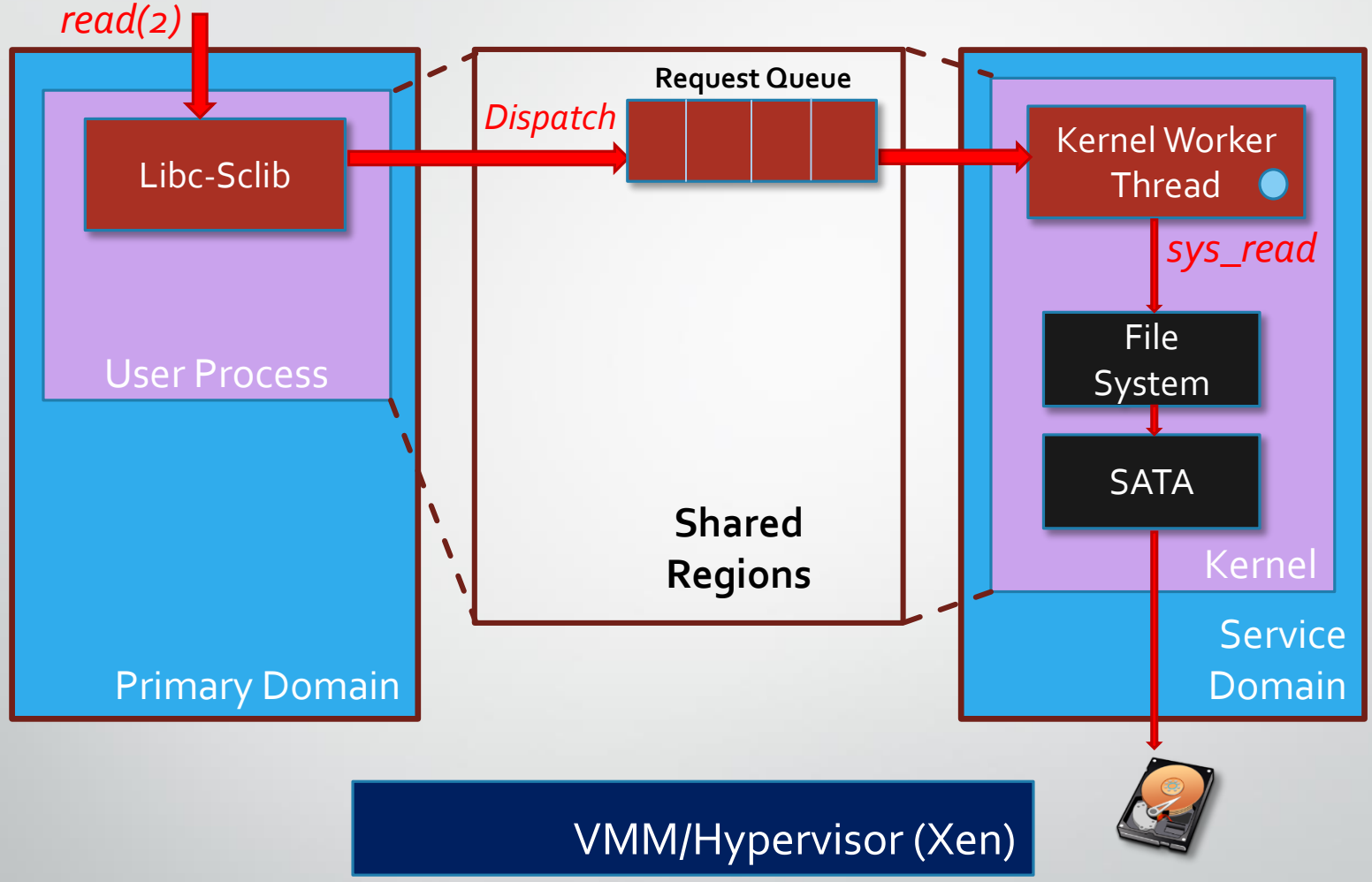

Exceptionless protocol. Shared pages for request data, lock-free request queues.

# Implementation: Shared Regions



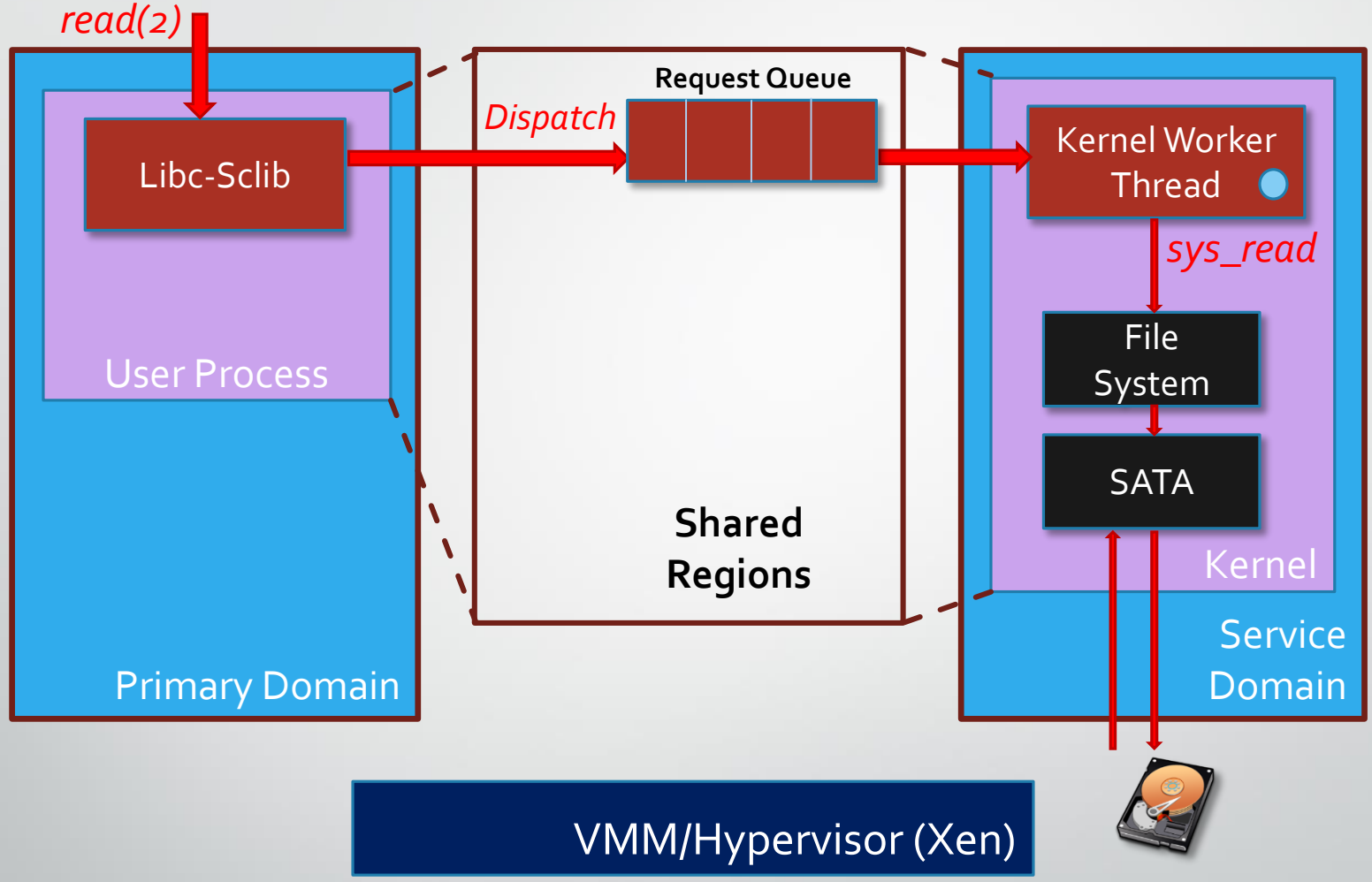

Exceptionless protocol. Shared pages for request data, lock-free request queues.

# Implementation: Shared Regions



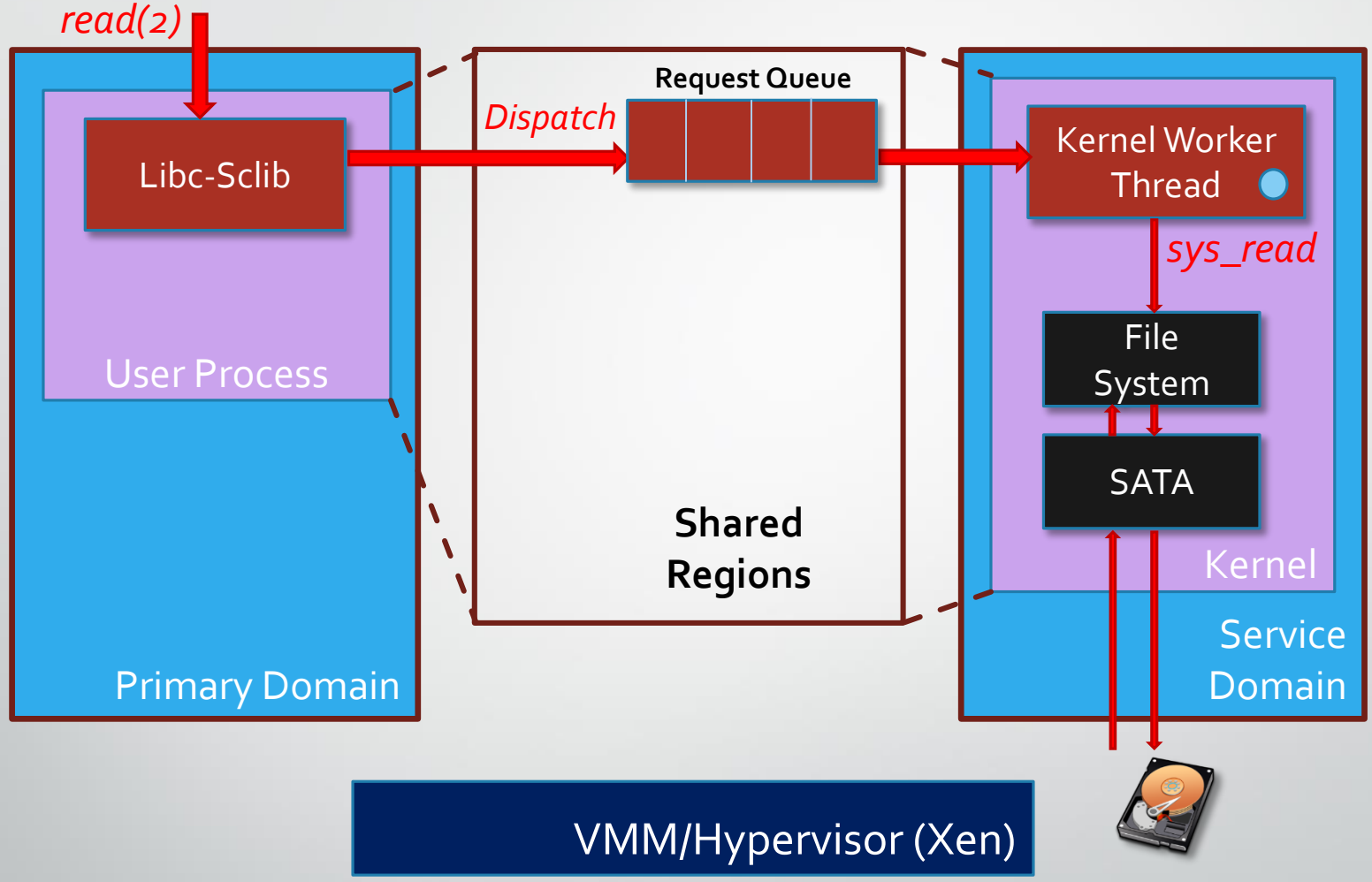

Exceptionless protocol. Shared pages for request data, lock-free request queues.

