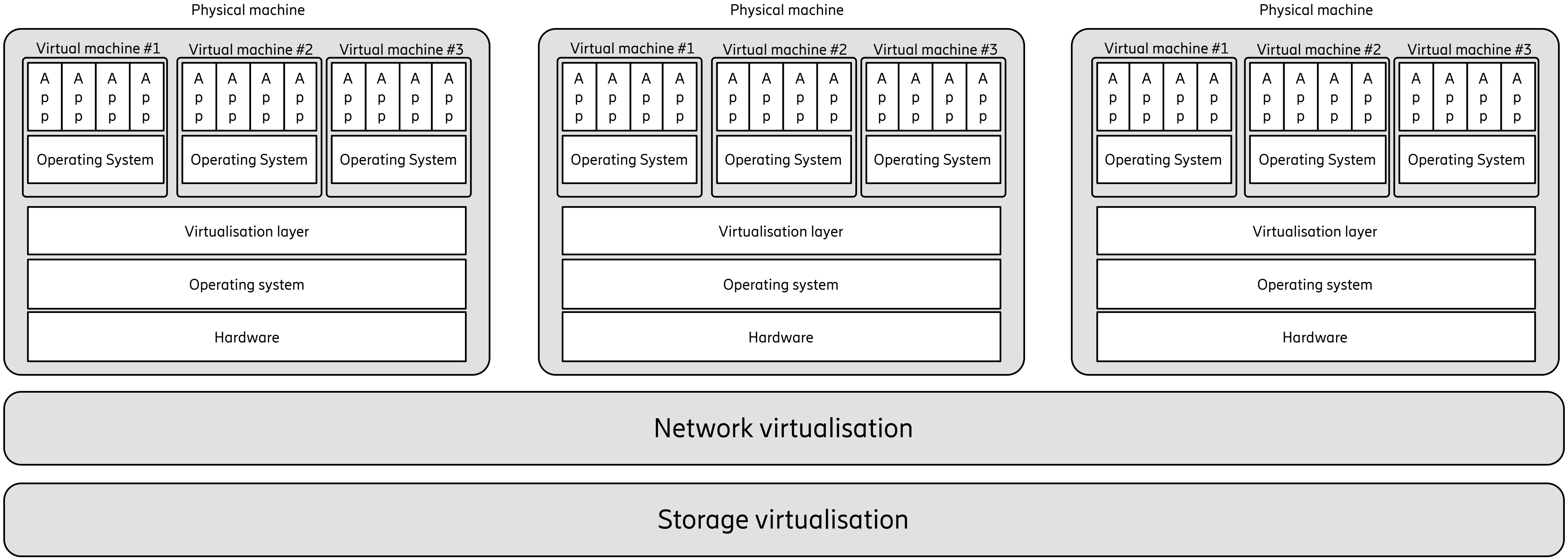
Cloud Computing
#2a - Virtualisation and Networking

Virtualization

From Wikipedia, the free encyclopedia

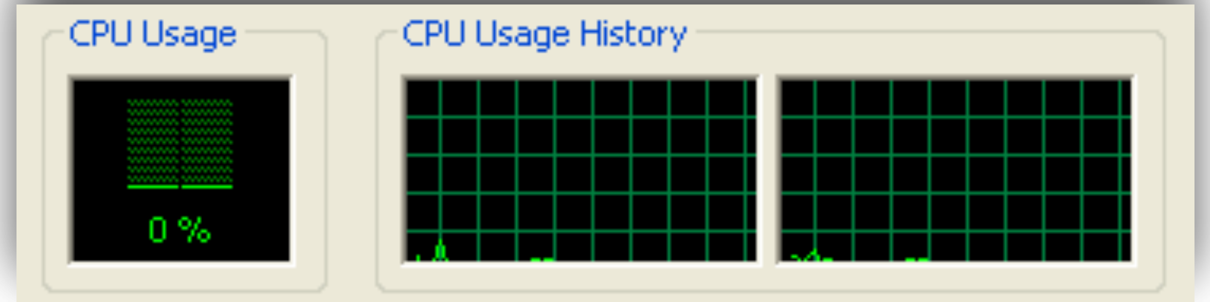
In computing, **virtualization** refers to the act of creating a virtual (rather than actual) version of something, including virtual computer hardware platforms, storage devices, and computer network resources.

Virtualization began in the 1960s, as a method of logically dividing the system resources provided by mainframe computers between different applications. Since then, the meaning of the term has broadened.[1]



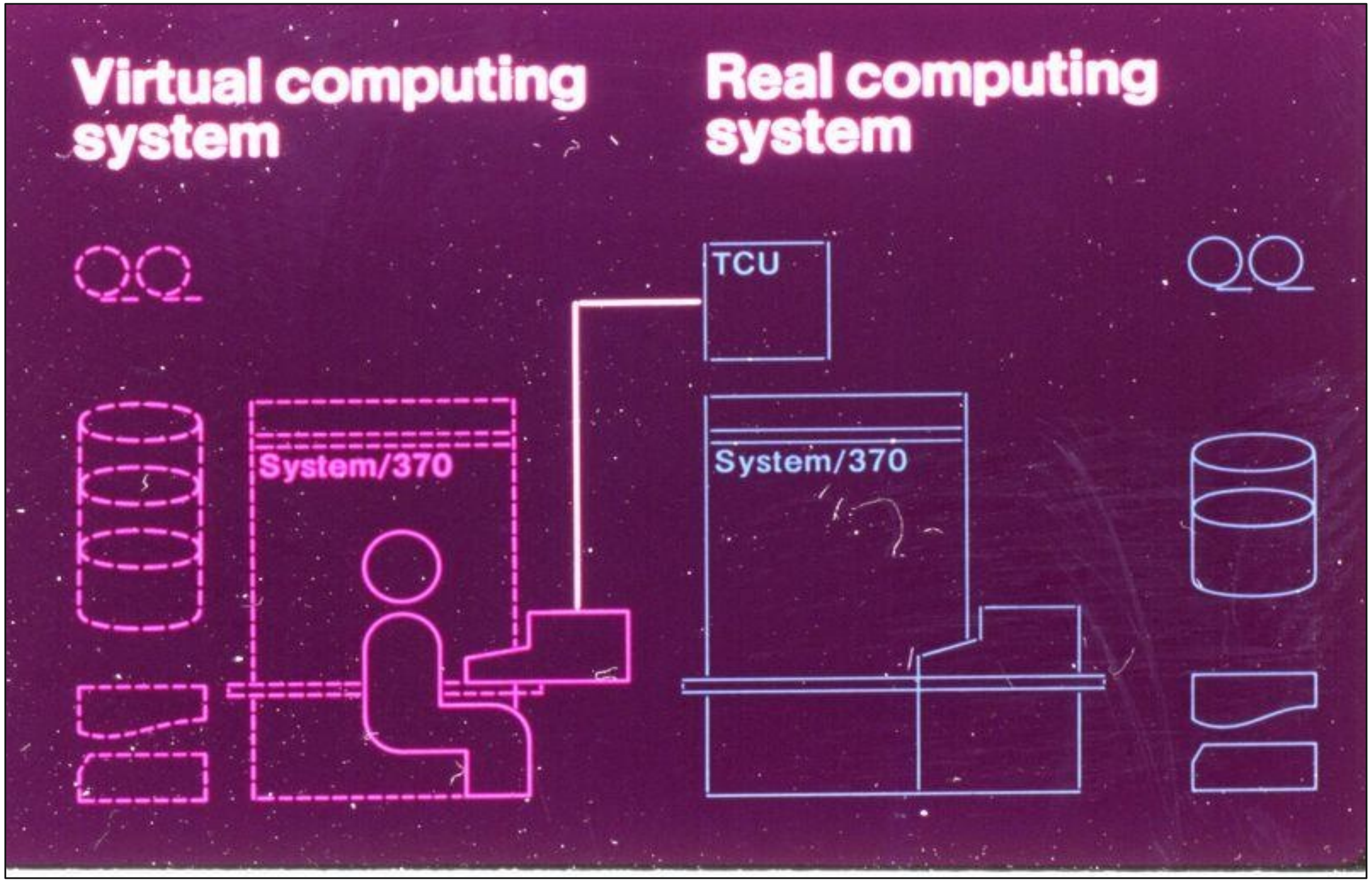
Cloud Motives

- Server Consolidation
  - Improve utilisation (possible to overcommit)
  - Significant cost savings (equipment, space, power)
- Simplified Management
  - Datacenter provisioning and monitoring
  - Dynamic load balancing
  - Migration (dead or alive)
- Improved Availability
  - Checkpointing
  - Fault tolerance
  - Disaster recovery
  - Replication
- Security
- Isolation
- Convenient for users



Yesterday's News

- Classical VMM
  - IBM S/360, IBM VM/370
  - Co-designed proprietary hardware, OS, VMM
- Applications
  - Timeshare several single-user OS instances on expensive hardware
  - Compatibility



