Achieving Both Security of VMs and Speed of Containers in Cloud Native

Hui Lu

Assistant Professor, SUNY Binghamton

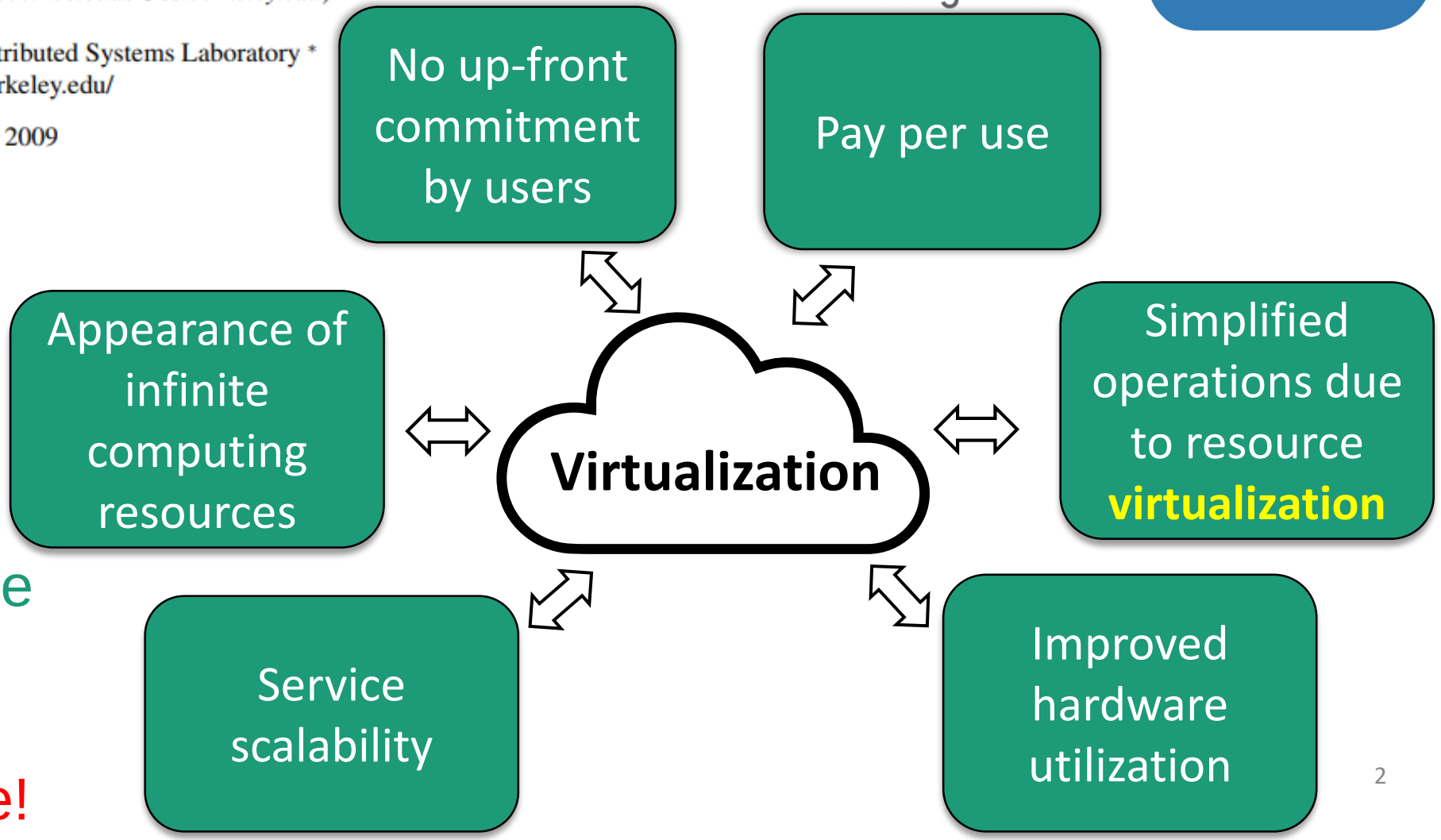
Cloud Computing 1.0

Above the Clouds: A Berkeley View of Cloud Computing

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz,
Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia
(Comments should be addressed to abovetheclouds@cs.berkeley.edu)

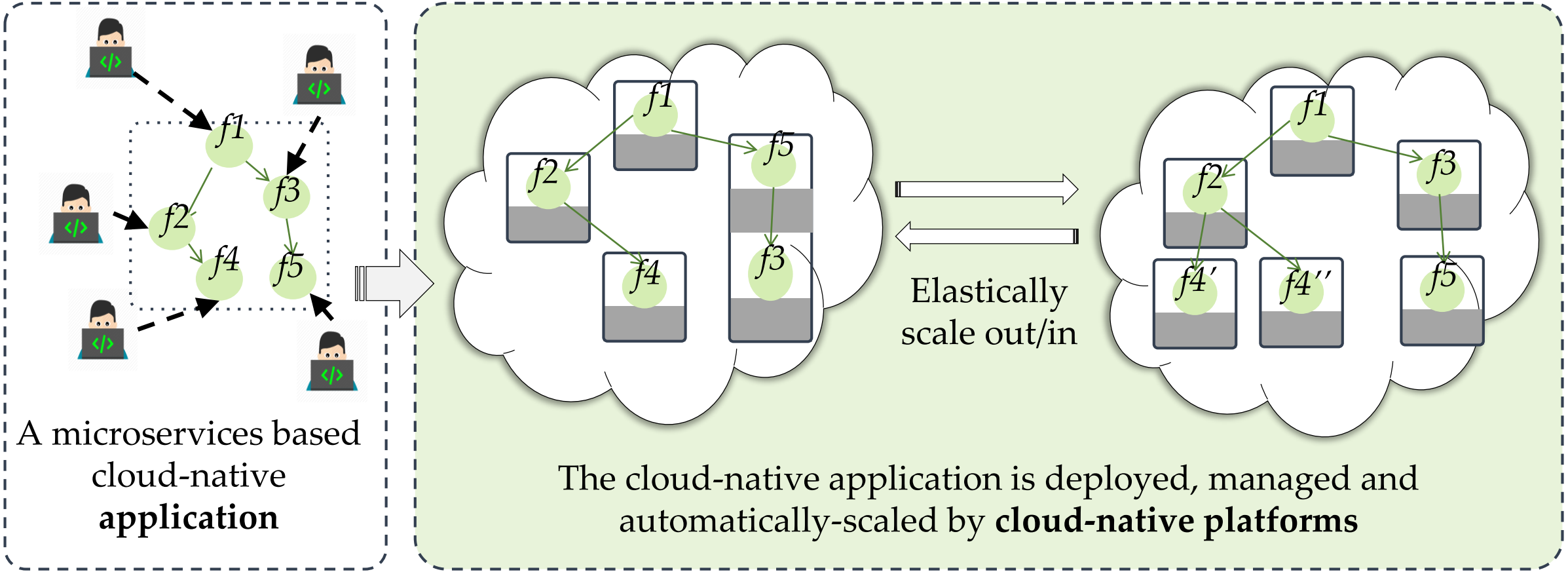
UC Berkeley Reliable Adaptive Distributed Systems Laboratory *
http://radlab.cs.berkeley.edu/

