barkhausen
institut

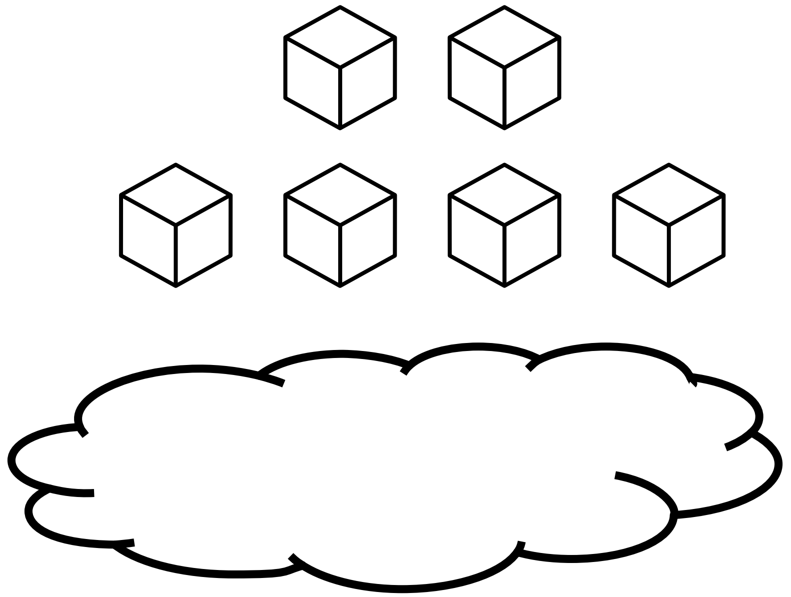
Tradeoffs for Virtualization Mechanisms

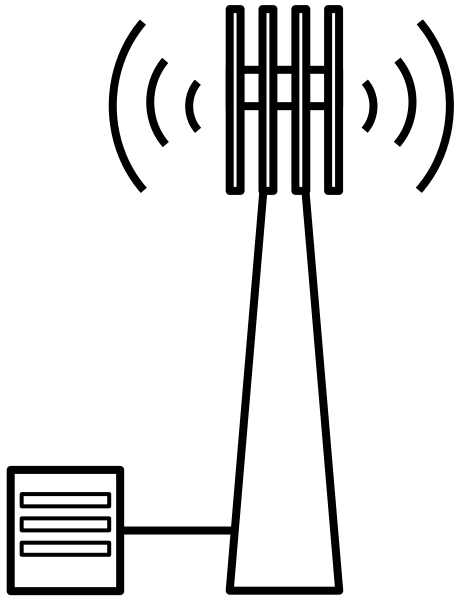
Lectures on Distributed Operating Systems (SS’25)

till.miemietz@barkhauseninstitut.org

Recap: Public Clouds

- Large-scale systems that run applications on behalf of *untrusted* users





- Outsourcing to cloud providers often due to (alleged) economic benefits

Requirements in Cloud Computing

- Users want their applications to run reliably.
  - Data protection (confidentiality / integrity), availability
  - Do not depend on third parties in any form

- Users want performance close to that of native deployment.
  - Get resources that you pay for

Requirements in Cloud Computing

- Users want their applications to run reliably.
  - Data protection (confidentiality / integrity), availability
  - Do not depend on third parties in any form

- Users want performance close to that of native deployment.
  - Get resources that you pay for

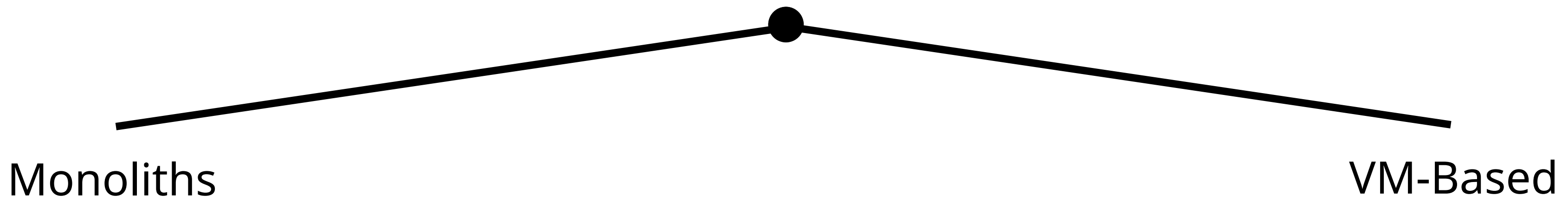
- Cloud providers want to maximize resource utilization.
  - If possible, customers should share physical hardware
  - Function as a service (FaaS) / serverless instead of static instances
  - Requires highly dynamic systems

Requirements in Cloud Computing

- Users want their applications to run reliably.
  - Data protection (confidentiality / integrity), availability
  - Do not depend on third parties in any form
- Users want performance close to that of native deployment.
  - Get resources that you pay for
- Cloud providers want to maximize resource utilization.
  - If possible, customers should share physical hardware
  - Function as a service (FaaS) / serverless instead of static instances
  - Requires highly dynamic systems

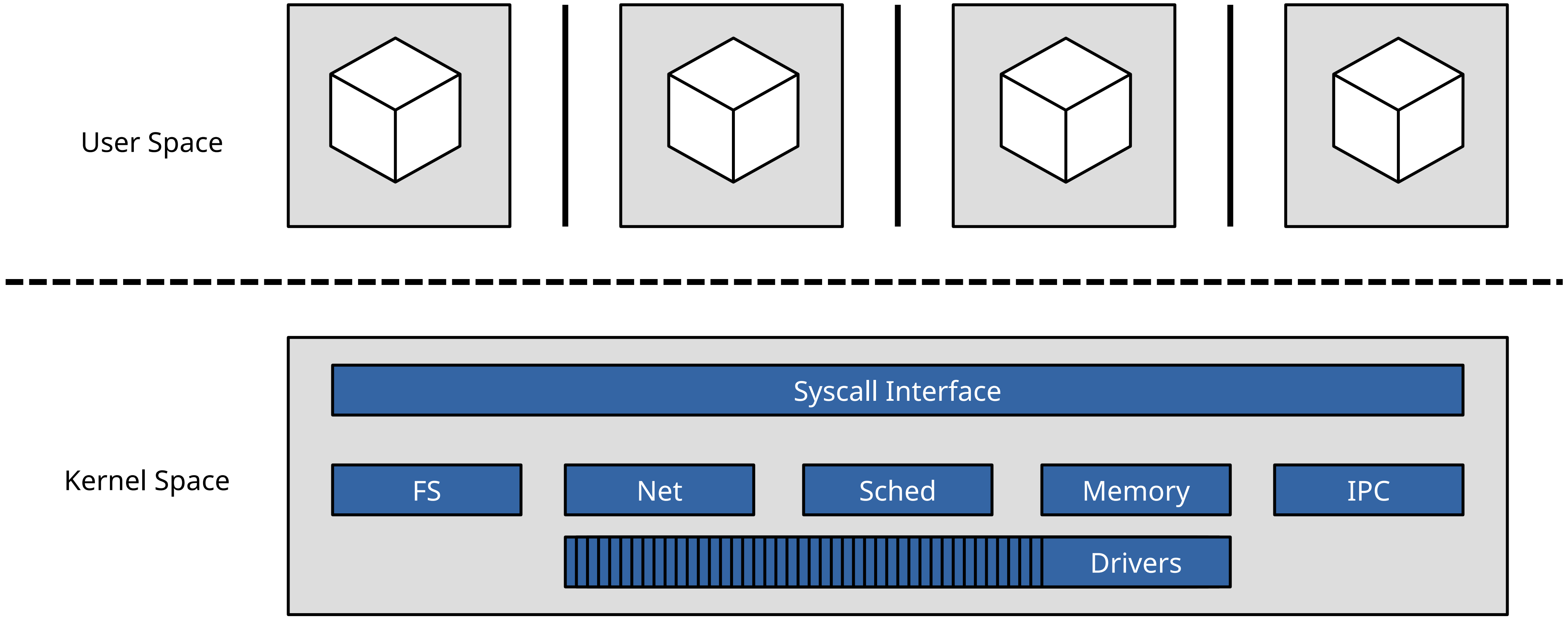
Isolating Cloud Applications – A Naive Approach

Hardening Operating Systems for the Cloud



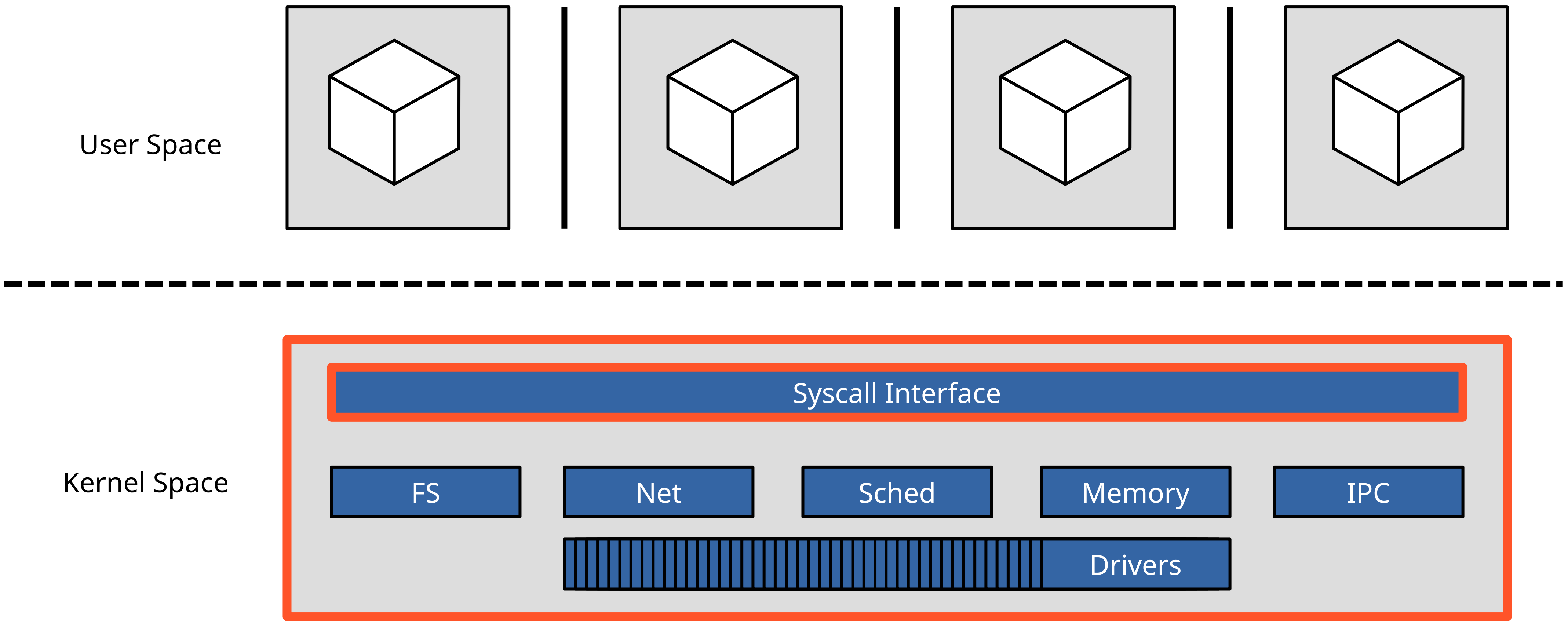
Cloud Servers from a Systems Perspective

