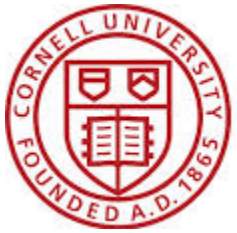


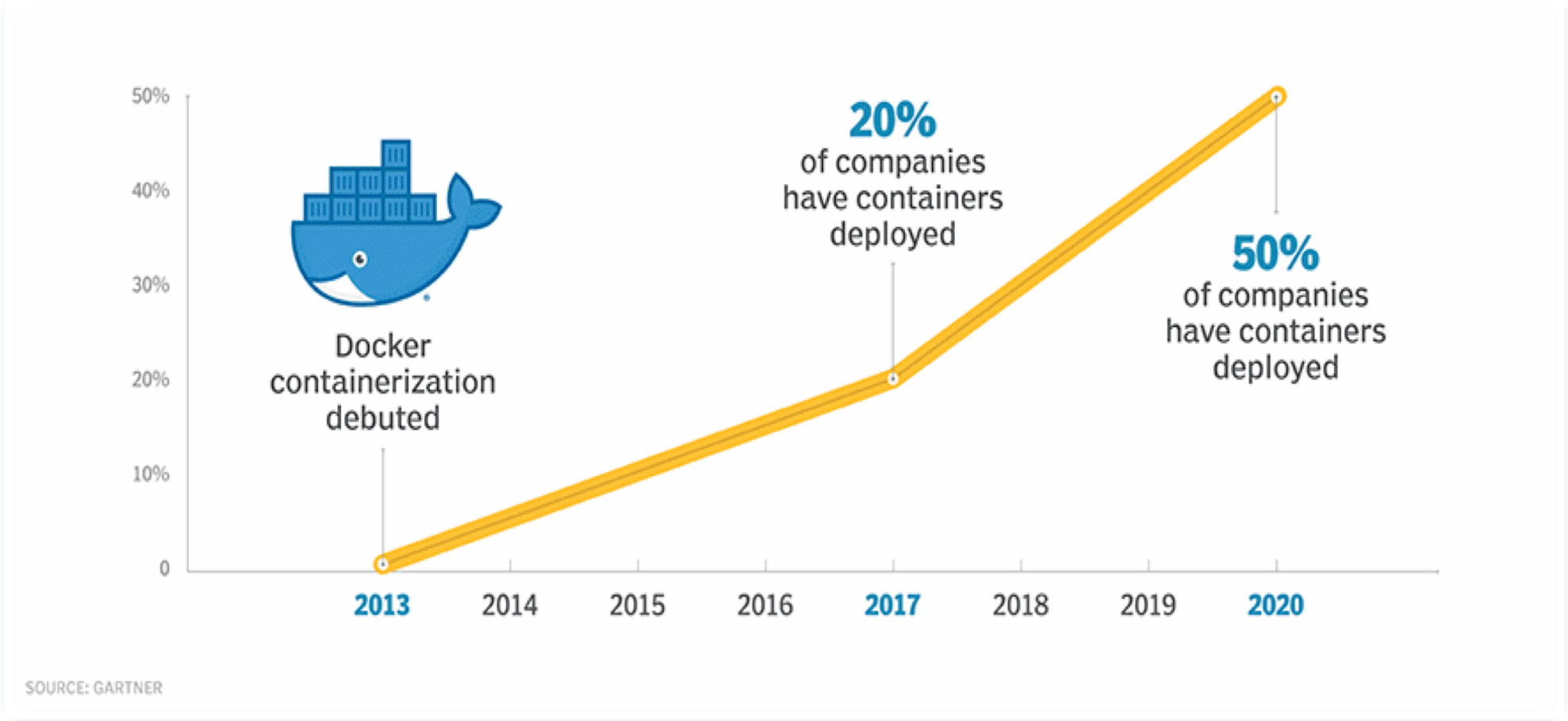
Cornell University

X-Containers: Breaking Down Barriers to Improve Performance and Isolation of Cloud-Native Containers

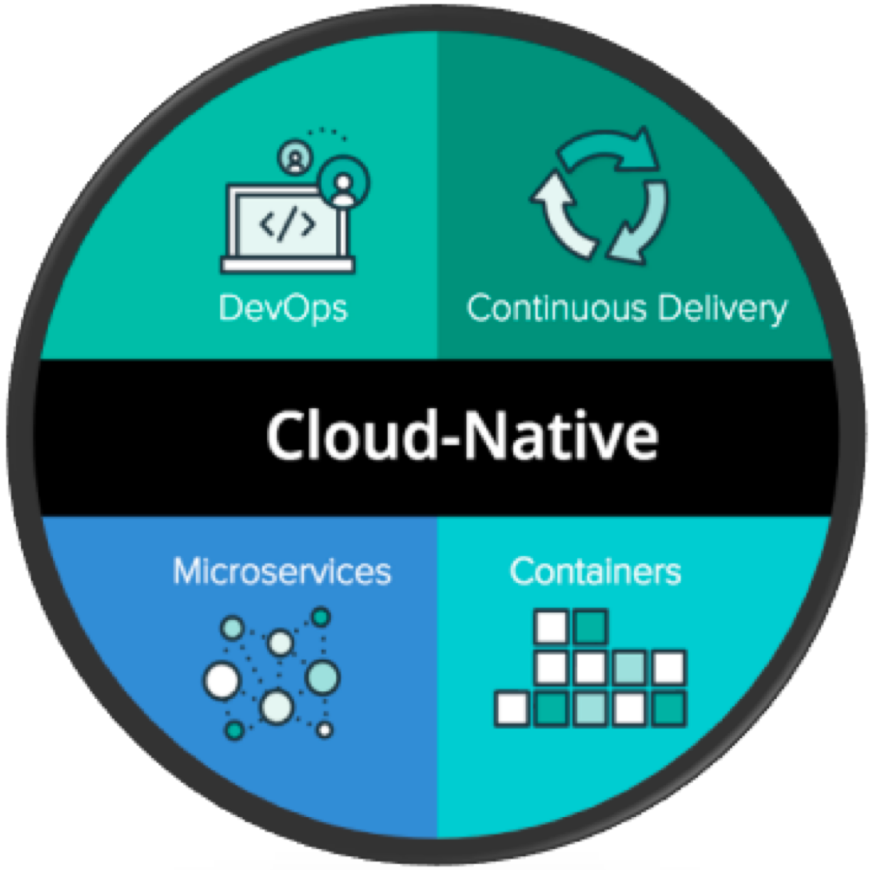
Zhiming Shen
Cornell University

Joint work with Zhen Sun, Gur-Eyal Sela, Eugene Bagdasaryan,
Christina Delimitrou, Robbert Van Renesse, Hakim Weatherspoon