February 10, 2009



Relieved users of physical infrastructure but left them with various virtual resources to manage!

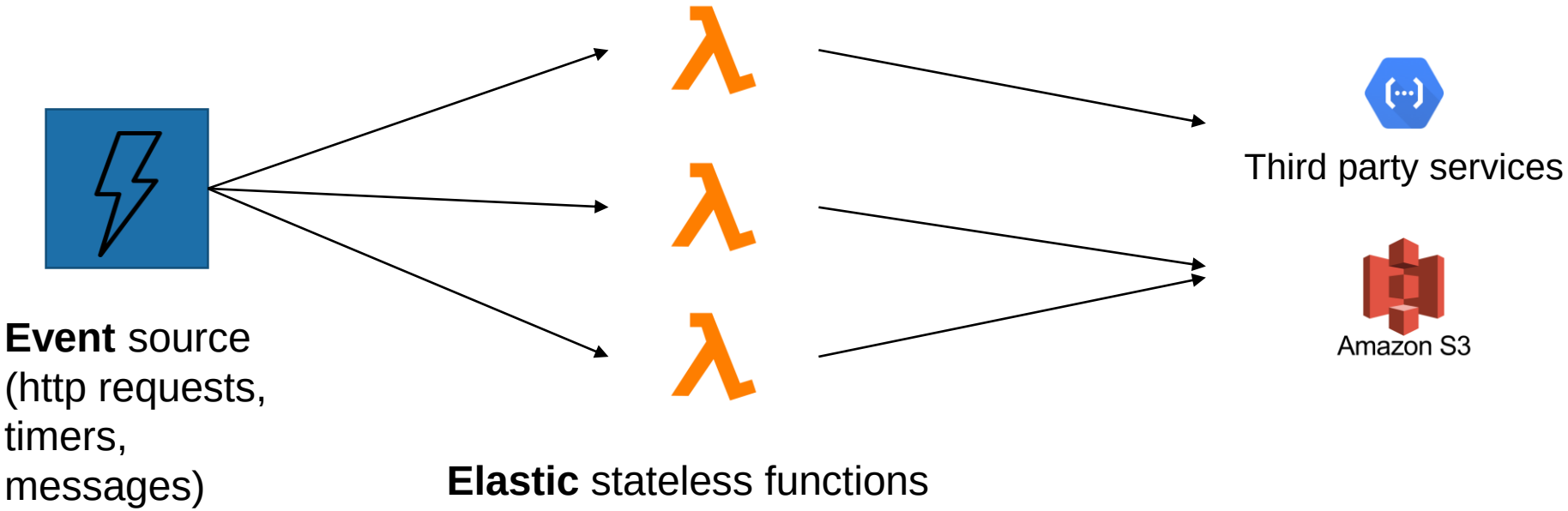
Cloud Native



Users focus on application development, while (virtual) resources are fully managed by cloud provider.