- Dominated by Linux, a monolithic OS design



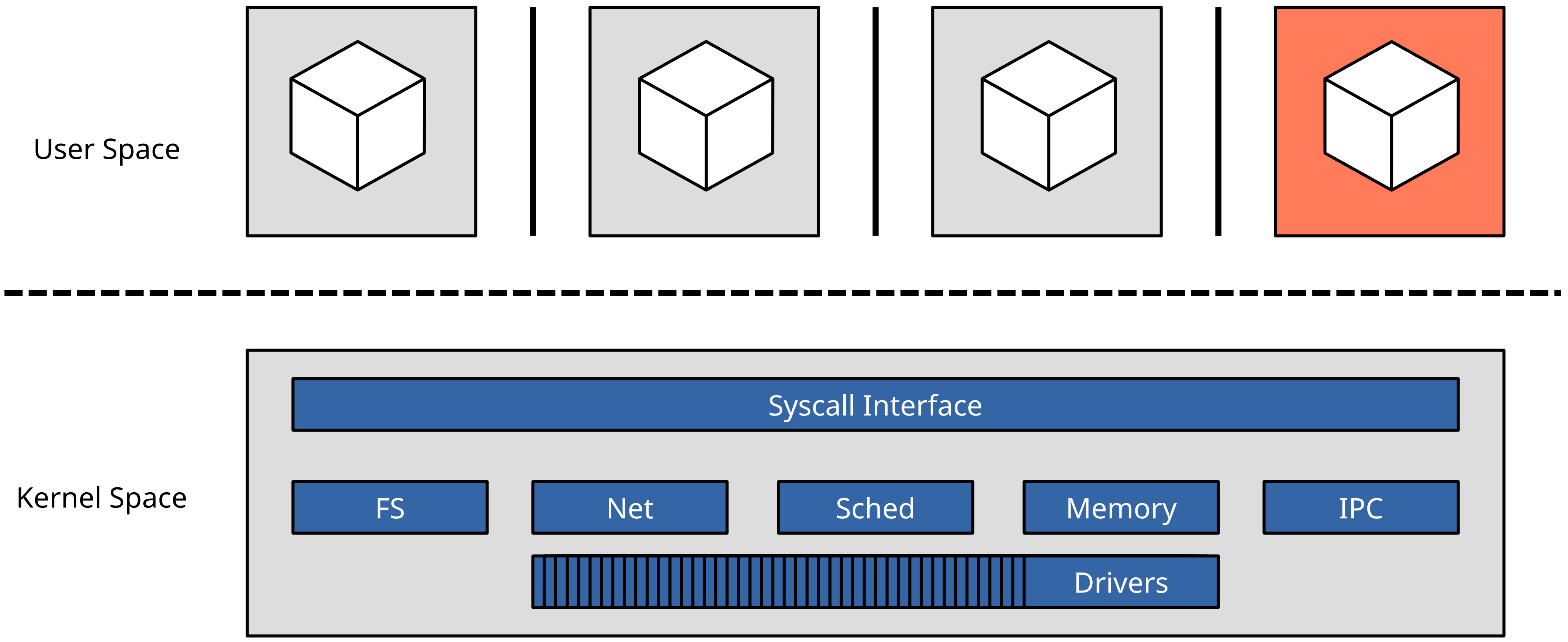
Cloud Servers from a Systems Perspective

- Monolithic OS: large API, large trusted computing base (TCB)



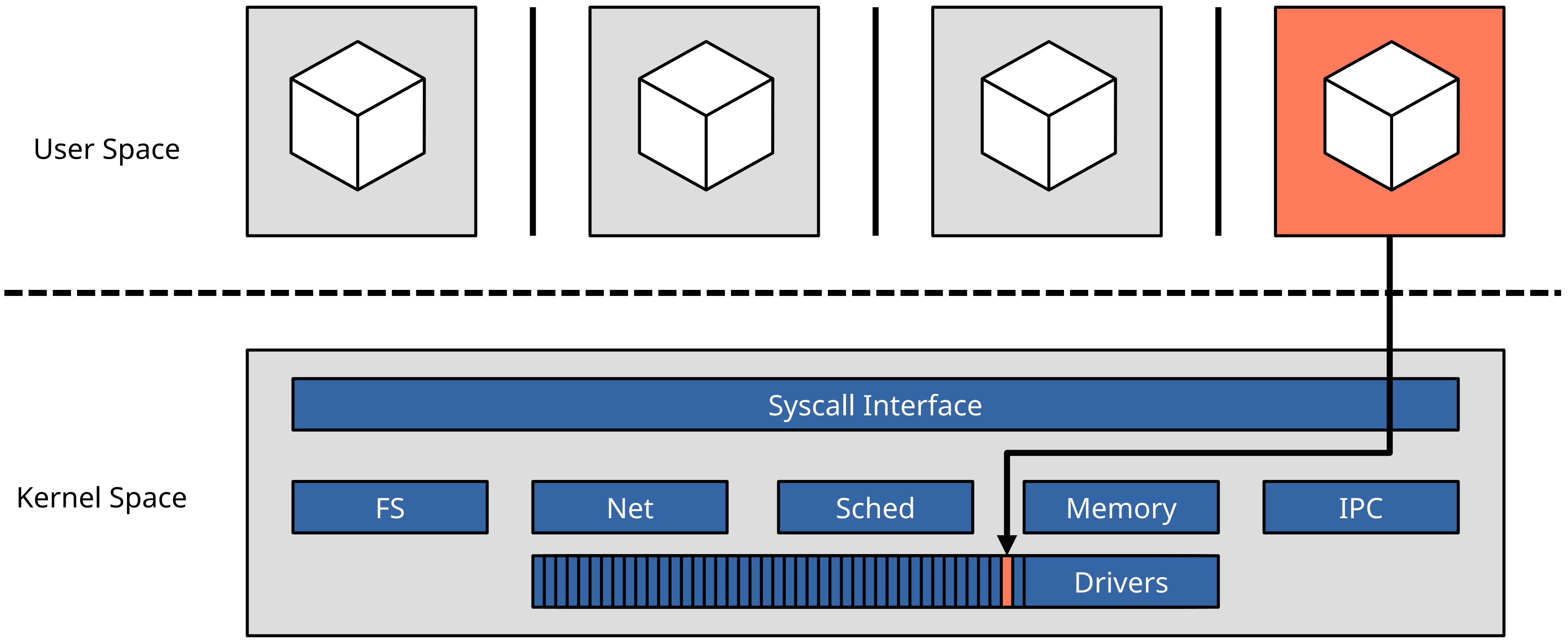
Cloud Servers from a Systems Perspective

- Monolithic architectures are prone to privilege escalation



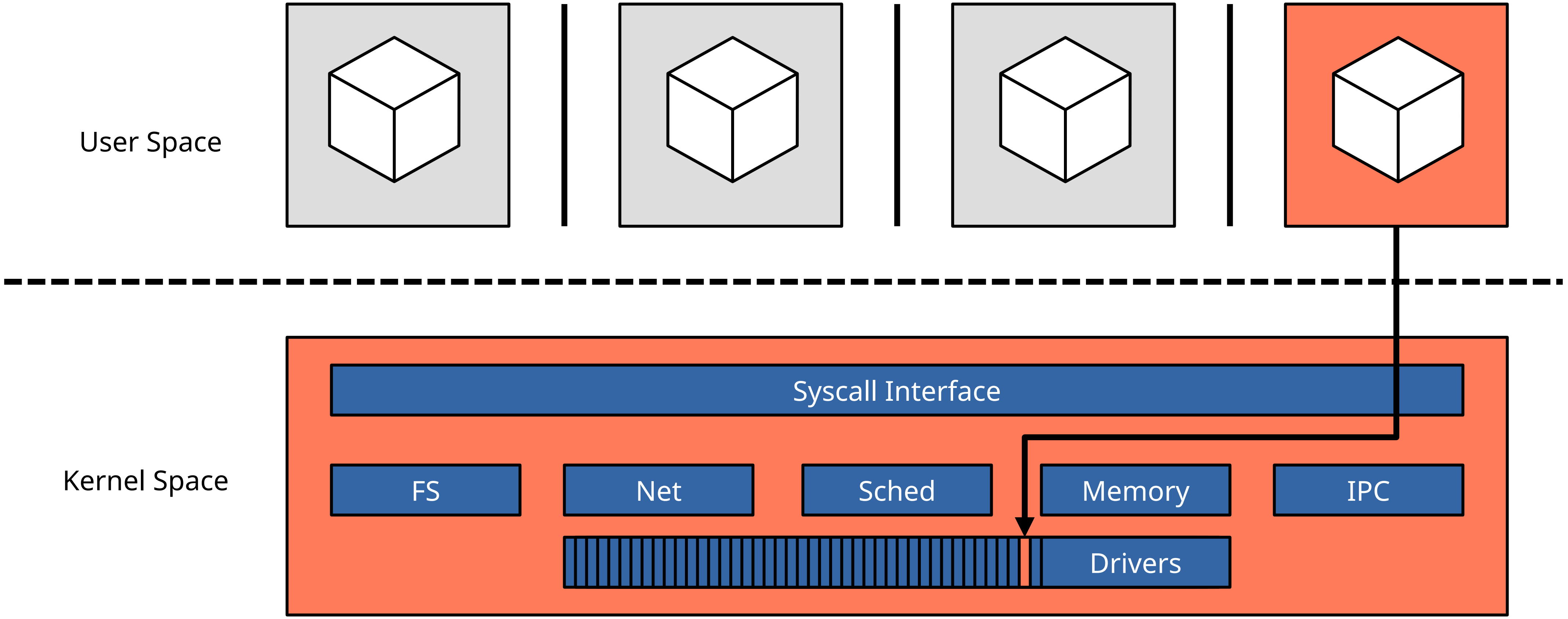
Cloud Servers from a Systems Perspective

- Monolithic architectures are prone to privilege escalation



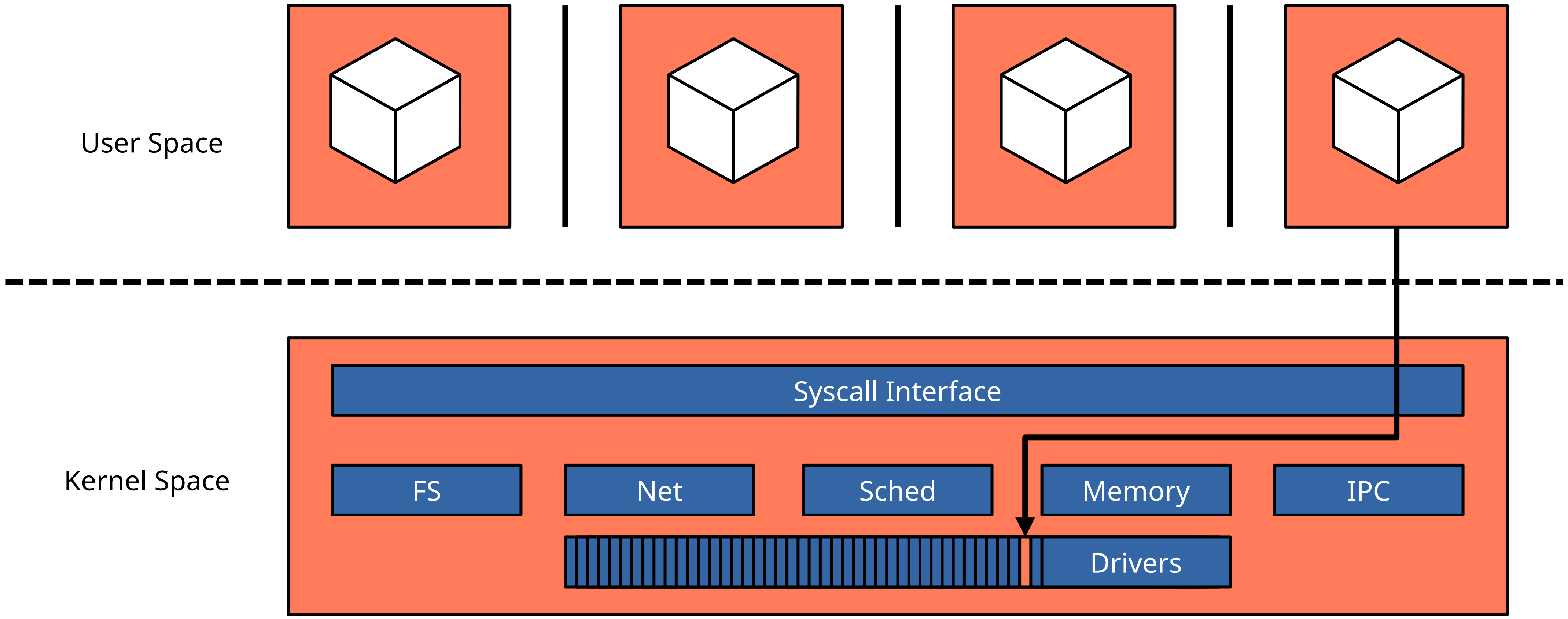
Cloud Servers from a Systems Perspective

- Monolithic architectures are prone to privilege escalation



Cloud Servers from a Systems Perspective

- Monolithic architectures are prone to privilege escalation

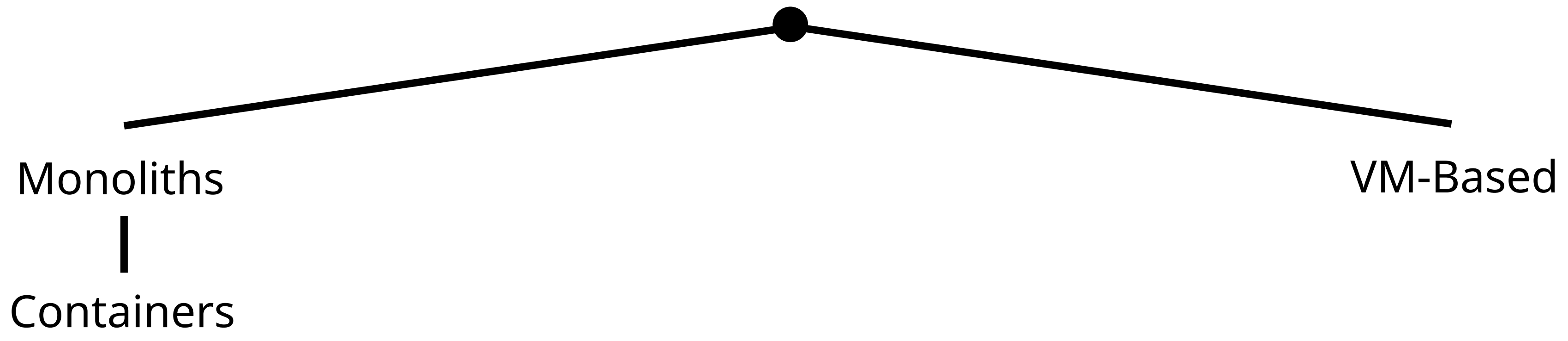


Why Is Process-Based Isolation on a Monolith Not Optimal?