# Implementation: Shared Regions



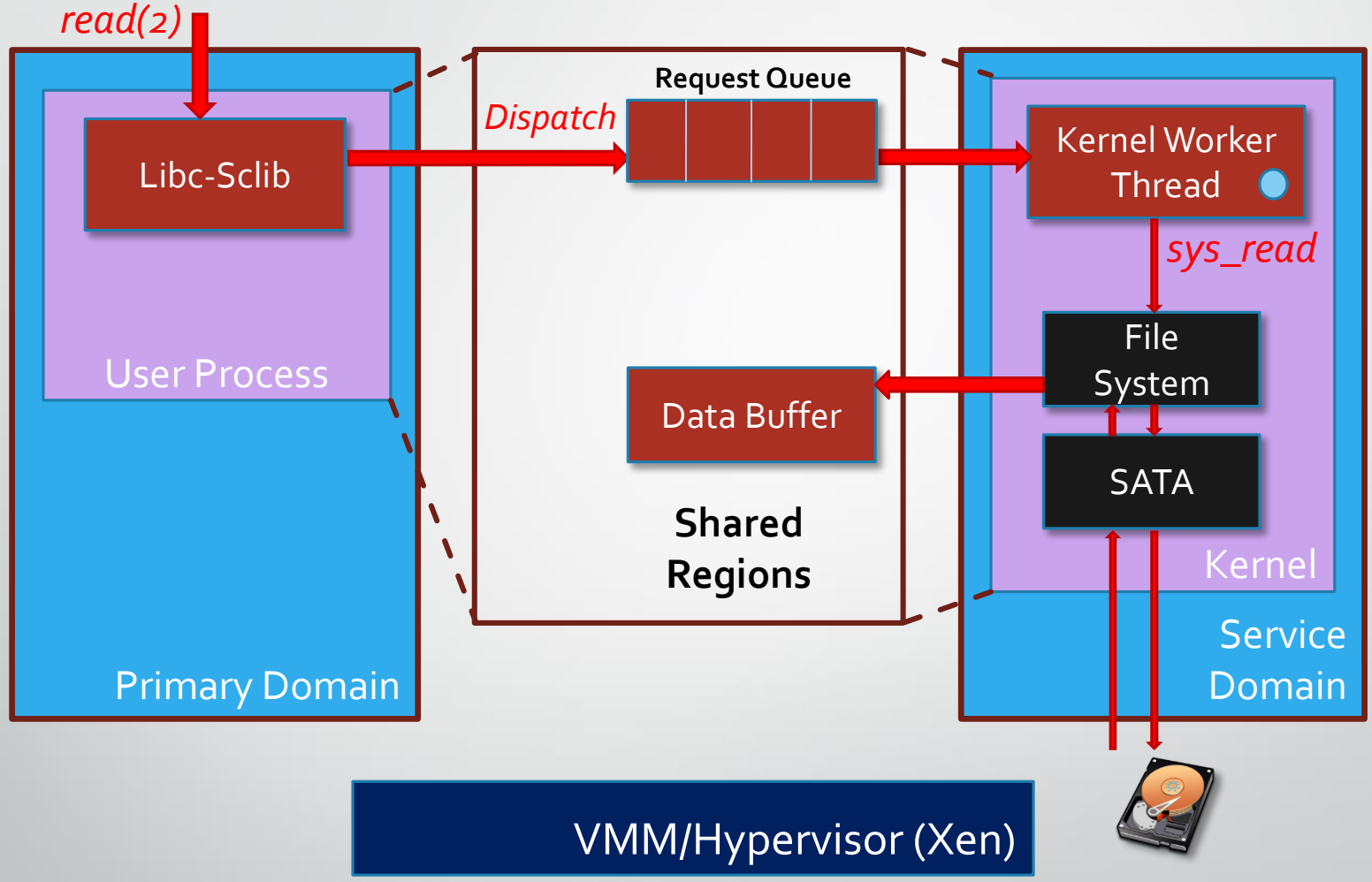

Exceptionless protocol. Shared pages for request data, lock-free request queues.

# Implementation: Shared Regions



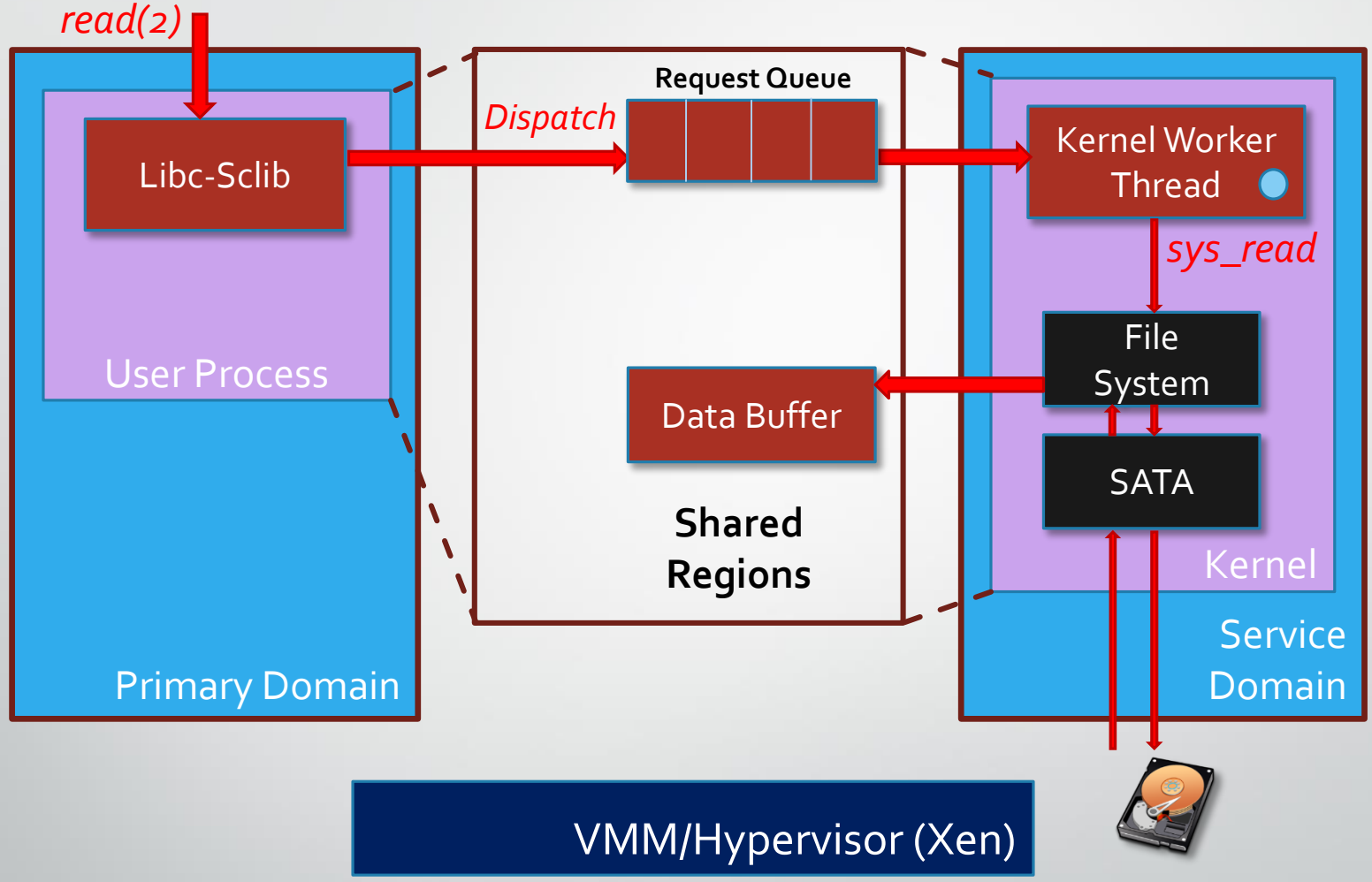

Exceptionless protocol. Shared pages for request data, lock-free request queues.

# Implementation: Shared Regions



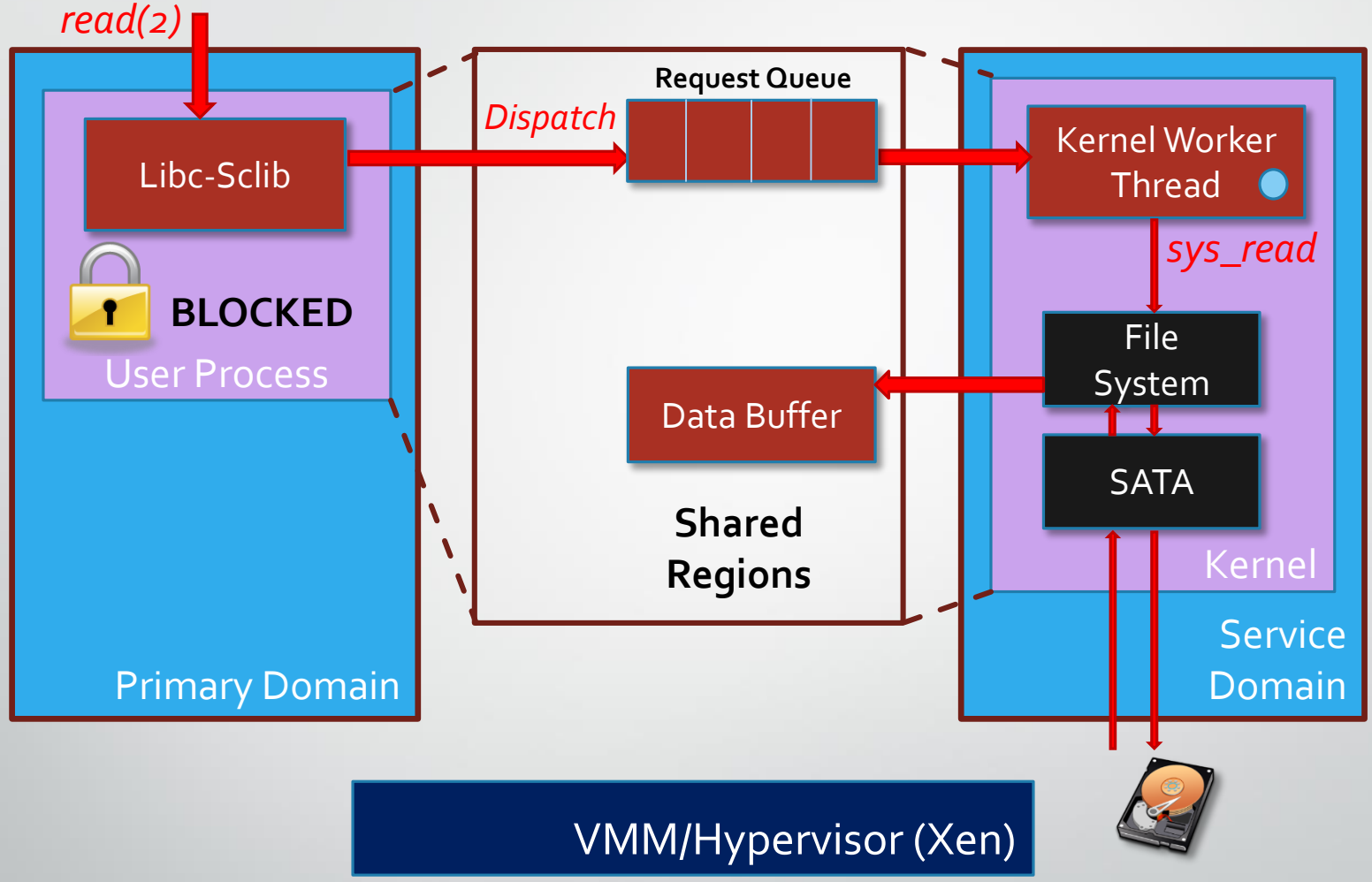

Exceptionless protocol. Shared pages for request data, lock-free request queues.

# Implementation: Threading



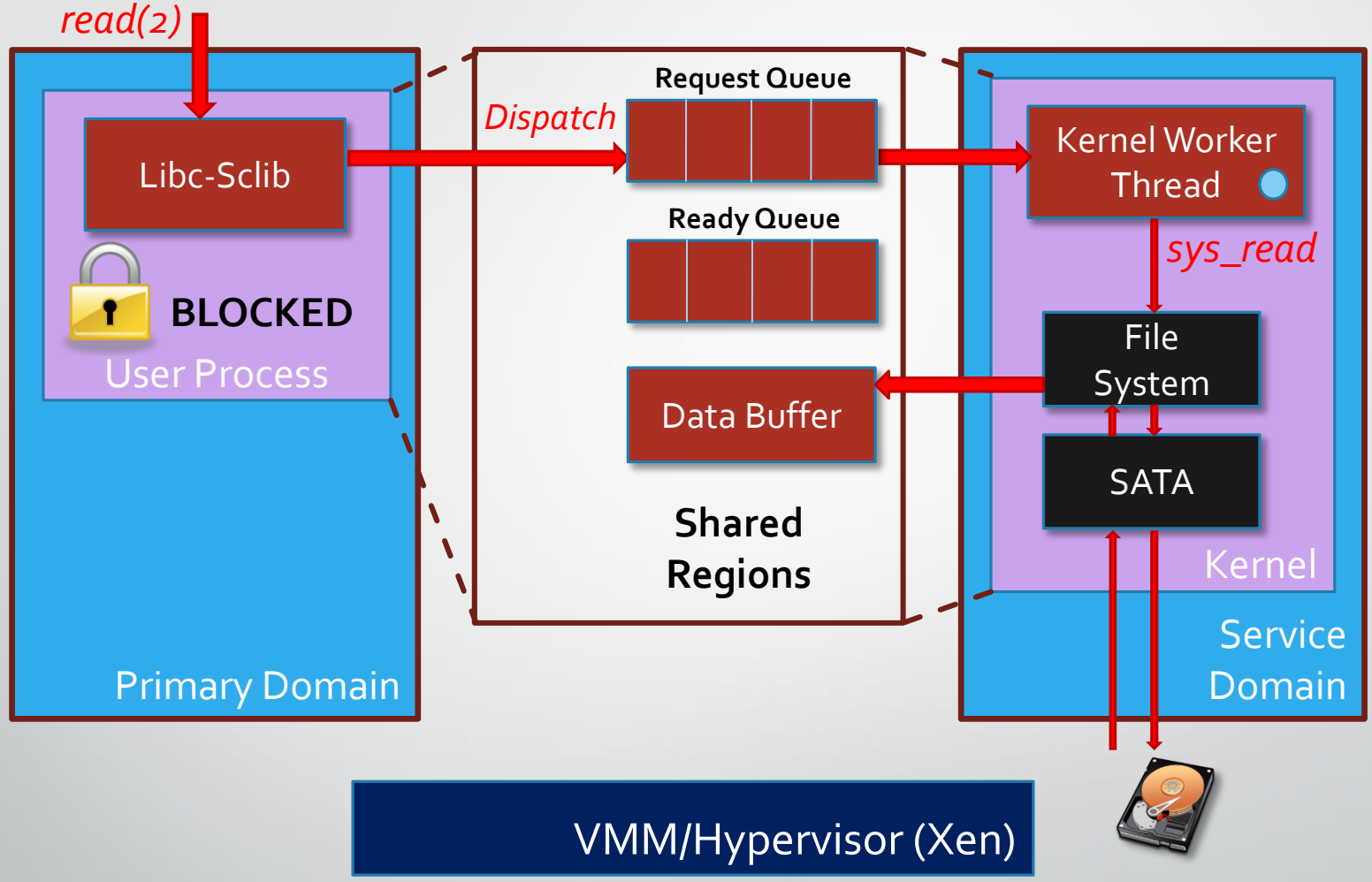

User process, context switch and M:N threading library.