Software Containers



Cloud-Native Container Platforms



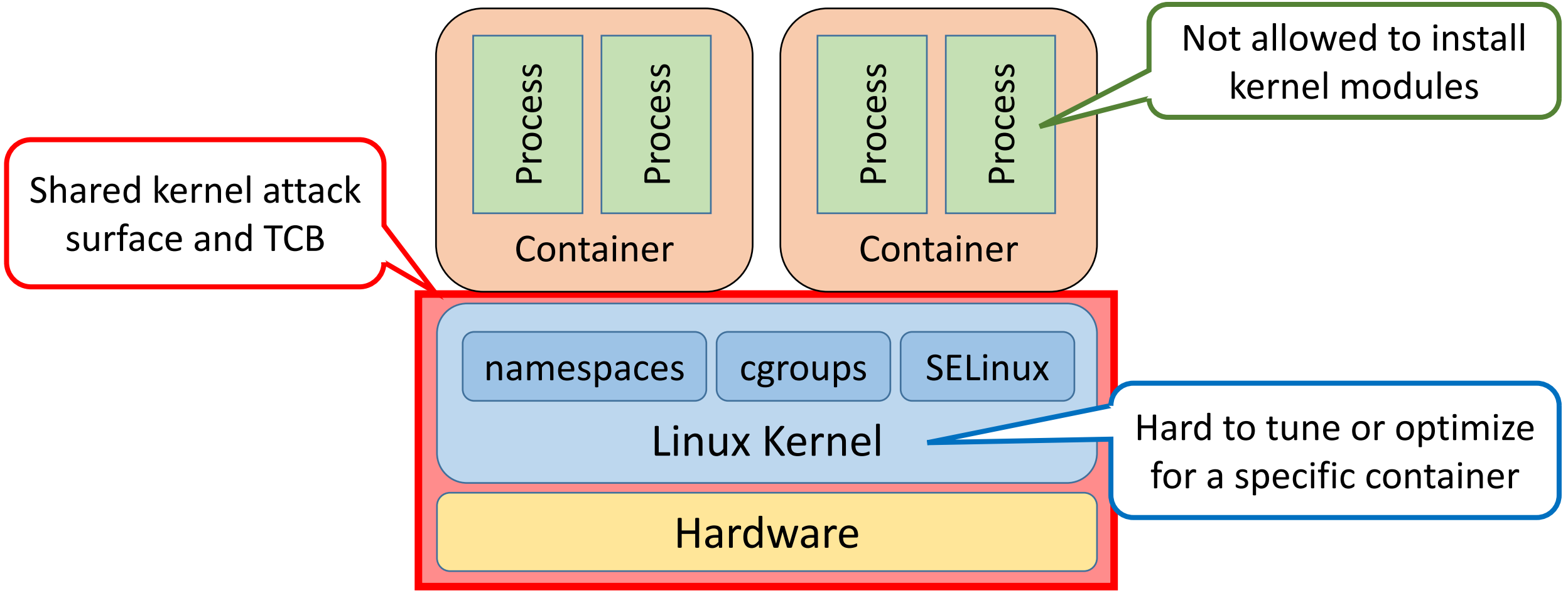
Img src: https://pivotal.io/cloud-native

- Single Concern Principle:

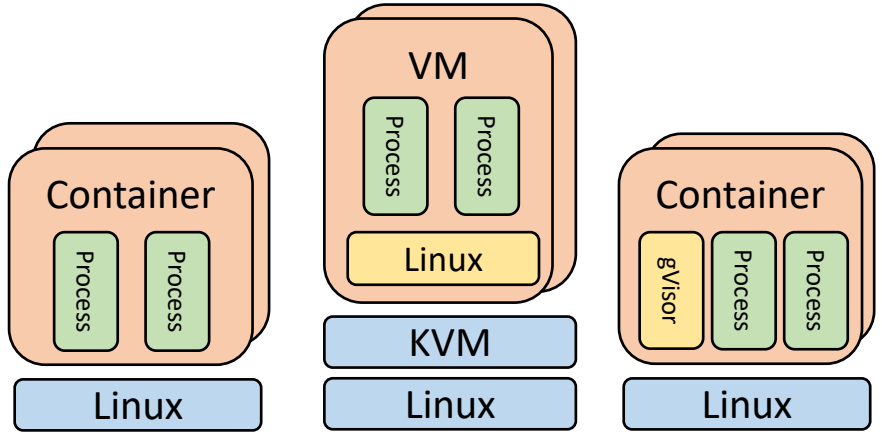
Every container should address a single concern and do it well.

- Making containers easier to
  - Replace, reuse, and upgrade transparently
  - Scale horizontally
  - Debug and troubleshoot

The Problem


Process
Process
Container
Process
Process
Container
Not allowed to install kernel modules
Shared kernel attack surface and TCB
namespaces
cgroups
SELinux
Linux Kernel
Hard to tune or optimize for a specific container
Hardware