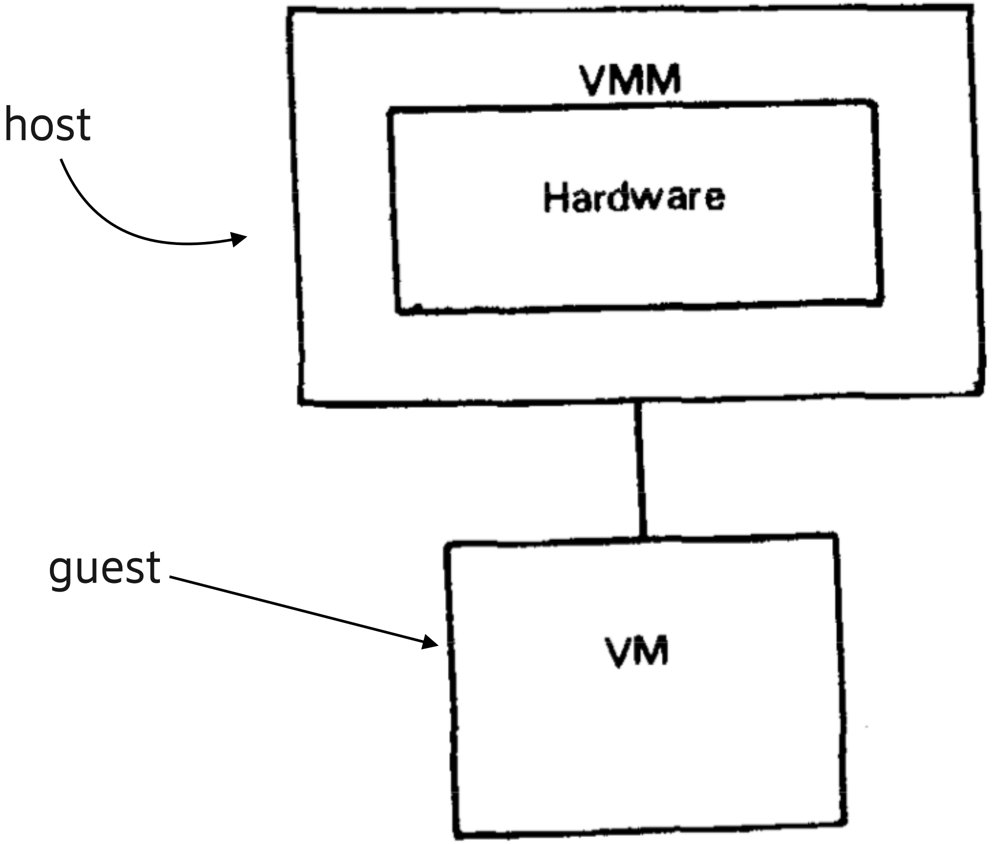
From IBM VM/370 product announcement, *ca*. 1972

Original Motives '65

- Multiprogramming
- Multiple single application VMs
- Multiple secure environments
- Managed application environments
- Mixed OS environments

- Legacy applications
- New systems transitions
- Software development
- OS training
- Help desk support
- Operating system instrumentation
- Event monitoring
- Check pointing

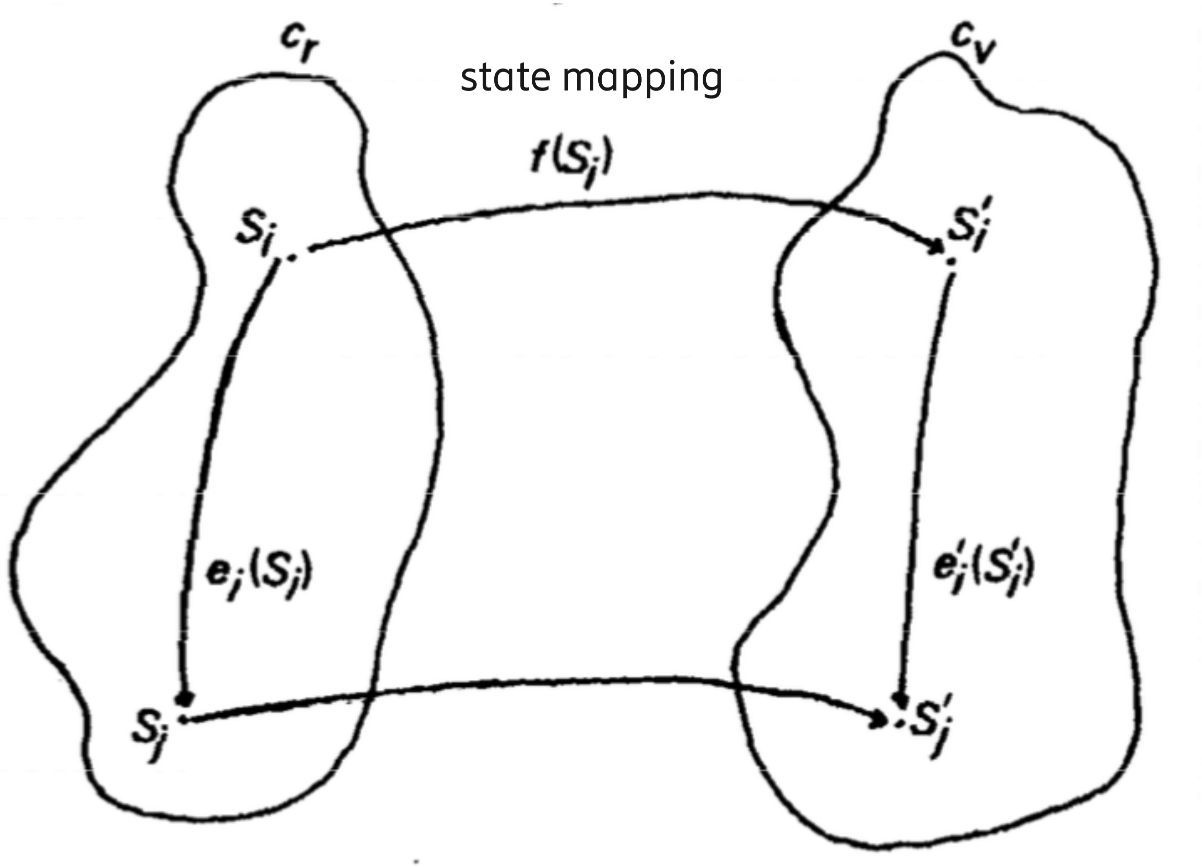
Popek & Goldberg '74



Hypervisor

A virtual machine is taken to be an *efficient, isolated duplicate* of the real machine. We explain these notions through the idea of a *virtual machine monitor* (VMM). See Figure 1. As a piece of software a VMM has three essential characteristics. First, the VMM provides an environment for programs which is essentially identical with the original machine; second, programs run in this environment show at worst only minor decreases in speed; and last, the VMM is in complete control of system resources.

Formally, virtualization involves the construction of an isomorphism that maps a virtual *guest* system to a real *host* guest host



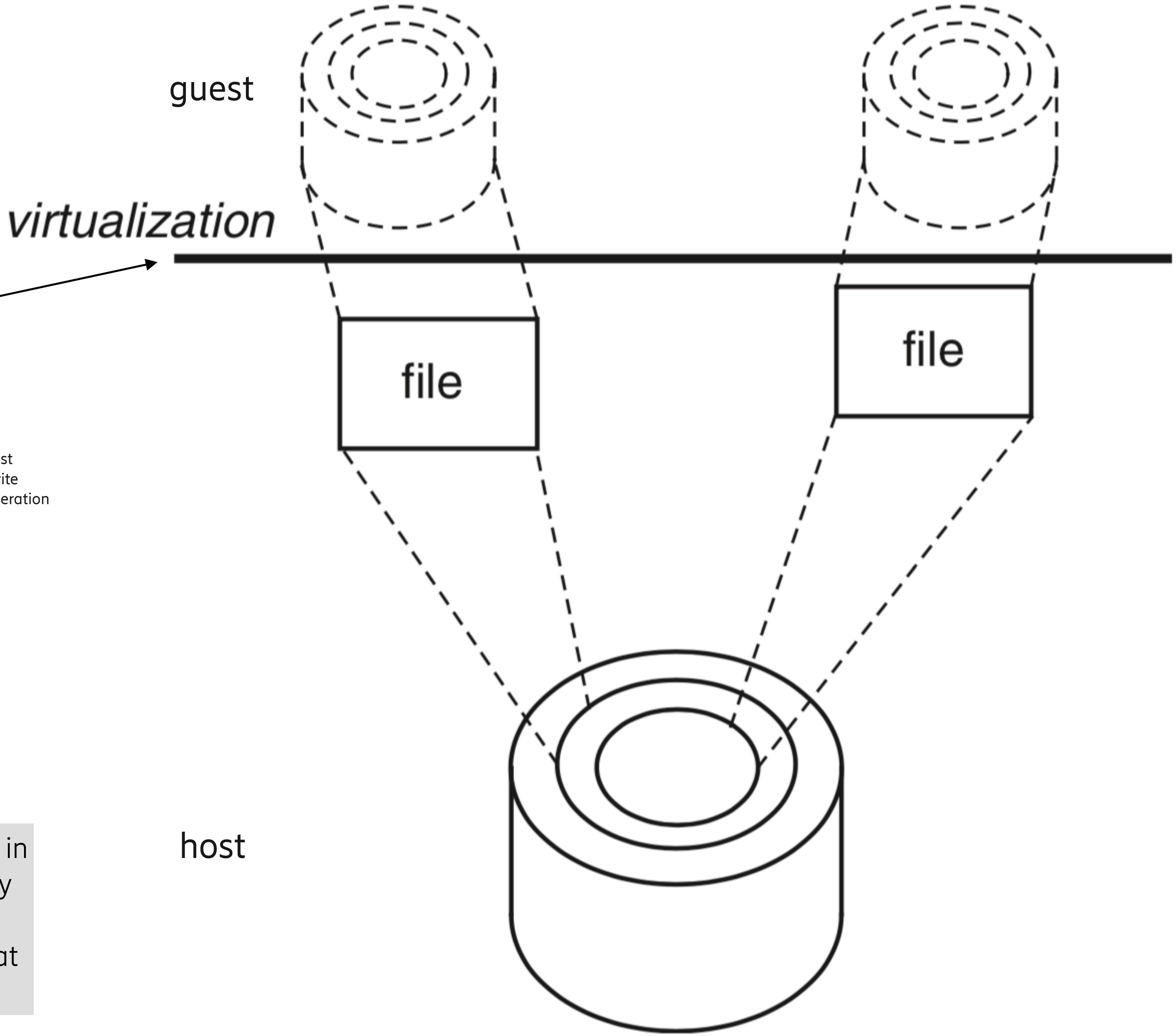
instruction sequence

existence of map & instruction sequences such that:

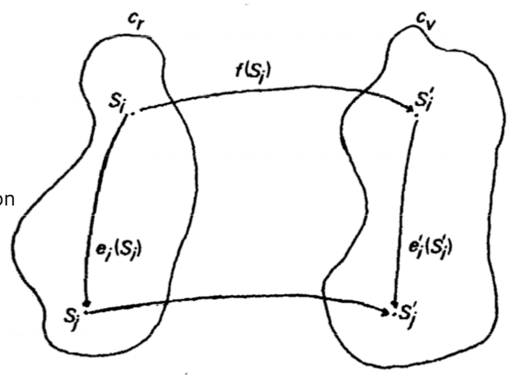
$$f(e_i(S_i) = e_i'(f(S_i))$$

Popek & Goldberg '74

Virtual disk



The function f()



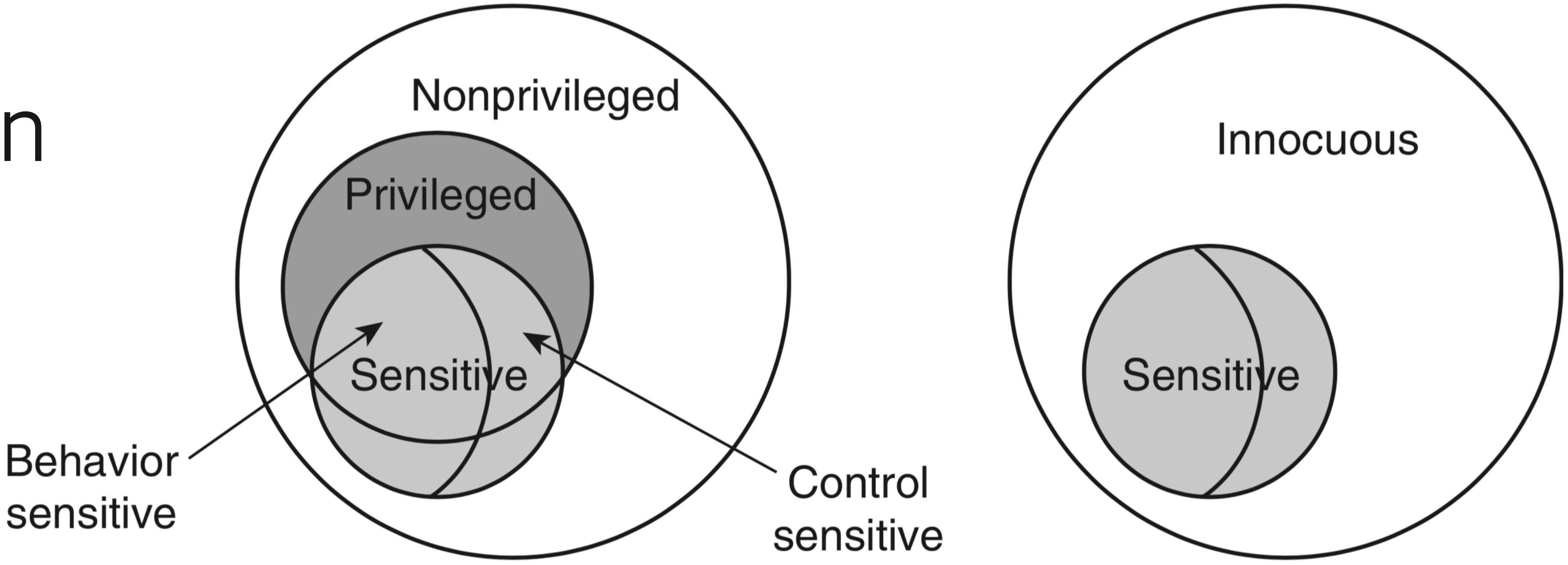
guest write operation

host write operation

Virtualization differs from abstraction in that virtualization does not necessarily hide details; the level of detail in a virtual system is often the same as that in the underlying real system.

CPU virtualisation

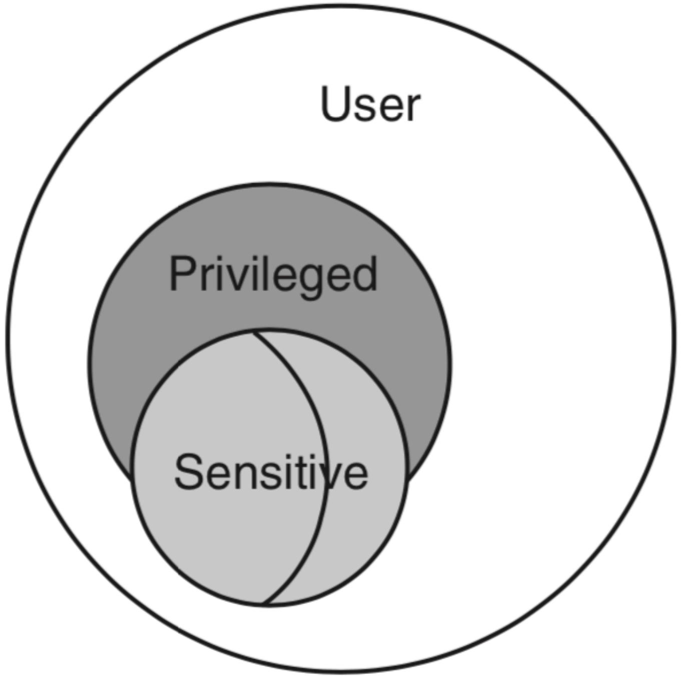
Popek & Goldberg '74



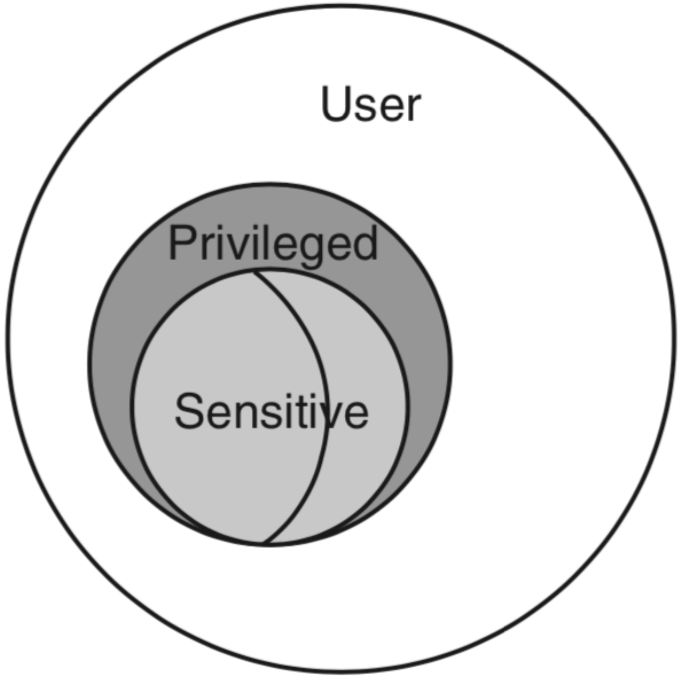
- Three types of instructions
  - Control sensitive
    - Change the configuration of resources
    - `Load PSW, Set CPU Timer (S/370)`
  - Behavior sensitive
    - Depend on the configuration of resources
    - `Load Real Address (S/370), Pop Stack into Flags Register (IA-32)`
  - Innocuous
    - The rest (klabbet)

CPU virtualisation

Popek & Goldberg '74



Does not satisfy condition



Satisfies condition — efficiently virtualizable

THEOREM 1. *For any conventional third generation computer, a virtual machine monitor may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.*