- Architectural weaknesses
  - No restriction of accessible kernel API (*ambient authority*)
- Insufficient means for enforcing resource limits
  - If applicable, `rlimit` too coarse-grained
- Processes share several global “namespaces”
  - E.g., process IDs, network devices, view on file system, …
- But: Excellent performance on standard hardware, compatible, easy to use

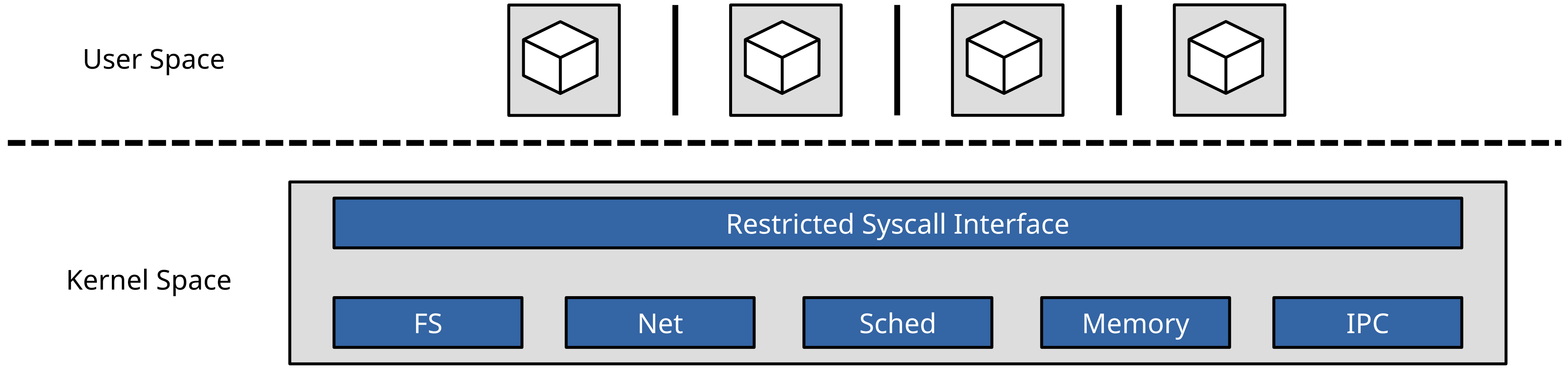
Isolating Cloud Workloads in Containers

Hardening Operating Systems for The Cloud



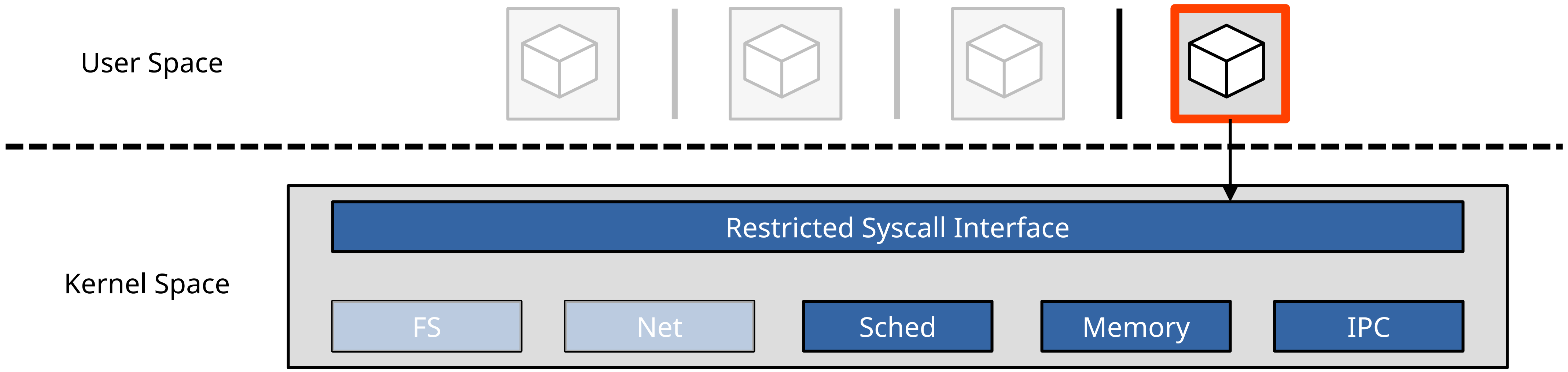
Containers

- Lightweight virtualization (BSD Jails [KW00], Solaris Zones [TC04], Linux [FF+15])
- Increase isolation level of groups of processes using in-kernel facilities
  - Comes with an entire ecosystem for deployment
  - Intuition: Provide separated userland instances on top of a shared kernel
  - No application modification and / or awareness needed, run images as-is



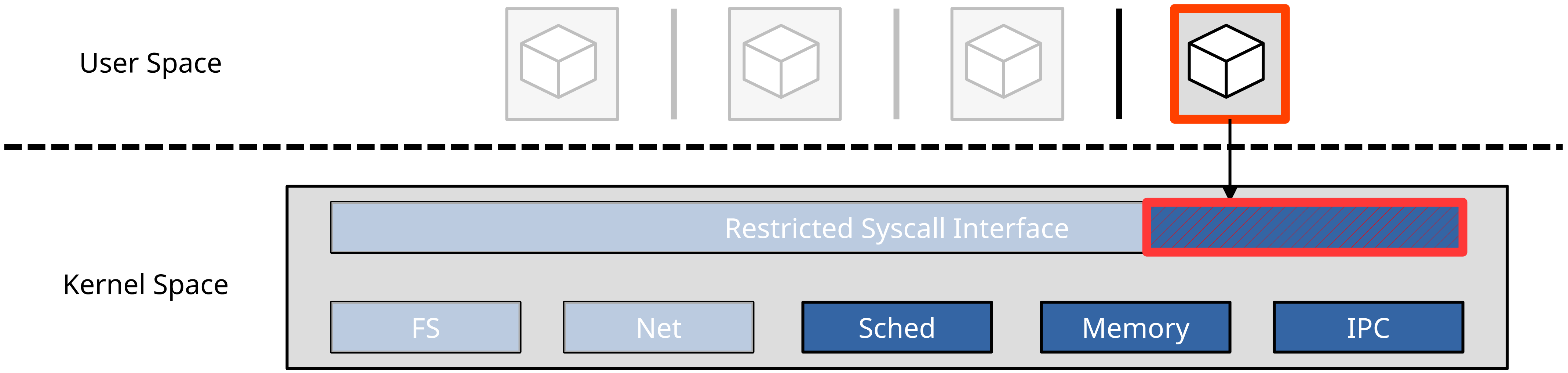
Linux Containers – Namespaces

- Restrict visibility of system components inside a container
- Used to virtualize a variety of kernel abstractions shared by stock processes
  - PIDs, mount points, network interfaces, IPC channels, …
- Often additional restrictions for visibility of file system subtrees (`chroot`, …)



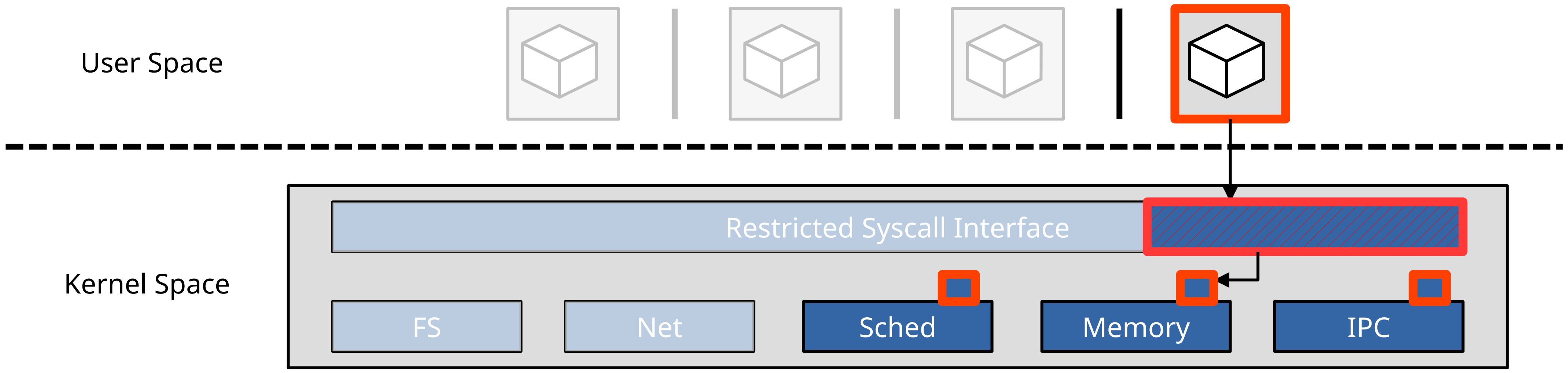
Linux Containers – Seccomp-BPF

- Enables administrator to limit execution of system calls
  - Inject BPF programs interpreted at certain probe points in the kernel
  - Possible to load / unload BPF programs at runtime
  - Also enables restriction of valid parameter range for system calls



Linux Containers – Cgroups

- Unified framework for resource restriction and accounting
  - CPU time / share, memory budgets, network bandwidth, …
  - Hierarchical organization possible → Pass fraction of own budget to other process
  - Offers priorities (e.g., differentiate between different customers)



Containers – Additional Benefits

- Execution inside virtual machines for performance reasons [SS+19]
  - Deploy customized system services per containerized application
- Implementation of Linux features in userspace container runtime [YZ+19]
  - Idea: Reduction of API used by the container
  - gVisor widely applied by Google
- Adding and removing container parts on demand [TB+18]
  - Reduce TCB inside containers

Hardening Containers

Everything Secure Now?

- Various approaches for further hardening container mechanisms
  - Automatic generation of system call filters [CW+21]
  - Transparent change of kernel substrate the containers run on [NS+23]
  - …

Everything Secure Now?