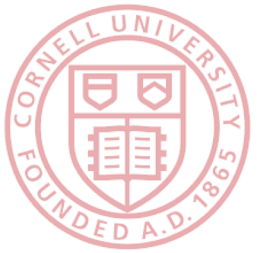
Existing Solutions



Isolation
Customization
Optimization
Portability
Performance

Require *nested* hardware virtualization support in the cloud

Ptrace mode: high overhead
KVM mode: require nested virtualization



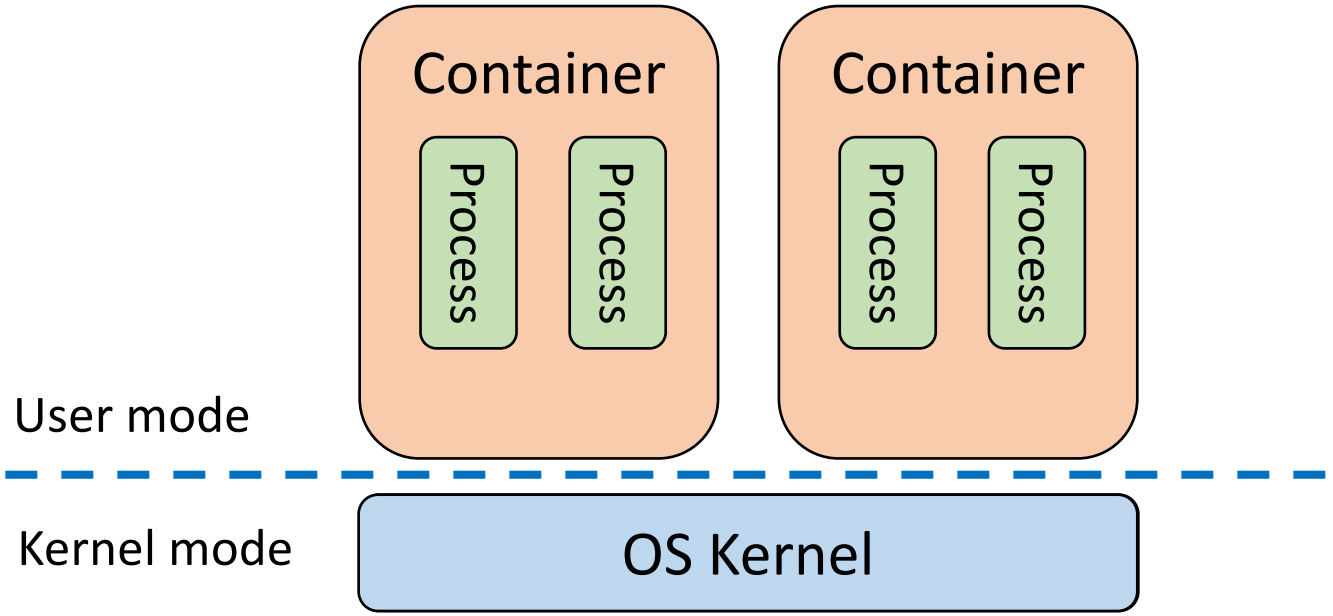
X-Containers achieve

- VM-level **Isolation**
- Support of Kernel **Customization**
- Support of Kernel **Optimization**
- Good **Portability** (without the need of hardware-assisted virtualization)
- High **Performance**

