Dependable Virtualised Systems

For improved reliability, security, performance, and functionality

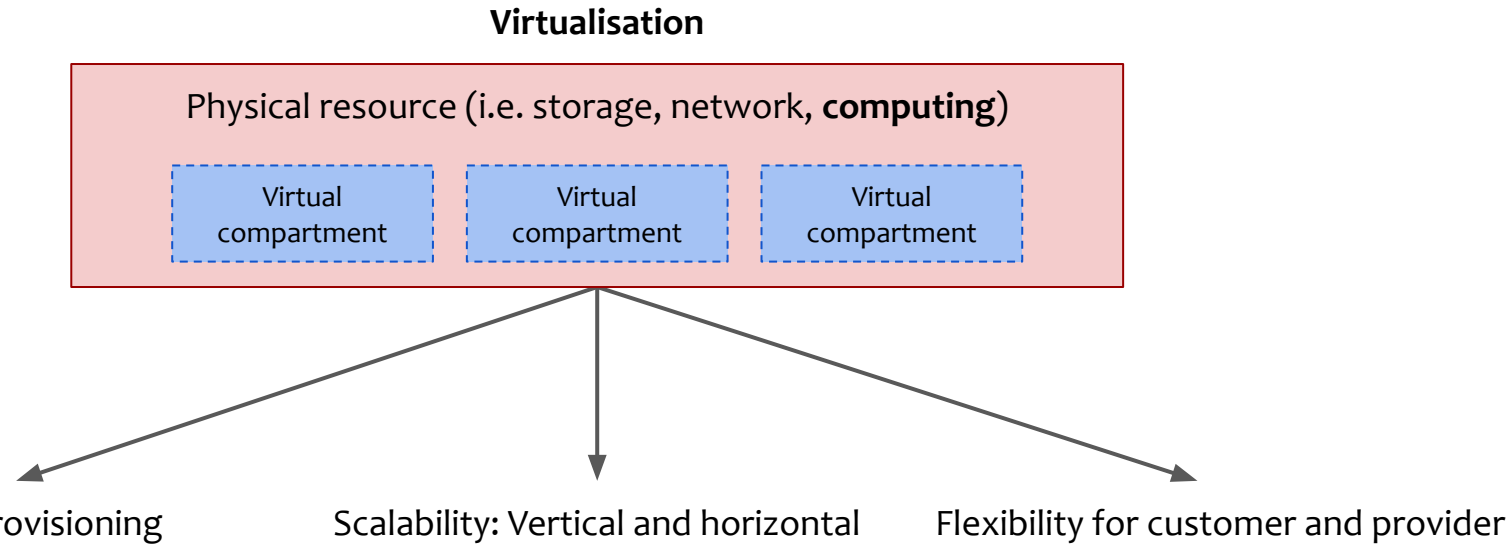
Jörg Thalheim

University of Edinburgh

THE UNIVERSITY *of* EDINBURGH

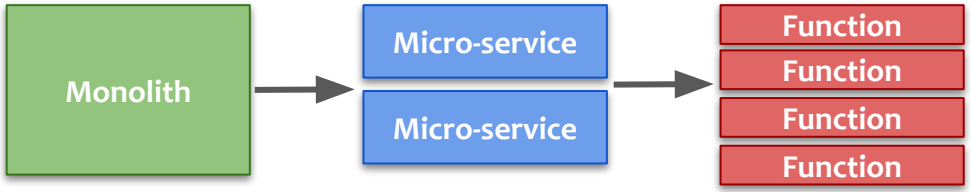
Virtualisation in the cloud

Modern data centers heavily depends on virtualisation



Trends in virtualized computing

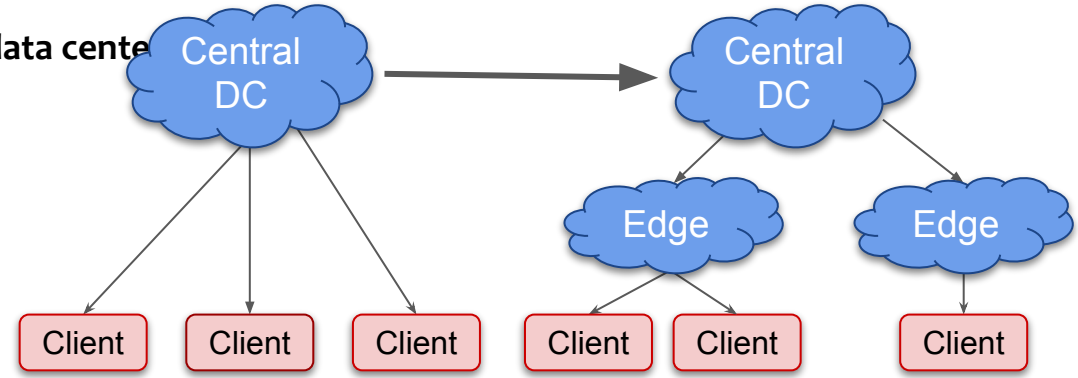
1. **Transition to new service architectures**



Utilisation

Operational cost

2. **Transition to new data cente architectures**



Latency

Reliability

New deployment models come with new challenges

Problem statement

THE UNIVERSITY of EDINBURGH

Challenges

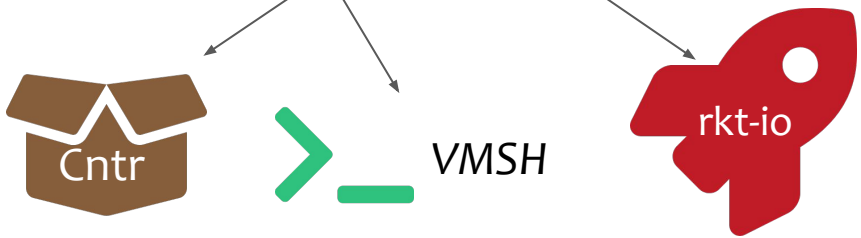
- Automatic placement by provider close to the client
- Scale-out as request come in

Physical hardware …

- … is shared with 1000s of other tenants
- … in potential less secure locations (*edge cloud*)