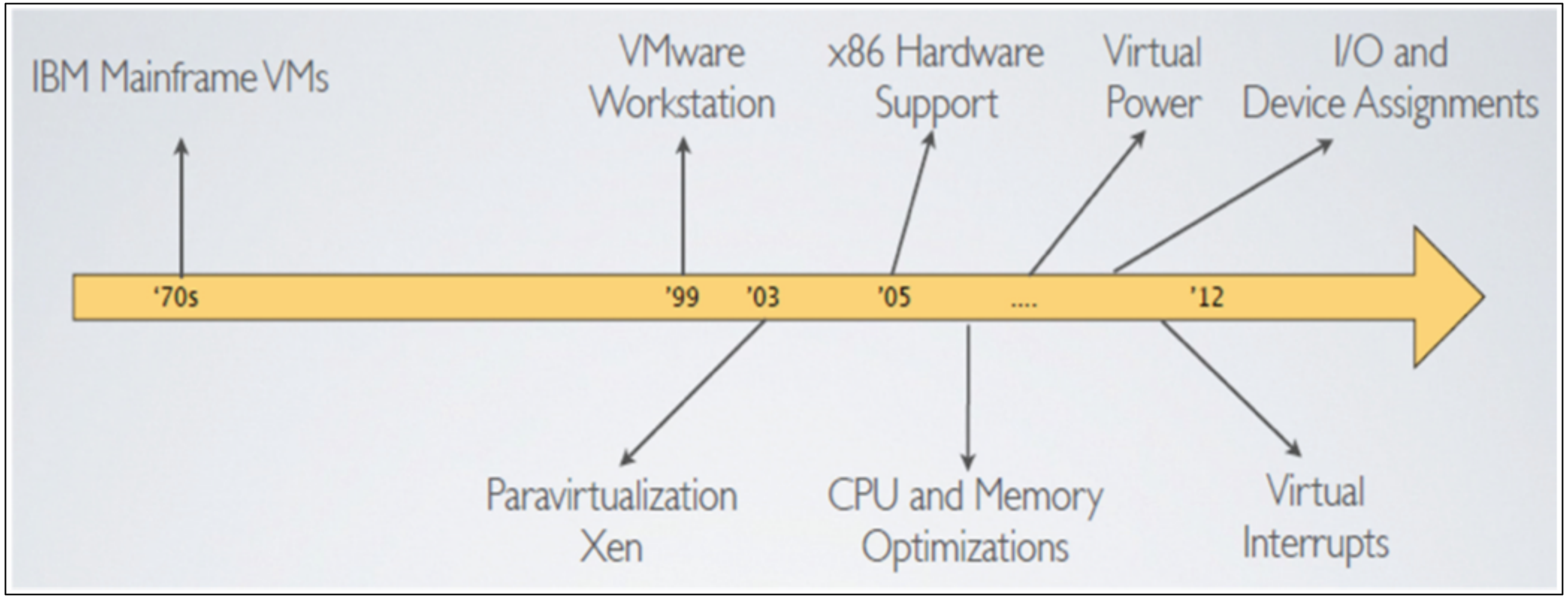
CPU virtualisation

Popek & Goldberg '74

- A VMM must satisfy three properties
  - Efficiency implies that all instructions that are innocuous must be executed natively on the hardware, with no intervention or emulation by the VMM.
  - Resource control implies that it should not be possible for guest software to directly change the configuration of any system resources available to it, e.g., real memory. The allocator must be invoked if the guest software makes any such attempt.
  - Equivalence implies that any program executing on a virtual machine must behave in a manner identical to the way it would have behaved when running directly on the native hardware, with only a few exceptions.

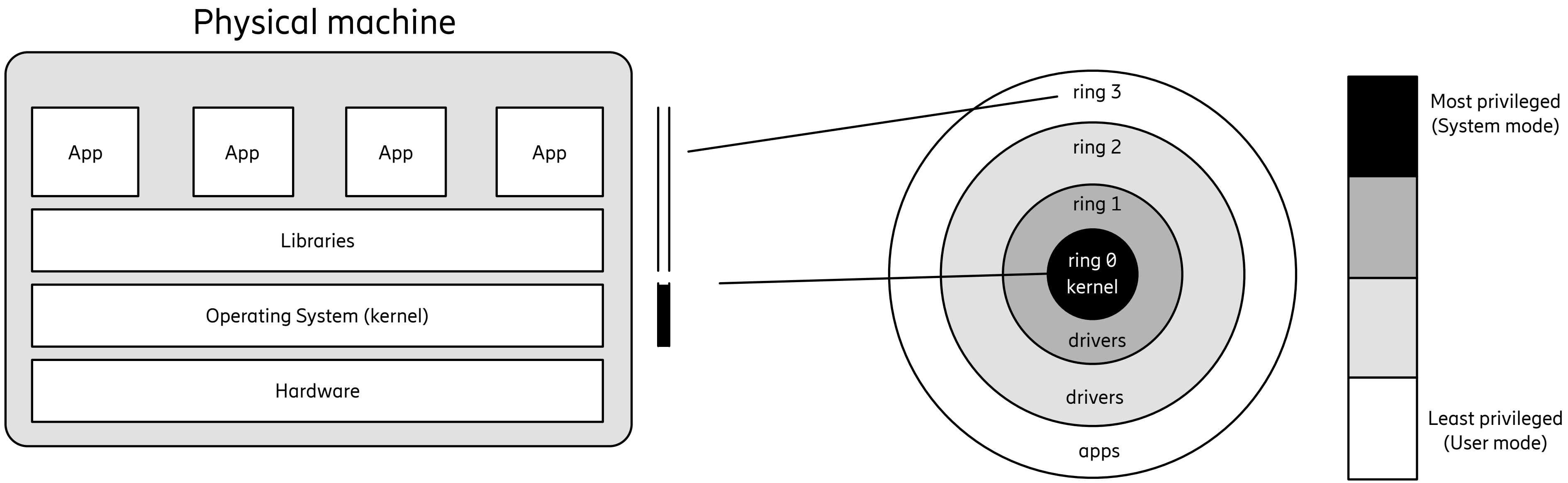
Virtualization Approaches

— Trap-and-emulate
— Binary translation
— Paravirtualization
— Hardware-assisted Virtualization

