>_ VMSH

Hypervisor-agnostic Guest Overlays for VMs

Jörg Thalheim, **Peter Okelmann**, Harshavardhan Unnibhavi,
Redha Gouicem, Pramod Bhatotia

Technische Universität München

THE UNIVERSITY of EDINBURGH

ACM EuroSys 2022

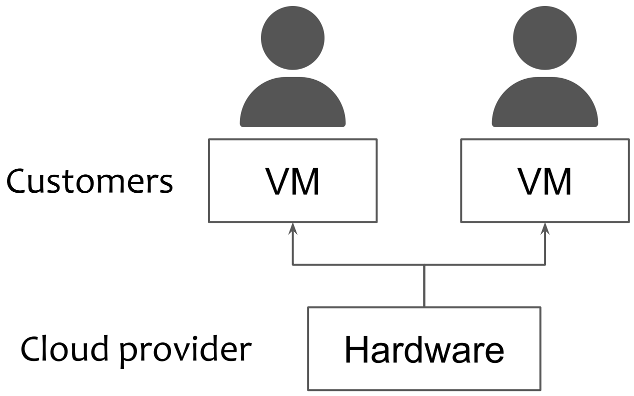
Virtual Machines (VMs)

VMs:

- Consolidation
- Cost-effectiveness

Optimized, lightweight VMs:

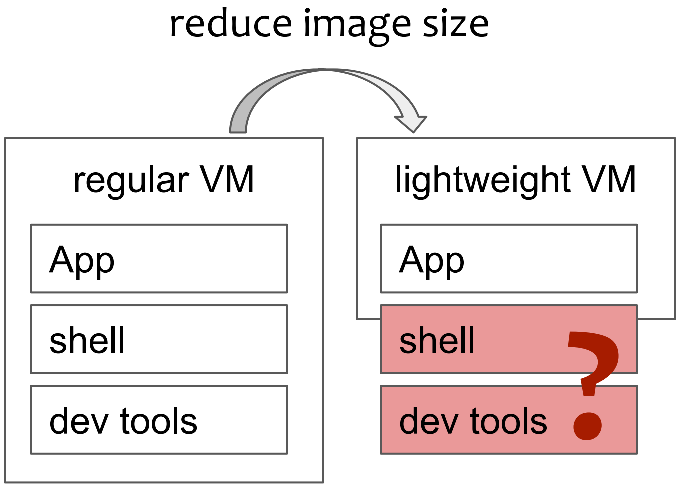
- **Small** memory footprint
- **Fast** bootup times
- Improve **dependability**: trust, reliability



Tradeoff: Lightweight VMs

Limited observability:

- No monitoring and inspection tools
- Disruptive: re-deployment for every change



Debugging, monitoring and repairing is time-consuming

Common solution: VM agents

Agent tasks:

- Provisioning
- Monitoring, Inspection
- Maintenance, Recovery

Multitude of implementations:

Amazon SSM, Google OS Config, Google Guest Agent, Microsoft OMI, QEMU Guest Agent, SSH, …

<u>Overheads for the customer:</u>

Devel & testing:
Provider, Hypervisor and OS distro specific

Infrastructure maintenance:
Management network, key management

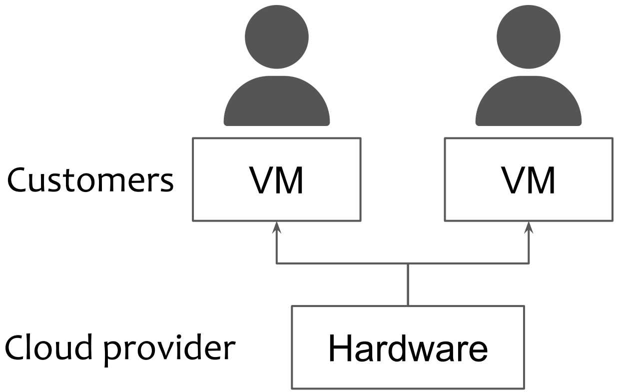
Complicated to use:
1600 pages of user manual

VM agents are an unsatisfactory solution

Beyond VM agents

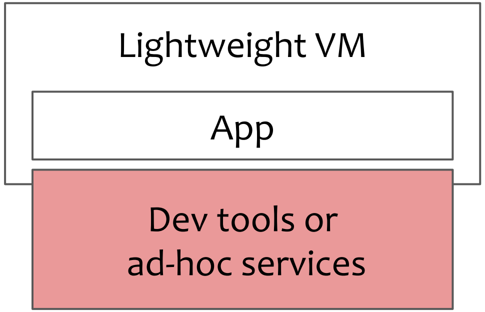
On monolithic servers, providers want to:

- Reduce overheads for customers
- Offer services to customers
    - Out-of-band management (~IPMI)
    - Update notifications
    - Security inspection



Out-of-band management with user-supplied tools?