# Implementation: Threading



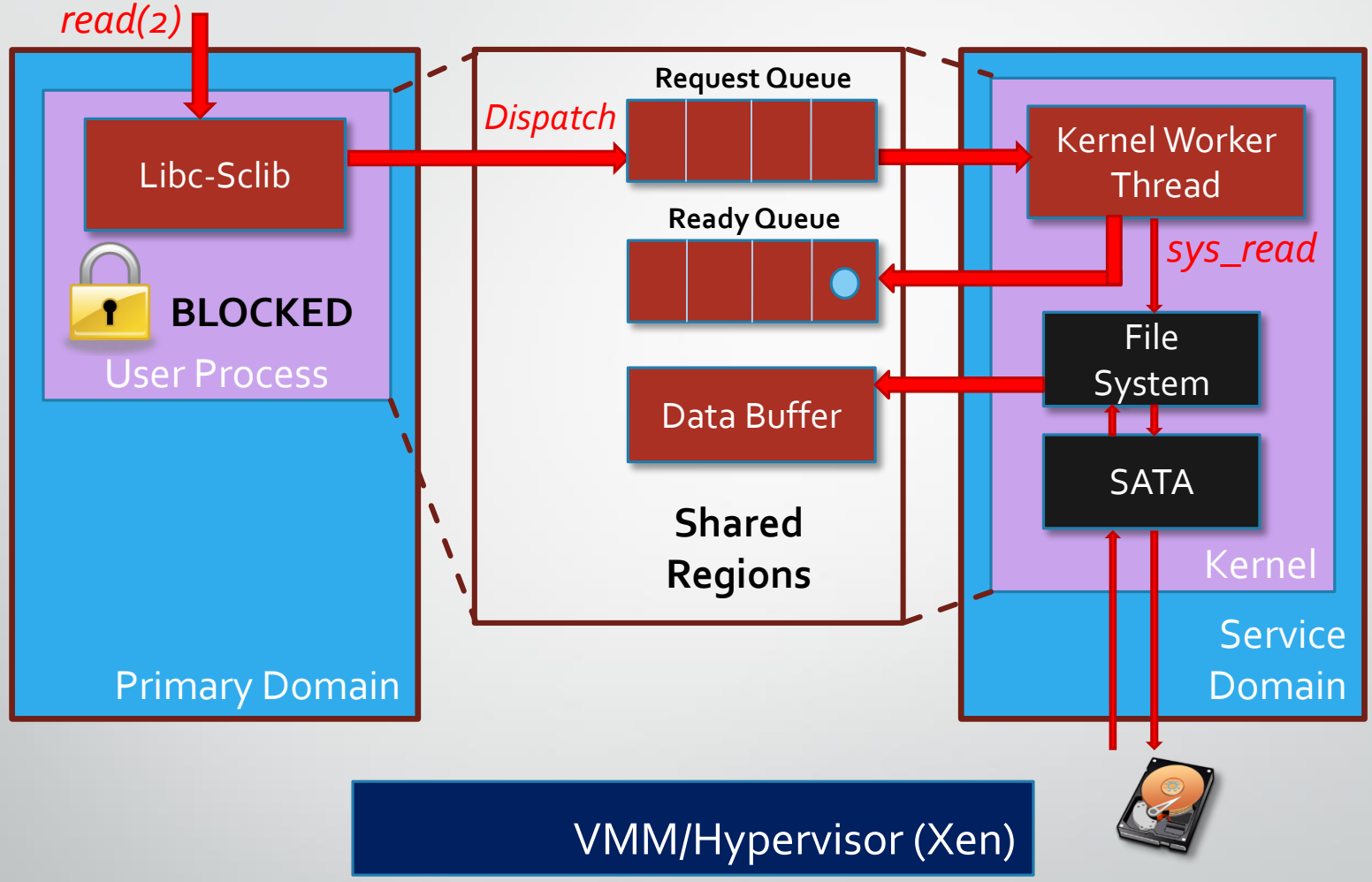

User process, context switch and M:N threading library.

# Implementation: Notification



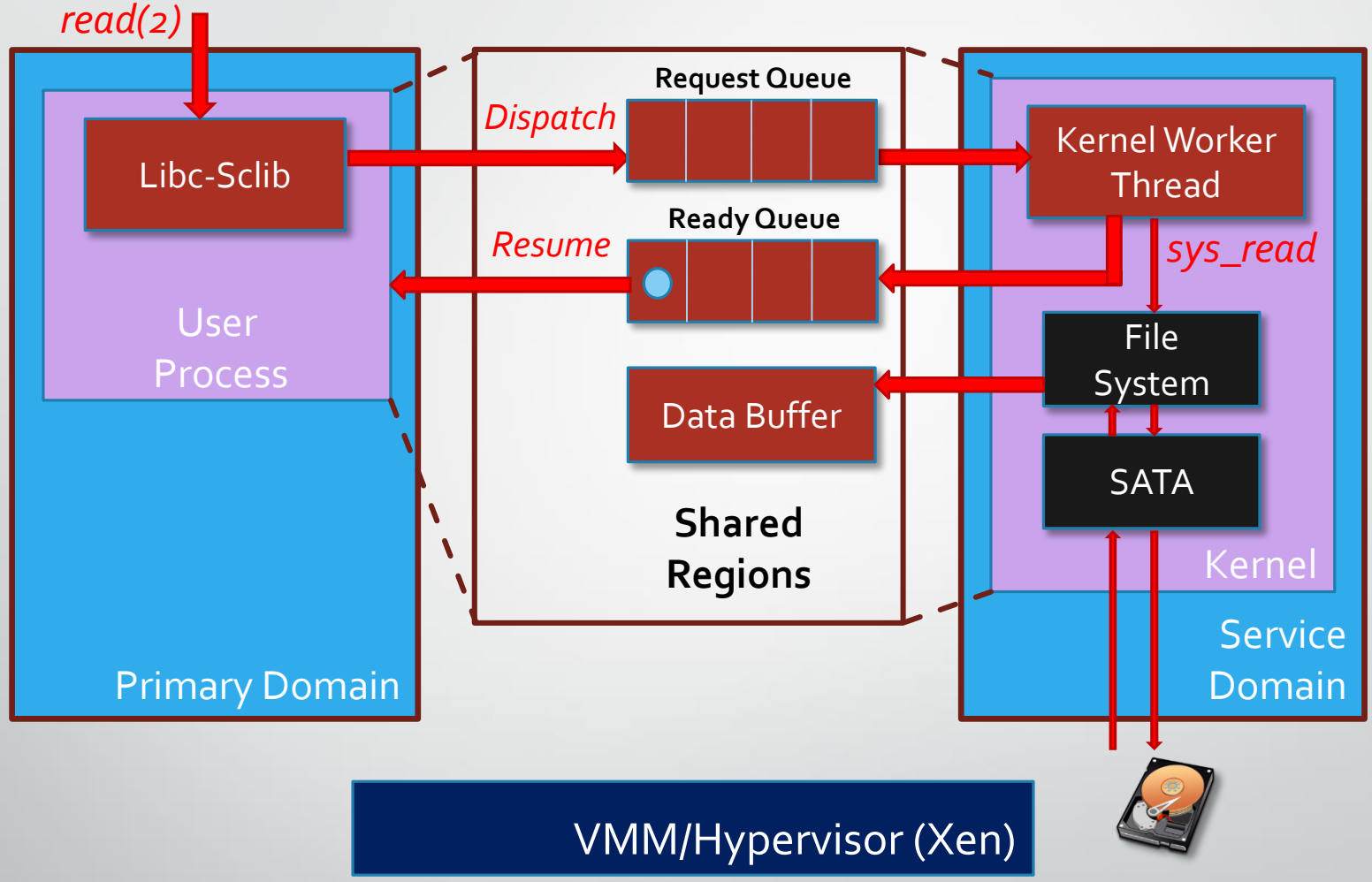

Shared lock-free ready queue. Service domains directly access process's ready queue.

# Implementation: Notification



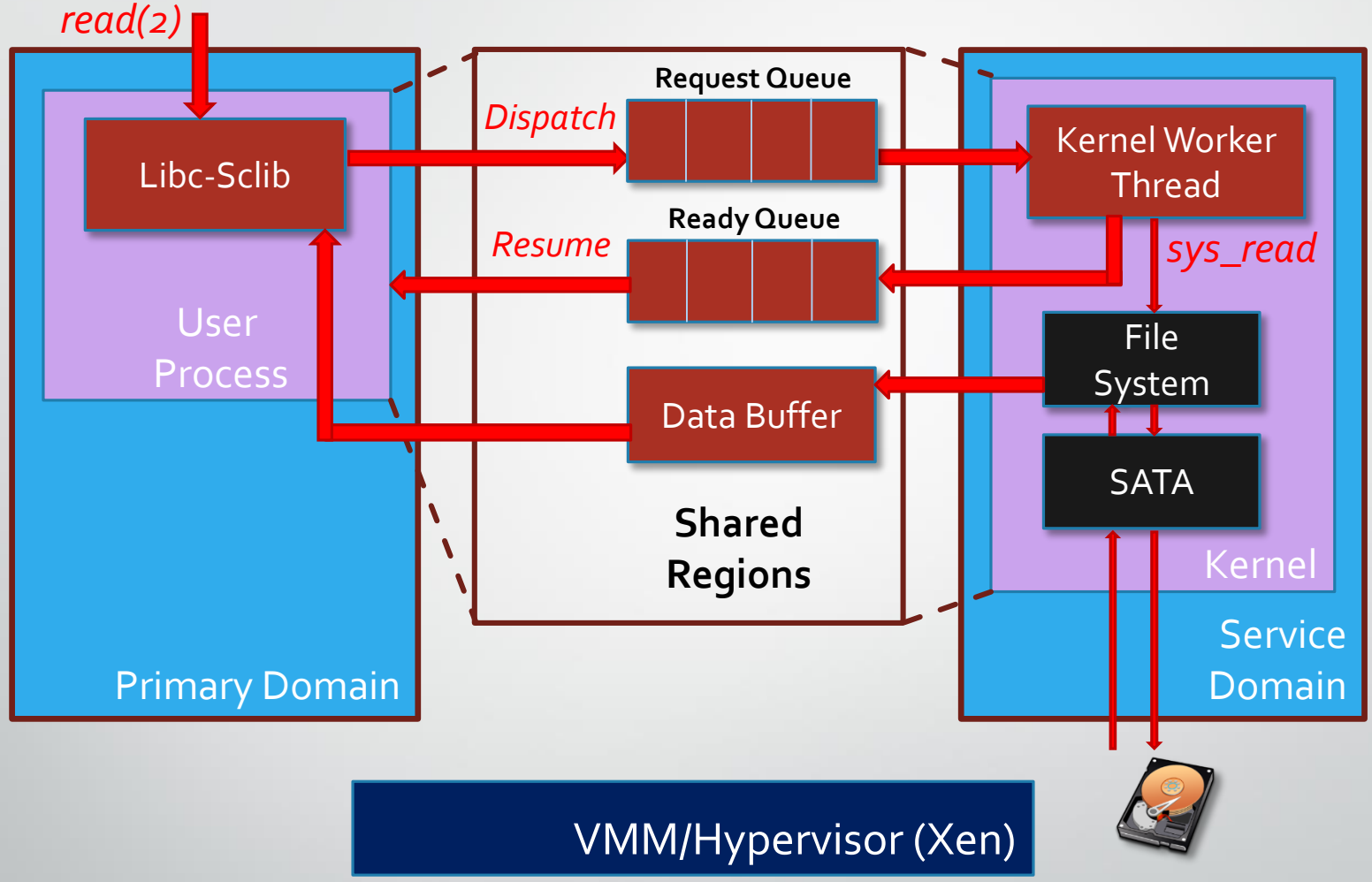

Shared lock-free ready queue. Service domains directly access process's ready queue.