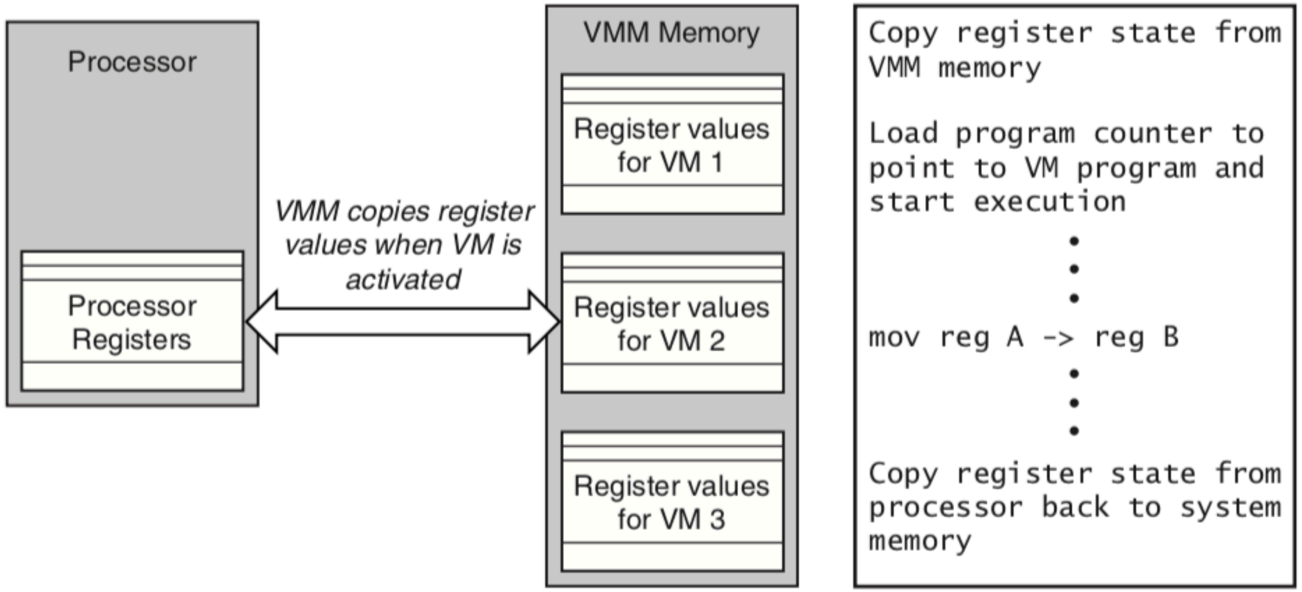


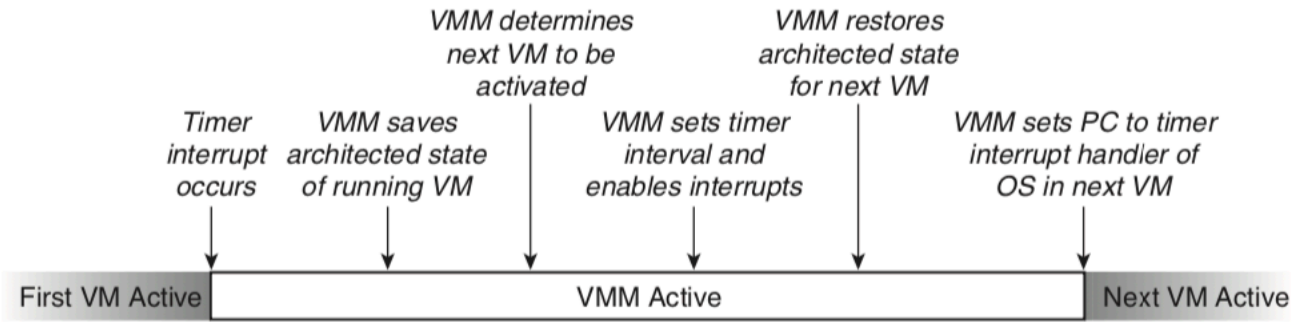
CPU virtualisation

Privileged instructions vs user instructions



Virtual State

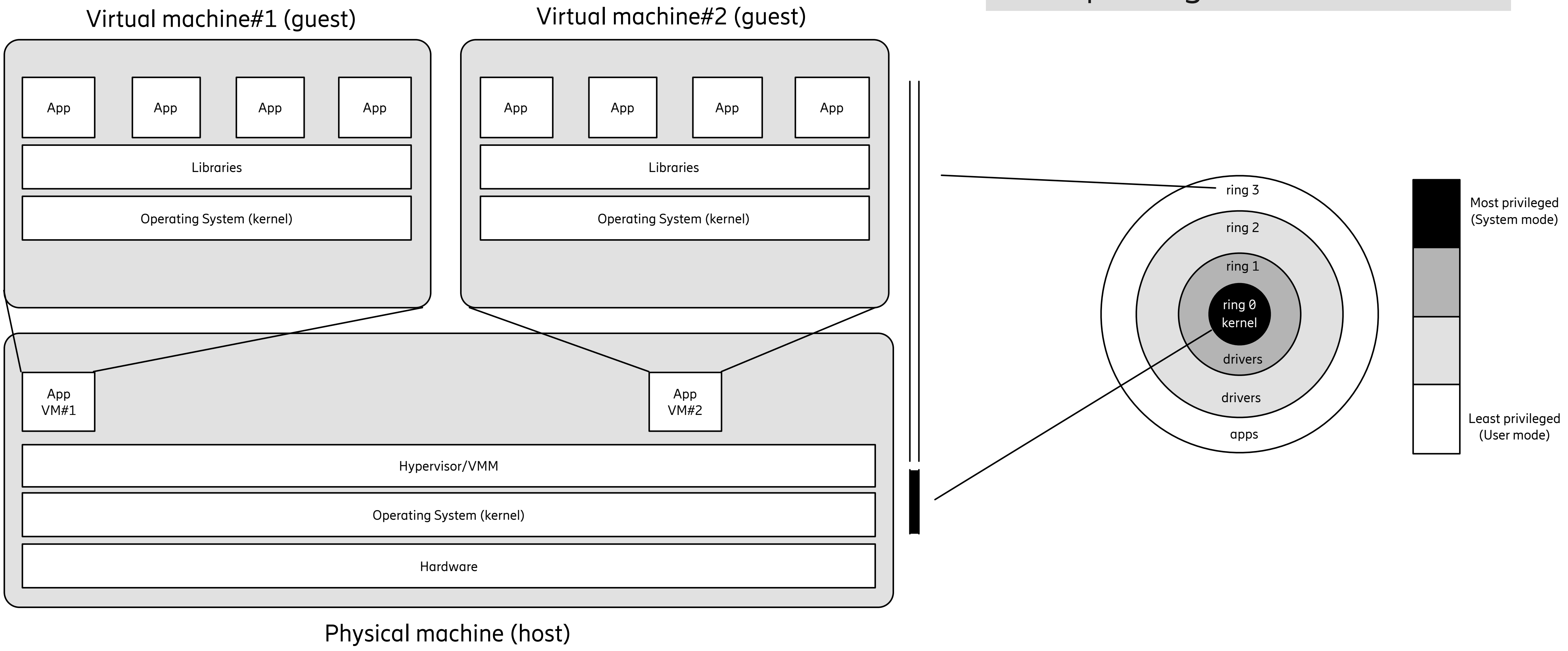




CPU virtualisation

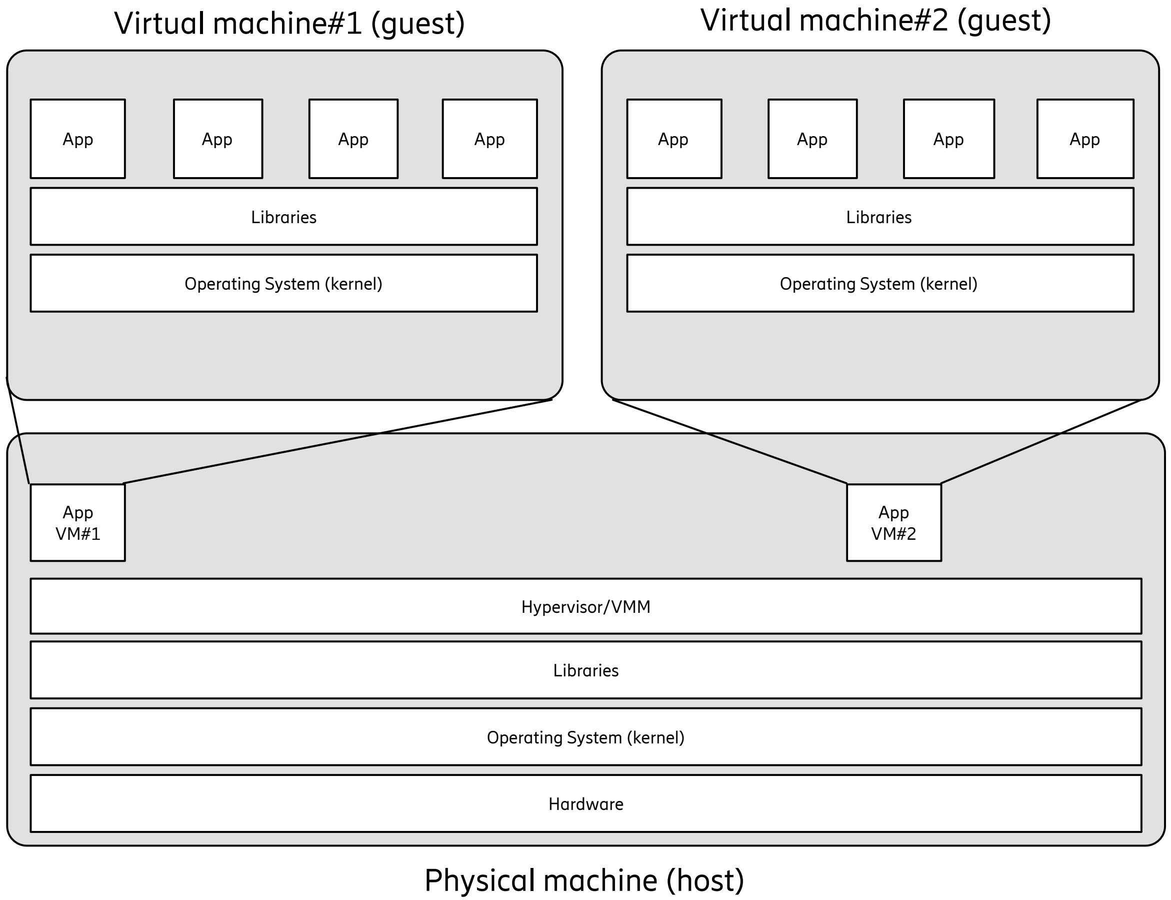
Privileged instructions vs user instructions