Need for **lightweight** and **secure** virtualisation architectures

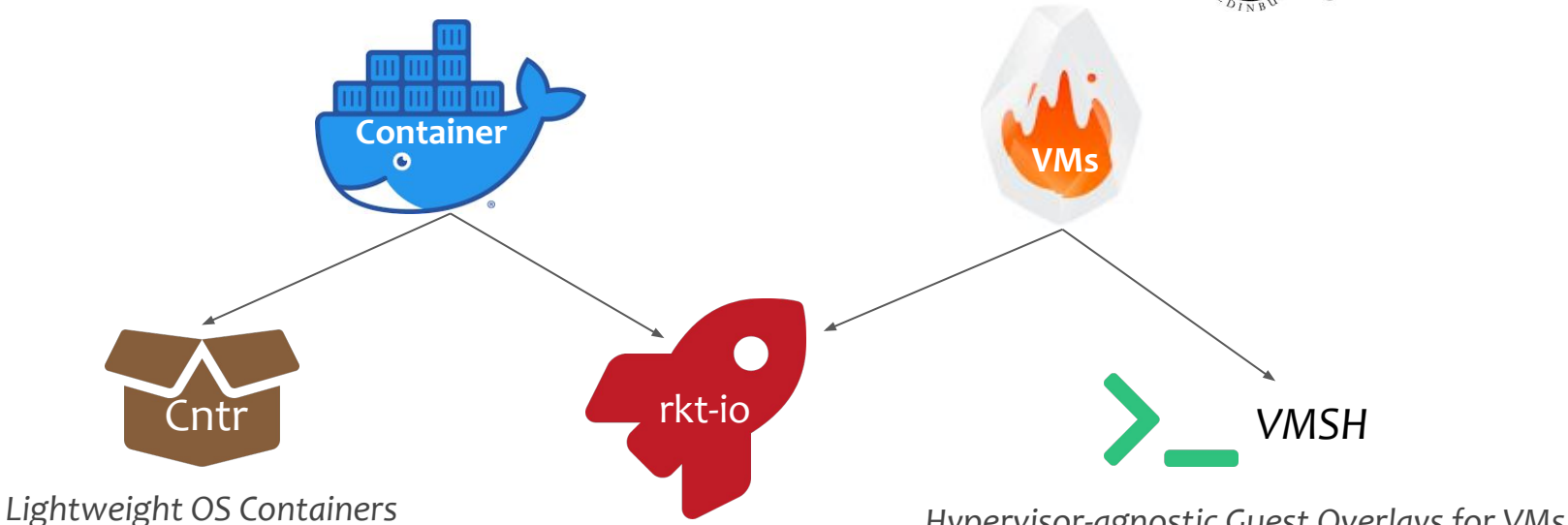
Our solutions



Our systems improve
dependability (maintainability and security) of virtualisation

Our contributions

THE UNIVERSITY of EDINBURGH



Lightweight OS Containers
USENIX ATC'18

Hypervisor-agnostic Guest Overlays for VMs
ACM EuroSys'22

A direct I/O stack for shielded execution
ACM EuroSys'21

Efficiency:
Allows developers to build & deploy smaller **container** images

Security:
LibOS for **virtual machines** and **containers** that encrypts data without sacrificing performance

Transparency:
Extends light**vm**'s with services at runtime

Outline

THE UNIVERSITY of EDINBURGH

- Motivation & Overview
- Dependable IO Stacks
  - [USENIX ATC'18] Cntr: Lightweight OS Containers
  - [EuroSys'21] rkt-io: A direct I/O stack for shielded execution
  - [EuroSys'22] vmsh: Hypervisor-agnostic Guest Overlays for VMs
- Conclusion

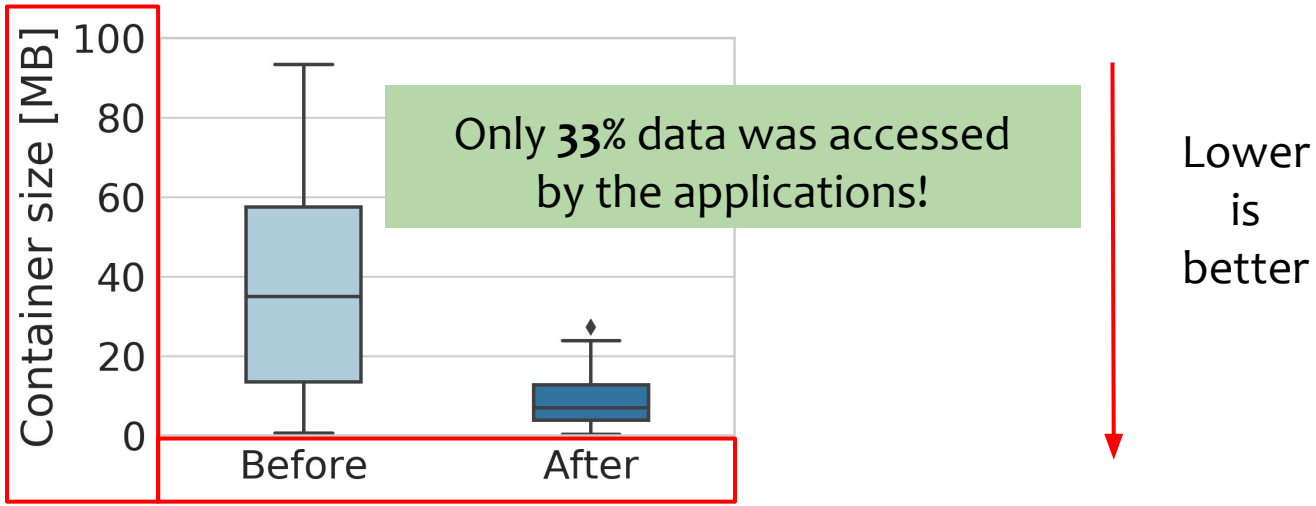
Cntr

Lightweight OS Containers

Containers are NOT lightweight anymore!