# Implementation: Completion



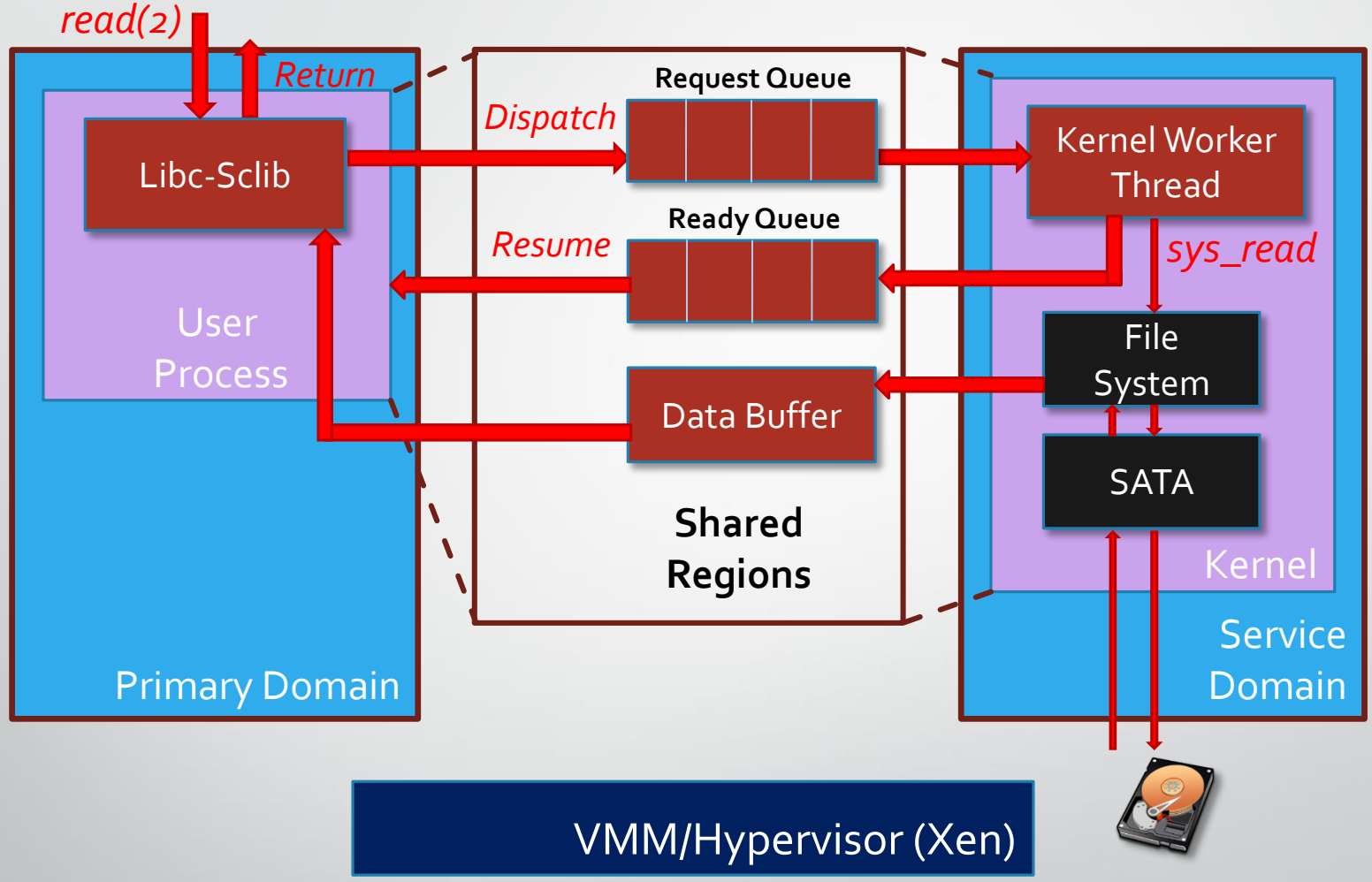

User thread is resumed, data copied, and call is returned.

# Implementation: Completion



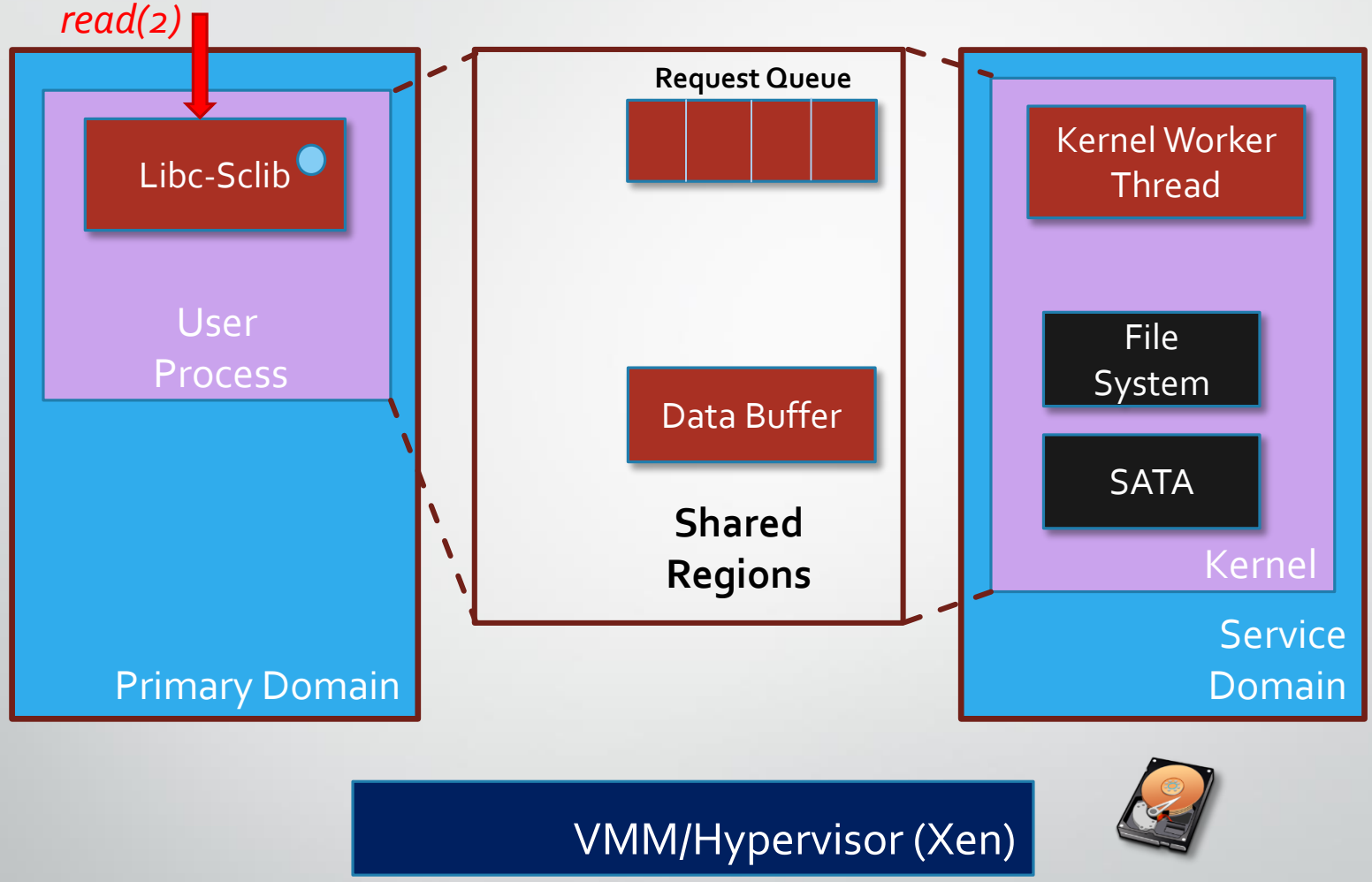

User thread is resumed, data copied, and call is returned.

# Implementation: Completion



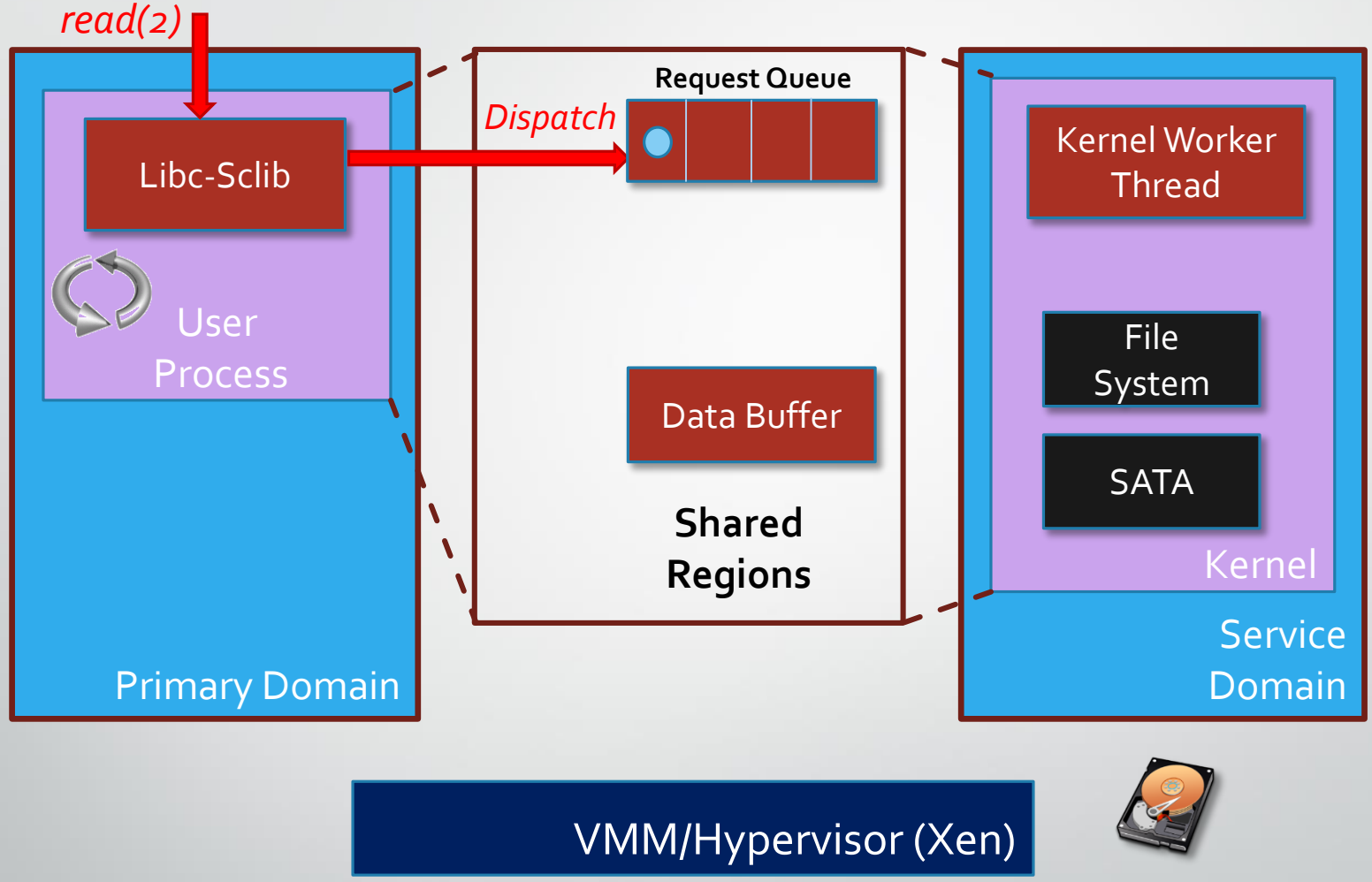

User thread is resumed, data copied, and call is returned.