De-privileging - Run guest OS in unprivileged mode



CPU virtualisation

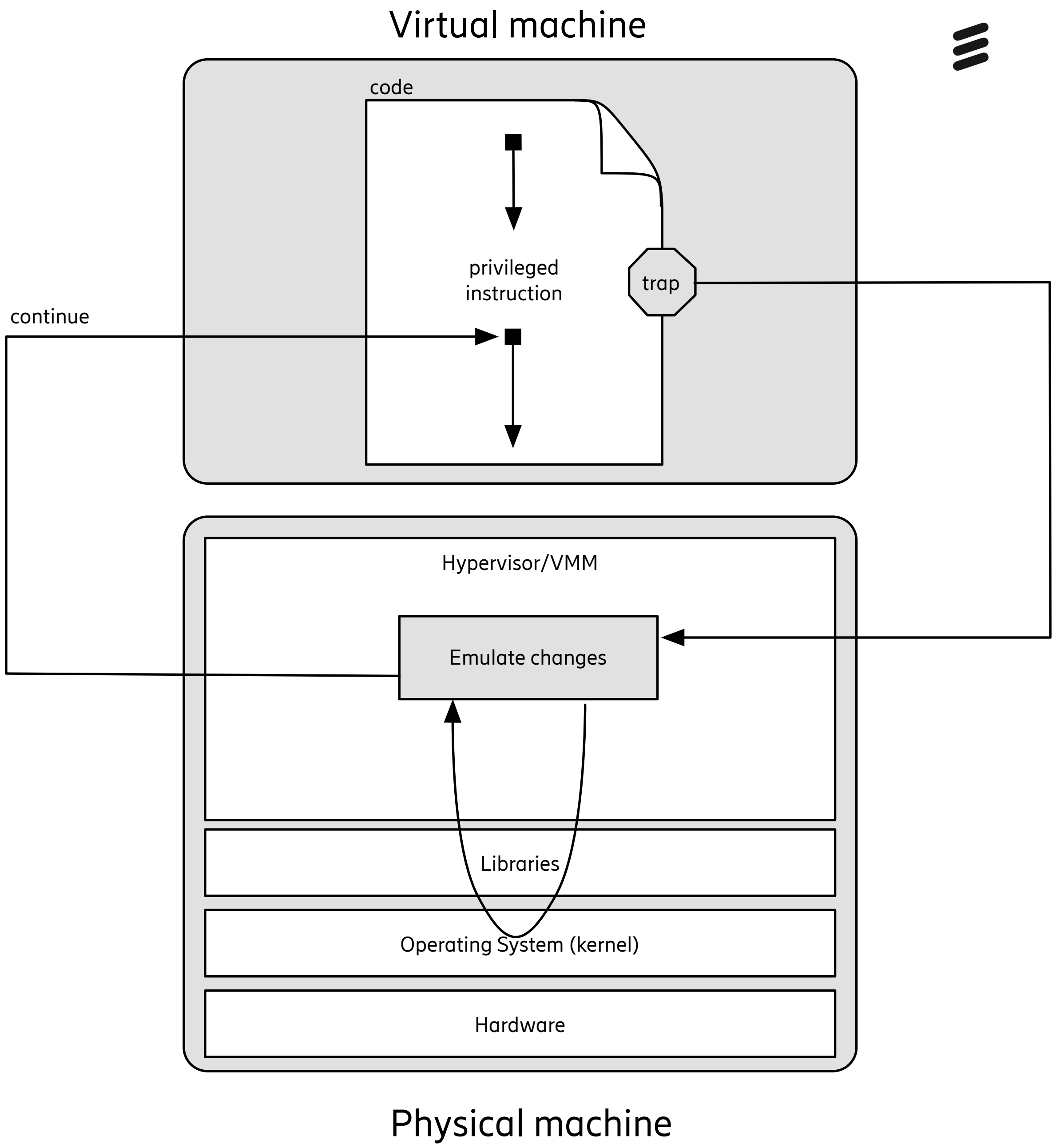
- The guest is typically just another user-level process (application)
- Facilitates processor sharing using standard operating system scheduling
- This allows for cloud providers to do overcommit, i.e. sell more compute power than is actually available.
  - Bet on that not everyone is running at the same time.



CPU virtualisation

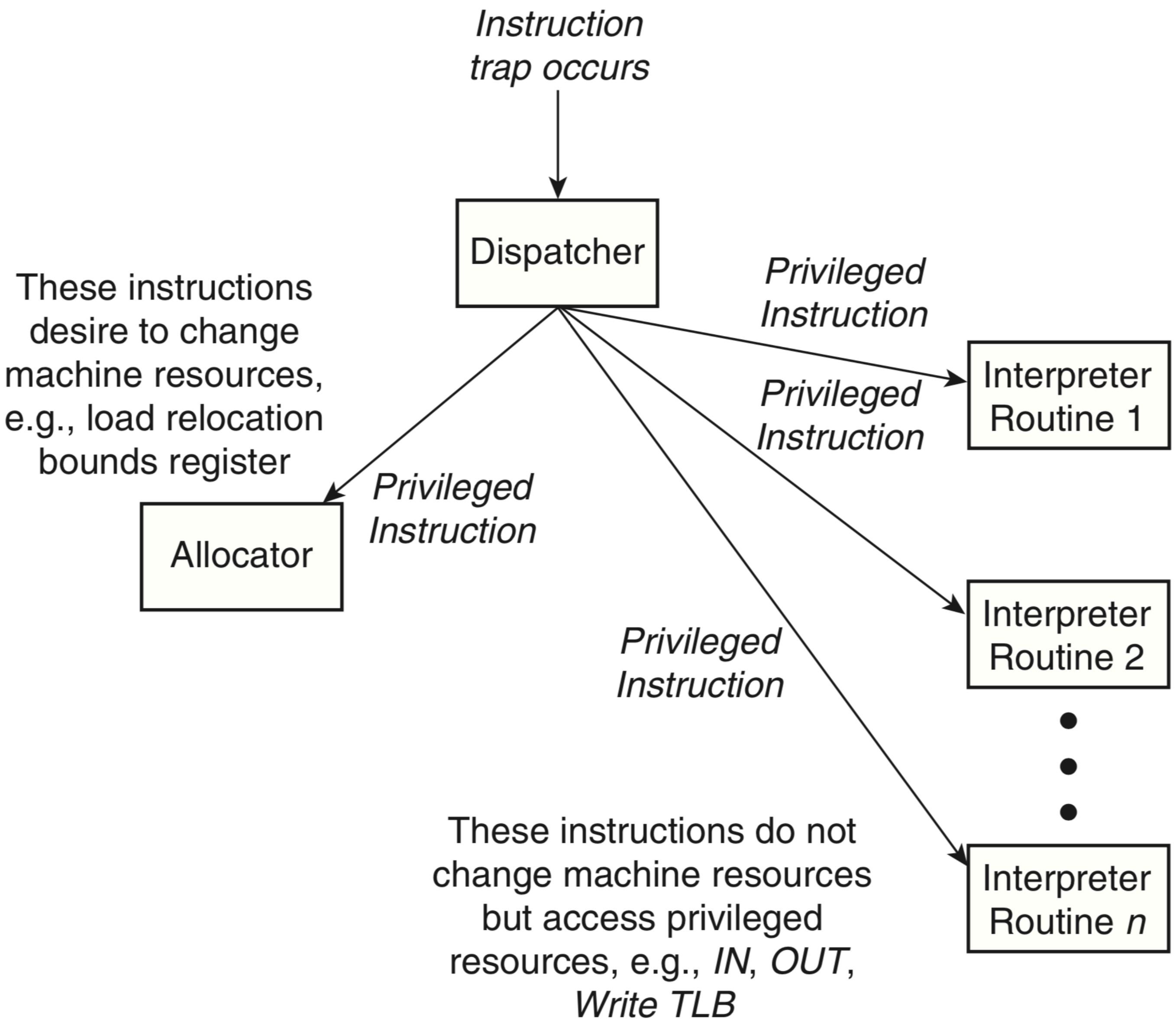
Trap and emulate

- Privileged instructions trap, and VMM emulates
  - E.g., movl %eax, %cr3 ; invalidate the TLB
  - Traps into VMM so the effect can be emulated
- Execute guest instructions on real CPU when possible
  - E.g., addl %eax, %ex