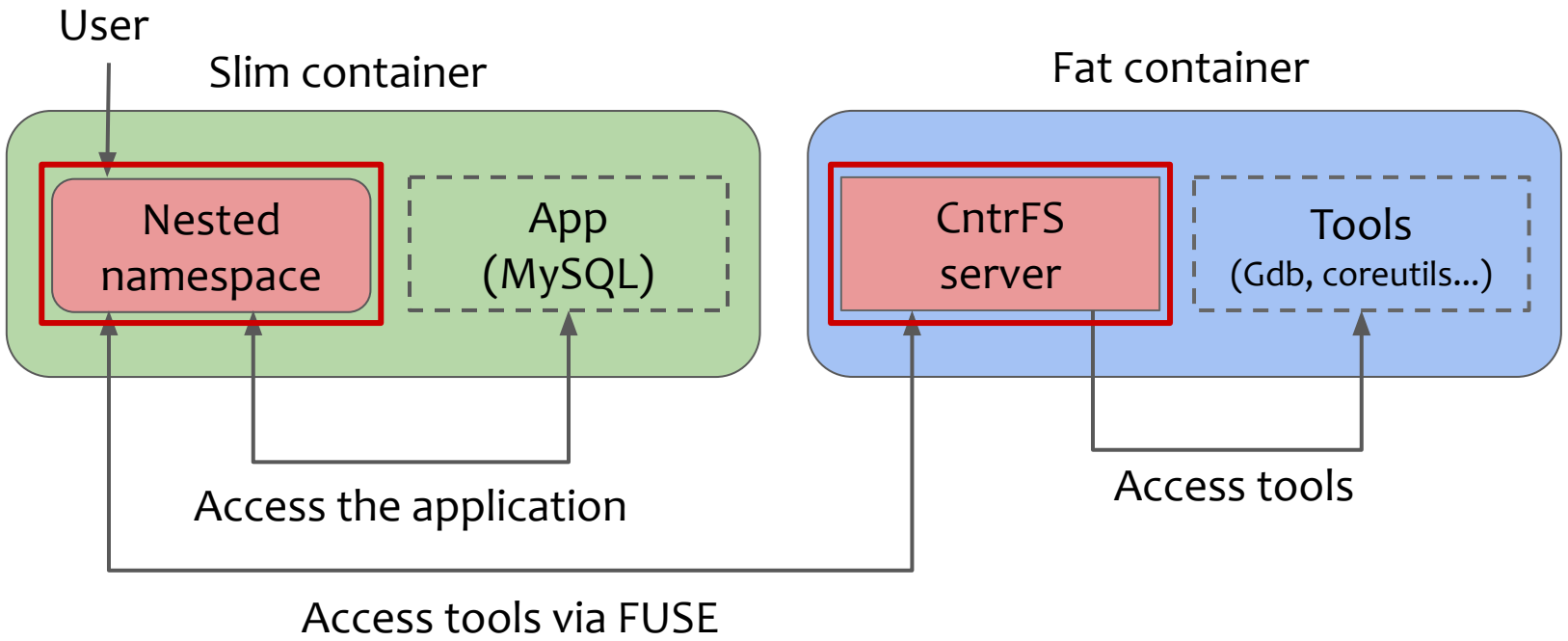
THE UNIVERSITY of EDINBURGH

Case study: Top 50 Docker Hub container images



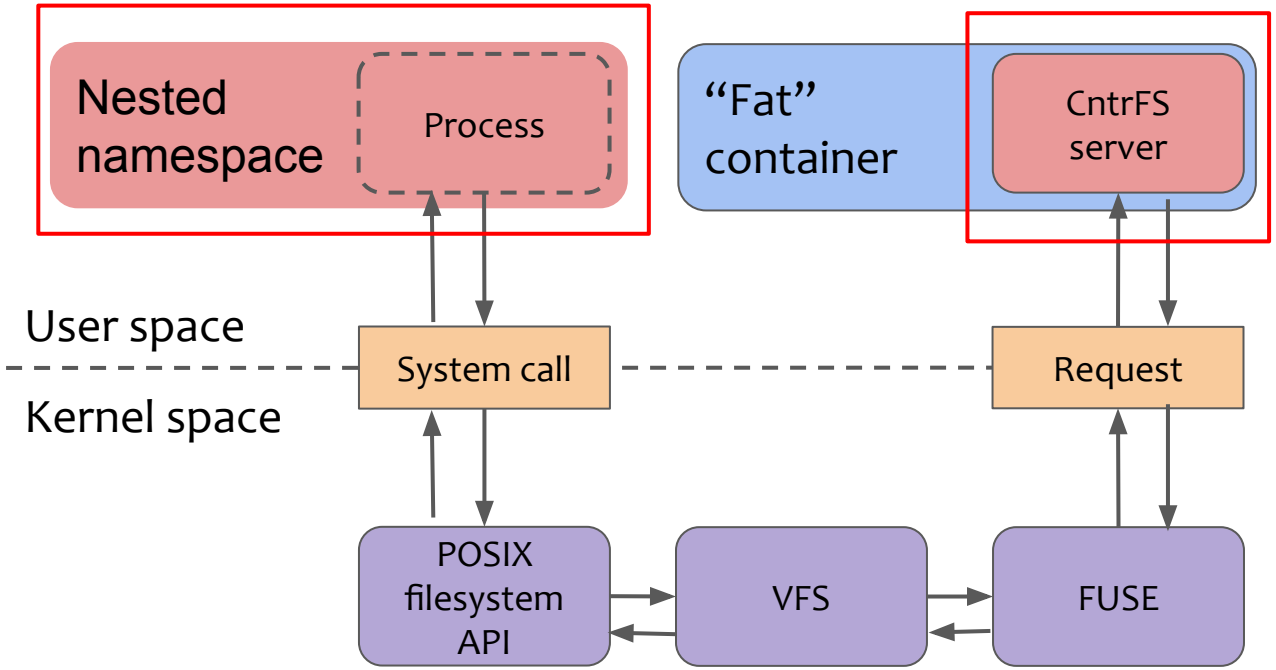
Limitations: Inefficient development and deployment of containers

Architecture



Split container into slim and fat part

Implementation



Process and CntrFS server can run in different namespaces (container)

Impact

THE UNIVERSITY of EDINBURGH

- Containers are NOT lightweight in practice
  - Limitation: Inefficient development and deployment of containers
- CNTR: Lightweight OS Containers
  - Splits the container image into **fat** and **slim** parts
  - Leverages **FUSE** to expose additional tools in a **nested namespace**
- Publication: USENIX ATC'18
- Project: https://github.com/Mic92/cntr
  - Popular and used in the container community (~370 stars)