VMSH: Guest overlays for VMs



VMSH: Guest overlays for VMs

Lightweight VM

App

Attach & run via VMSH

Independent from guest userspace

Fat image

Dev tools or ad-hoc services

Only when needed (like a thumb drive)

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

Design

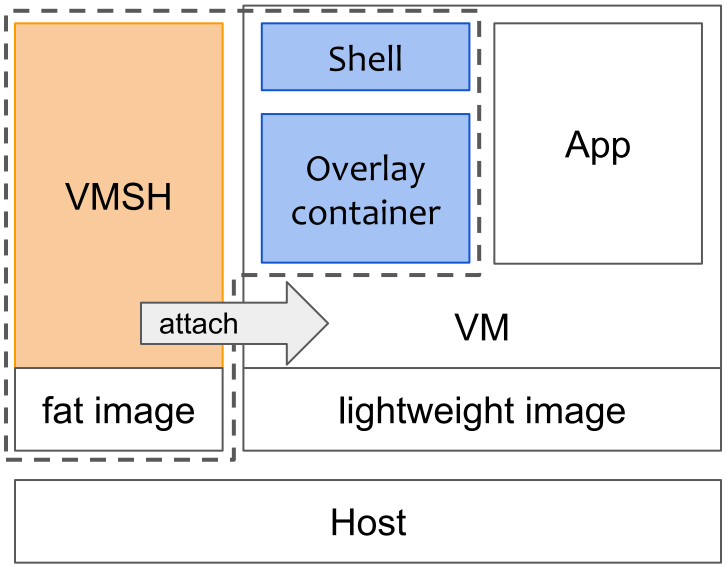
Design Goals
Overview

Design goals

- Non-cooperativeness
  - No guest agents
- Generality
  - No hypervisor specific APIs
  - Many Linux kernels
- Performance
  - No degradation of guest processes

Overview

- Non-cooperativeness
  - Attach to any VM
- Generality
  - Side-load overlay container
- Performance
  - VMSH serves fat image



Implementation

Side-loading a kernel-agnostic library
Container-based system overlay

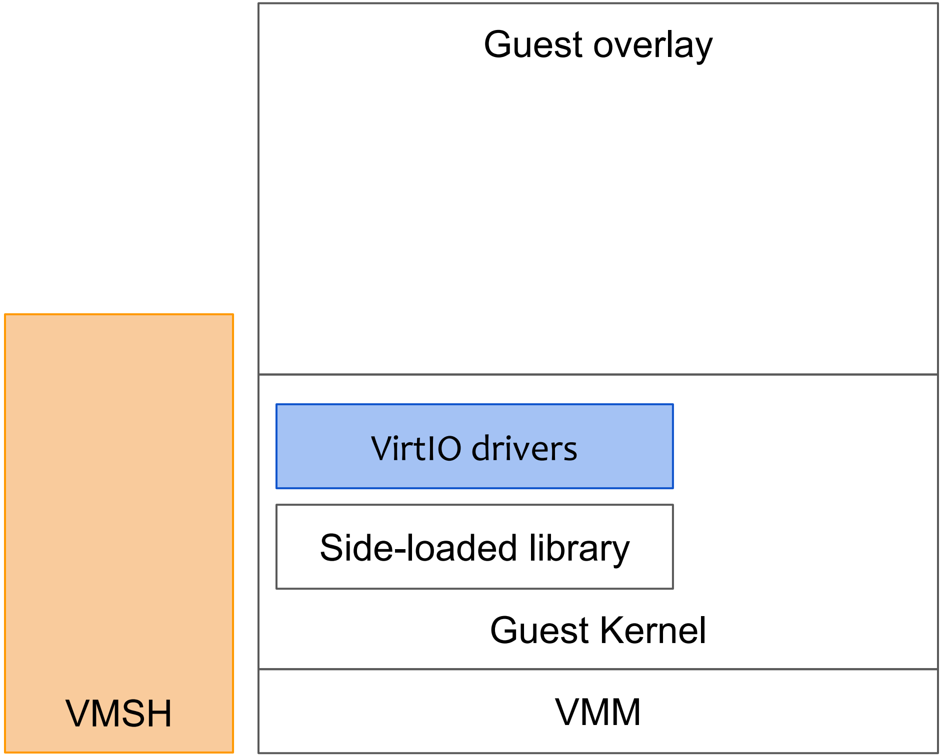
Side-loading a kernel-agnostic library

Side-loading:

- Side-load executable page into guest kernel
- Find kernel and parse its function table

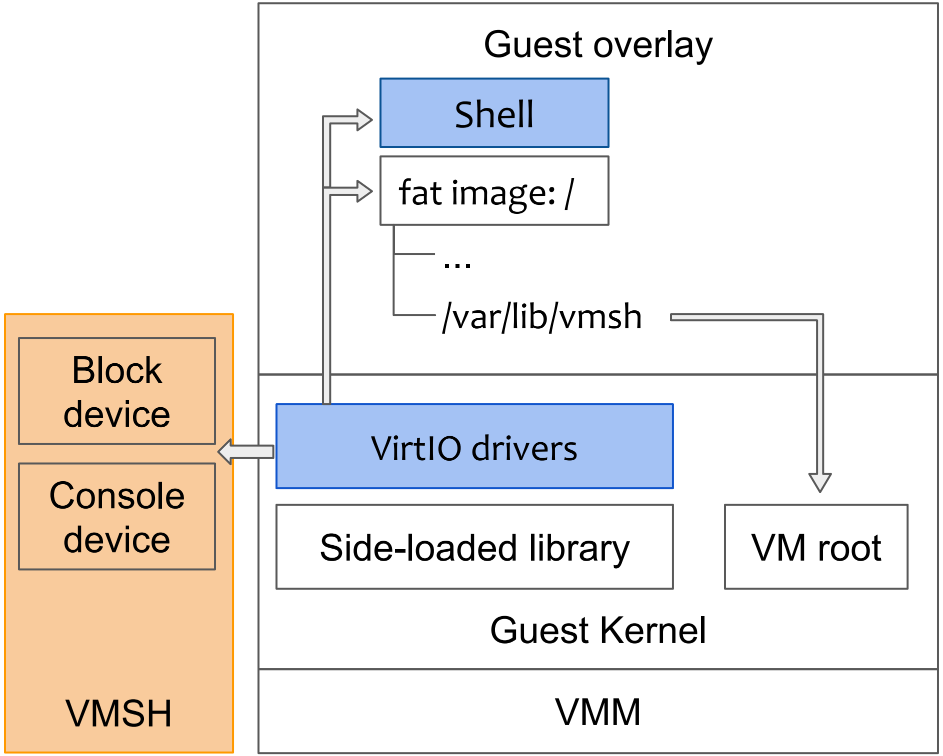
The kernel library…

- Starts overlay container
- Starts VirtIO drivers



Container-based system overlay

- Overlay for attached tools
- Overlay with VMSH's block device as fs root
- Communication to outside world via VMSH devices
- VMSH VirtIO devices via ptrace and ioregionfd



Evaluation

Evaluation

Questions:

1. Is the implementation robust?

2. Is our approach general?

3. Does VMSH impact performance?

Experimental Testbed:

- Intel Core i9-9900K CPU
- 64GB RAM
- Intel P4600 NVMe 2TB

1. Is the implementation robust?

Xfstests [3]:

- File system testing
- Widely adopted by Linux devs
- Regression tests, fuzzing

Block device	Passing tests
Qemu	**616**
VMSH	**616**

VMSH's block device is as robust as Qemu's

2. Is our approach general?

4 KVM Hypervisors:

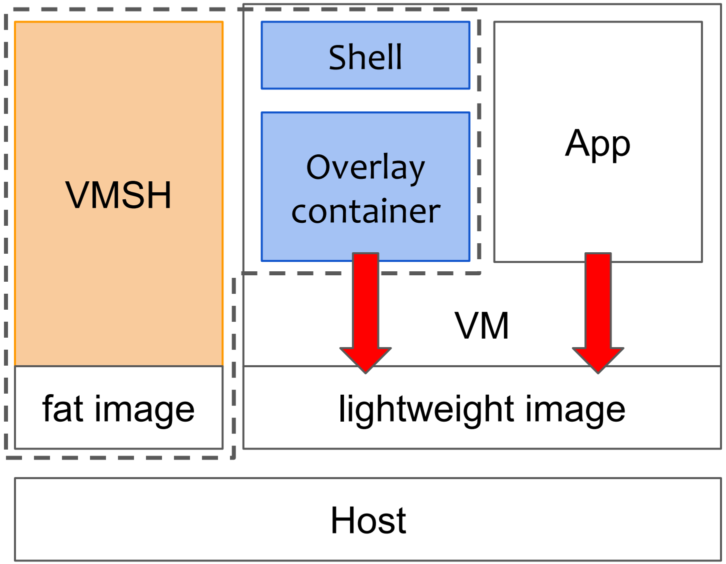
- QEMU
- kvmtool
- Firecracker
- crosVM

All Linux LTS kernels:

~40h to cover 5 years of kernel development

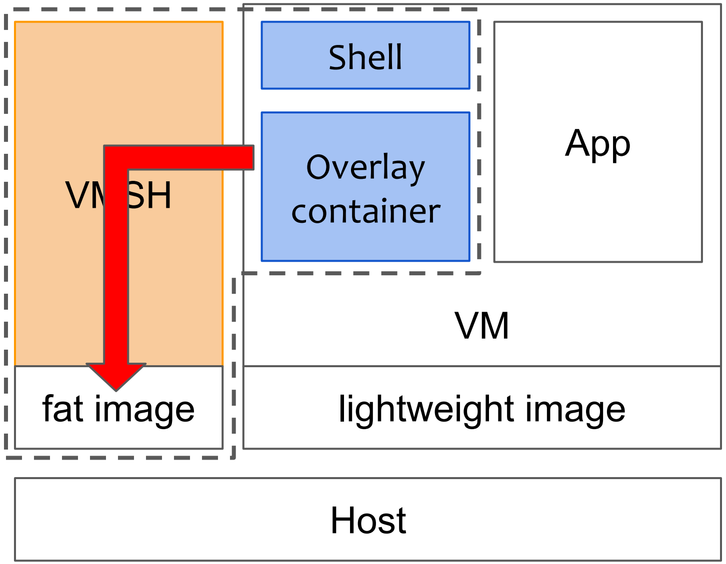
- v5.10, v5.4, v4.19, v4.14, v4.9, v4.4

3. Does VMSH impact performance?



3a. Common case: access original VM

3. Does VMSH impact performance?



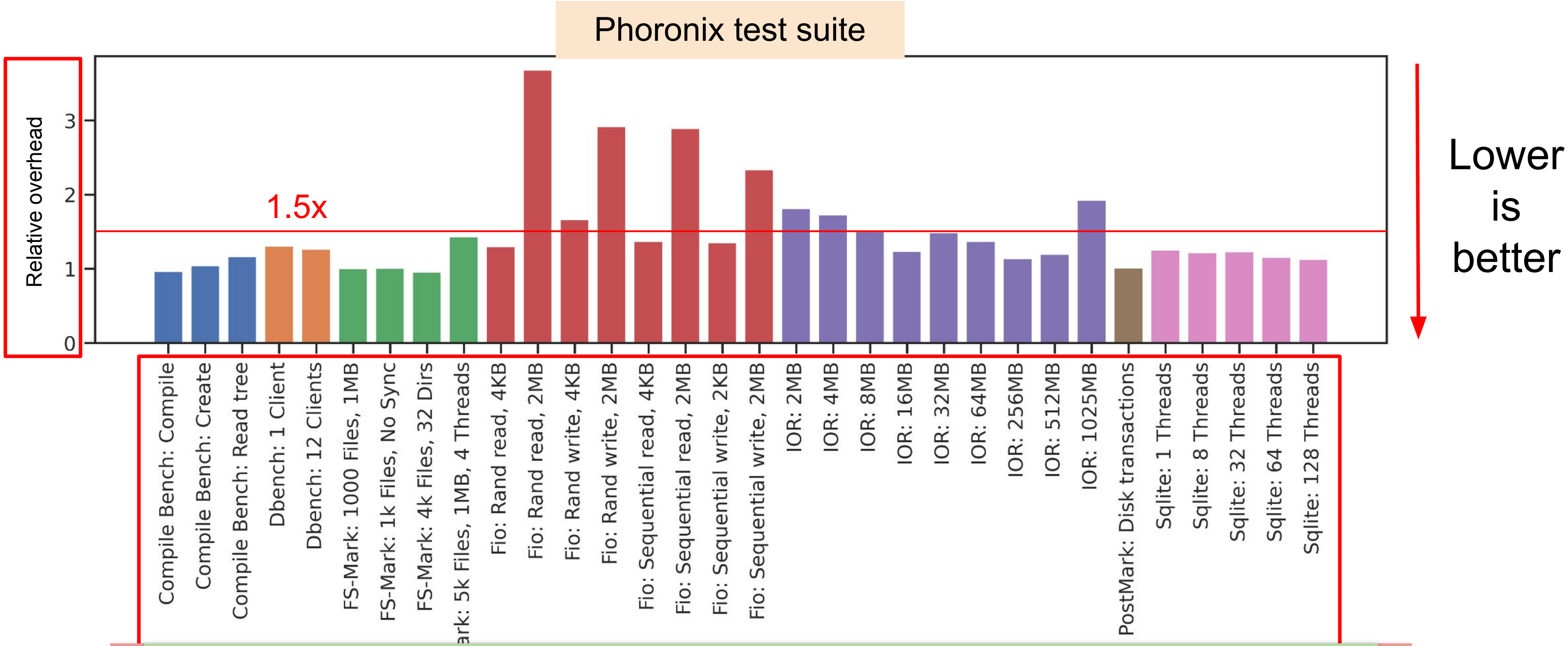
3b. Attached tools: VMSH devices

3a. Overhead for the lightweight image

0x

For the common case of
accessing the original VM

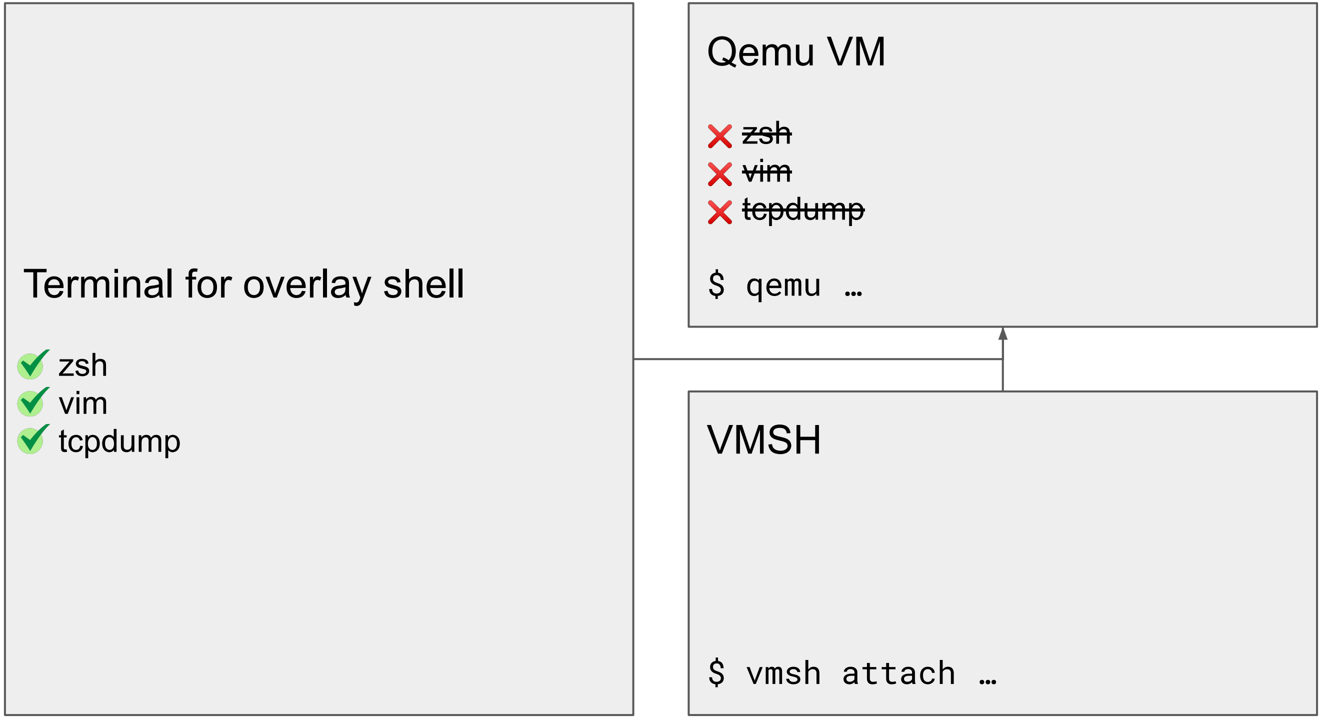
3b. Overhead: VMSH devices



VMSH incurs reasonable overhead for management tasks

Demo

Demo


Terminal for overlay shell
zsh
vim
tcpdump
Qemu VM
~~zsh~~
~~vim~~
~~tcpdump~~
$ qemu …
VMSH
$ vmsh attach …

Conclusion

VMSH extends lightweight VMs with external functionality

- on-demand
- non-disruptively

VMSH provides...