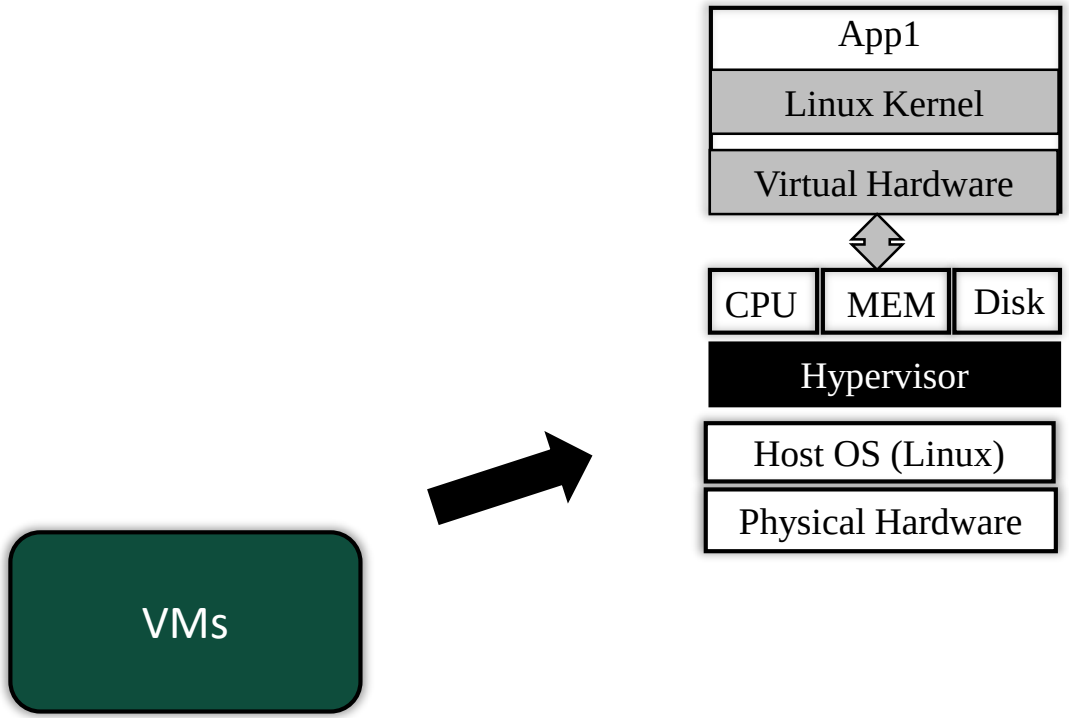
An example: Serverless Computing

- Program cloud applications as suites of loosely-coupled components
- Attractive attention (from cloud users)
  - no provisioning, infinite elasticity, fine-grained pay-per-use
- Most popular offering: Function-as-a-service (e.g., Lambda functions)



Isolation - VMs

- Cloud native relies on performance and security isolation to make multi-tenant hardware sharing possible.



Strong isolation

- A narrow attack interface between VMs and hypervisor
- Defense in depth including both guest OS and hypervisor protection

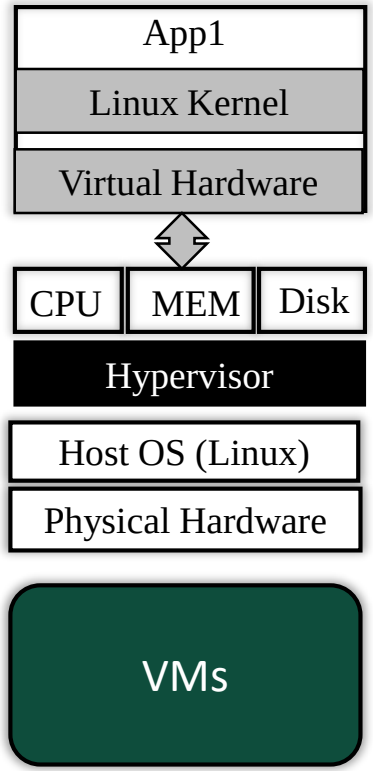
High overhead due to

- Multiple software layers

Strong isolation
High overhead

Weak isolation
Low overhead

Containers

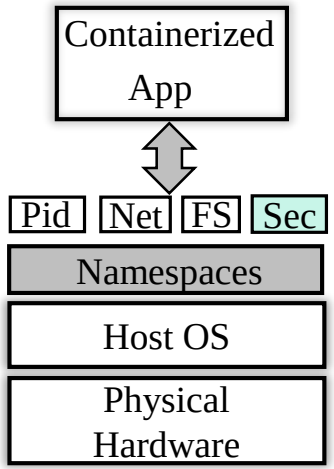


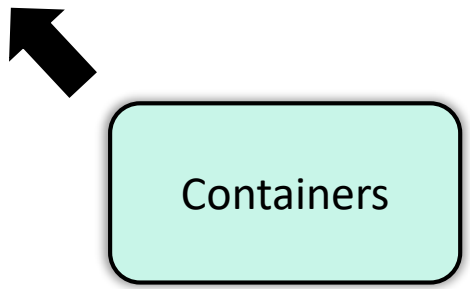
High performance

- Running directly upon native host
- Close to native performance

Weak isolation

- Wide interface between containerized applications and native host (i.e., via system calls)





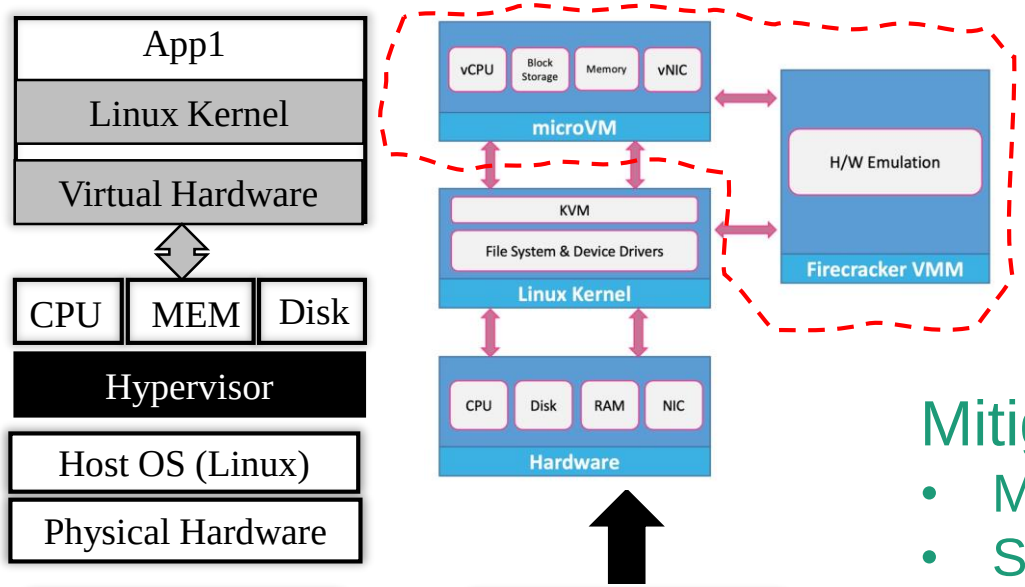
Strong isolation

High overhead

Weak isolation

Low overhead

Isolation - MicroVMs



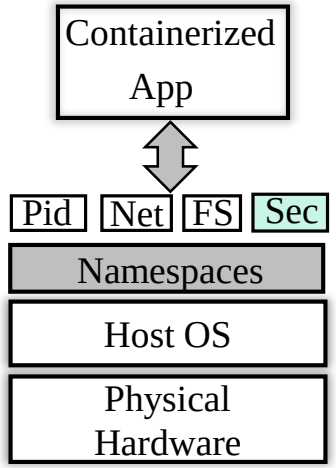
VMs

microVMs (Firecracker)