Outline

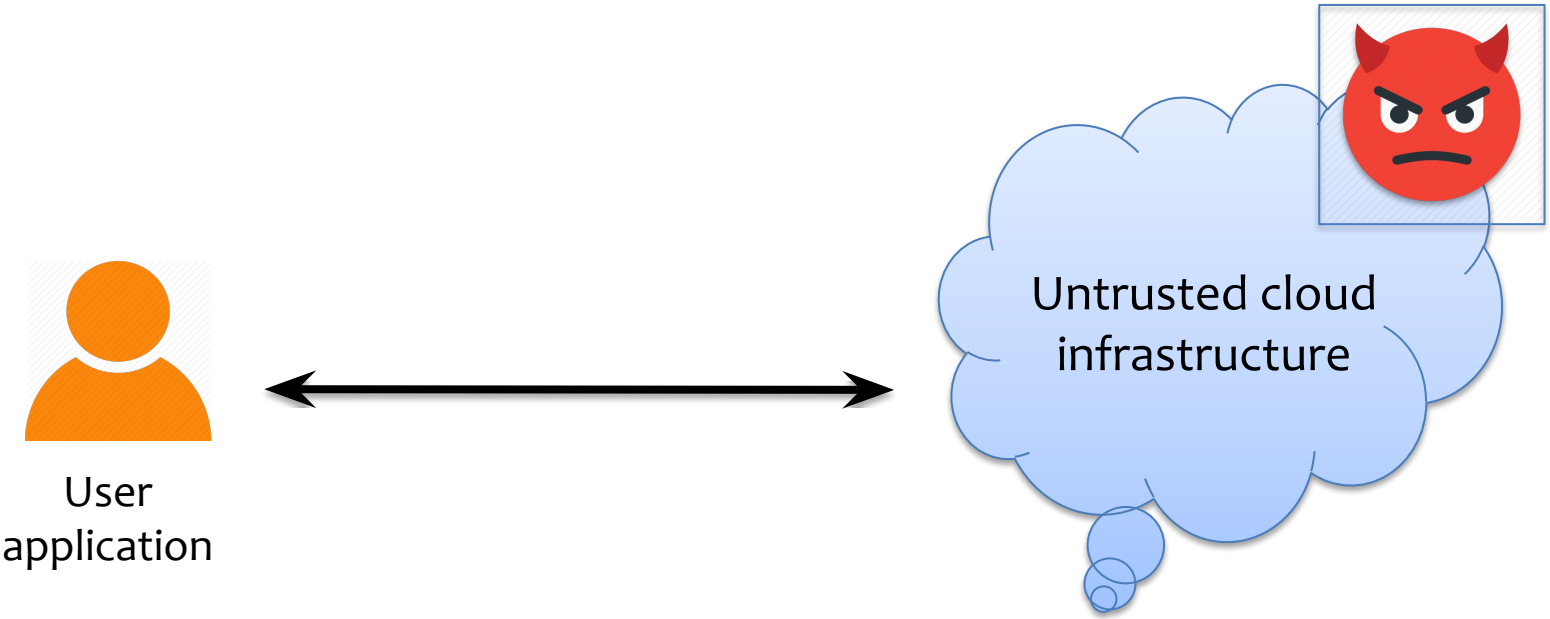
THE UNIVERSITY of EDINBURGH

- Motivation & Overview
- System architectures
  - [USENIX ATC'18] Cntr: Lightweight OS Containers
  - [EuroSys'21] rkt-io: A direct I/O stack for shielded execution
  - [EuroSys'22] vmsh: Hypervisor-agnostic Guest Overlays for VMs
- Conclusion

rkt-io

A direct I/O stack for shielded execution

Security in the untrusted infrastructure



How do we ensure application security in untrusted cloud environments?

Trusted computing

THE UNIVERSITY of EDINBURGH

Hardware-assisted
trusted execution environments (TEEs)