- Various approaches for further hardening container mechanisms
  - Automatic generation of system call filters [CW+21]
  - Transparent change of kernel substrate the containers run on [NS+23]
  - …

- The complexity of container mechanisms remains.
  - Various CVEs (e.g., CVE-2018-18955 for namespaces, CVE-2022-0492 for cgroups)
  - Executing user-provided code in kernel dangerous [HG+23]
  - Huge TCB for confidentiality, integrity, and availability

Running Containers Inside TEEs

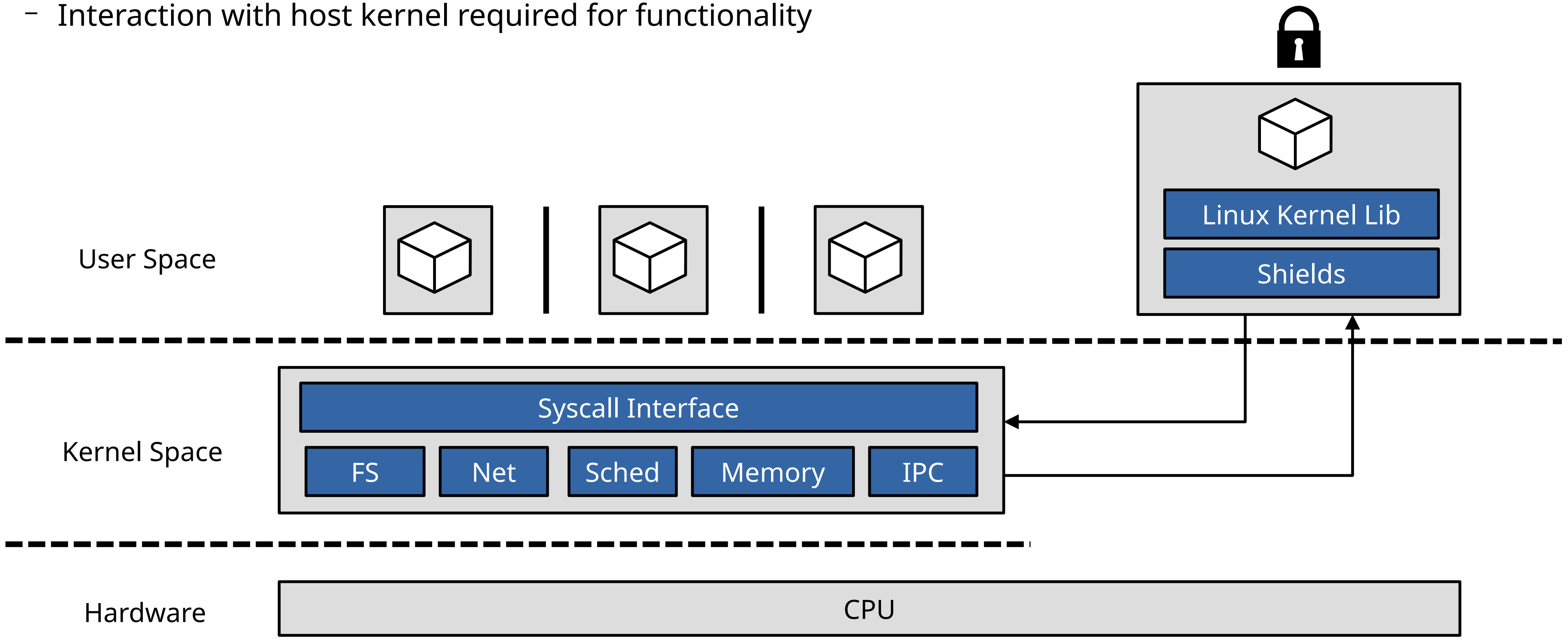
- SCONE: Trusted execution environment (TEE) to shield container [AT+16]
  - Based on Intel SGX
  - Idea: Let hardware provide secure compartments for executing applications

- SGX provides extensive security guarantees ($\rightarrow$ see lecture on trusted execution)
  - Enclaves provide encrypted execution of applications in ring 3 (user space)
  - Enclave set up by OS, inaccessible from any software after initialization (including SMM and DMA)
  - OS, BIOS, firmware are *not* a part of a container’s TCB
  - Only need to trust CPU vendor (for attestation)

- TEEs like SGX can incur significant performance overhead

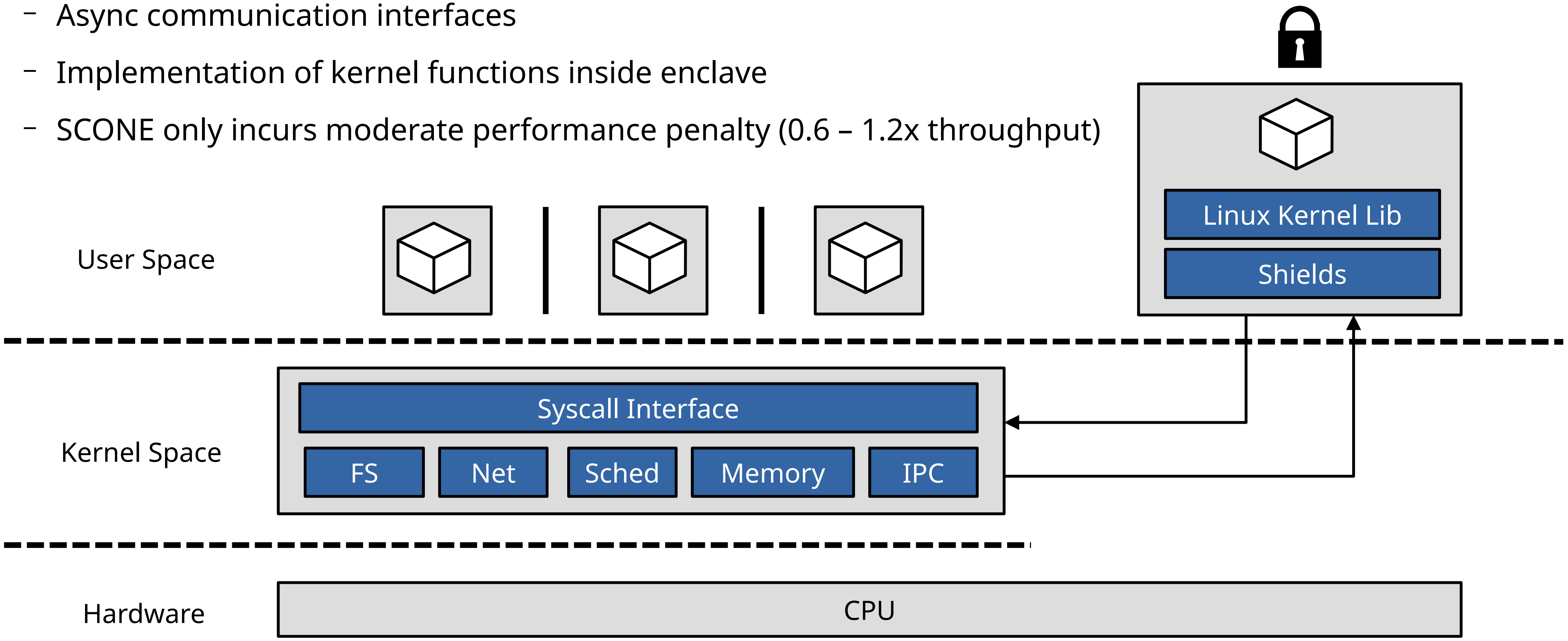
Running Containers Inside TEEs

- Container in TEE runs isolated from the rest of the system
  - Interaction with host kernel required for functionality



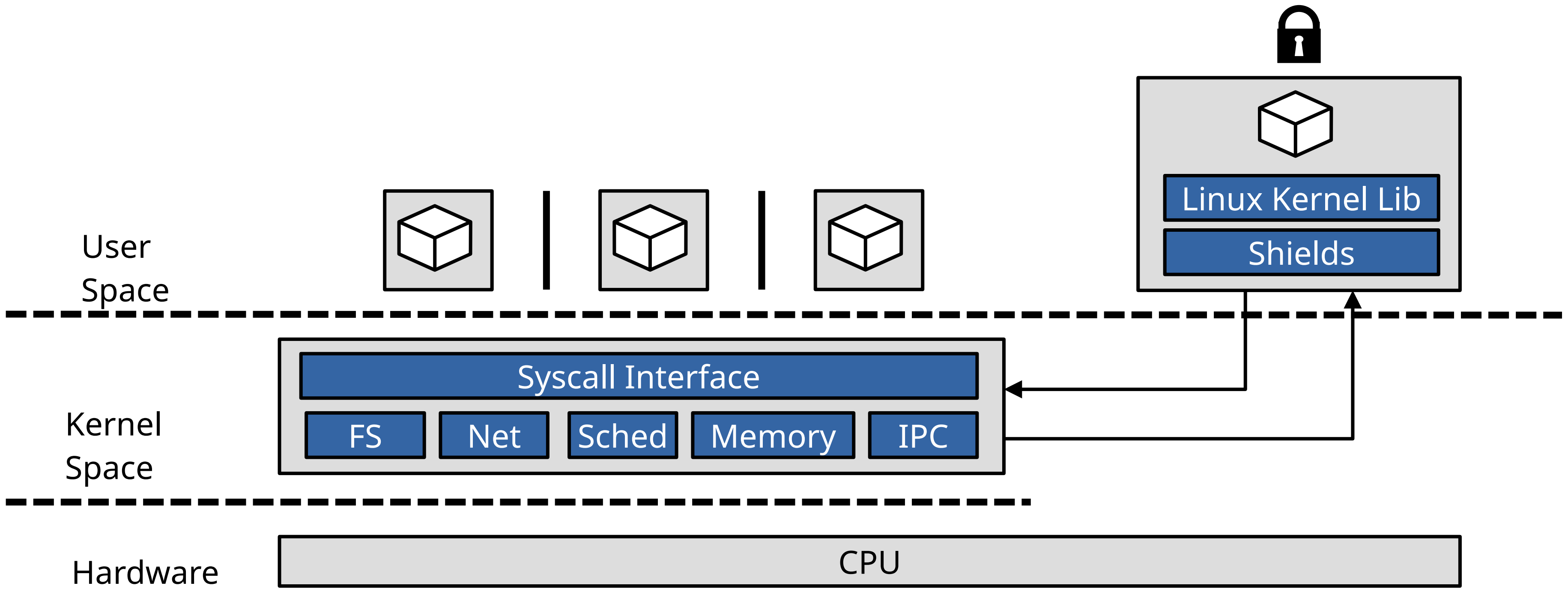
Running Containers Inside TEEs

- SCONE applies optimizations to reduce SGX overhead
  - Async communication interfaces
  - Implementation of kernel functions inside enclave
  - SCONE only incurs moderate performance penalty (0.6 – 1.2x throughput)



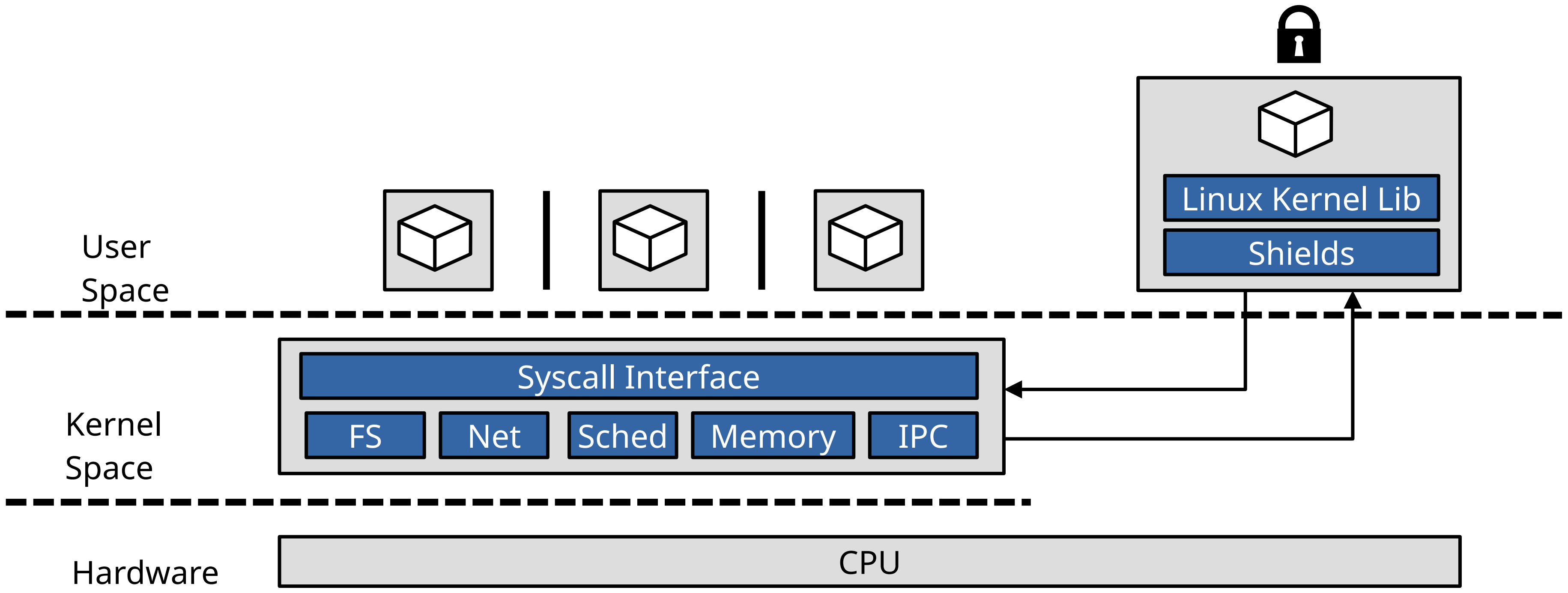
Running Containers Inside TEEs

- What are the security implications of TEEs?