Mitigated overhead

- Minimized guest functionality
- Simplified I/O model

No layer reduction



Containers

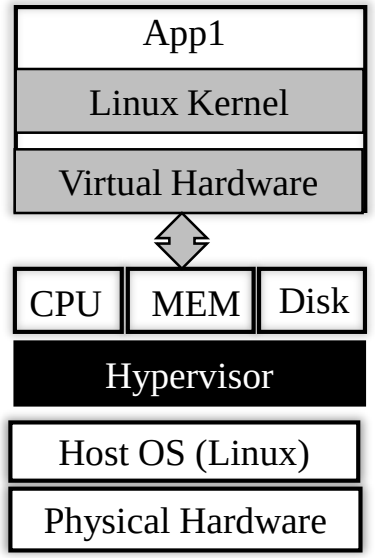
Strong isolation

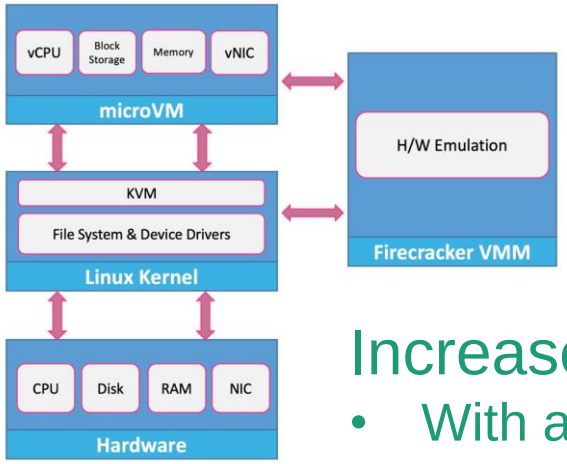
High overhead

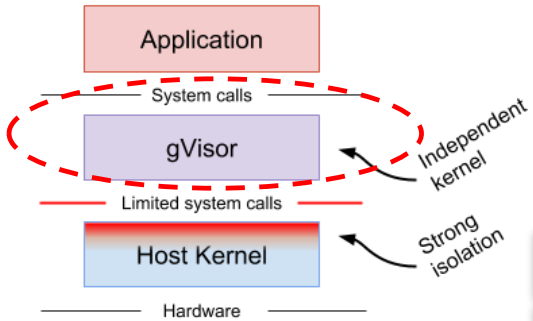
Weak isolation

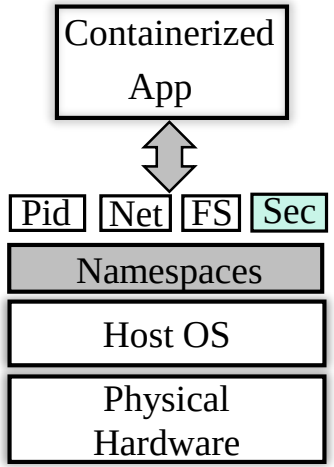
Low overhead

Isolation - Gvisor









Increased Isolation
- With a userspace guest kernel

Overhead also increases
- With indirection

VMs

microVMs (Firecracker)

Gvisor (userspace kernel)

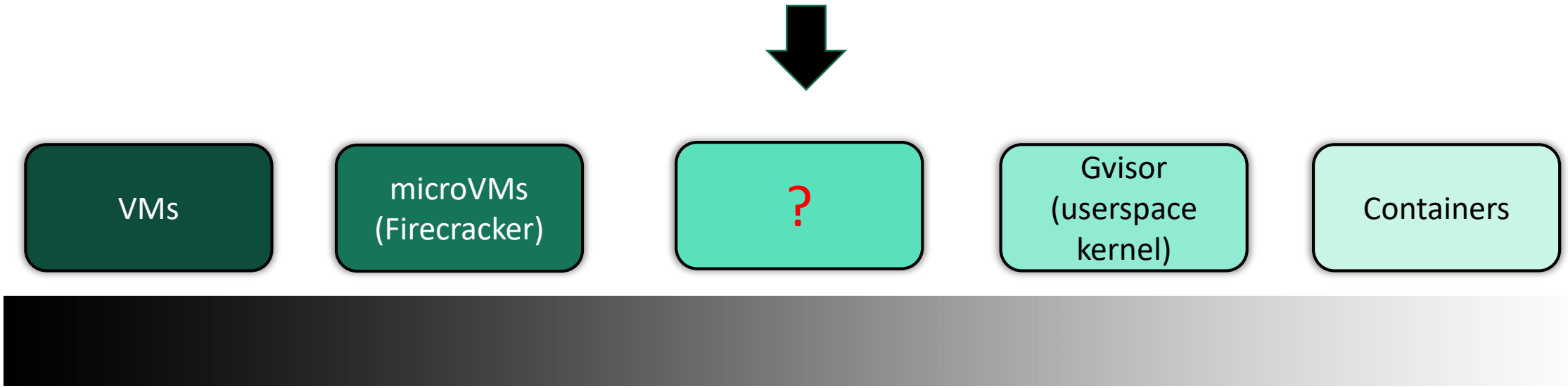
Containers

Strong isolation
High overhead

Weak isolation
Low overhead

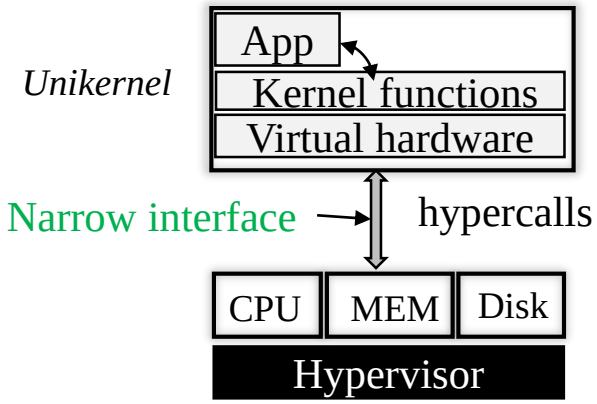
Other Isolation Approach?