AND

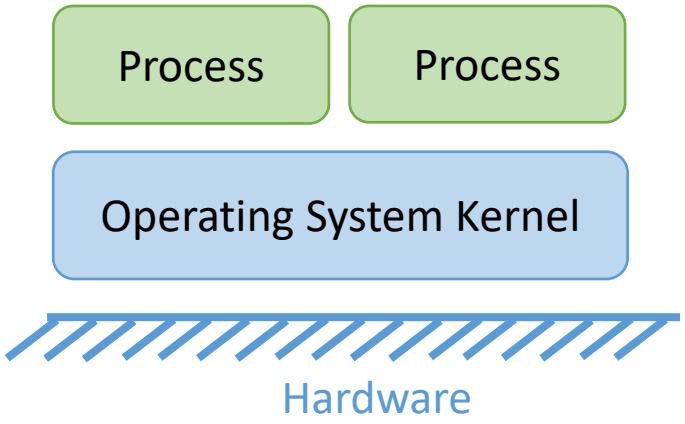
- Backward **Compatibility**

X-Containers

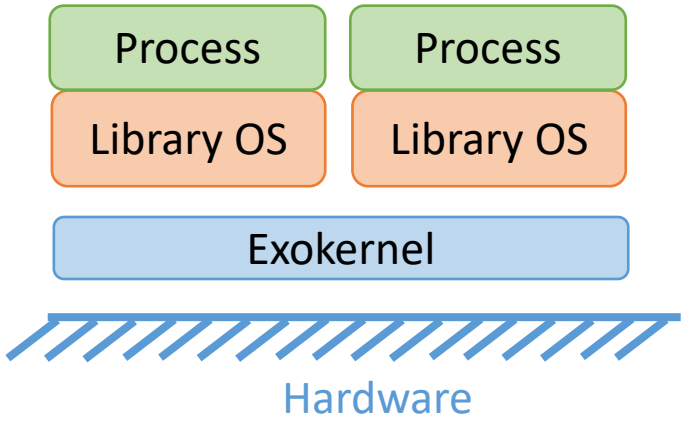


The Exokernel Approach

- Separating protection and management

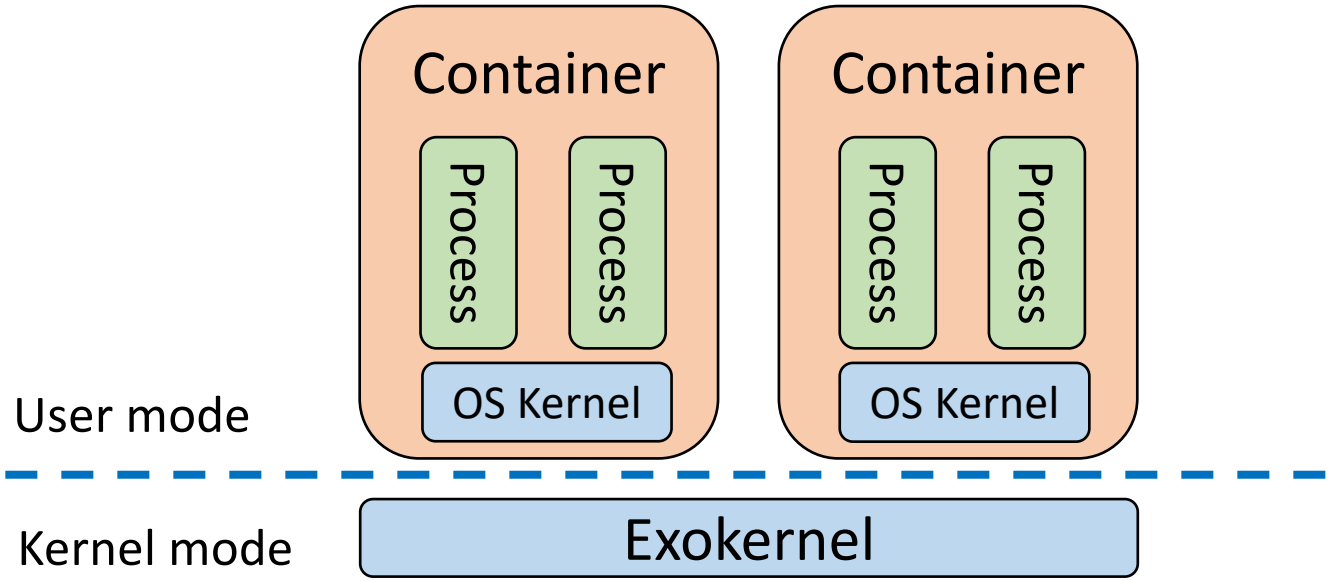


Monolithic OS Kernel

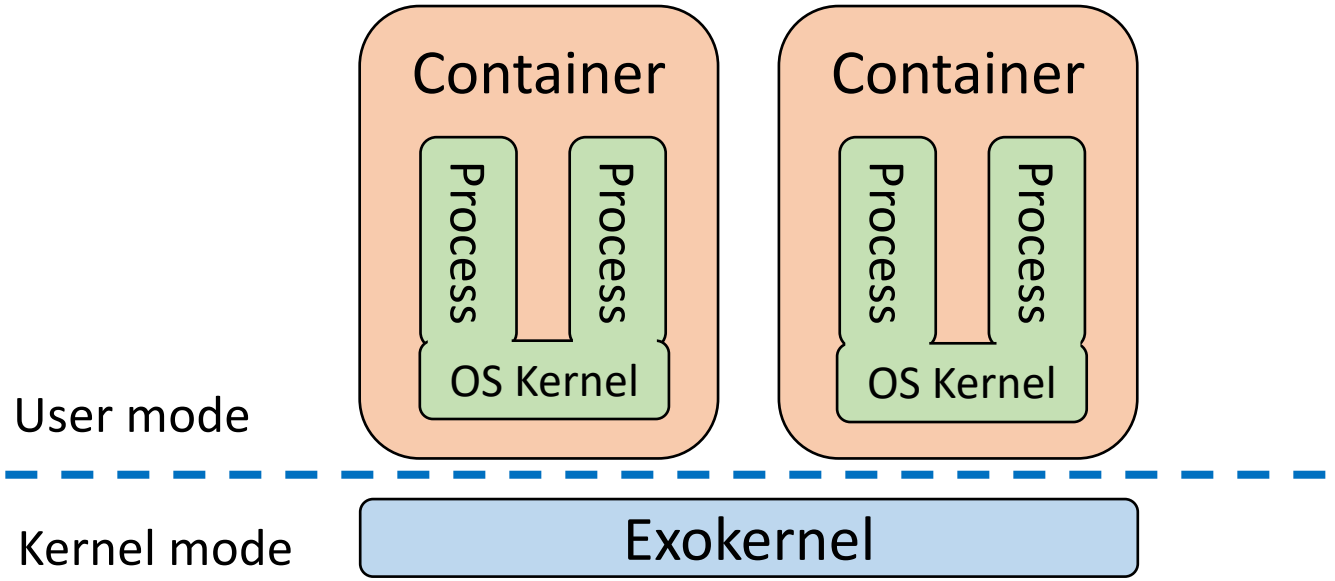


Exokernel

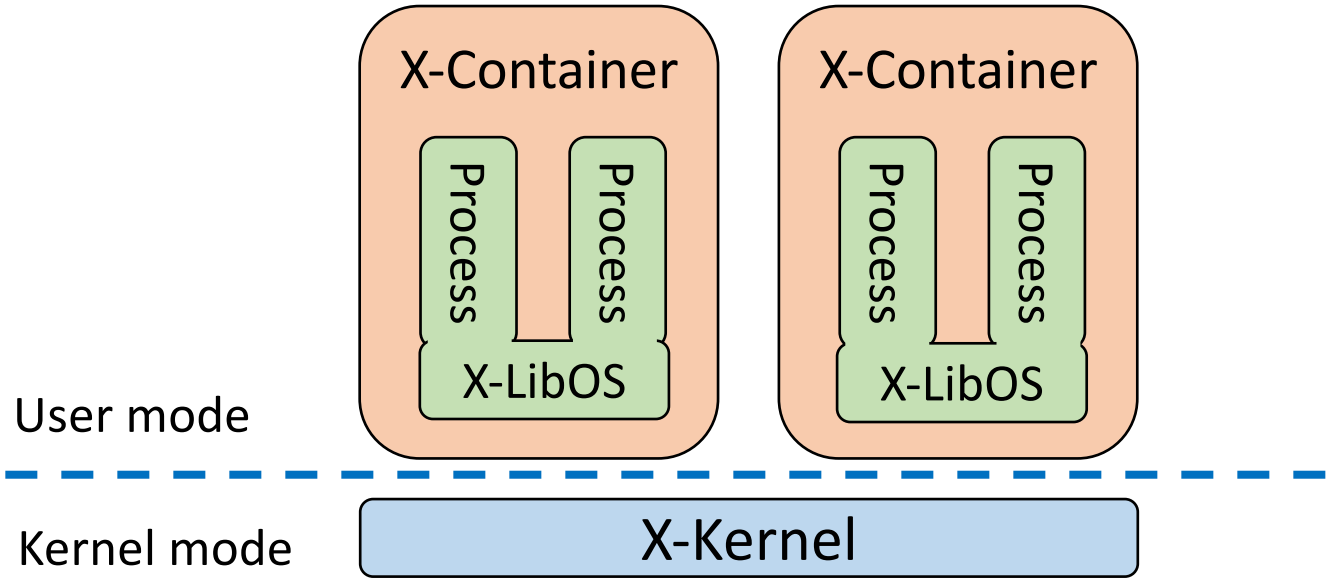
X-Containers



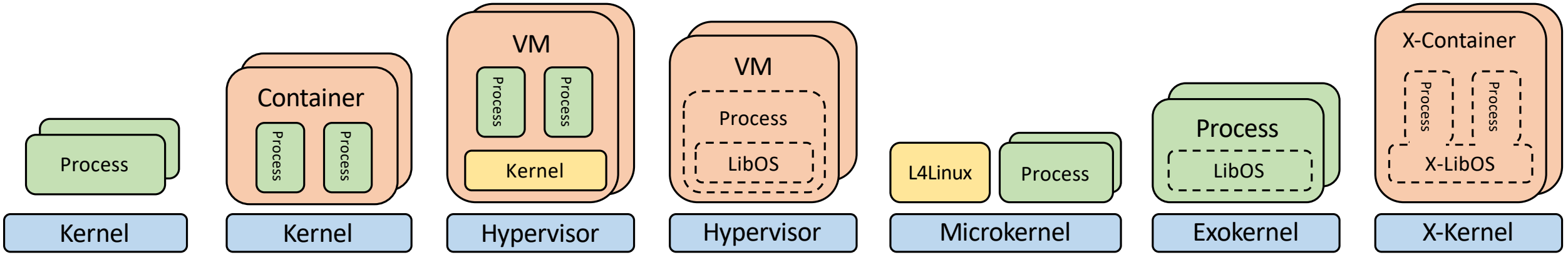
X-Containers



X-Containers



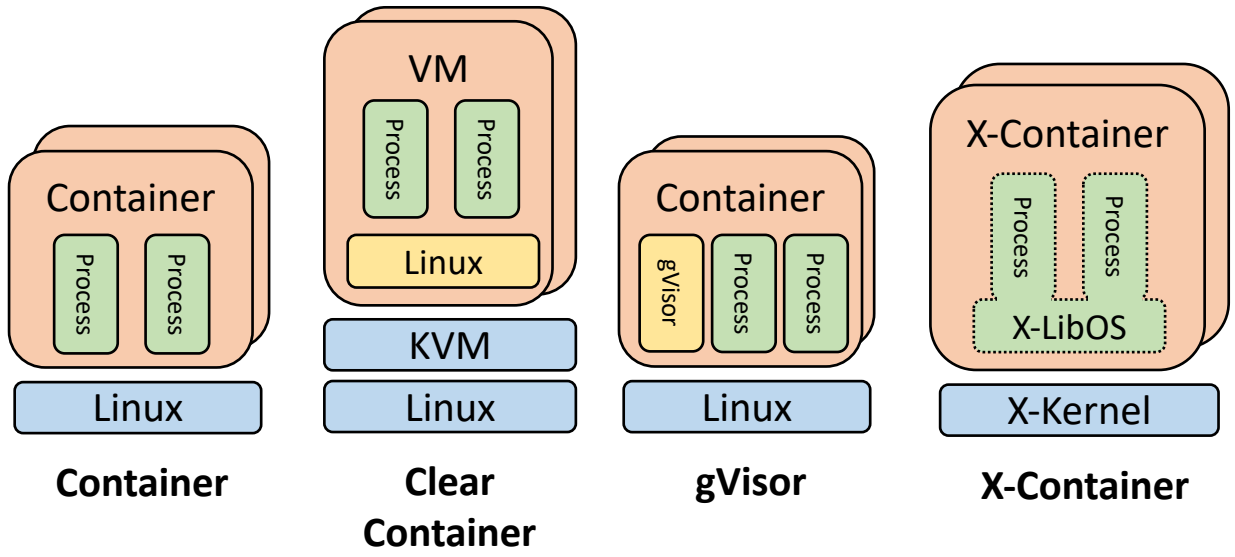
Comparing Isolation Boundaries



Process | Container | Virtual Machine | Unikernel, Dune, EbbRT, OSᵛ | L4Linux (Microkernel) | Library OS (Exokernel) | X-Container