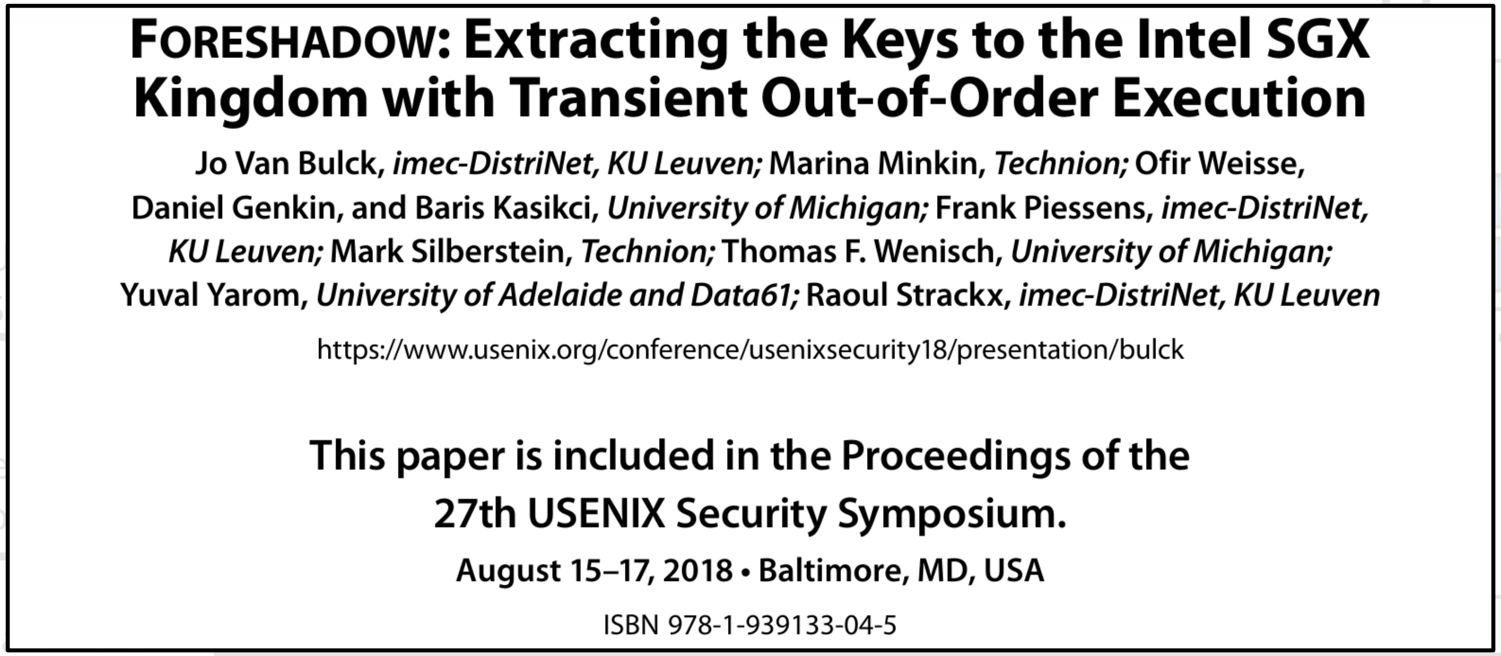
Running Containers Inside TEEs

- What are the security implications of TEEs?
  - Reliant on hardware vendor
  - Complexity moved from soft- to hardware
  - Effective TCB?



Running Containers Inside TEEs

- What are the security implications of TEEs?
  - Reliant on hardware vendor
  - Complexity moved from soft- to hardware
  - Effective TCB?



FORESHADOW: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution

Jo Van Bulck, *imec-DistriNet, KU Leuven;* Marina Minkin, *Technion;* Ofir Weisse, Daniel Genkin, and Baris Kasikci, *University of Michigan;* Frank Piessens, *imec-DistriNet, KU Leuven;* Mark Silberstein, *Technion;* Thomas F. Wenisch, *University of Michigan;* Yuval Yarom, *University of Adelaide and Data61;* Raoul Strackx, *imec-DistriNet, KU Leuven*

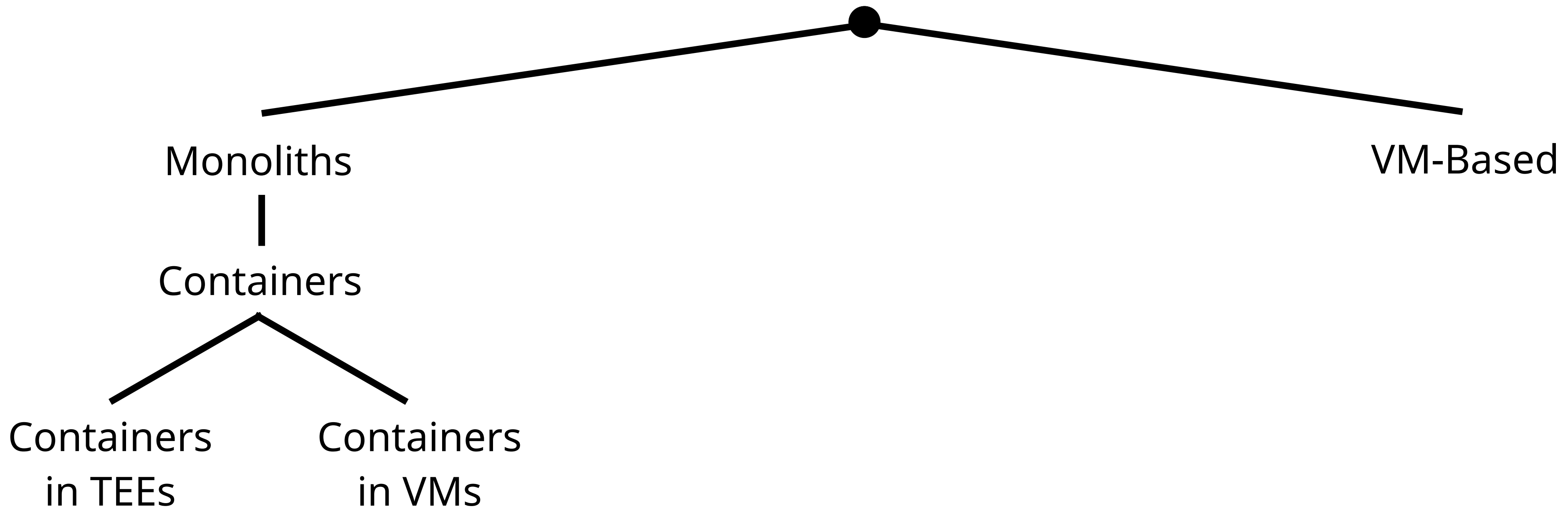
https://www.usenix.org/conference/usenixsecurity18/presentation/bulck

This paper is included in the Proceedings of the 27th USENIX Security Symposium.

August 15–17, 2018 • Baltimore, MD, USA

ISBN 978-1-939133-04-5

Hardening Operating Systems For The Cloud



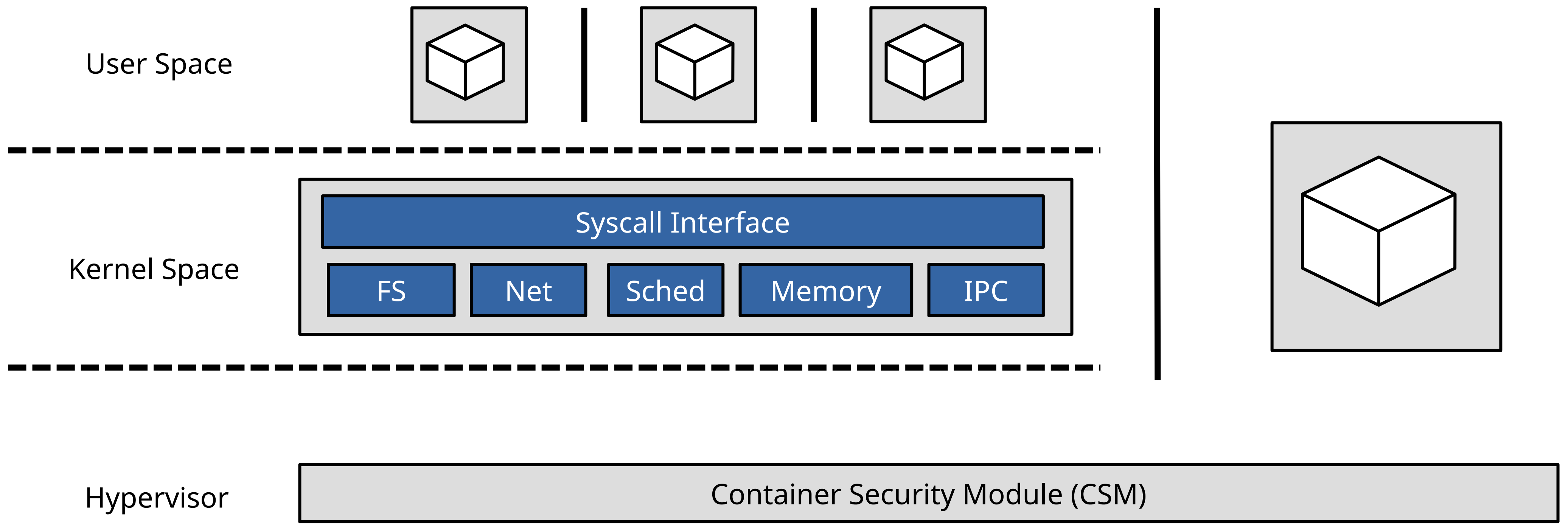
Running Containers as Virtual Machines

- Simpler hardware, yet strong isolation possible (BlackBox [VN22])
  - Runs on standard virtualization hardware

- Run containers on top of a *container security module* (CSM)
  - Linux VM with drivers etc. running beside application containers
  - CSM shields application containers
  - Applications should not have to trust function donor VM

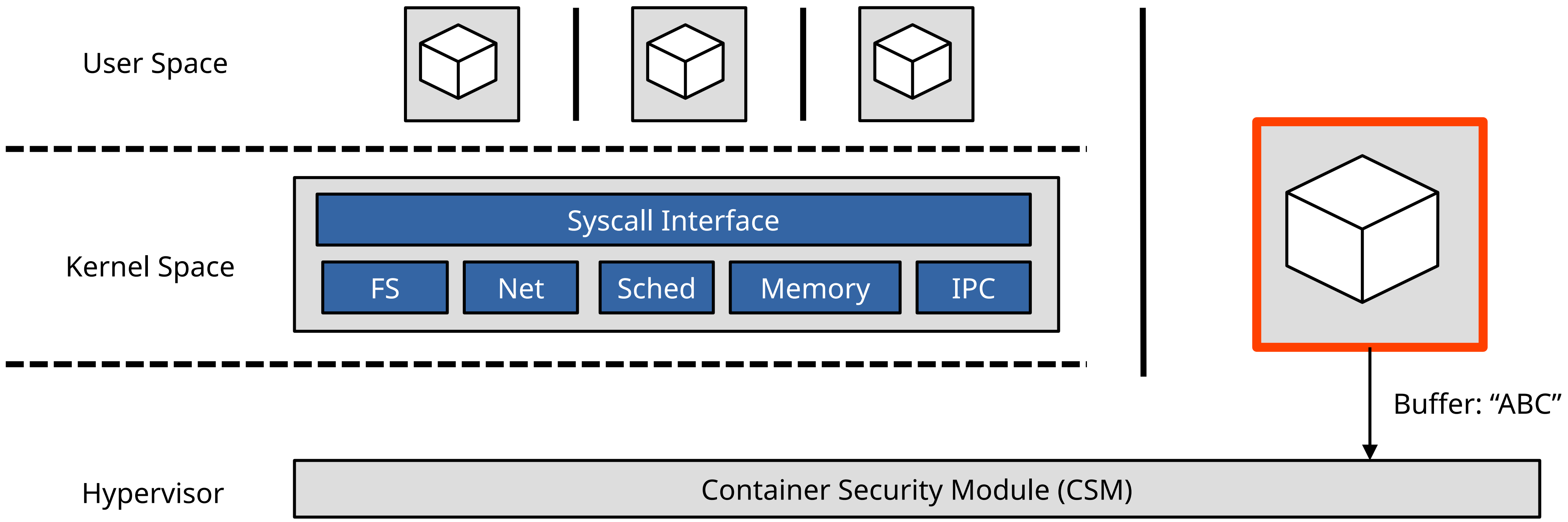
Running Containers as Virtual Machines

- Simple hardware, yet strong isolation possible (BlackBox [VN22])
  - Only needs to trust CSM for security, integrity (~10k LoC)
  - Moderate performance overhead compared to native containers