Achieving Both Security of VMs and Speed of Containers?



Isolation - Unikernel

- The best of two worlds?



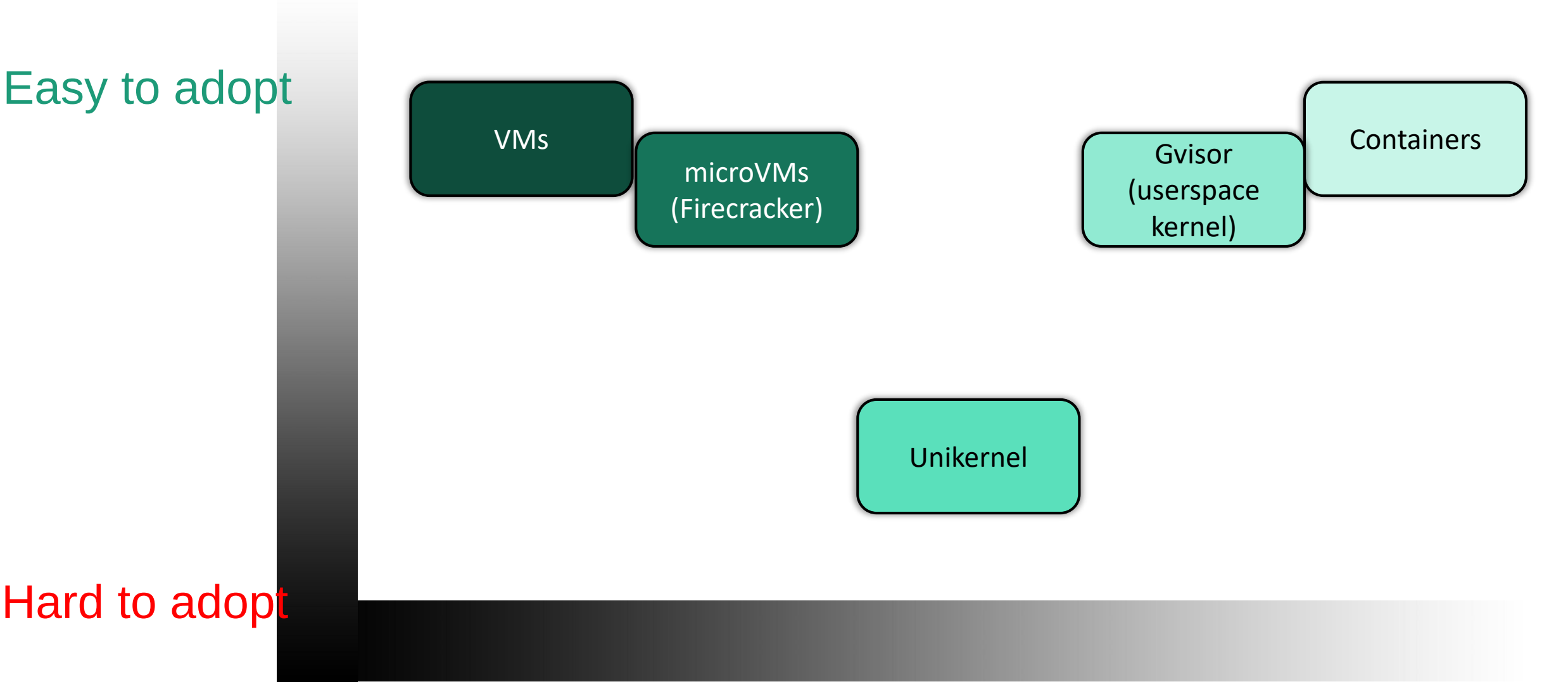
Strong isolation
- Narrow interface

Low performance overhead
- Deprivileged guest kernel (i.e., serving as libraries)
- Single address space
- Minimal kernel functionality