X-Containers

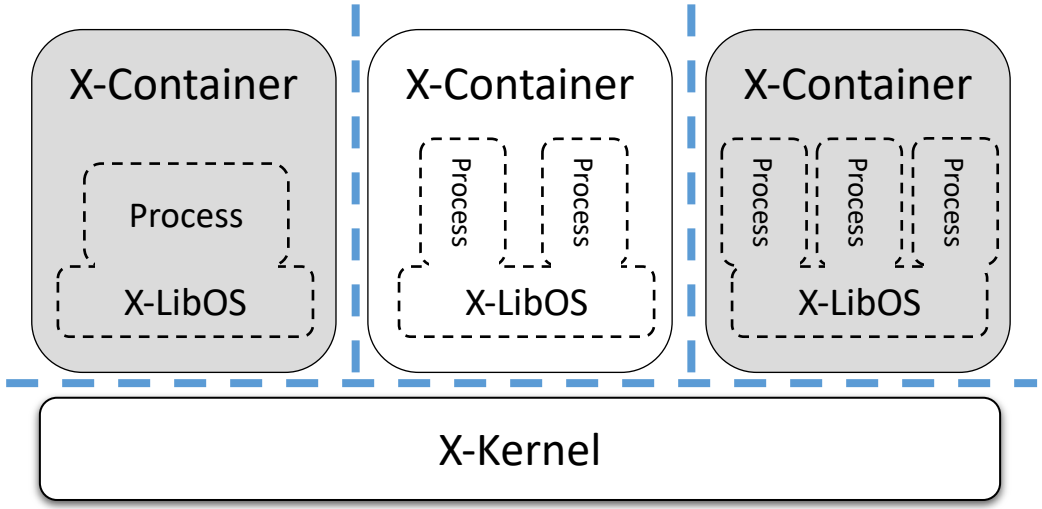
- A new security paradigm for cloud-native containers



- X-Kernel: an exokernel with a small attack surface and TCB
- X-LibOS: a LibOS that decouples security isolation from the process model

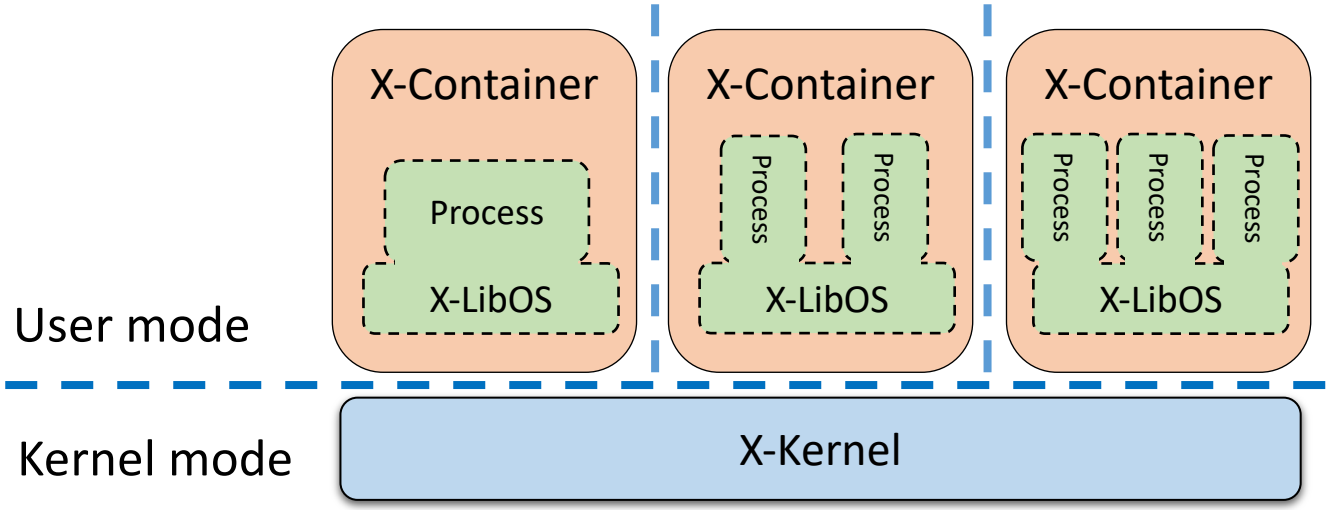
Threat Model and Design Trade-offs

- Threat model



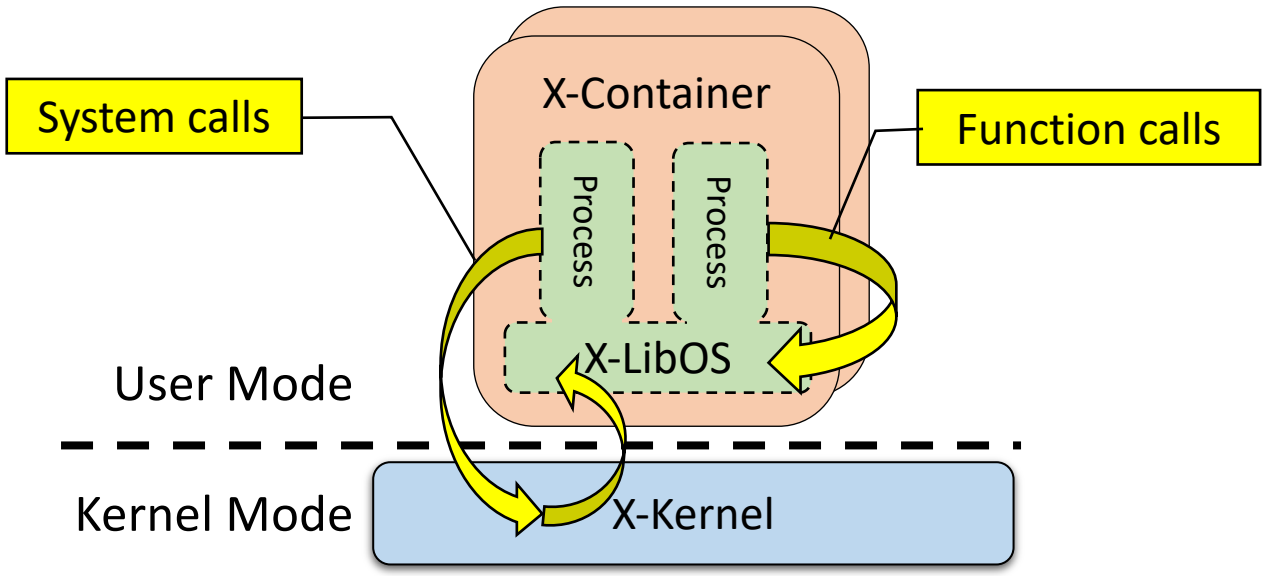
- Trade-offs
  - Reduced intra-container isolation
  - Improved inter-container isolation and performance
  - Process isolation and kernel-supported security features are not effective