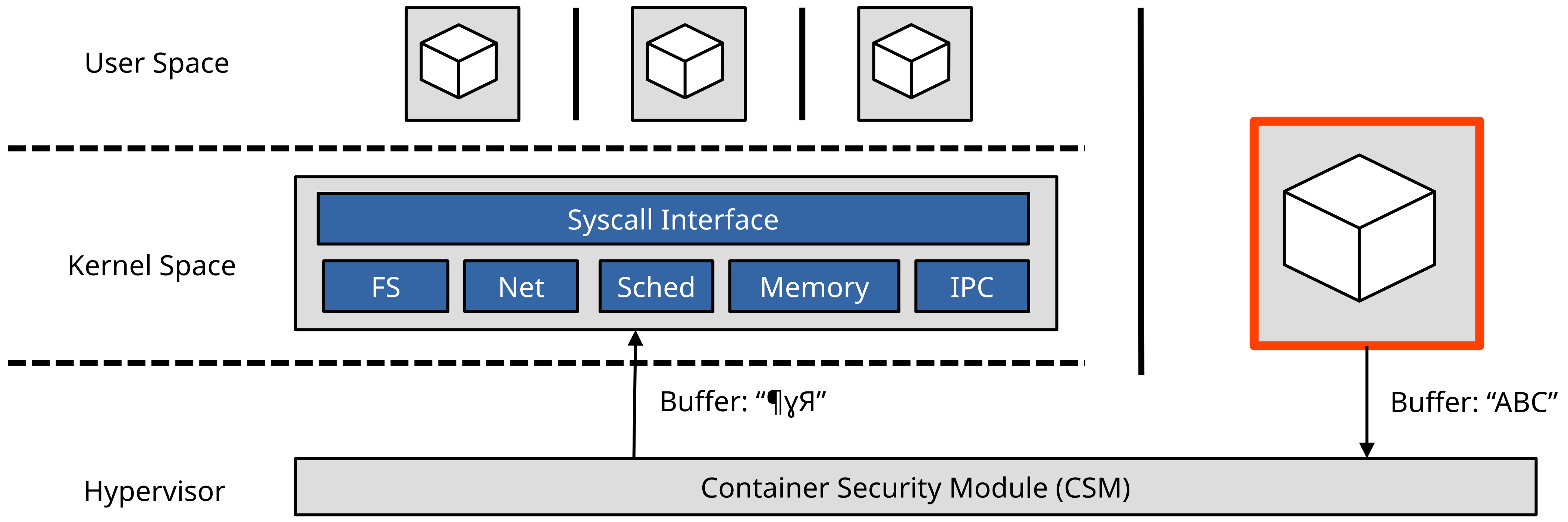
Running Containers as Virtual Machines

- Simple hardware, yet strong isolation possible (BlackBox [VN22])
  - Only needs to trust CSM for security, integrity (~10k LoC)
  - Moderate performance overhead compared to native containers



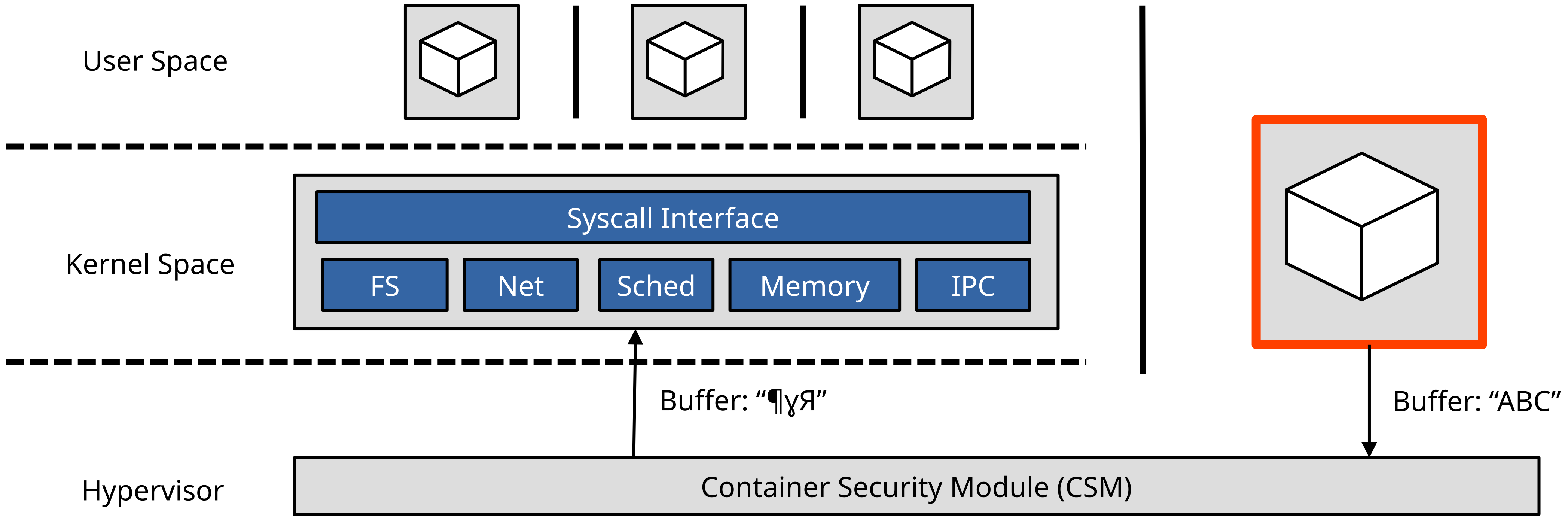
Running Containers as Virtual Machines

- Simple hardware, yet strong isolation possible (BlackBox [VN22])
  - Only needs to trust CSM for security, integrity (~10k LoC)
  - Moderate performance overhead compared to native containers



Running Containers as Virtual Machines

- Simple hardware, yet strong isolation possible (BlackBox [VN22])
  - What about the TCB size regarding availability?



Lightweight Virtual Machines

Virtual Machines in Dynamic Cloud Settings

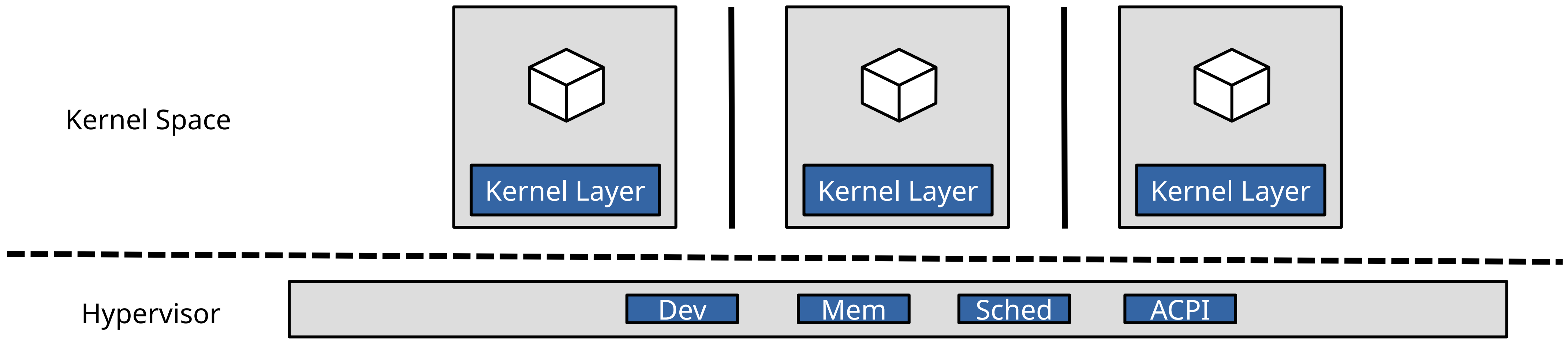
- Strong isolation between VMs
  - Hypervisor has a smaller interface to provide than an OS kernel
  - Hardware acceleration is wide-spread, also for many devices (SR-IOV [YY+08])

- Performance overhead cannot be avoided.
  - Nested paging
  - Device emulation

- Load complete kernel, startup considered slow [ML+17]
  - Partially caused by device initialization
  - Cost for being able to run arbitrary operating systems

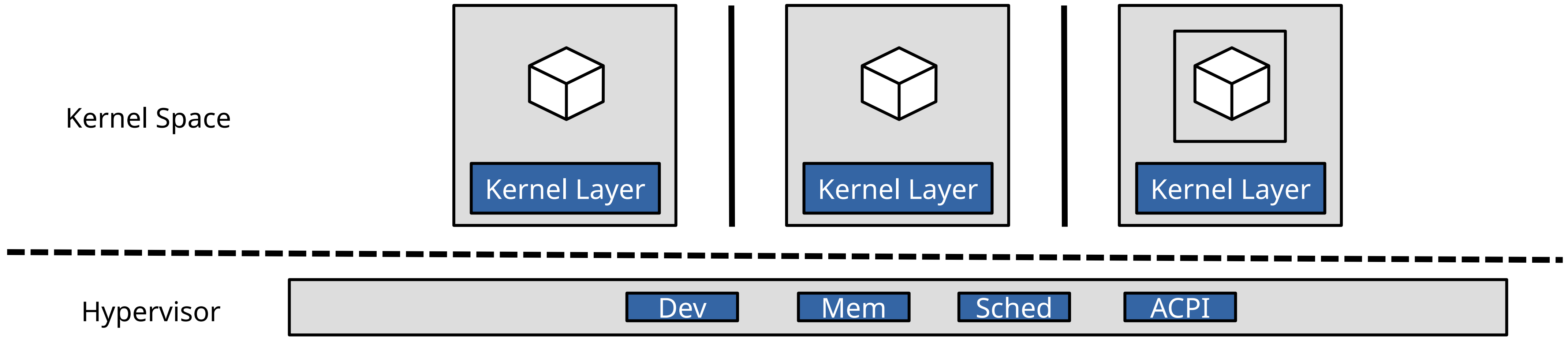
Lightweight Virtual Machines

- Reduce VM startup overhead, but retain isolation (e.g. LightVM [ML+17])
  - Remove unused devices etc.
  - Boot time as low as a couple of milliseconds
- Dedicated hypervisors offering tight feature set
  - E.g., Firecracker [AB+20] (used by AWS)



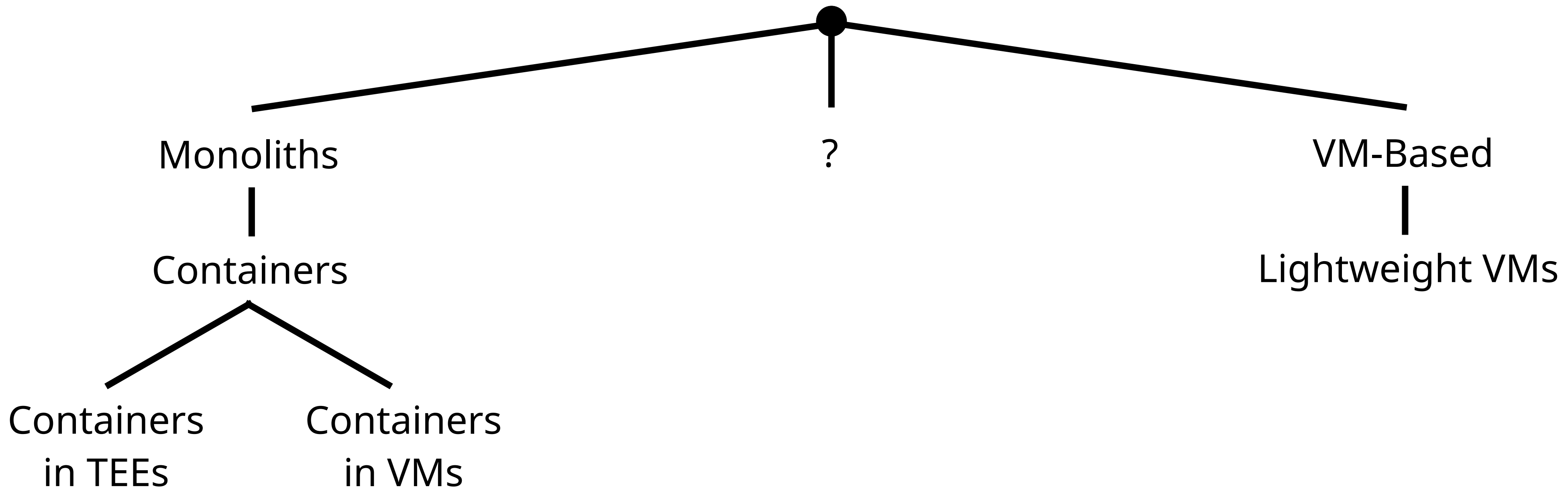
Lightweight Virtual Machines

- Flavor I: Unikernel guests (e.g., Unikraft [KB+21])
  - Performance and TCB advantages possible
- Flavor II: Containers inside lightweight VM (e.g., Kata containers [KC24])
  - Second layer of defense, used with Alibaba Cloud [LC+22]