arm

AMD

RISC-V
Keystone

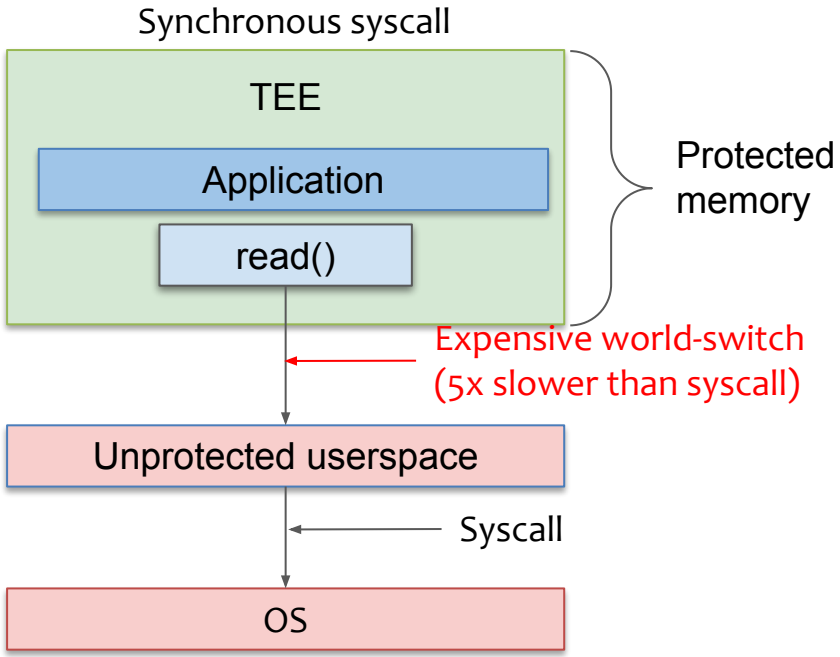
Google Cloud

Azure

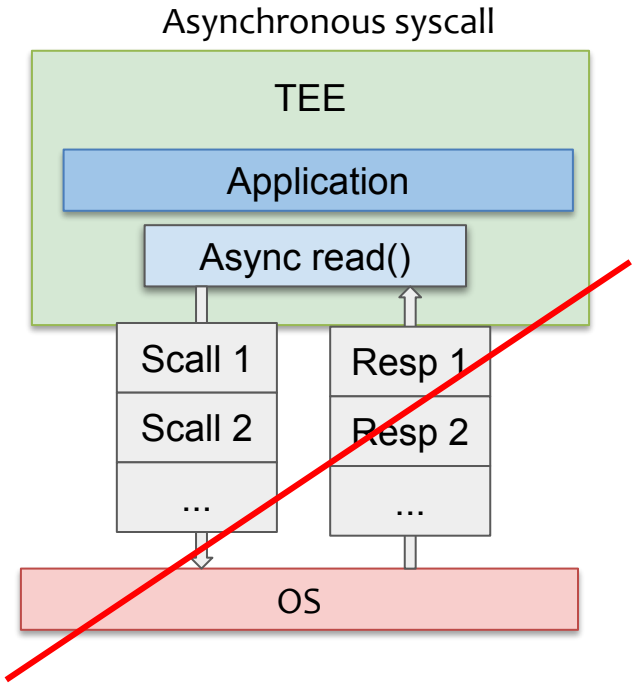
Alibaba Cloud

Offered by major cloud providers

IO in TEEs: Current approaches



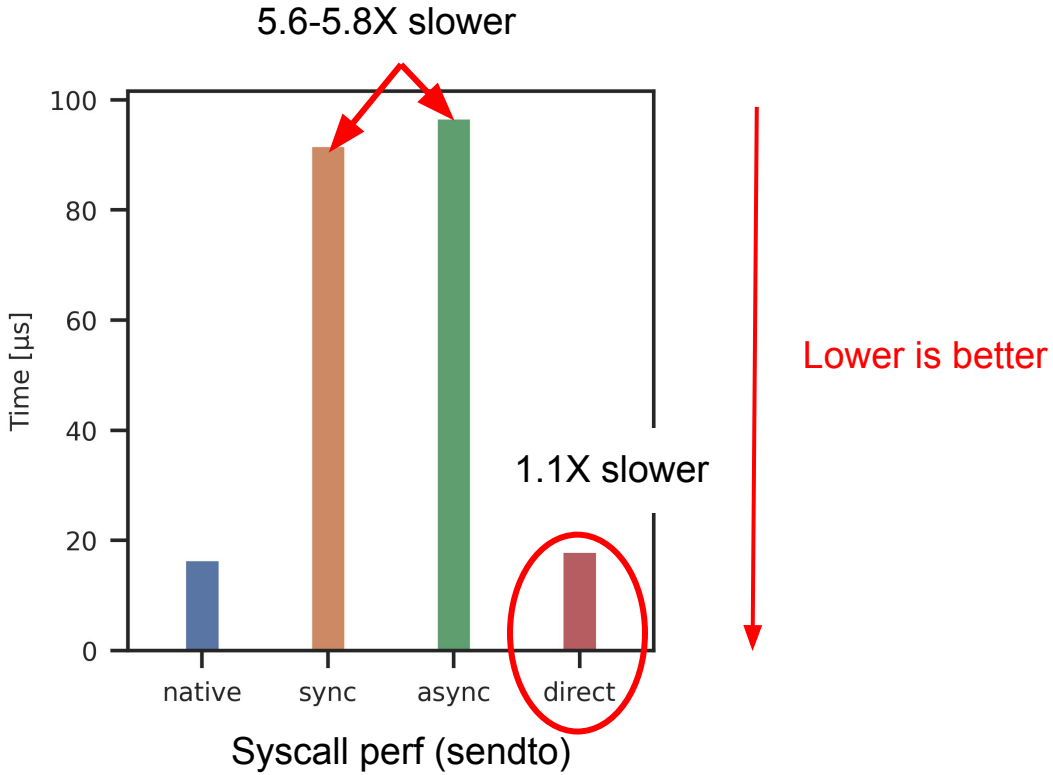
IO in TEEs: Current approaches



- **OS bottlenecks:**
  - The OS is still on the I/O path
- **I/O threads:**
  - Needs dedicated I/O threads, require tuning to find optimal number of threads
- **Data copies:**
  - Additional data copies between TEE ↔ IO threads ↔ OS

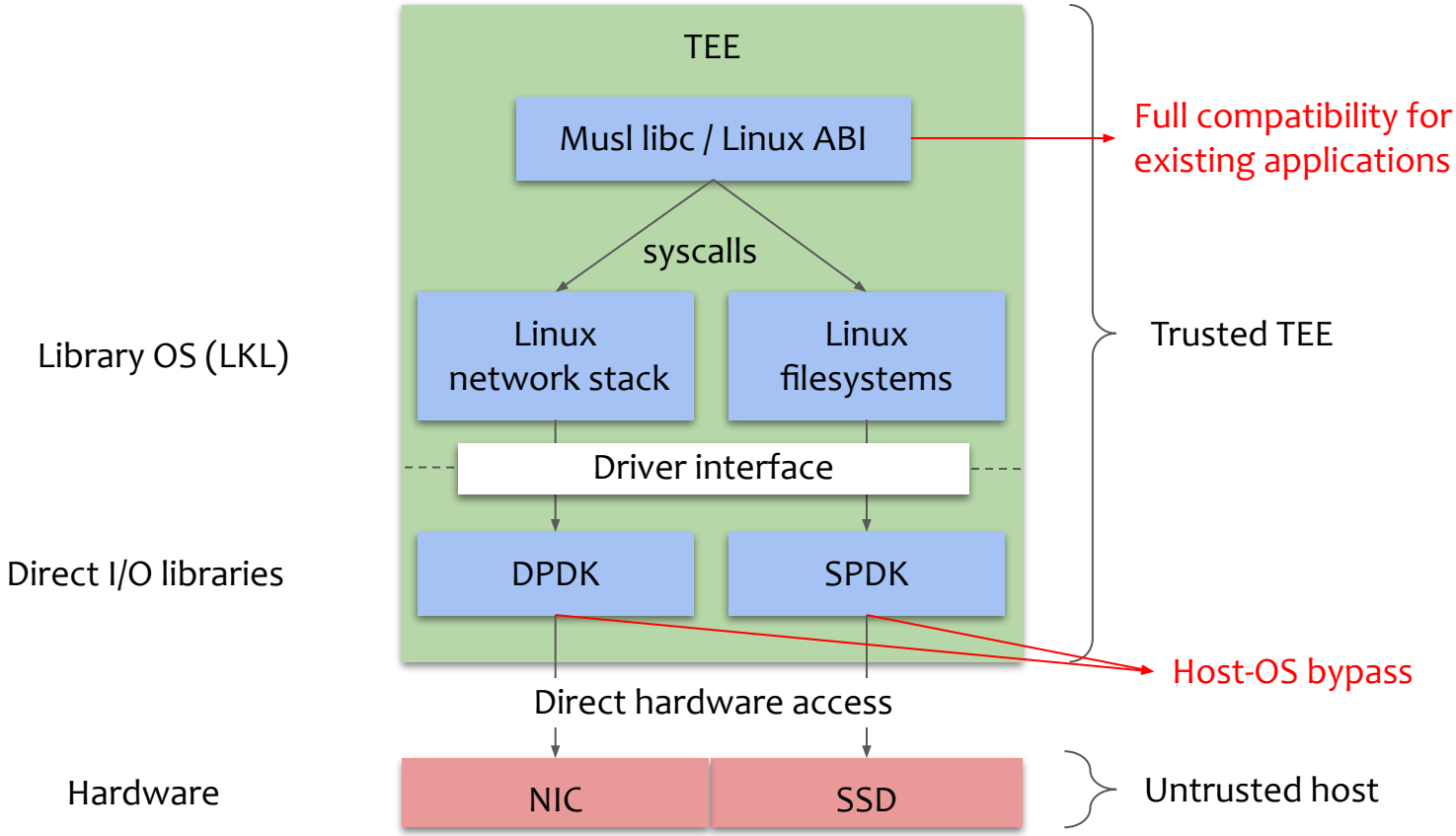
What if we access hardware directly?