Implementation



- X-LibOS from Linux kernel
  - Binary compatibility
  - Highly customizable
- X-Kernel from Xen
  - Para-virtualization interface
  - Concurrent multi-processing
- Limitations
  - Memory management
  - Spawning time

Optimizing System Calls



- Existing solutions
  - Patch source code
  - Link to another library
- Our solution
  - Automatic Binary Optimization Module (ABOM)
  - Binary level equivalence
  - Position-independence

For many applications, more than 90% of syscalls are turned into function calls

Automatic Binary Optimization Module (ABOM)

```
00000000000eb6a0 <__read>:
eb6a9:       b8 00 00 00 00          mov    $0x0,%eax
eb6ae:       0f 05                   syscall
```

7-Byte Replacement (Case 1)

```
00000000000eb6a0 <__read>:
eb6a9:       ff 14 25 08 00 60 ff    callq  *0xffffffffff600008
```

```
000000000007f400 < syscall.Syscall>:
7f41d:       48 8b 44 24 08          mov    0x8(%rsp),%eax
7f422:       0f 05                   syscall
```

7-Byte Replacement (Case 2)

```
000000000007f400 < syscall.Syscall>:
7f41d:       ff 14 25 08 0c 60 ff    callq  *0xffffffffff600c08
```

```
0000000000010330 <__restore_rt>:
10330:       48 c7 c0 0f 00 00 00    mov    $0xf,%rax
10337:       0f 05                   syscall
```

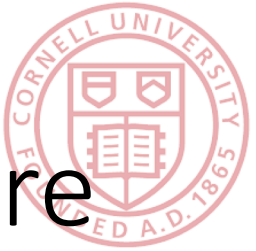
9-Byte Replacement (Phase-1)

```
0000000000010330 <__restore_rt>:
10330:       ff 14 25 80 00 60 ff    callq  *0xffffffffff600080
10337:       0f 05                   syscall
```

9-Byte Replacement (Phase-2)

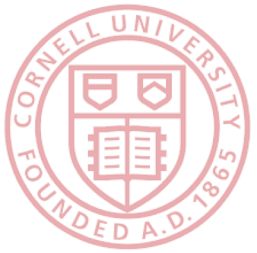
```
0000000000010330 <__restore_rt>:
10330:       ff 14 25 80 00 60 ff    callq  *0xffffffffff600080
10337:       eb f7                   jmp 0x10330
```

Pros and Cons of the X-Container Architecture



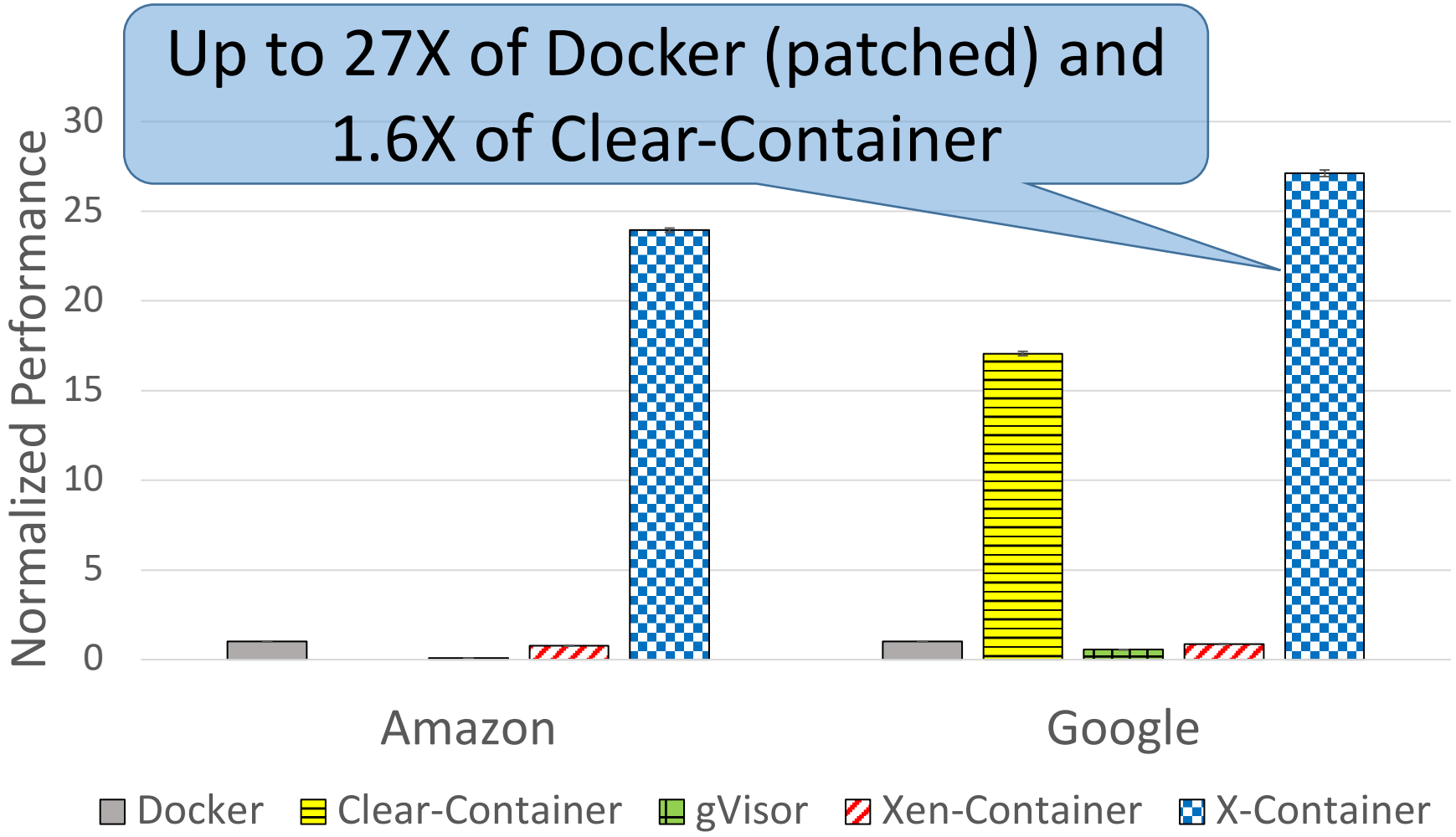
	Container	gVisor	Clear-Container	LightVM	X-Container
Inter-container isolation	Poor	Good	Good	Good	Good
System call performance	Limited	Poor	Limited	Poor	Good
Portability	Good	Good	Limited	Good	Good
Compatibility	Good	Limited	Good	Good	Good
Intra-container isolation	Good	Good	Good	Good	Reduced
Memory efficiency	Good	Good	Limited	Limited	Limited
Spawning time	Short	Short	Moderate	Moderate	Moderate
Software licensing	Clean	Clean	Clean	Clean	Need discussion