Strong isolation
High overhead

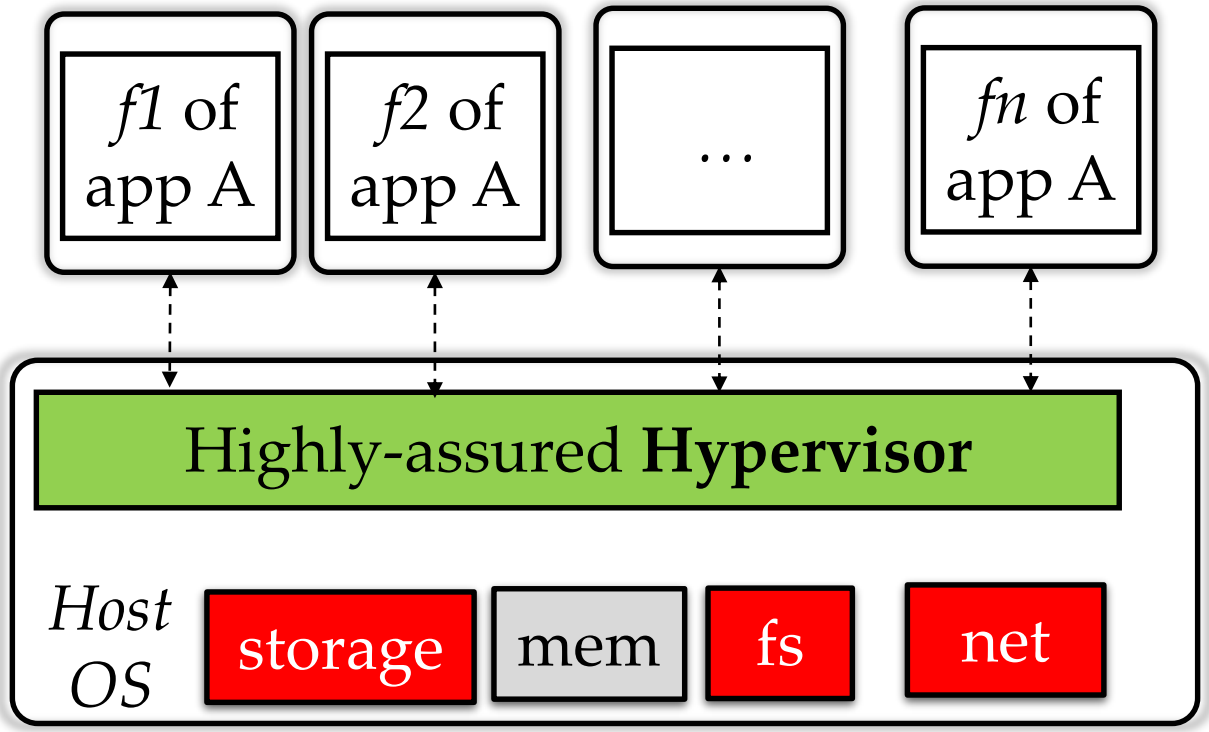
Weak isolation
Low overhead

Adoptability



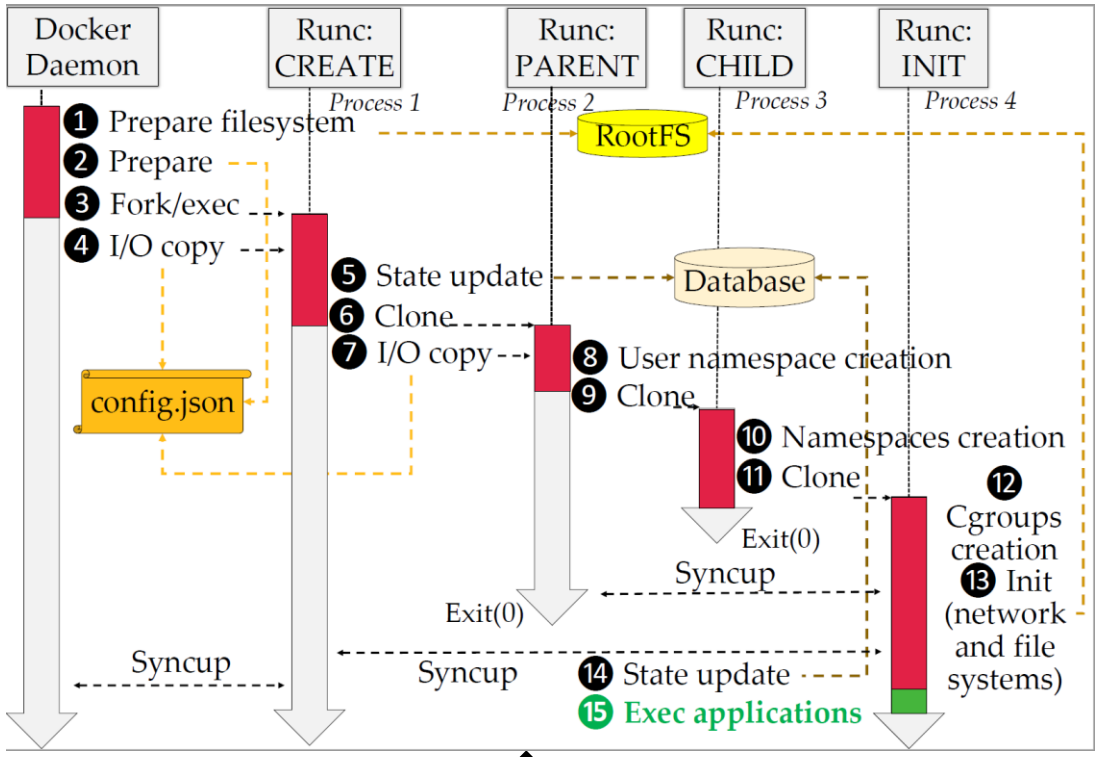
Key Challenges

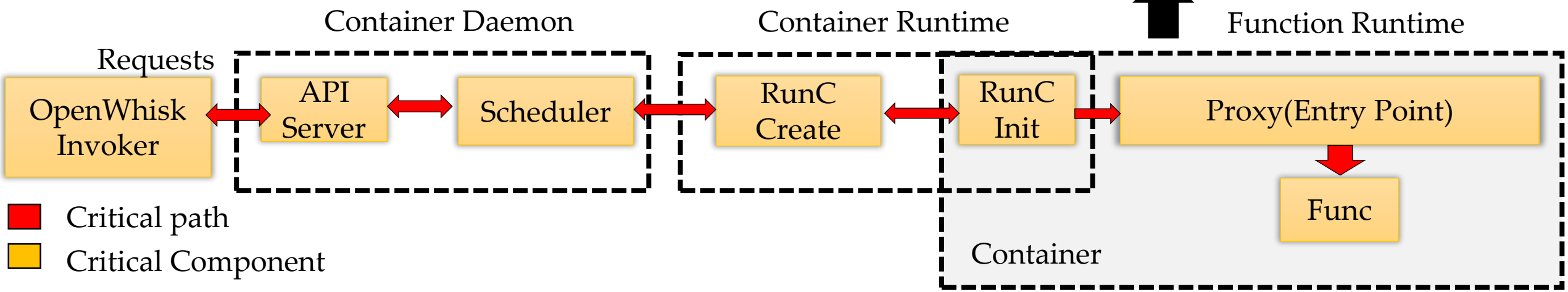
- Increase the adoptability of unikernel
- Hypervisor could go “wrong”
  - According to CVE, there were 184 vulnerability records involving main hypervisors (e.g., Xen, KVM, Hyper-V) since 2007
  - 33% from the past 1.5 years
- Unikernel approaches might still be heavy for microservices-based cloud-native applications
- The “peripheral” components may not be streamlined causing nontrivial performance overhead