Direct I/O in TEEs



Direct I/O improves IO performance significantly

Architecture



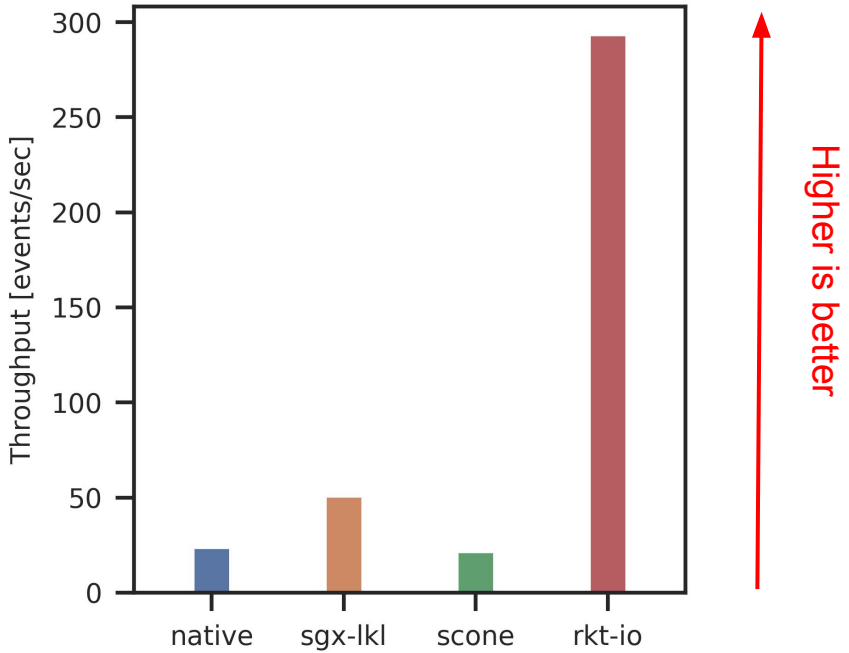
Benchmarks and applications

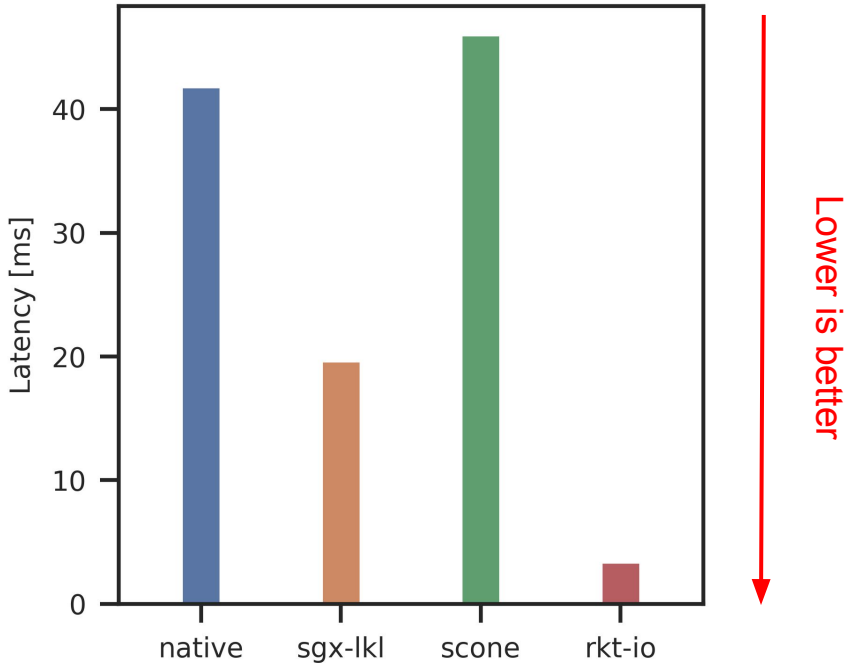
THE UNIVERSITY of EDINBURGH

- Synthetic benchmarks:
  - Storage (fio) and network (iPerf)
- Real-world applications:
  - Sqlite (Speedtest), nginx (wrk), Redis (YCSB), MySQL (Sysbench)
- Baselines
  - **Non-secure:** Native Linux
  - **Secure:** SCONE (host OS) & SGX-LKL (library OS)

Evaluation of MySQL (sysbench)

THE UNIVERSITY of EDINBURGH





Both SGX-LKL and rkt-io are faster than native

Analysis of futexes in MySQL

THE UNIVERSITY of EDINBURGH

Top 5	Syscall	Count	Time (µs)	Total (%)
#1	futex	64	4.20e+07	69.4
#2	read	24728	9.40e+06	15.5
#3	select	9	8.99e+06	14.8
#4	fsync	436	6.03e+04	0.1
#5	write	8243	3.48e+04	0.06

Breakdown of Top-5 syscalls in MySQL native execution

MySQL performance benefits from LibOS futexes / context switching

Impact

THE UNIVERSITY of EDINBURGH

- Current SGX-implementation are not designed for high-performance I/O
  - **OS bottlenecks:** The OS is still on the I/O path
  - **I/O threads:** Require tuning to find optimal number of I/O threads
  - **Data copies:** Additional data copies between TEE ↔ IO threads ↔ OS
- rkt-io provides
  - Transparent and fast access to the I/O devices
  - Linux ABI-compatibility for applications in TEEs
- Publication: EuroSys'21
- Project: https://github.com/Mic92/rkt-io

Outline

THE UNIVERSITY of EDINBURGH

- Motivation & Overview
- System architectures
  - [USENIX ATC'18] Cntr: Lightweight OS Containers
  - [EuroSys'21] rkt-io: A direct I/O stack for shielded execution
  - [EuroSys'22] vmsh: Hypervisor-agnostic Guest Overlays for VMs
- Conclusion

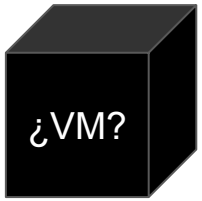
VMSH

Hypervisor-agnostic Guest Overlays for VMs

Motivation

THE UNIVERSITY of EDINBURGH

VMs are blackboxes: No API to attach services on-demand



Current solution:

- Cloud agents/Management tools inside the guest

Limitations for the provider

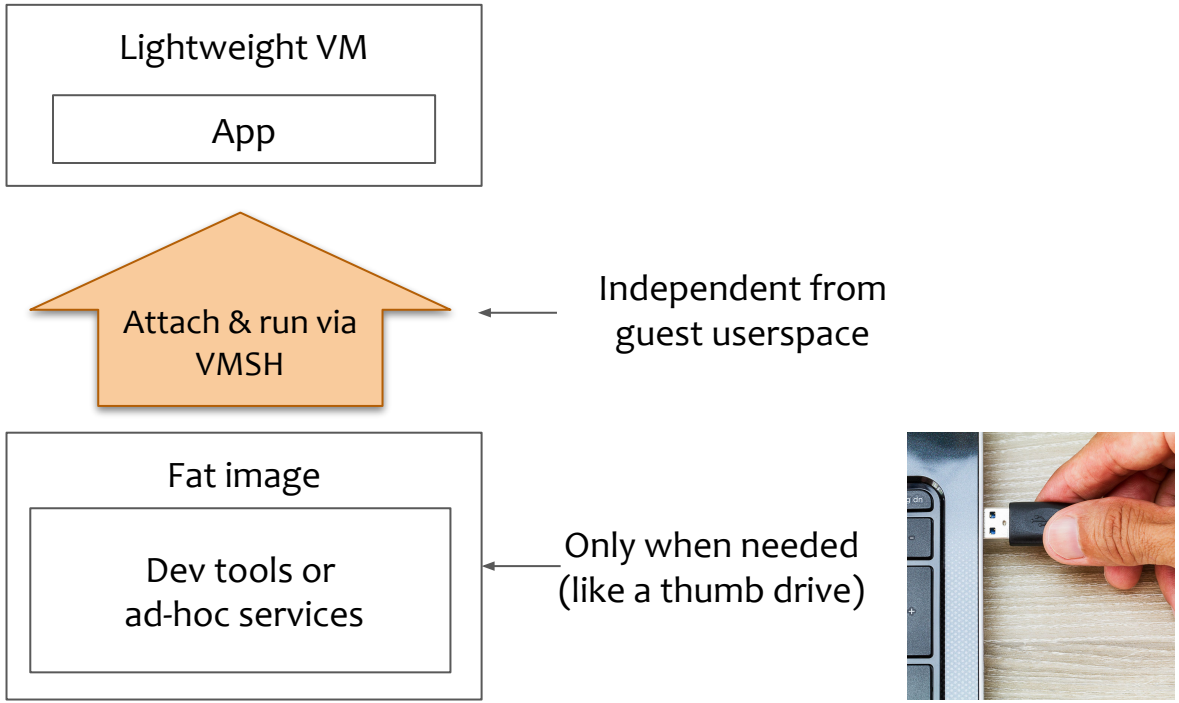
- Long innovation cycles
- Integration pain: Compatibility with different distributions required

Limitations for the user

- Bloat FS image size
- (potentially) require management network
- Source of misconfiguration:
  - Safety, i.e. lock-out
  - Security, i.e. remote code execution in cloud agents

How to keep VMs lightweight but make them extensible?

VMSH: The missing abstraction

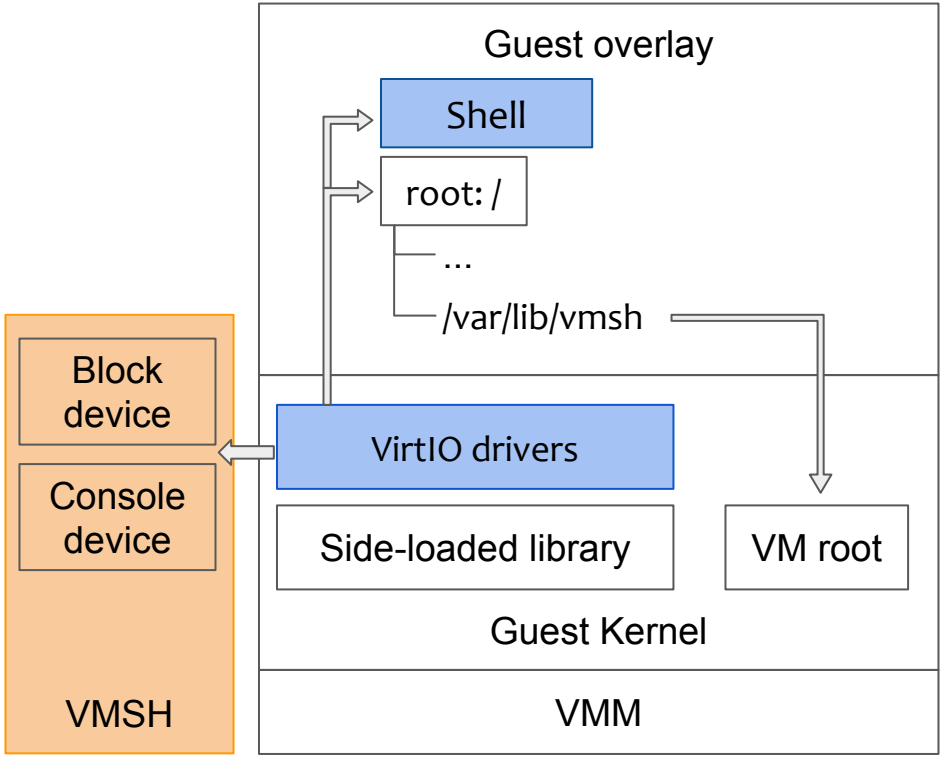


VMSH attaches to VM **on demand** & without guest agents

Architecture

THE UNIVERSITY of EDINBURGH

1. Find guest kernel symbols in memory and load kernel-version agnostic library into guest VMs
2. Kernel library sets up devices
3. Container-based system overlay runs user defined command