Current Research: Microkernels as Data Center Substrate

Hardening Operating Systems for the Cloud



Containers on Microkernels

- General idea of microkernels: Minimize code running in kernel mode
  - Drivers, file systems, network etc. implemented in user space
  - Only requires basic CPU architecture (standard virtual memory and threads)

- Modern microkernels use capabilities for access control
  - Processes can only access resources for that they have been granted permission explicitly
  - No ambient authority

- Inter process communication (IPC) important for performance
  - Microkernels by design have performance disadvantages compared to monoliths

Till Miemietz, Viktor Reusch, Matthias Hille, Max Kurze, Adam Lackorzynski, Michael Roitzsch, Hermann Härtig: A Perfect Fit? - Towards Containers on Microkernels. WOC@Middleware 2024

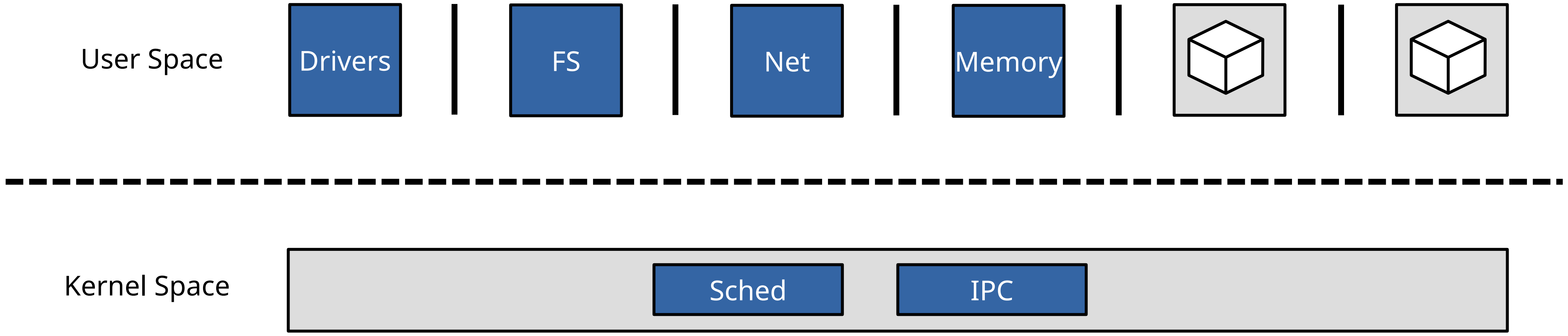
Containers on Microkernels

- What if we used an OS with proper process isolation to begin with?
  - Capability-based access control
  - Small TCB for confidentiality, integrity, **and** availability

Till Miemietz, Viktor Reusch, Matthias Hille, Max Kurze, Adam Lackorzynski, Michael Roitzsch, Hermann Härtig: A Perfect Fit? - Towards Containers on Microkernels. WOC@Middleware 2024

Containers on Microkernels

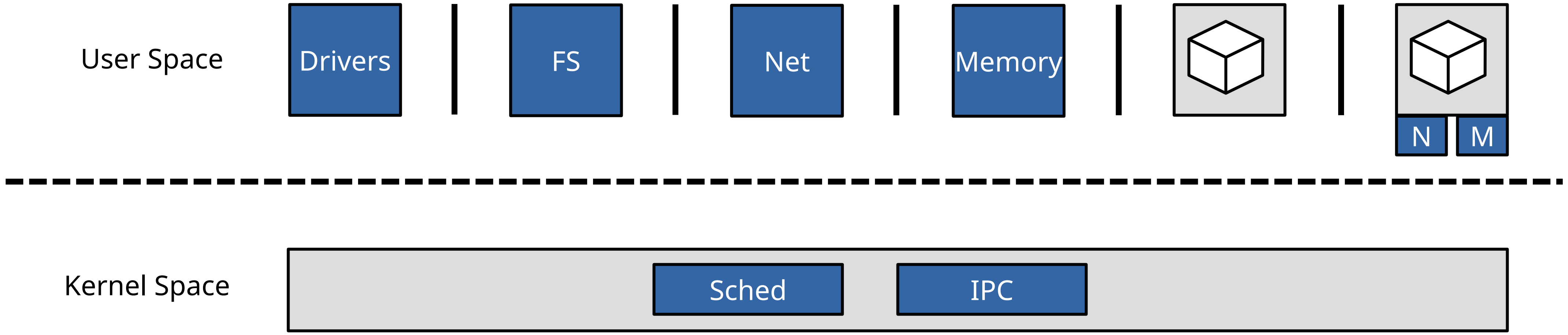
- What if we used an OS with proper process isolation to begin with?
  - Capability-based access control
  - Small TCB for confidentiality, integrity, **and** availability



Till Miemietz, Viktor Reusch, Matthias Hille, Max Kurze, Adam Lackorzynski, Michael Roitzsch, Hermann Härtig: A Perfect Fit? - Towards Containers on Microkernels. WOC@Middleware 2024

Containers on Microkernels

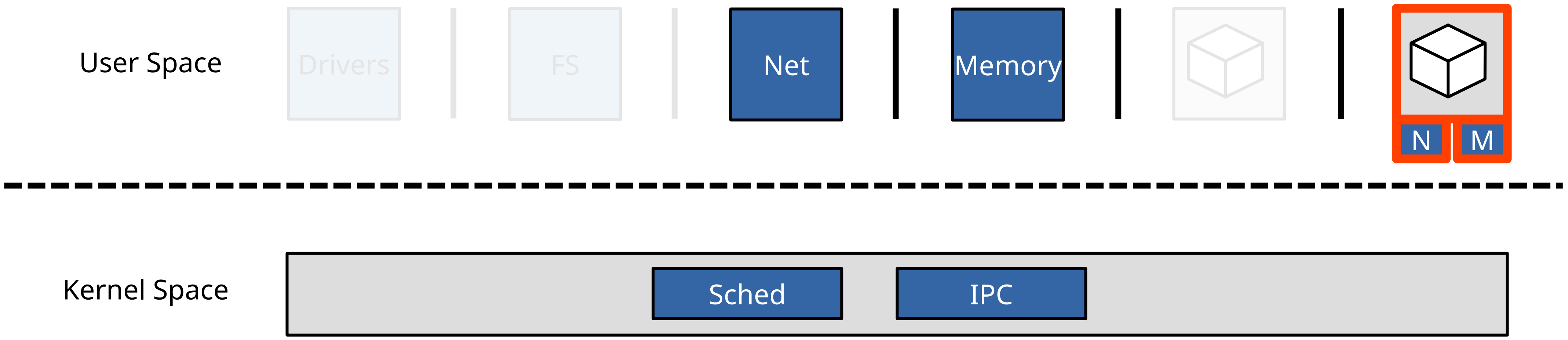
- What if we used an OS with proper process isolation to begin with?
  - Capability-based access control
  - Small TCB for confidentiality, integrity, **and** availability



Till Miemietz, Viktor Reusch, Matthias Hille, Max Kurze, Adam Lackorzynski, Michael Roitzsch, Hermann Härtig: A Perfect Fit? - Towards Containers on Microkernels. WOC@Middleware 2024

Containers on Microkernels

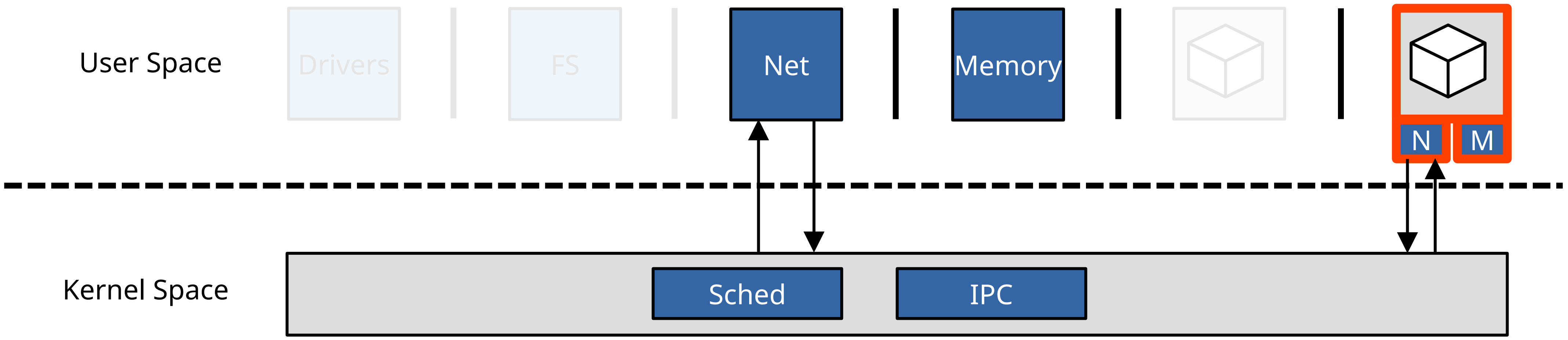
- What if we used an OS with proper process isolation to begin with?
  - Capability-based access control
  - Small TCB for confidentiality, integrity, **and** availability



Till Miemietz, Viktor Reusch, Matthias Hille, Max Kurze, Adam Lackorzynski, Michael Roitzsch, Hermann Härtig: A Perfect Fit? - Towards Containers on Microkernels. WOC@Middleware 2024

Containers on Microkernels

- What if we used an OS with proper process isolation to begin with?
  - Capability-based access control
  - Small TCB for confidentiality, integrity, **and** availability
  - Interaction with system services more expensive



Till Miemietz, Viktor Reusch, Matthias Hille, Max Kurze, Adam Lackorzynski, Michael Roitzsch, Hermann Härtig: A Perfect Fit? - Towards Containers on Microkernels. WOC@Middleware 2024

Containers on Microkernels

- Visibility restrictions
  - Integrated in capability system
  - Namespace abstraction also exists on L4Re

- Interface restrictions: Containers can only access a minimal interface by design
  - There is no ambient authority.

- Resource restrictions
  - Tied to capability
  - To be implemented by individual service

Till Miemietz, Viktor Reusch, Matthias Hille, Max Kurze, Adam Lackorzynski, Michael Roitzsch, Hermann Härtig: A Perfect Fit? - Towards Containers on Microkernels. WOC@Middleware 2024

Does It Work in Practice?

- Simplicity reduces attack surface by orders of magnitude

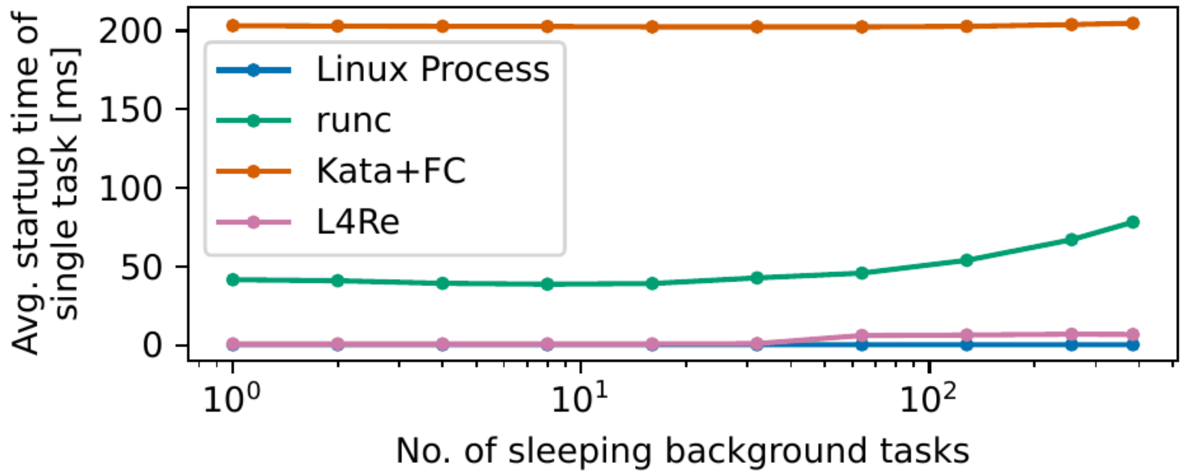
- What about performance?