# Implementation: Spinning



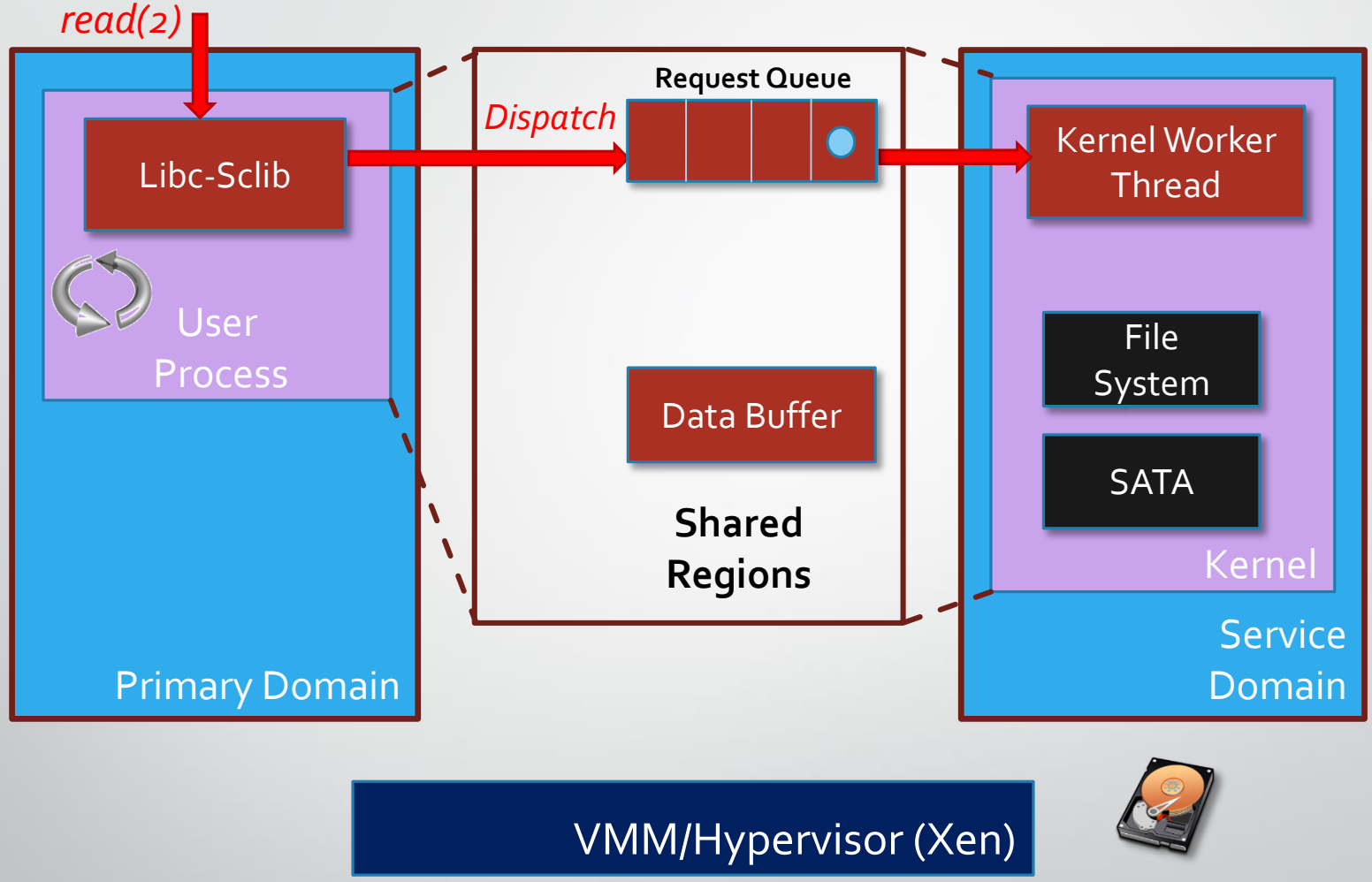

Single-threaded applications. Spinning.

# Implementation: Spinning



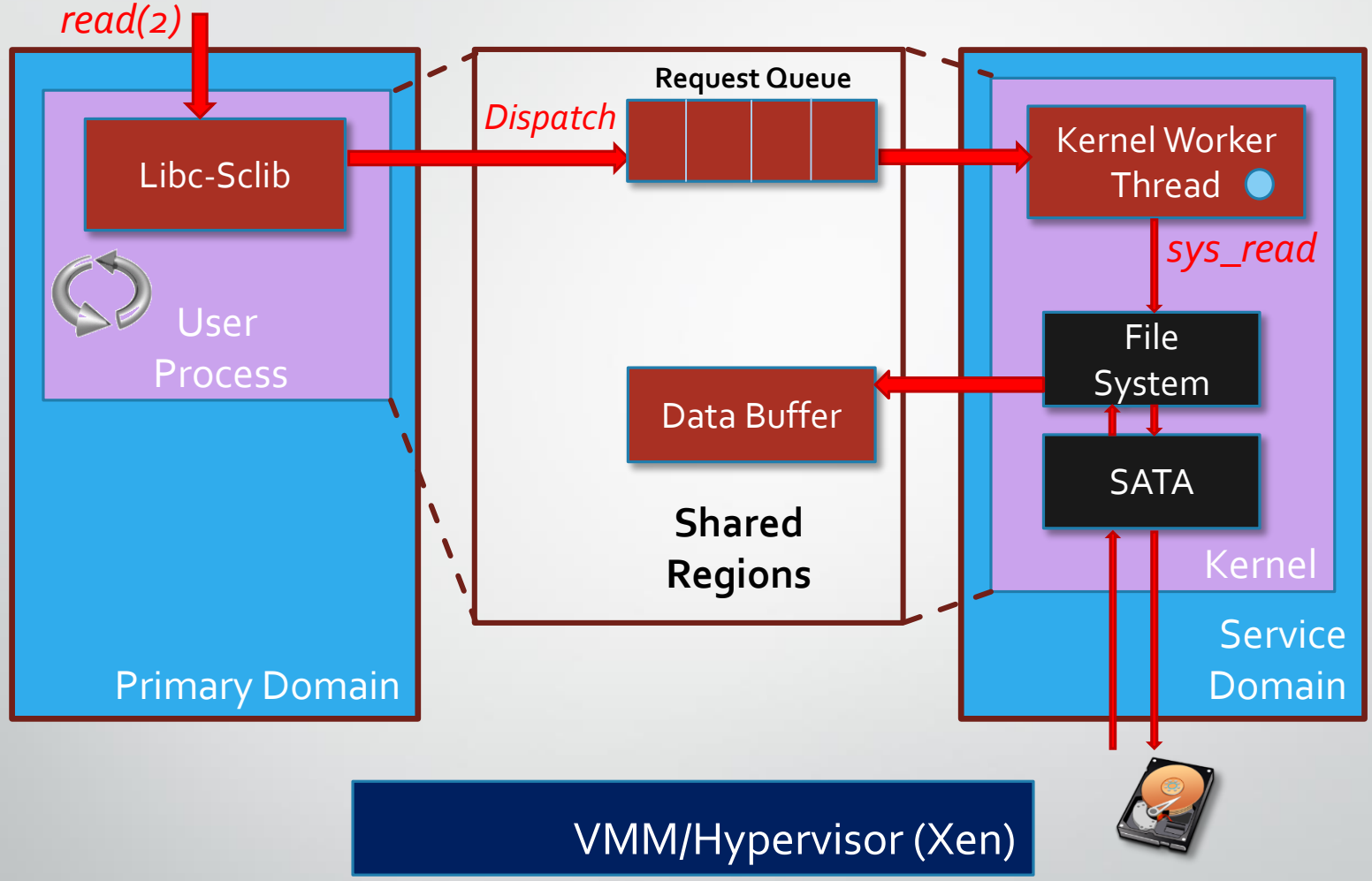

Single-threaded applications. Spinning.

# Implementation: Spinning



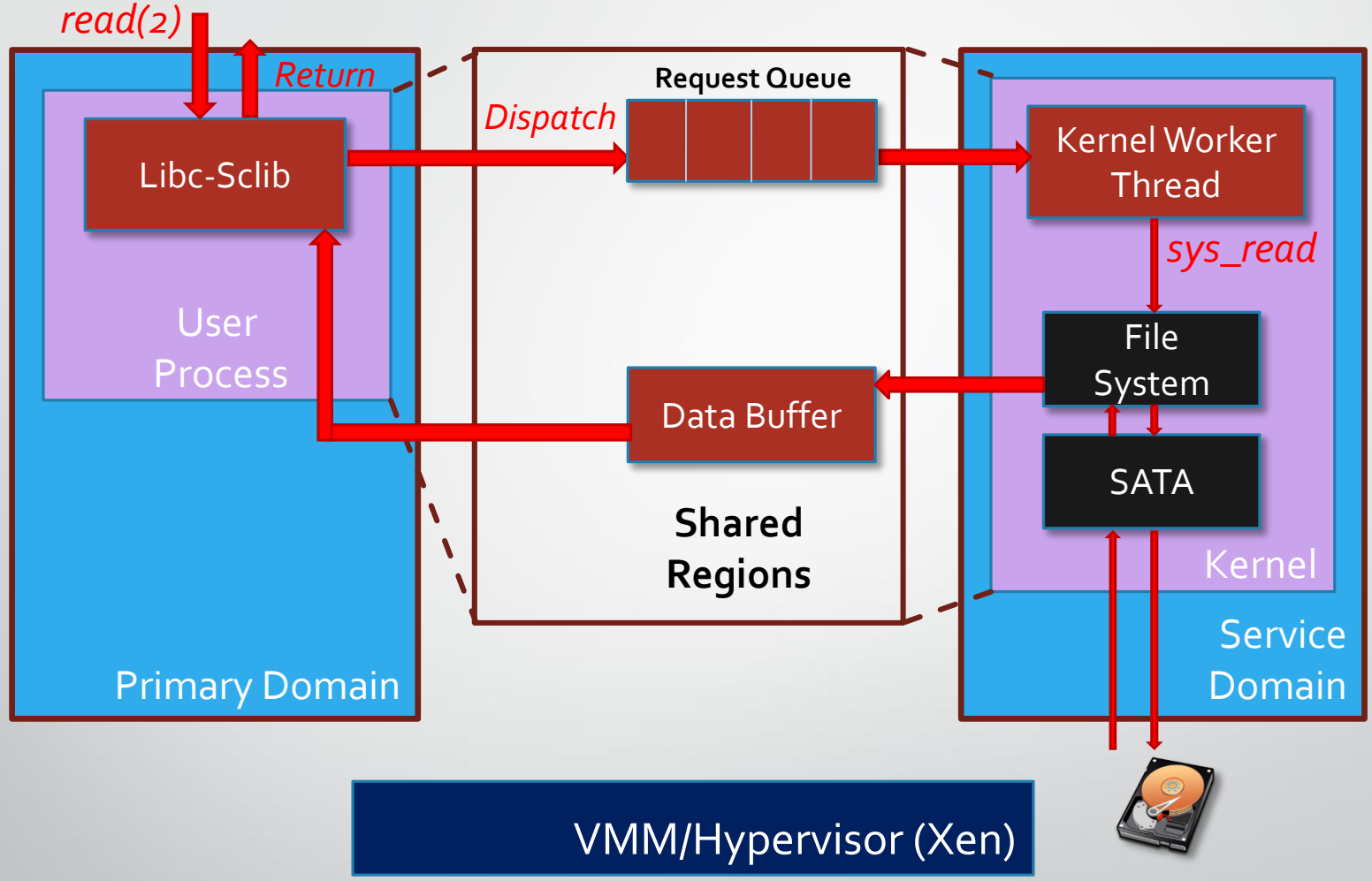

Single-threaded applications. Spinning.

# Implementation: Spinning



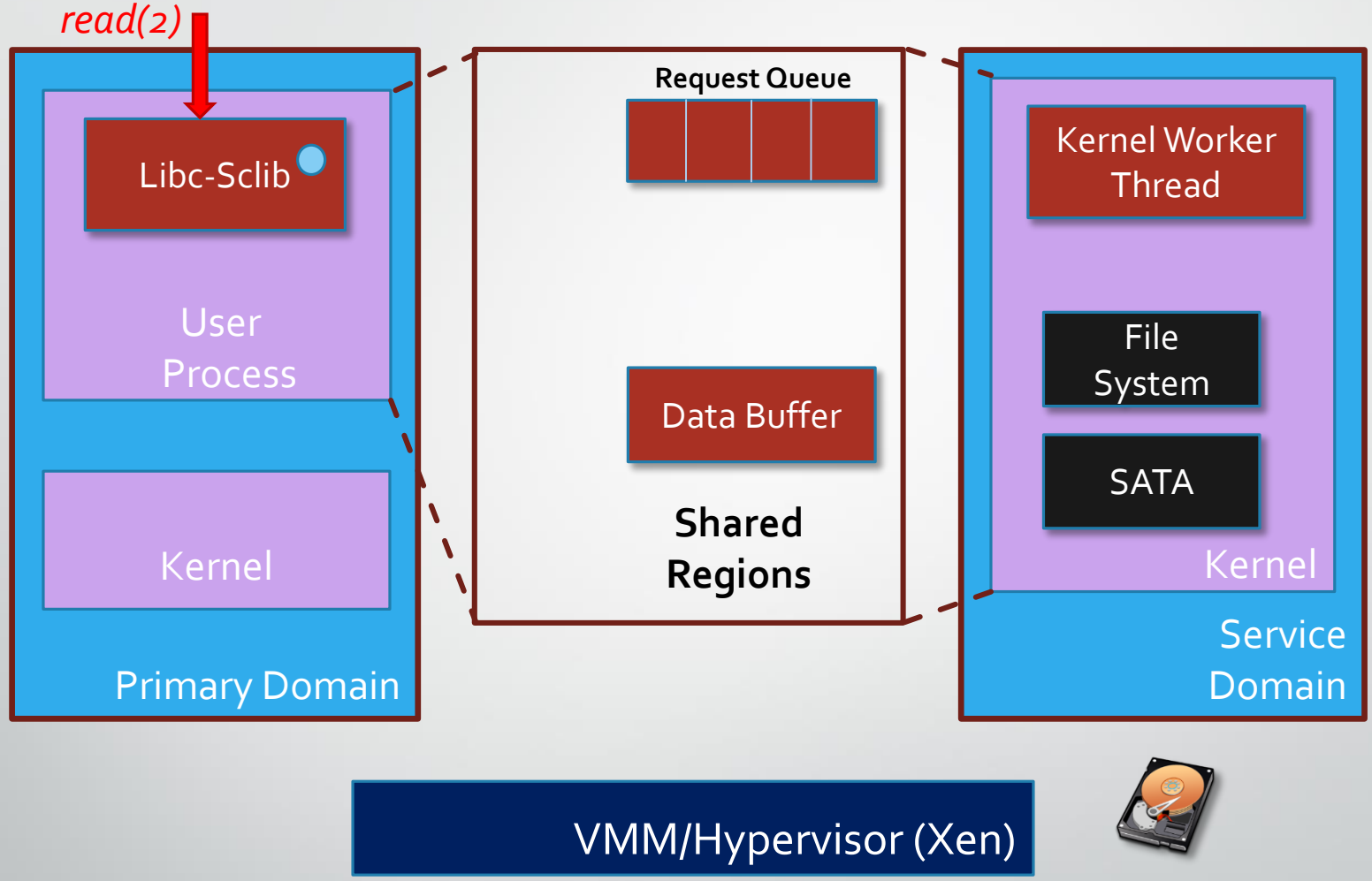

Single-threaded applications. Spinning.