1. A generic guest-overlay
2. Hypervisor-independent VirtIO devices
3. An OS-independent code side-loading into VM guests

Try it on https://vmsh.org

References

[1] Maintenance icons created by kerismaker - Flaticon, https://www.flaticon.com/free-icons/maintenance
[2] Cube icons created by Freepik - Flaticon, https://www.flaticon.com/free-icons/cube
[3] xfstests-dev https://git.kernel.org/pub/scm/fs/xfs/xfstests-dev.git/

Backup Slides

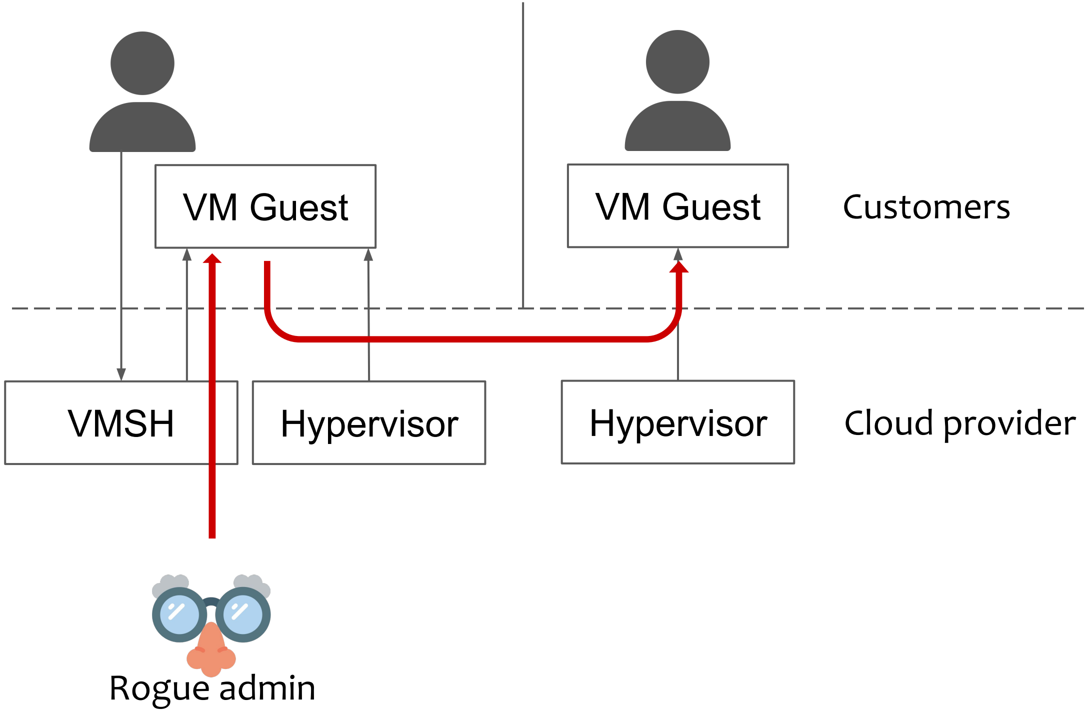
Threat model

VMSH control:

- Direct: provider
- Indirect: customer

Threats:

1. Inter-VM attack
   Difficult: Attached services run in guest domain
2. Rogue admin
   Unlikely: Providers have incentive for prevention



VMSH leaves the responsibility of authentication to the provider