Evaluation Setup

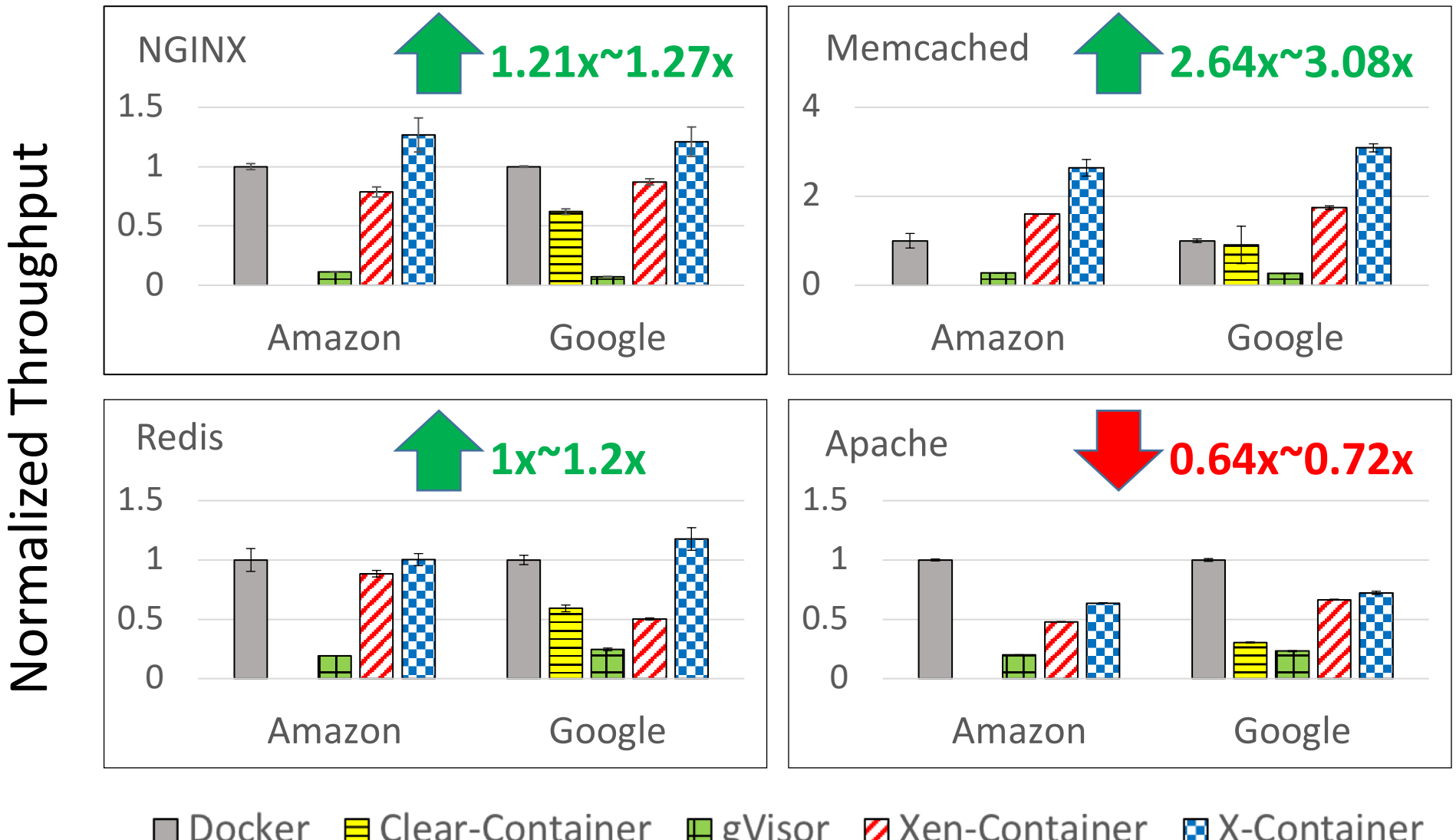


- Testbed
  - Amazon EC2
  - Google Compute Engine
- Compared container runtimes
  - Docker
  - gVisor (Ptrace in Amazon, and KVM in Google)
  - Clear-Container (only in Google)
  - Xen-Container
  - X-Container
- Configurations
  - Patched for Meltdown