# Implementation: Spinning



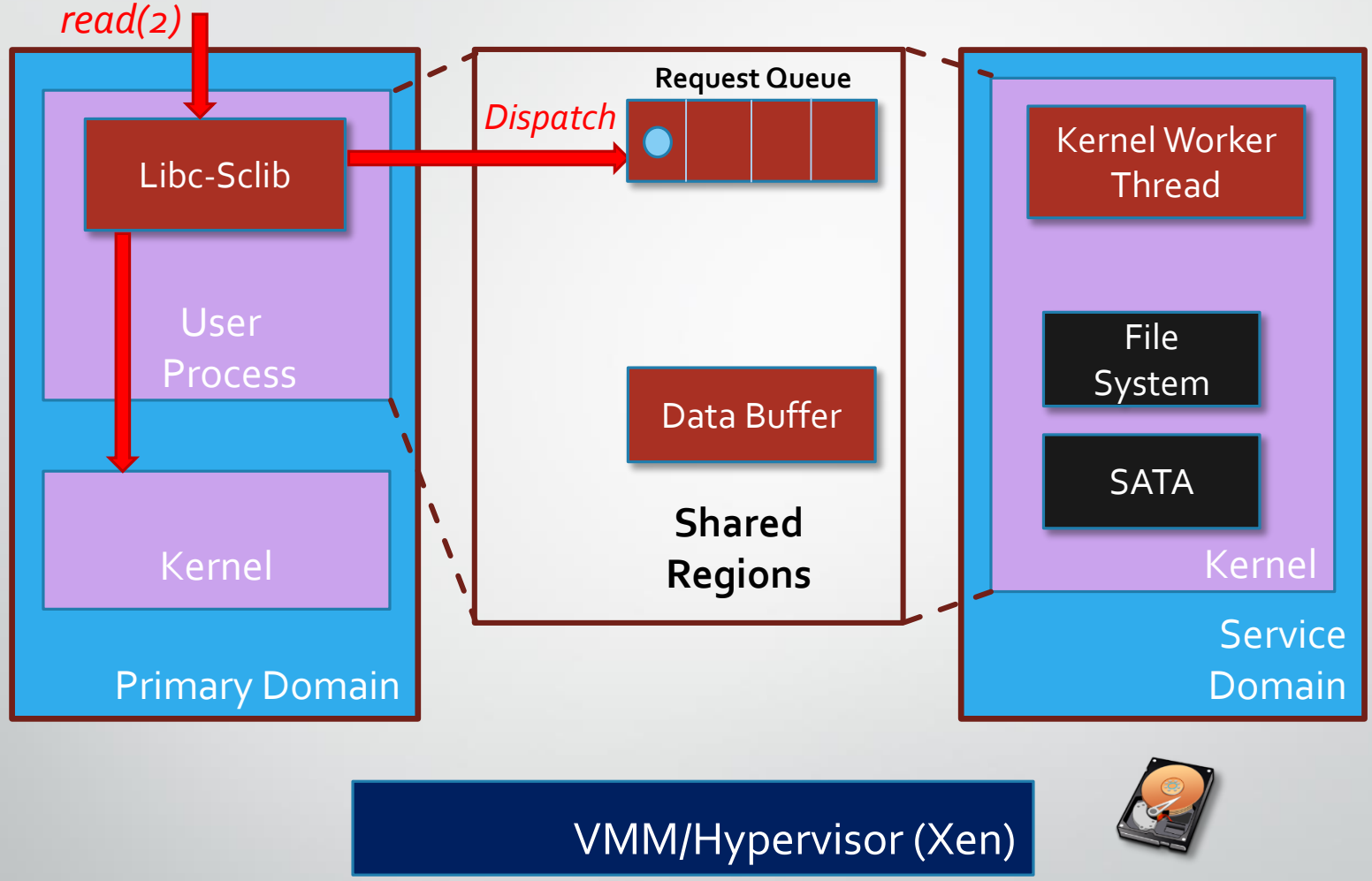

Single-threaded applications. Spinning.

# Implementation: Kernel Switch



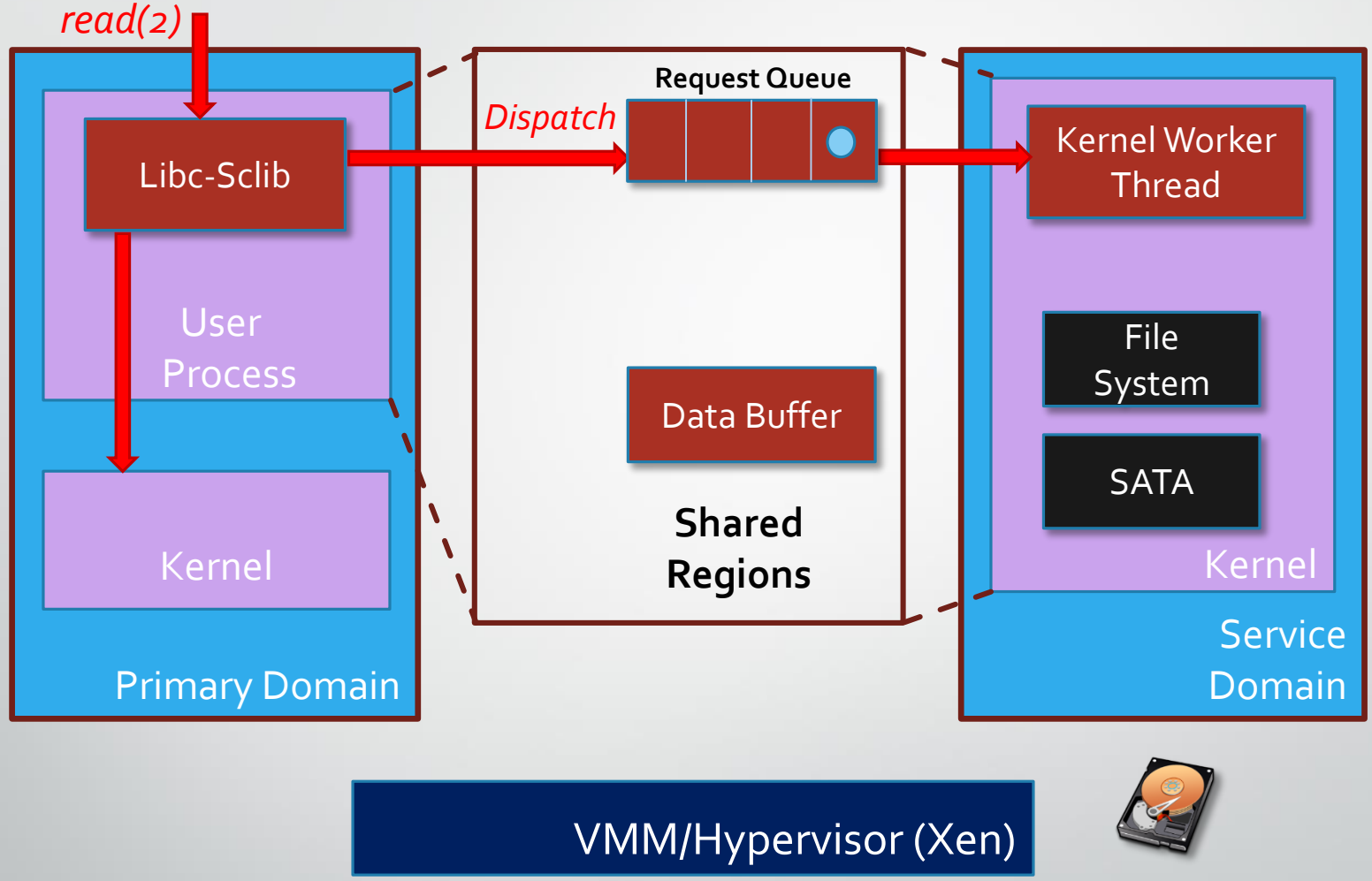

Single-threaded applications. Kernel context switch.

# Implementation: Kernel Switch



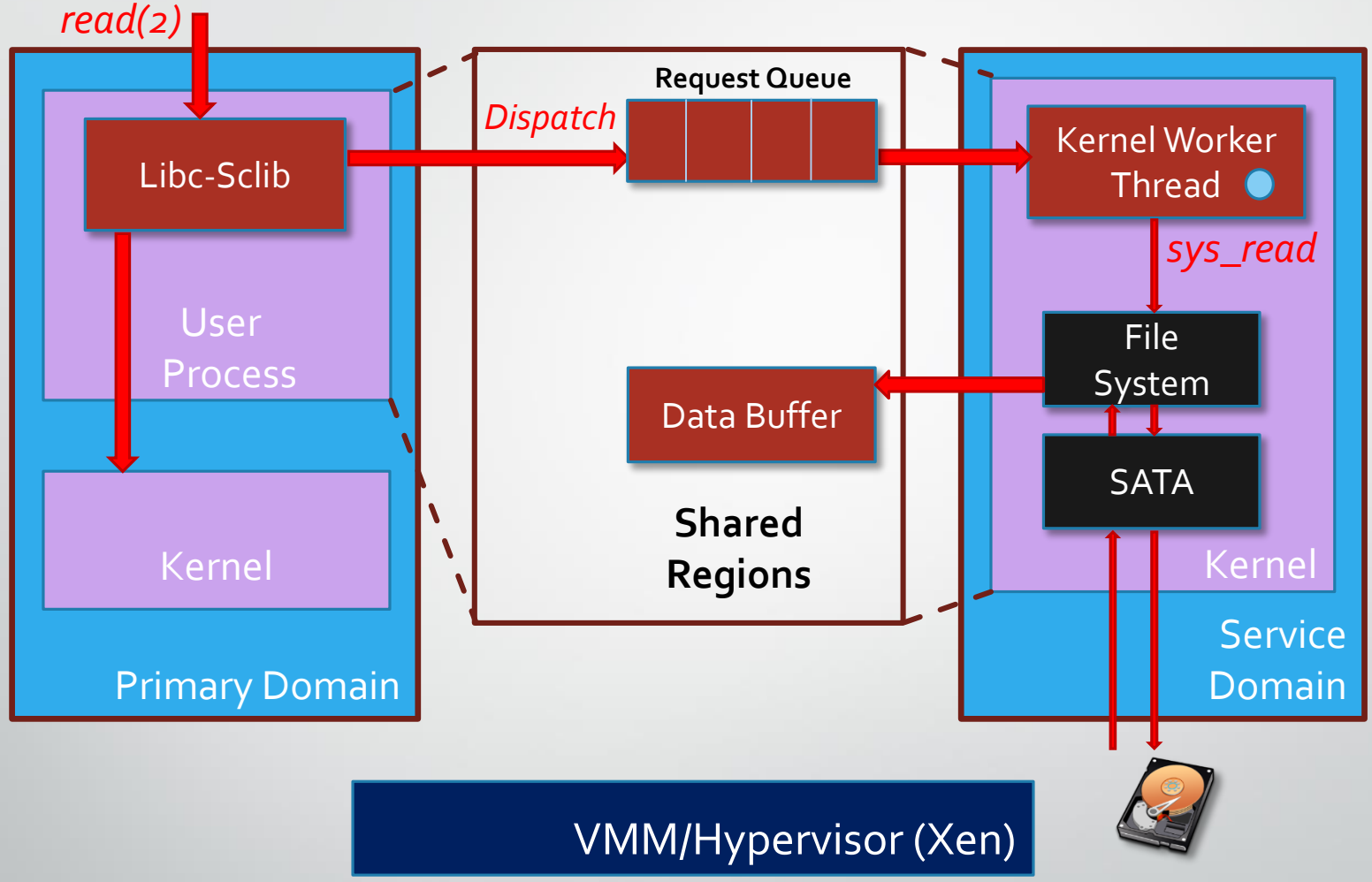

Single-threaded applications. Kernel context switch.