Till Miemietz, Viktor Reusch, Matthias Hille, Max Kurze, Adam Lackorzynski, Michael Roitzsch, Hermann Härtig: A Perfect Fit? - Towards Containers on Microkernels. WOC@Middleware 2024

Does It Work in Practice?

- Simplicity reduces attack surface by orders of magnitude

- Starting a single container

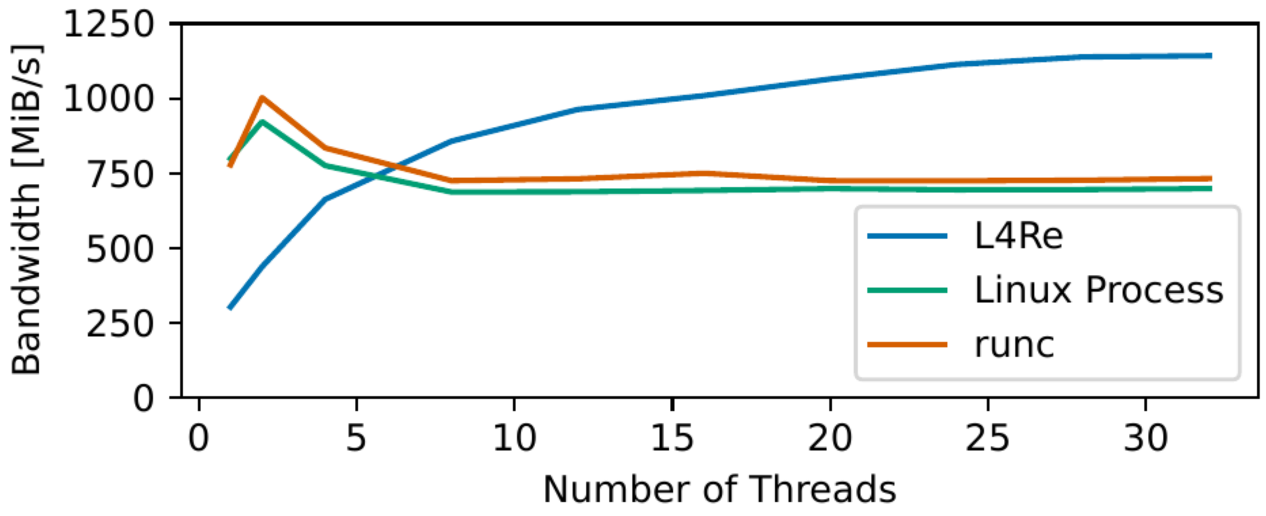


Till Miemietz, Viktor Reusch, Matthias Hille, Max Kurze, Adam Lackorzynski, Michael Roitzsch, Hermann Härtig: A Perfect Fit? - Towards Containers on Microkernels. WOC@Middleware 2024

Does It Work in Practice?

- Simplicity reduces attack surface by orders of magnitude

- I/O performance on a 10G NIC

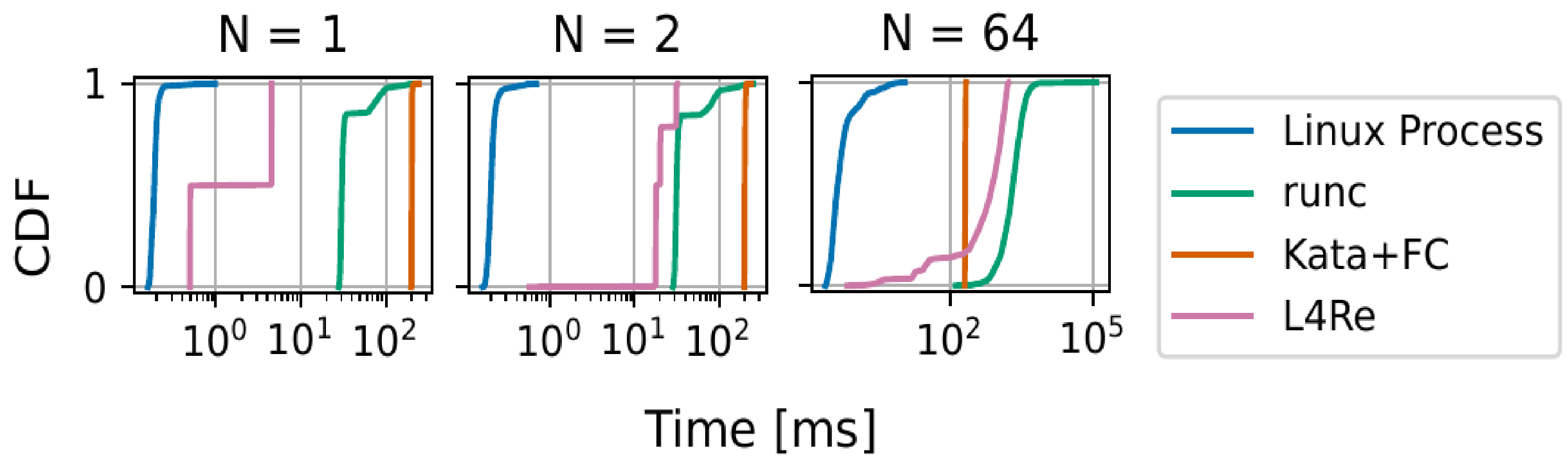


Till Miemietz, Viktor Reusch, Matthias Hille, Max Kurze, Adam Lackorzynski, Michael Roitzsch, Hermann Härtig: A Perfect Fit? - Towards Containers on Microkernels. WOC@Middleware 2024

Does It Work in Practice?

- Simplicity reduces attack surface by orders of magnitude

- Parallel container startup



Till Miemietz, Viktor Reusch, Matthias Hille, Max Kurze, Adam Lackorzynski, Michael Roitzsch, Hermann Härtig: A Perfect Fit? - Towards Containers on Microkernels. WOC@Middleware 2024

Summary: Mechanisms for Isolating Cloud Applications

Approach	Isolation	Small TCB	Low Overhead	Scalability	Compatibility
Monolith	X	X	✓	✓	✓

Summary: Mechanisms for Isolating Cloud Applications

Approach	Isolation	Small TCB	Low Overhead	Scalability	Compatibility
Monolith	X	X	✓	✓	✓
Container	O	X	✓	✓	✓

Summary: Mechanisms for Isolating Cloud Applications

Approach	Isolation	Small TCB	Low Overhead	Scalability	Compatibility
Monolith	X	X	✓	✓	✓
Container	O	X	✓	✓	✓
TEE	✓	X	X	?	O

Summary: Mechanisms for Isolating Cloud Applications

Approach	Isolation	Small TCB	Low Overhead	Scalability	Compatibility
Monolith	X	X	✓	✓	✓
Container	O	X	✓	✓	✓
TEE	✓	X	X	?	O
Light VM	✓	?	O	✓	O

Summary: Mechanisms for Isolating Cloud Applications

Approach	Isolation	Small TCB	Low Overhead	Scalability	Compatibility
Monolith	X	X	✓	✓	✓
Container	O	X	✓	✓	✓
TEE	✓	X	X	?	O
Light VM	✓	?	O	✓	O
Microkernel	✓	✓	?	?	?

Summary: Mechanisms for Isolating Cloud Applications

Approach	Isolation	Small TCB	Low Overhead	Scalability	Compatibility
Monolith	X	X	✓	✓	✓
Container	O	X	✓	✓	✓
TEE	✓	X	X	?	O
Light VM	✓	?	O	✓	O
Microkernel	✓	✓	?	?	?

Summary

- Current isolation mechanisms in dynamic cloud settings complex
  - Requirements: Strong isolation between clients, fast startup, good performance
  - Large attack surface hard to avoid with standard system design

- Capability-based microkernels with strong built-in isolation
  - Container isolation is conceptually simple on a microkernel

- Simplicity is key for building secure systems
  - Holds for both software and hardware
  - Modularity to have the choice of including complex, high-performance implementations in TCB

References for Further Reading I

[KW00] Peter Van Der Kamp and Robert N. M. Watson. Jails: confining the omnipotent root. In Proceedings of the 2nd International SANE Conference, 2000.

[TC04] Andrew Tucker and David Comay. Solaris zones: Operating system support for server consolidation. In Proceedings of the 3rd Virtual Machine Research and Technology Symposium, May 6-7, 2004, San Jose, CA, USA. USENIX, 2004.

[FF+15] Wes Felter, Alexandre Ferreira, Ram Rajamony, and Juan Rubio. An updated performance comparison of virtual machines and linux containers. In 2015 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2015, Philadelphia, PA, USA.

[HG+23] Yi He, Roland Guo, Yunlong Xing, Xijia Che, Kun Sun, Zhuotao Liu, Ke Xu, Qi Li: Cross Container Attacks: The Bewildered eBPF on Clouds. USENIX Security Symposium 2023

[CW+21] Claudio Canella, Mario Werner, Daniel Gruss, and Michael Schwarz. Automating seccomp filter generation for linux applications. In CCSW@CCS'21: Proceedings of the 2021 on Cloud Computing Security Workshop, 2021

[NS+23] Yuki Nakata, Shintaro Suzuki, and Katsuya Matsubara. Reducing attack surface with container transplantation for lightweight sandboxing. In Proceedings of the 14th ACM SIGOPS Asia-Pacific Workshop on Systems, APSys 2023

[AT+16] Sergei Arnautov, Bohdan Trach, Franz Gregor, Thomas Knauth, André Martin, Christian Priebe, Joshua Lind, Divya Muthukumaran, Dan O'Keeffe, Mark Stillwell, David Goltzsche, David M. Eyers, Rüdiger Kapitza, Peter R. Pietzuch, Christof Fetzer: SCONE: Secure Linux Containers with Intel SGX. OSDI 2016

[VN22] Alexander Van't Hof, Jason Nieh: BlackBox: A Container Security Monitor for Protecting Containers on Untrusted Operating Systems. OSDI 2022

References for Further Reading II

[SS+19] Zhiming Shen, Zhen Sun, Gur-Eyal Sela, Eugene Bagdasaryan, Christina Delimitrou, Robbert van Renesse, Hakim Weatherspoon: X-Containers: Breaking Down Barriers to Improve Performance and Isolation of Cloud-Native Containers. ASPLOS 2019

[YZ+10] Ethan G. Young, Pengfei Zhu, Tyler Caraza-Harter, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau: The True Cost of Containing: A gVisor Case Study. HotCloud 2019

[TB+18] Jörg Thalheim, Pramod Bhatotia, Pedro Fonseca, Baris Kasikci: Cntr: Lightweight OS Containers. USENIX ATC 2018

[ML+17] Filipe Manco, Costin Lupu, Florian Schmidt, Jose Mendes, Simon Kuenzer, Sumit Sati, Kenichi Yasukata, Costin Raiciu, Felipe Huici: My VM is Lighter (and Safer) than your Container. SOSP 2017

[YY+08] Yaozu Dong, Zhao Yu, Greg Rose: SR-IOV Networking in Xen: Architecture, Design and Implementation. Workshop on I/O Virtualization 2008

[KB+21] Simon Kuenzer, Vlad-Andrei Badoiu, Hugo Lefeuvre, Sharan Santhanam, Alexander Jung, Gaulthier Gain, Cyril Soldani, Costin Lupu, Stefan Teodorescu, Costi Raducanu, Cristian Banu, Laurent Mathy, Razvan Deaconescu, Costin Raiciu, Felipe Huici: Unikraft: fast, specialized unikernels the easy way. EuroSys 2021

[AB+20] Alexandru Agache, Marc Brooker, Alexandra Iordache, Anthony Liguori, Rolf Neugebauer, Phil Piwonka, Diana-Maria Popa: Firecracker: Lightweight Virtualization for Serverless Applications. NSDI 2020

[KC24] OpenInfra Foundation. Kata containers. https://katacontainers.io/, 2024. [Online; last accessed on November 28, 2024]

[LC+22] Zijun Li, Jiagan Cheng, Quan Chen, Eryu Guan, Zizheng Bian, Yi Tao, Bin Zha, Qiang Wang, Weidong Han, Minyi Guo: RunD: A Lightweight Secure Container Runtime for High-density Deployment and High-concurrency Startup in Serverless Computing. USENIX ATC 2022