CPU virtualisation

Trap and emulate

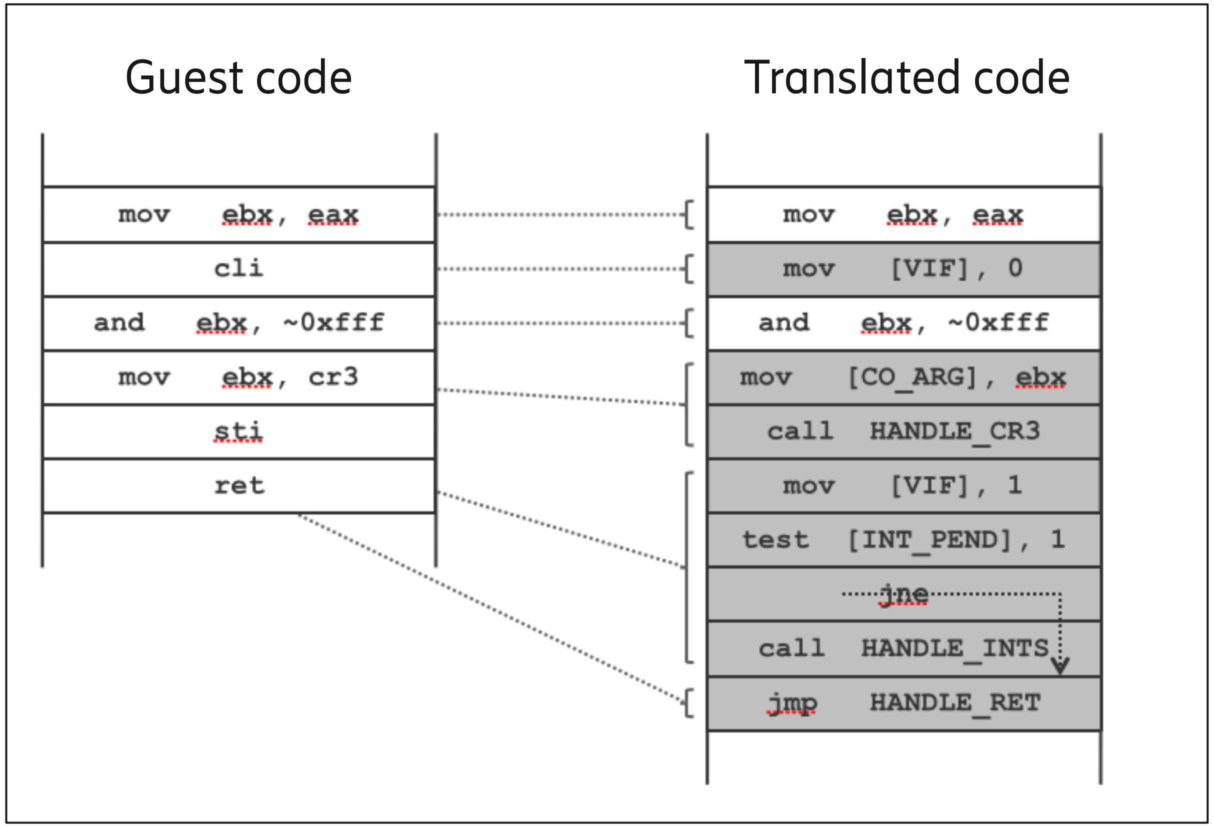


- VMM has three parts
  - Dispatcher
  - Allocater
  - Interpretor routines

CPU virtualisation

Binary translation

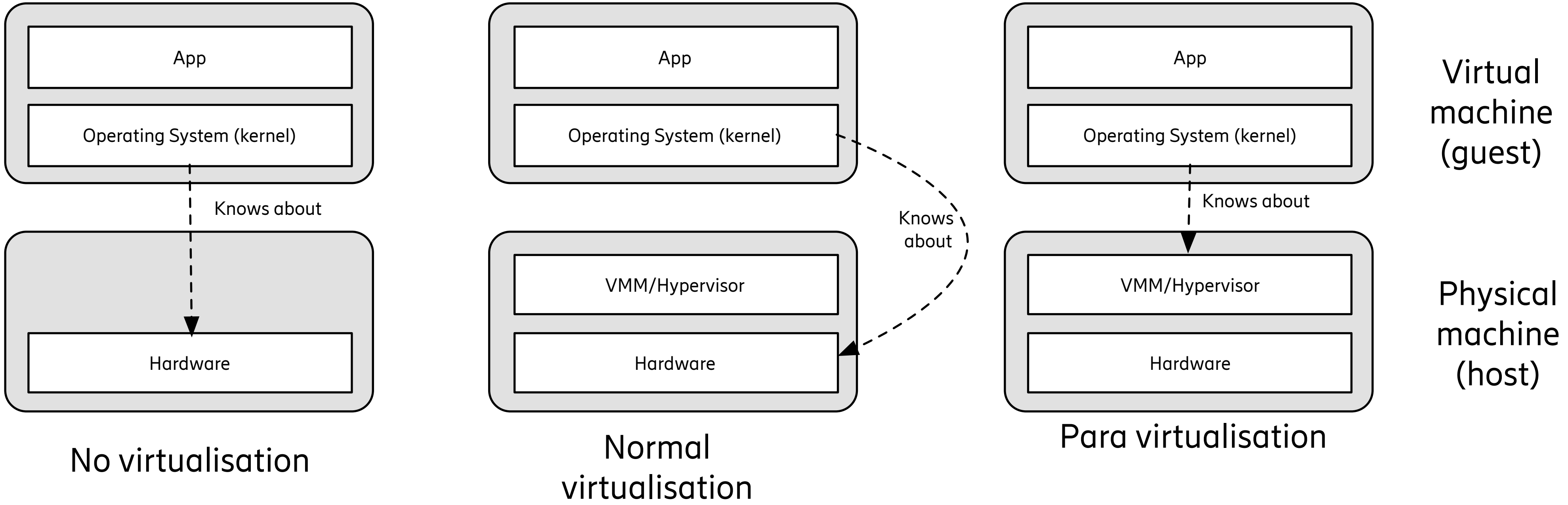
- Interpret the binary code
  - Replace privileged instructions
  - Avoids traps, which can be expensive
  - Most instructions remain identical, except control flow (calls, jumps, branches, ret, etc.), and privileged instructions
  - Dynamic or static
- Use cache to speed up
- Popularised by VMWare on x86



CPU virtualisation

Paravirtualisation

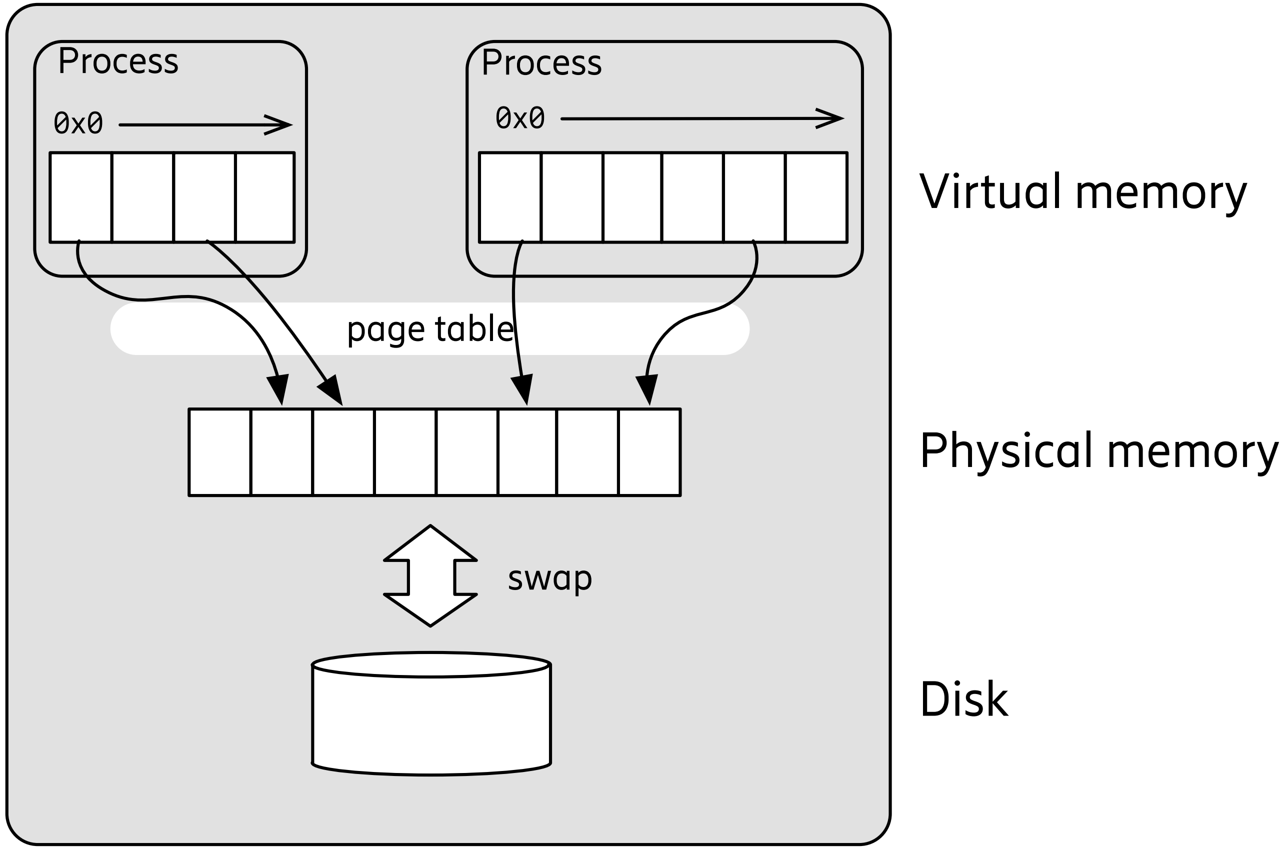
- OS or system devices are virtualization aware
  - Requires recompilation of the OS
  - Guest applications unaffected
  - In general good performance
- Popularised by XEN for x86