# Implementation: Kernel Switch



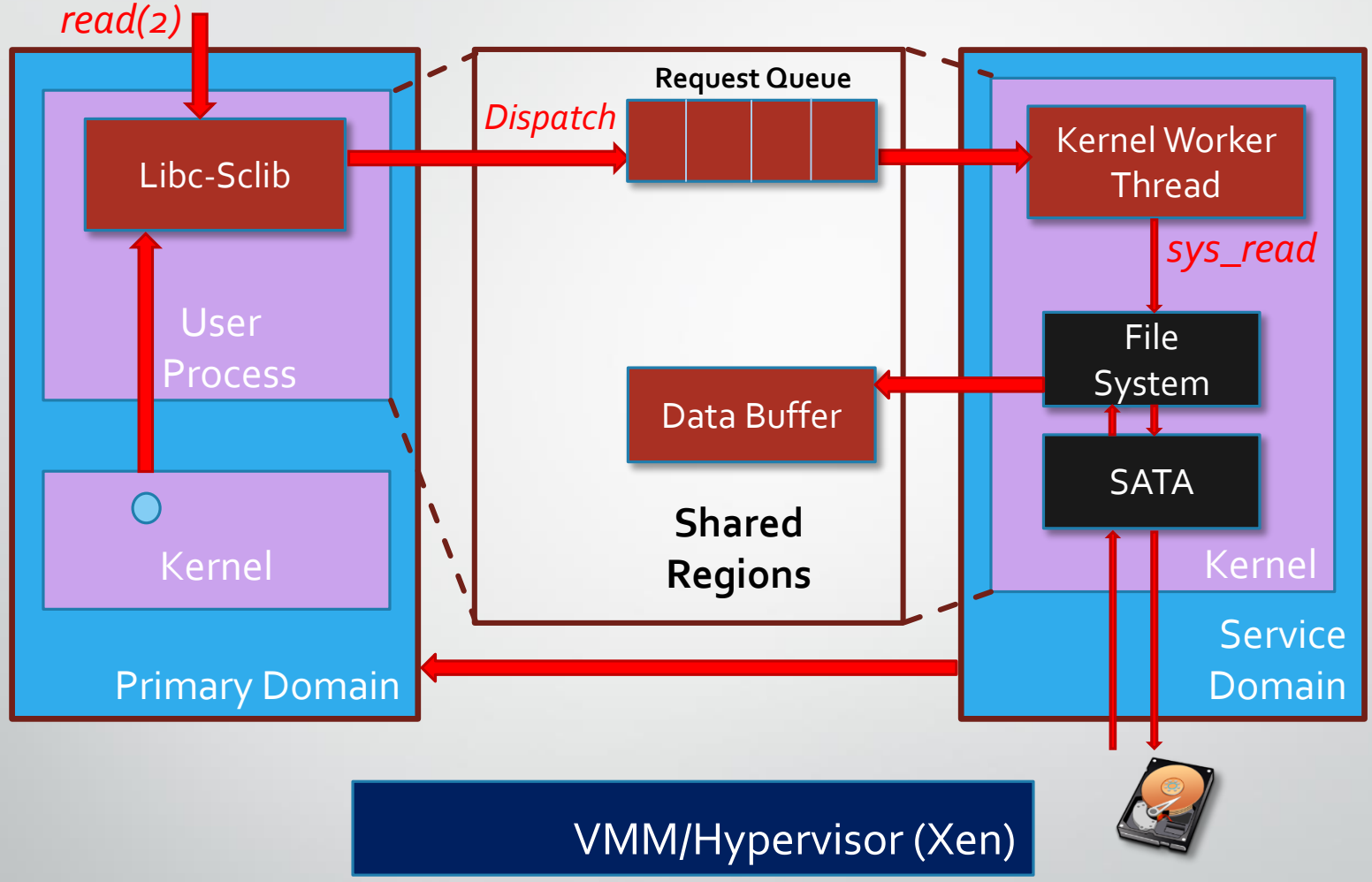

Single-threaded applications. Kernel context switch.

# Implementation: Kernel Switch



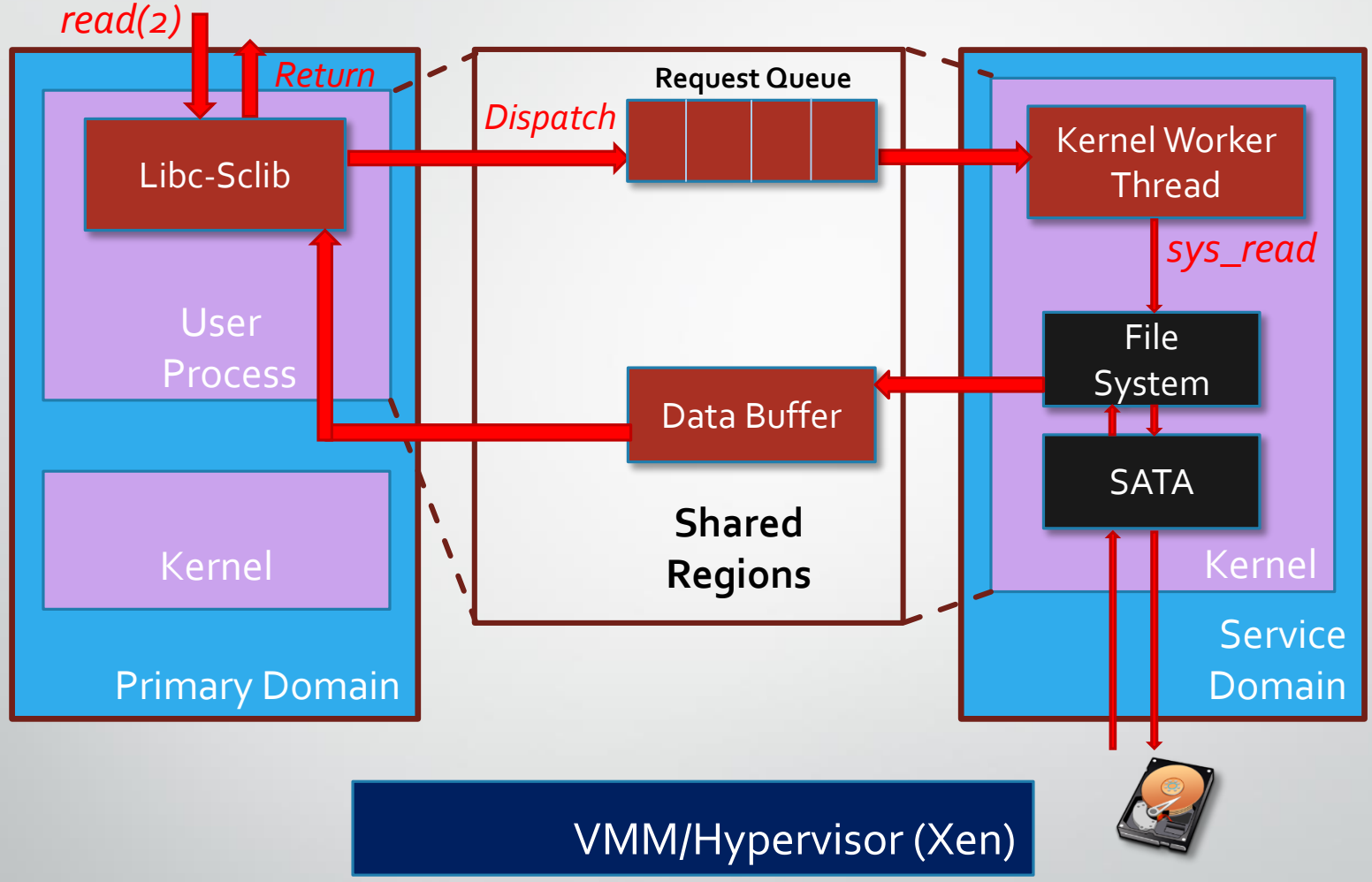

Single-threaded applications. Kernel context switch.

# Implementation: Kernel Switch




Single-threaded applications. Kernel context switch.

# Implementation: Kernel Switch




Single-threaded applications. Kernel context switch.

# Evaluation

- Compatibility
  - System code compatibility – network & storage service domain implementations
  - Application compatibility with existing applications, particularly server-based workloads

| Component | Number of Lines |
|---|---|
| Backend/Frontend Driver | 3115/2157 |
| uClibc+NPTL/libaio | 11152/2290 |
| Linux kernel/Xen | 1610/468 |
| **Total:** | **20792** |

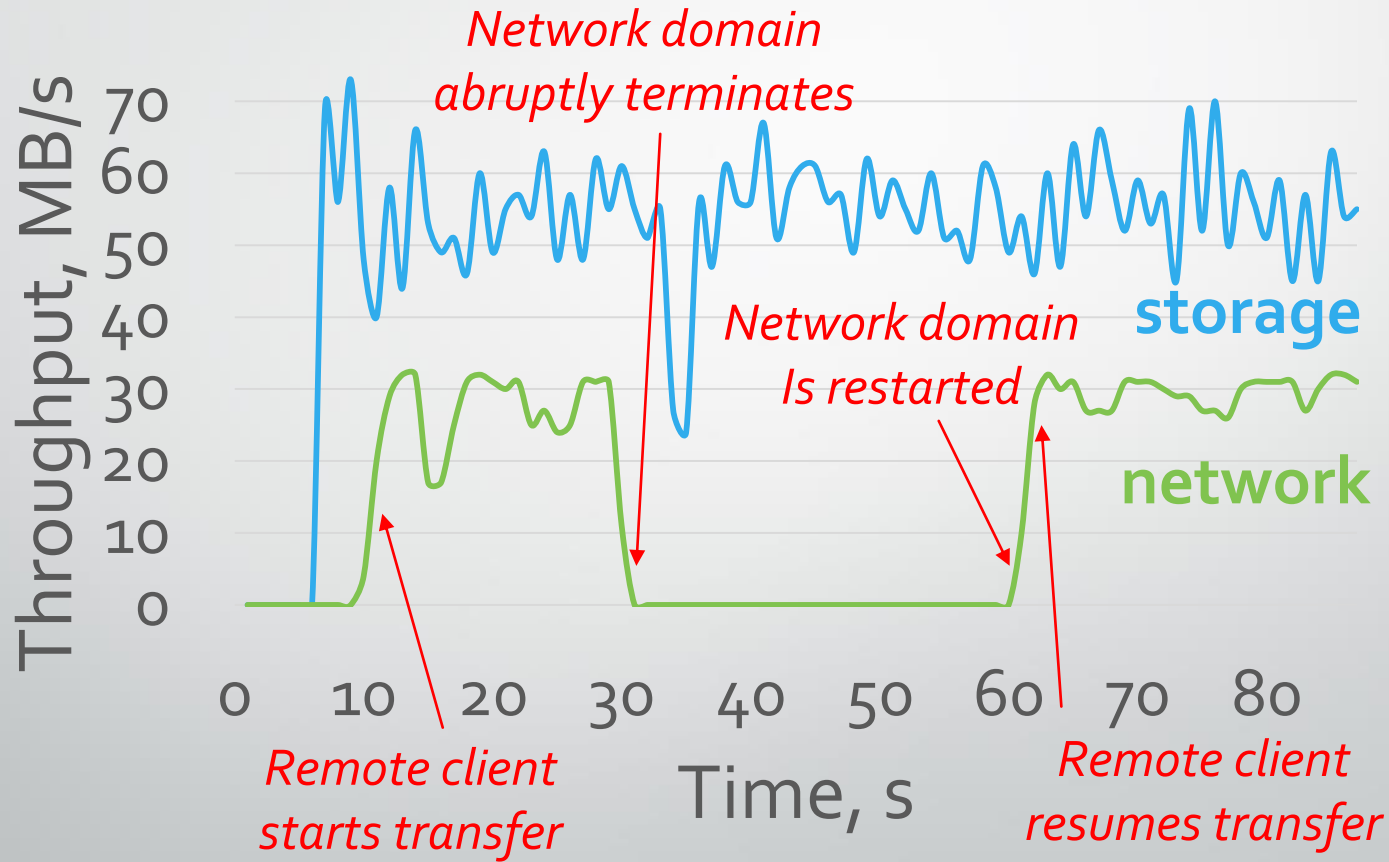
# Evaluation

- Tested fault containment and failure recovery
- Estimated overheads
  - System call overhead
  - Copying
  - Process coordination
- Macrobenchmark performance
  - Multithreaded applications
  - Multiprocess applications