Case 1: Startup Latency

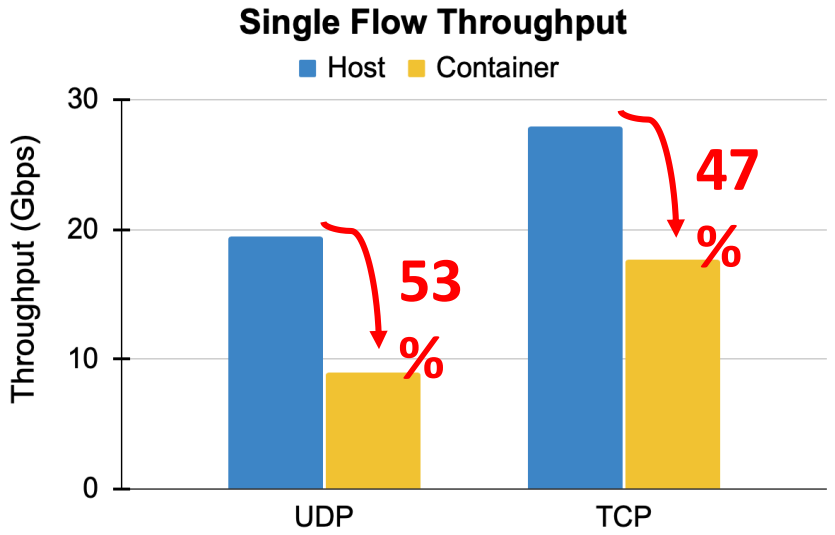
- To start a container instance, a complex serialized initialization pipeline is involed
- Such **cold start** latency range from 100 us to several seconds (worst-case scenario)