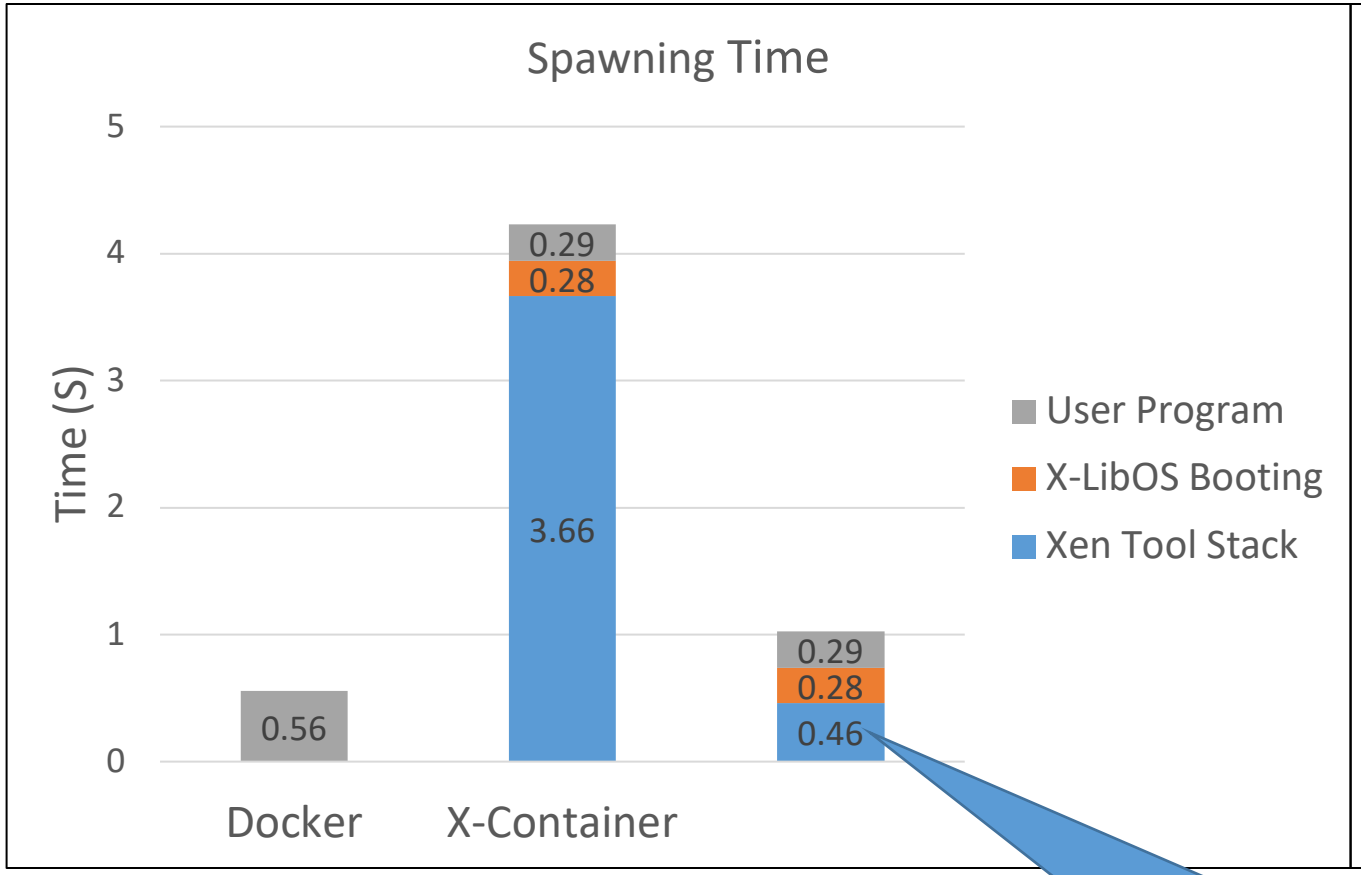
System Call Performance

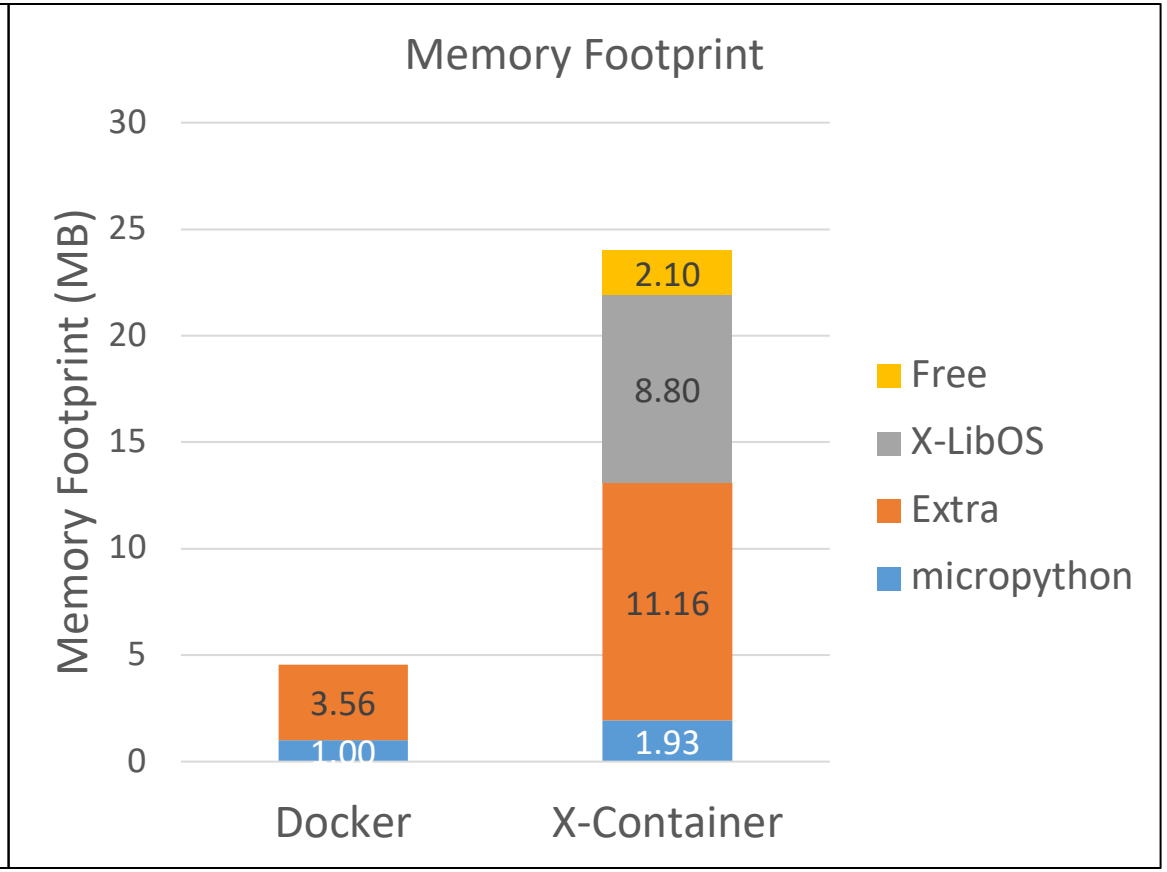


Real Application Performance



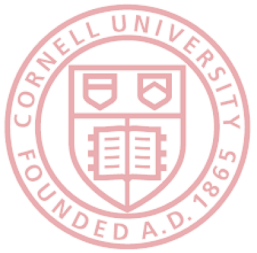
Spawning Time and Memory Footprint





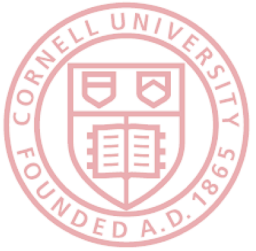
Reduced to 460ms.
Can be further reduced to <10ms.

More Evaluations in the Paper



- More micro/macro benchmarks
- Patched and unpatched for Meltdown
- Comparing to Unikernel and Graphene
- Scalability (up to 400 containers on a single host)

Conclusion



- X-Containers: a new security paradigm for isolating single-concerned cloud-native containers
  - X-Kernel: an exokernel with a small attack surface and TCB
  - X-LibOS: A LibOS that decouples security isolation from the process model
  - Trade-off: intra-container isolation vs. inter-container isolation
- Implemented with Xen and Linux
  - Binary compatibility
  - Concurrent multi-processing
- More at http://x-containers.org

Thank You. Questions?