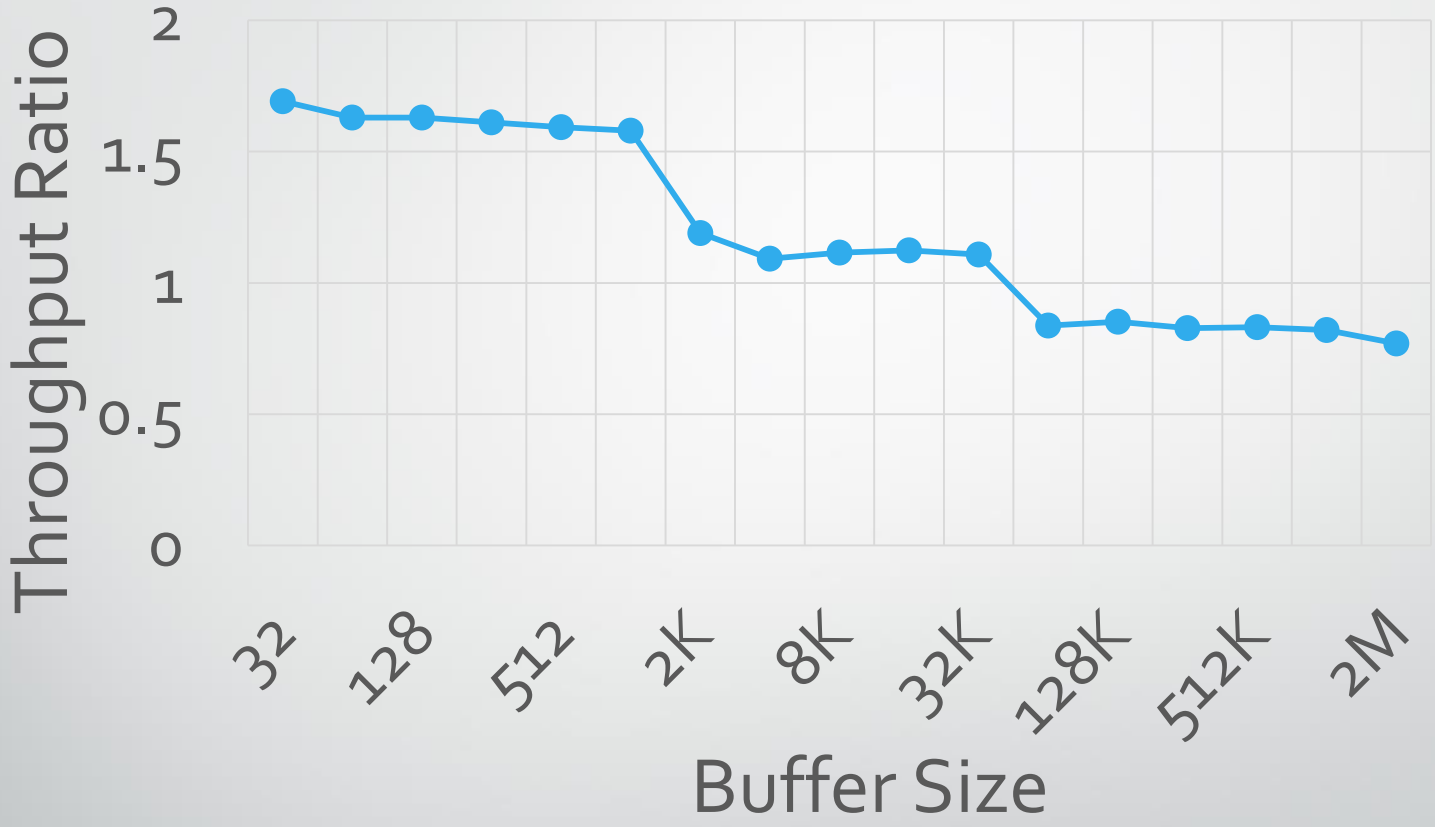
# Evaluation: System Configuration

- 2 x Intel Xeon E5520, 2.27GHz
- 2x4 cores (HyperThreading: OFF, TurboBoost: OFF)
- L1/L2 cache: 64K/256K per core
- L3 cache: 2x8MB
- Main memory: 12GB
- Network: Gigabit Ethernet, PCI Express
- HDD: SATA 7200RPM

# Failure Recovery

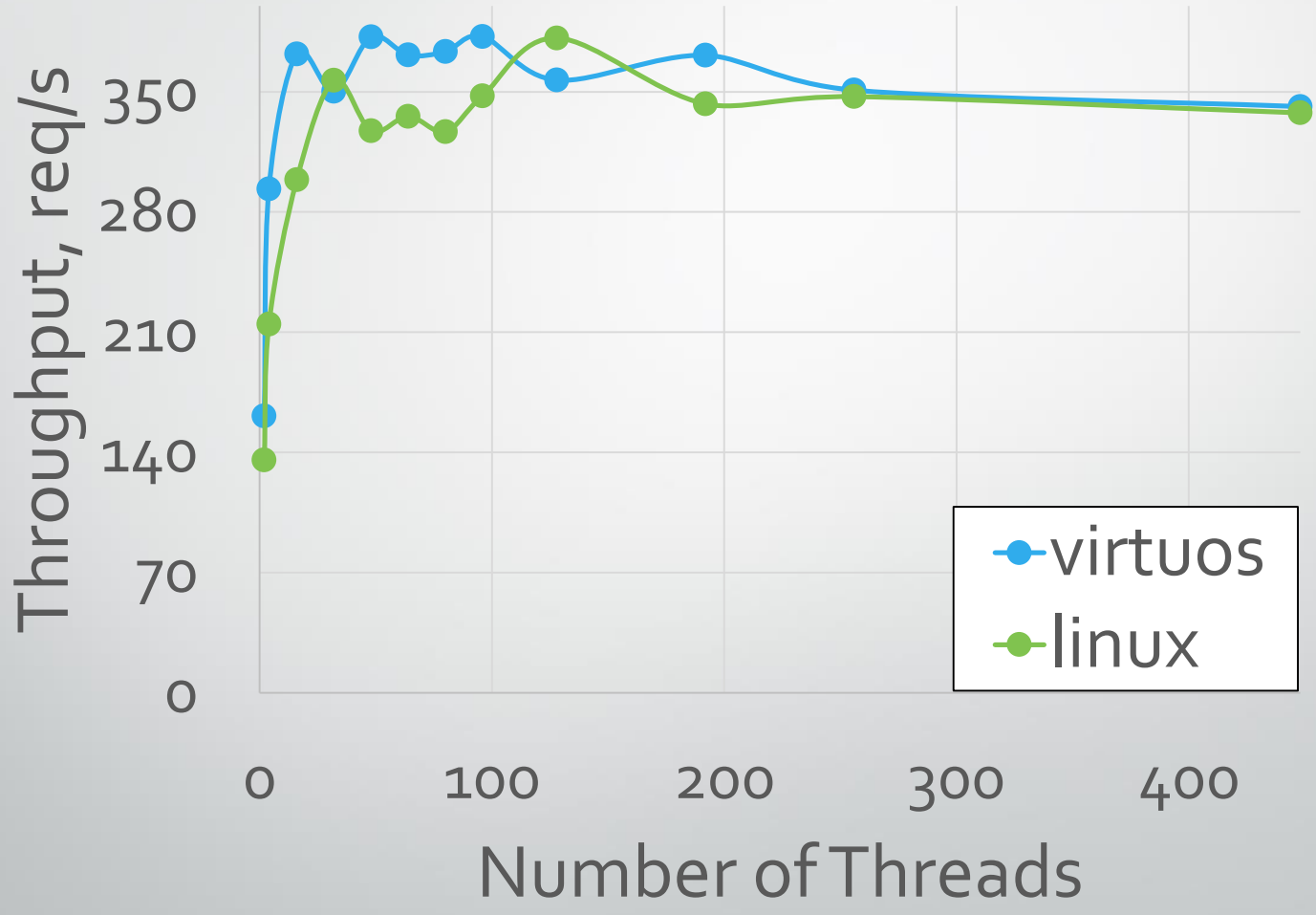


# Memory Copying Overhead

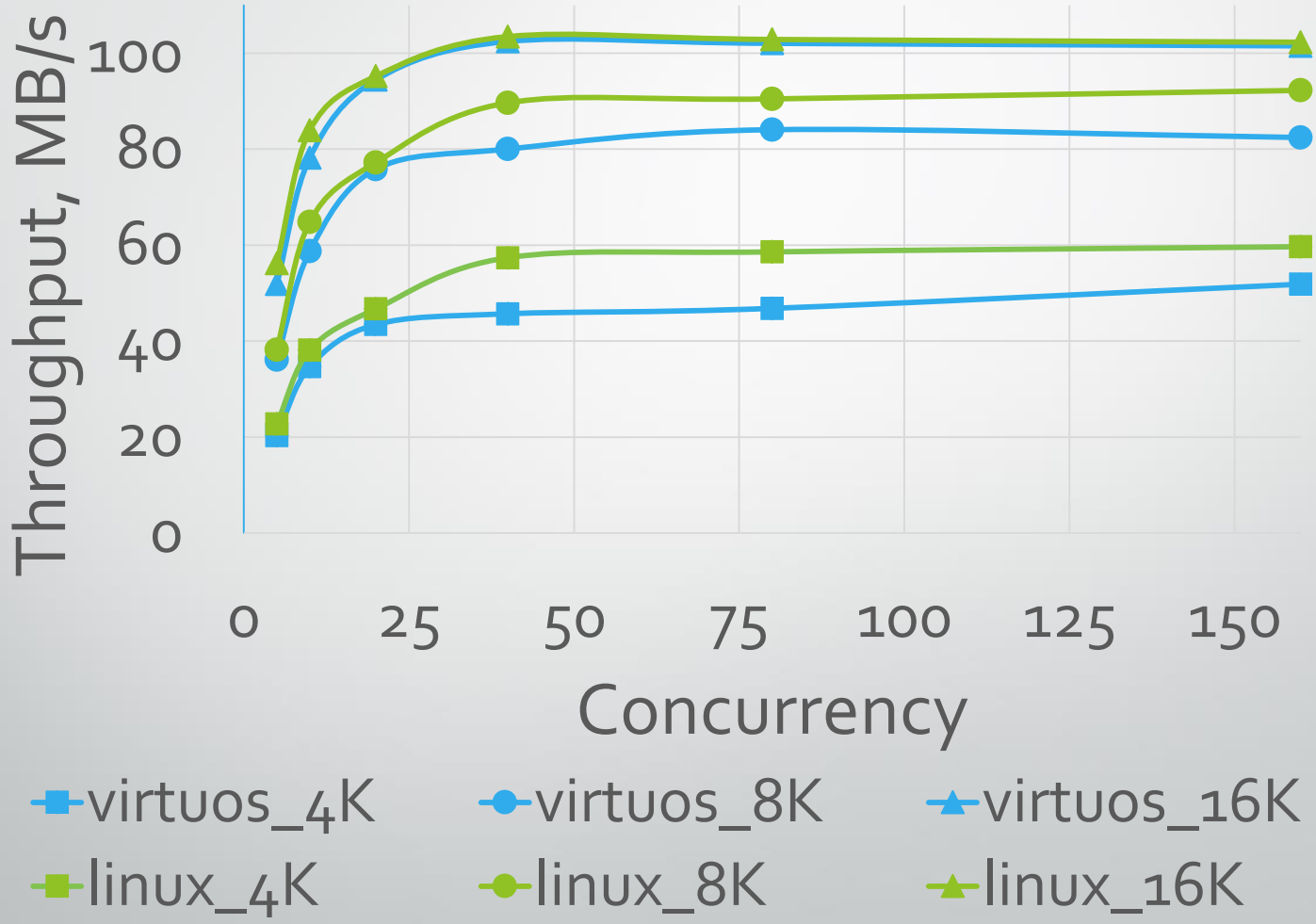



- VirtuOS (storage) vs. Linux
- 16MB tmpfs transfer

# OLTP/Sysbench mySQL



# Apache


Throughput, MB/s
0
20
40
60
80
100
0
25
50
75
100
125
150
Concurrency
virtuos_4K
virtuos_8K
virtuos_16K
linux_4K
linux_8K
linux_16K

# Related Work

- Microkernels/Multiservers
  - L4 [Liedtke 1995]
  - Mach [Accetta, et al. 1986]
  - MINIX 3 [Herder, et al. 2006]
  - SawMill [Gefflaut, et. al 2000]
- Driver Protection
  - Nooks [Swift, et al. 2003]
  - MicroDrivers [Ganapathy, et al. 2008]
  - SUD [Boyd-Wickizer, et al. 2010]

# Related Work

- VM-based solutions
  - Device Driver OS [LeVasseur, et al. 2004]
  - Xen Driver Domains
- Xen hypervisor [Barham, et al. 2003]
- Exception-less mechanism
  - FlexSC [Soares 2010]
  - fos [Wentzlaff 2009]
  - Corey [Boyd-Wickizer 2008]

# Conclusions

- Isolation and Failure Recovery
  - VirtuOS provides isolation of vertical slices of a monolithic kernel into domains
  - Exploits strong isolation & device protection provided by hardware-based virtual machines
  - Supports separate failure & recovery of such domains
- Performance
  - VirtuOS uses fast interdomain communication to achieve good performance
- Compatibility
  - VirtuOS requires few changes to system code
  - Is transparent to POSIX applications

# Availability

- Source code

  people.cs.vt.edu/~rnikola

# THANK YOU!

***Presentation artwork attribution:***

http://TpdkDesign.net (Refresh CL)
http://3xhumed.deviantart.com (Tools Hardware Pack 4 Icons)
http://www.devcom.com (DevCom Network Set 1 Icons)
http://preloaders.net
Office 2013 Online Clipart