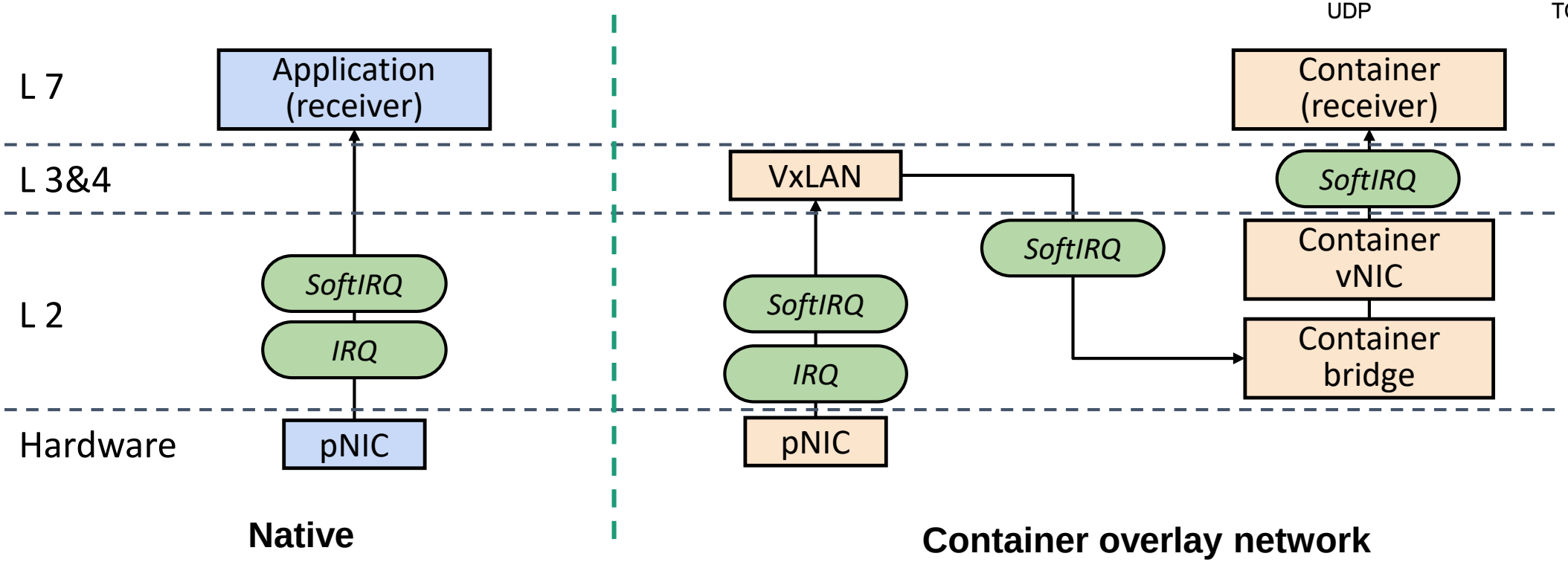




Case 2: Prolonged Data Path

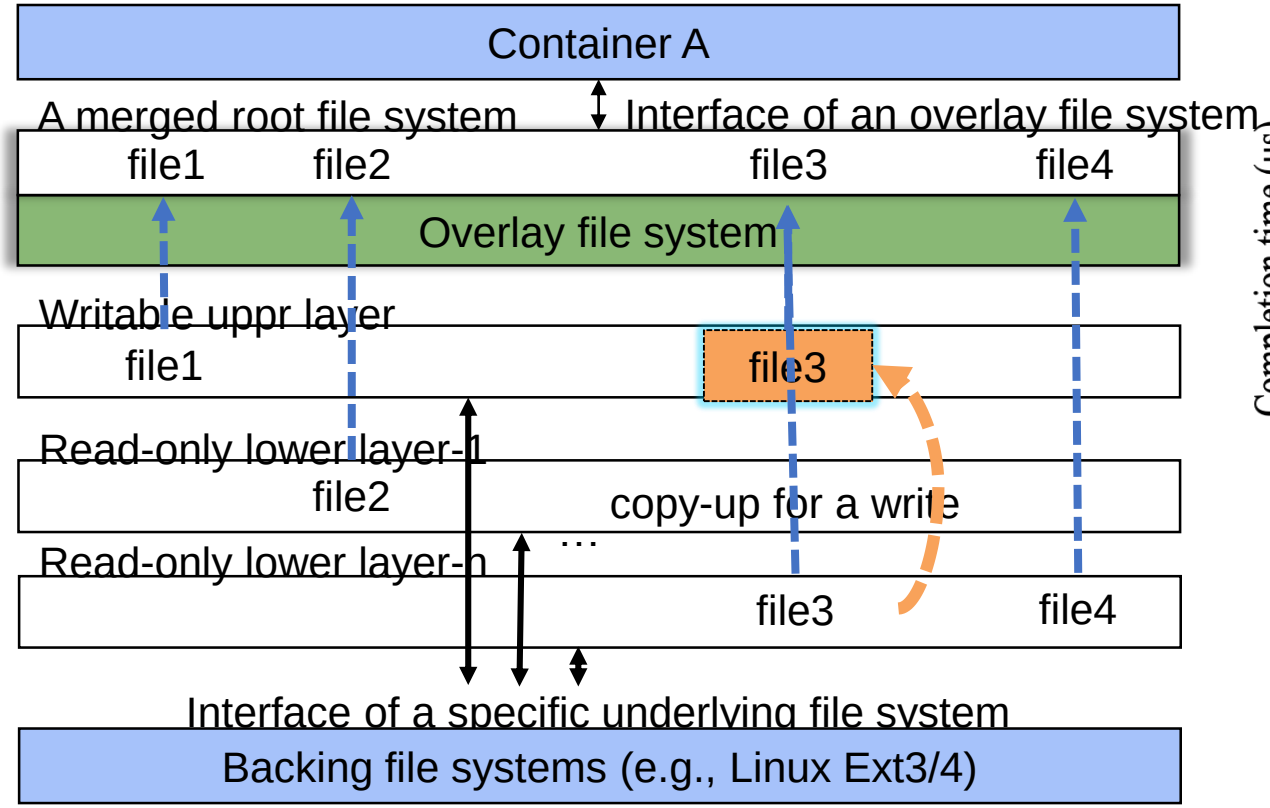
- Compared to a physical network, container overlay networks incur prolonged data path with both performance loss and longer latency

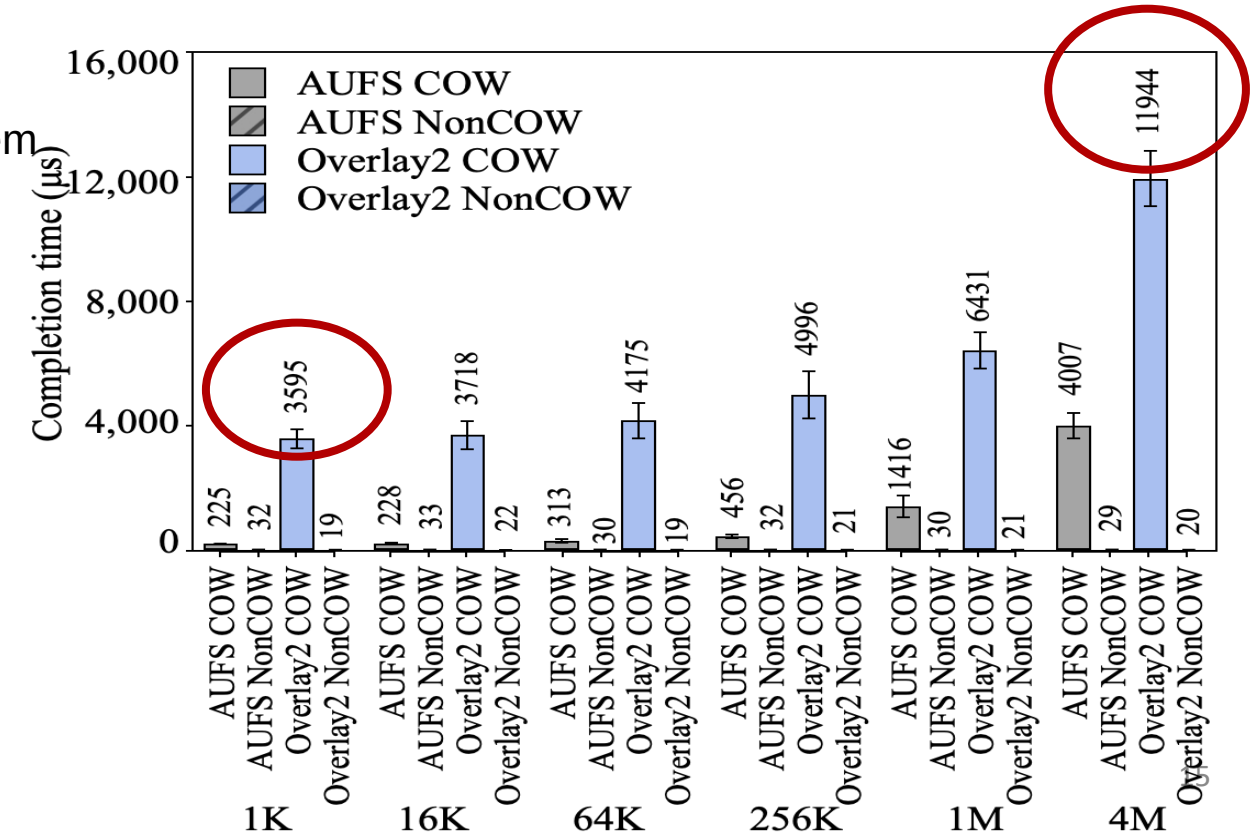




Case 3: Access Granularity

- Container storage commonly relies on overlay file systems to interpose **read-only** container images upon backing file systems
- File-based **copy-on-write** result in long write latency and inefficient use of container storage





Conclusions

- As cloud is moving along a long tradition of “ever-higher-level” abstractions with lower-complexity user-facing interfaces
  - It also creates critical challenges in existing virtualization techniques
- **Isolation** serves as the foundation in securing multi-tenancy clouds for safe resource sharing
  - However, existing virtualization techniques does not seem to fit well in microservices-based cloud-native environments
- We are seeking the fitting isolation technique with minimal performance cost, meanwhile retaining good adoptability
  - A unikernel-based sandboxing technique seems promising but with useability challenges
- Inefficiencies in data communication and processing attached to the isolation sandboxes were commonly observed and should be addressed