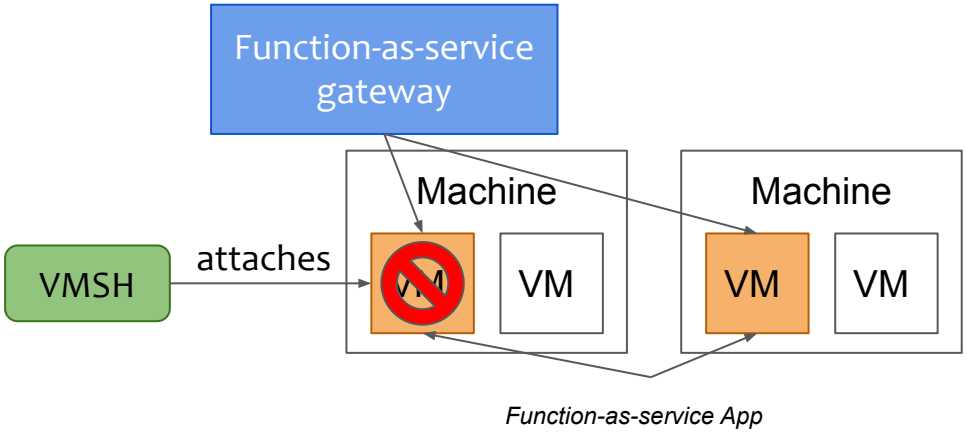


Based on Ptrace and VirtIO, VMSH can spawn new services in VMs

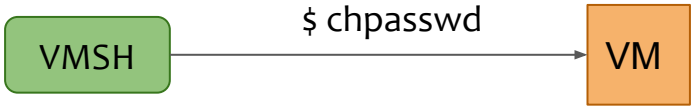
Implemented use cases

THE UNIVERSITY of EDINBURGH

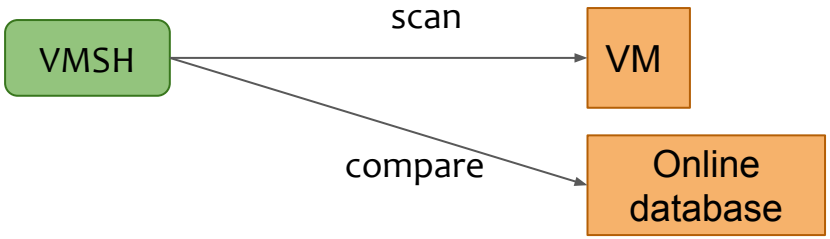
#1: Serverless debug shell



#2: Online password recovery



#3: Package security scanner



Impact

THE UNIVERSITY of EDINBURGH

Can lightweight VMs be extended with external functionality

- on-demand and
- non-disruptively?

VMSH

1. Hypervisor-independent guest code side-loading for KVM
2. OS-independent virtio-devices implementation
3. Generic guest-overlay

Publication: EuroSys'22

Project: https://github.com/Mic92/vmsh

- Commercial support for providers planed

Outline

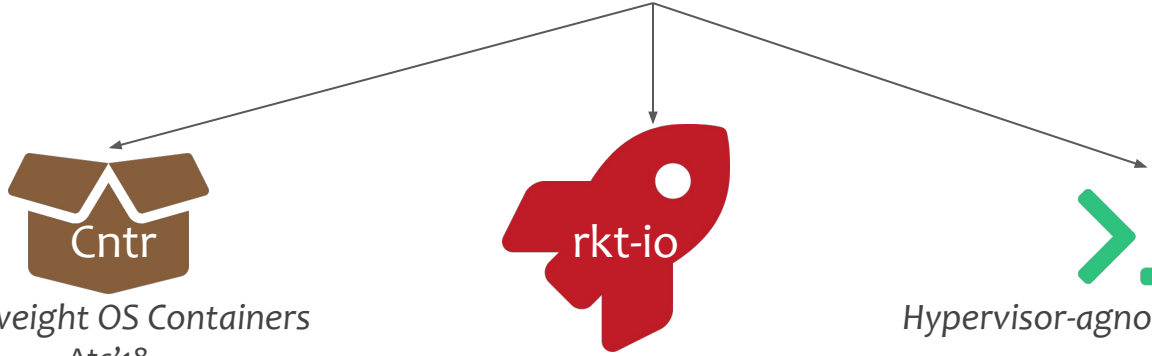
THE UNIVERSITY of EDINBURGH

- Motivation & Overview
- System architectures
  - [USENIX ATC'18] Cntr : Lightweight OS Containers
  - [EuroSys'21] rkt-io: A direct I/O stack for shielded execution
  - [EuroSys'22] vmsh: Hypervisor-agnostic Guest Overlays for VMs
- Conclusion

Summary

THE UNIVERSITY of EDINBURGH

Dependable Virtualized Systems



Lightweight OS Containers
Atc'18

A direct I/O stack for shielded execution
Eurosys'21

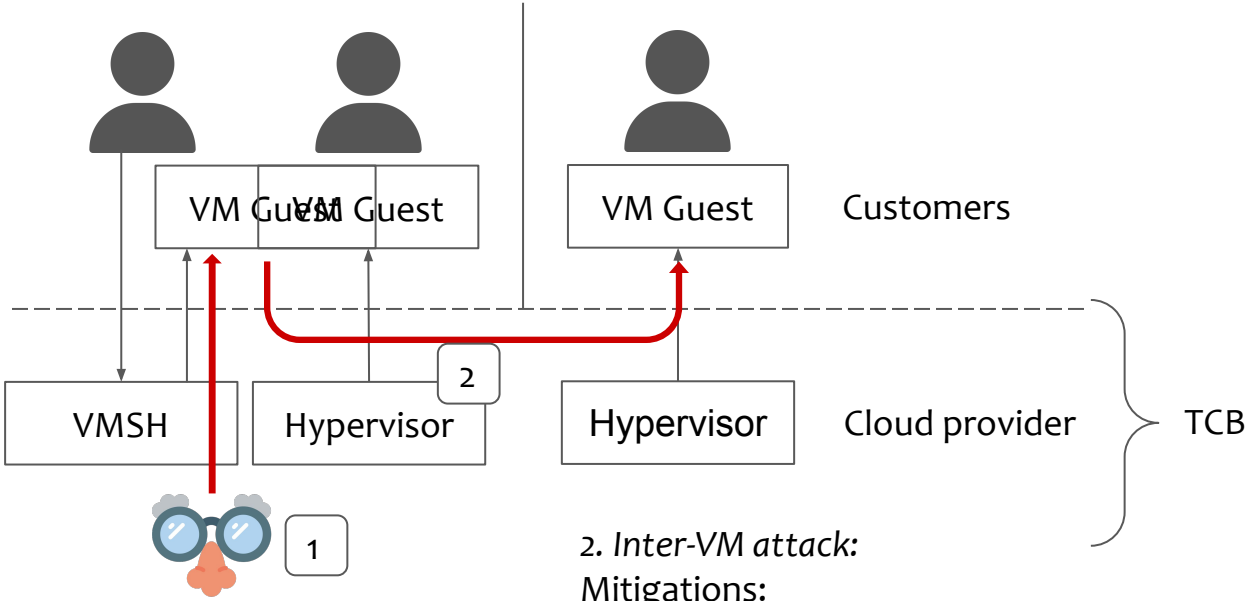
Hypervisor-agnostic Guest Overlays for VMs
Eurosys'22

Efficiency:
Allows developers to build & deploy smaller **container** images

Security:
LibOS for **virtual machines** and **containers** that encrypts data without sacrificing performance

Transparency:
Extends light**vm**'s with services at runtime

Threat model



1. Rogue admin
Mitigations:

- None, but providers are incentivised not to compromise their customer

2. Inter-VM attack:
Mitigations:

- Memory-safe device code based on production libraries from firecracker/crosvm
- VMSH runs with same privileges as hypervisor on the host