Memory virtualisation

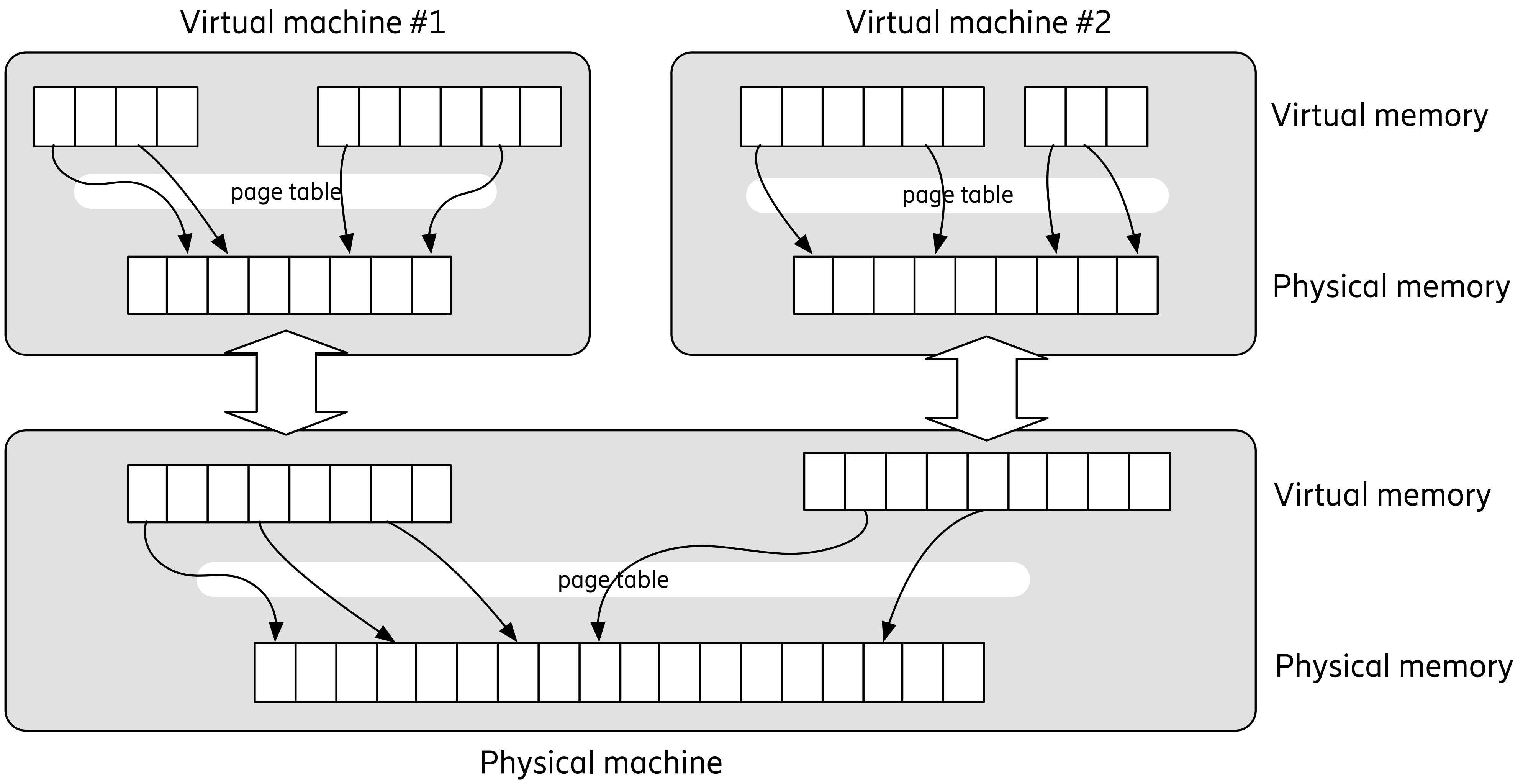
Virtual Memory 101

- Each process has its own space (usually starting at `0x0`)
- The page table keeps map of virtual memory to physical memory
- TBL is the page mapping cache
- Virtual memory enables memory isolation between user processes



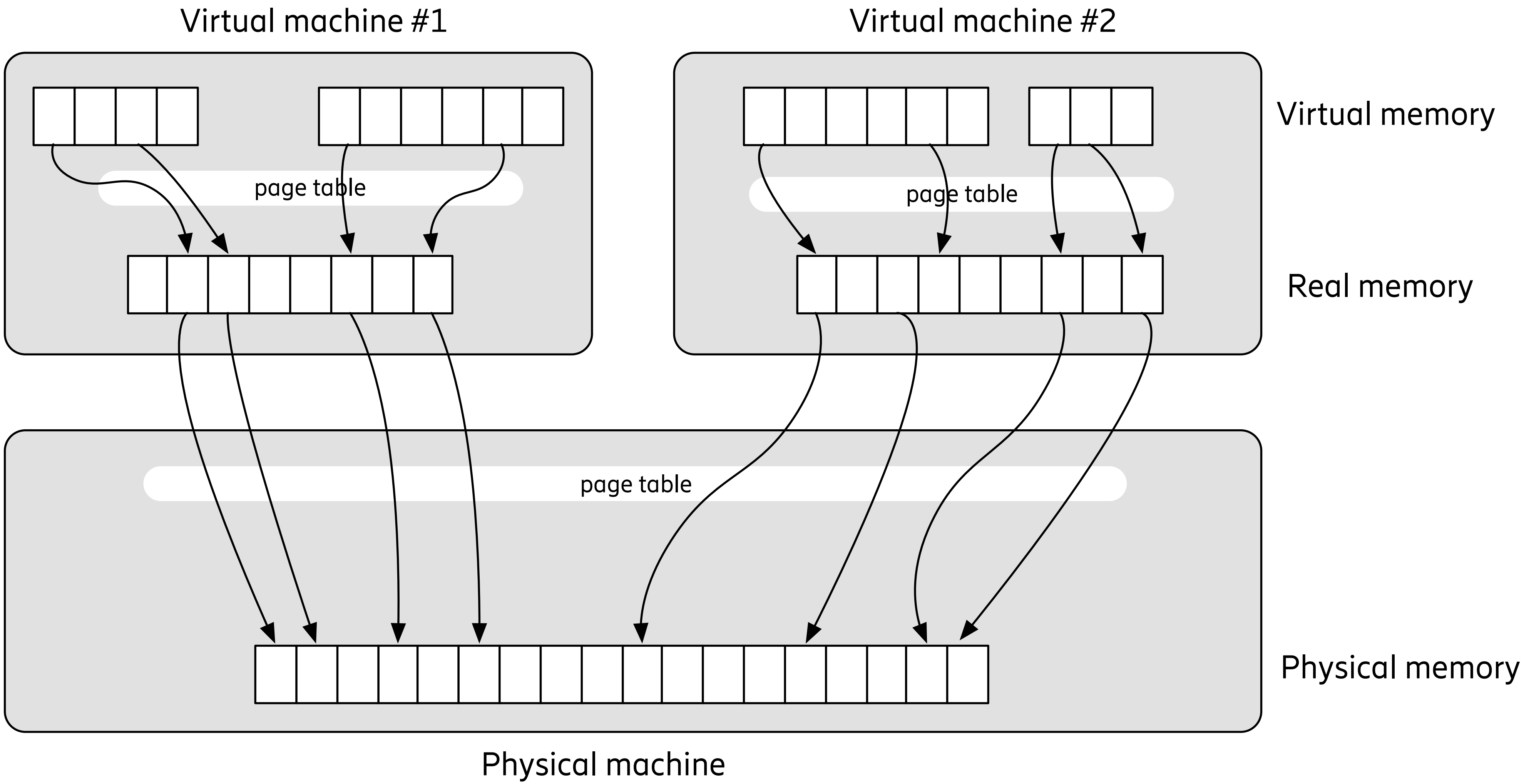
Memory virtualisation

When virtual memory is virtually virtualised



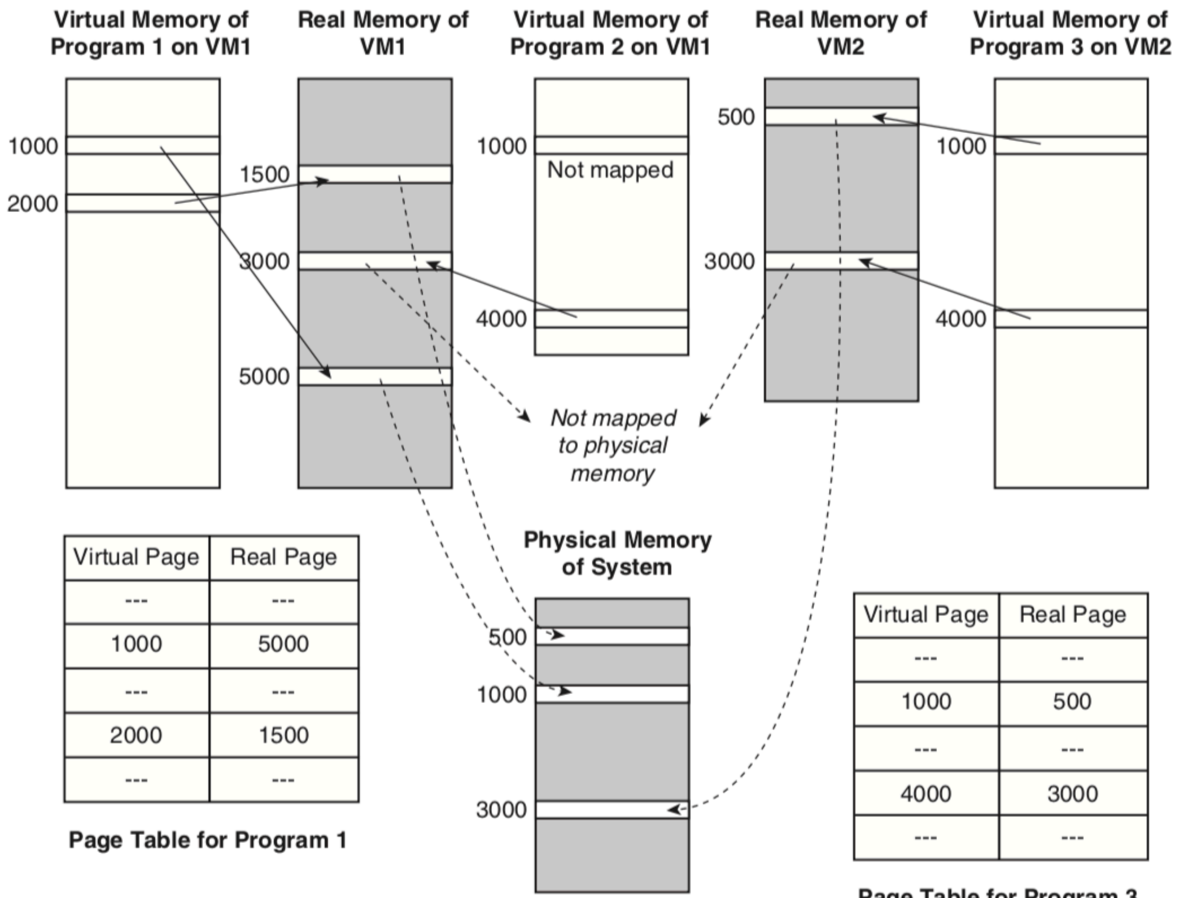
Memory virtualisation

When virtual memory is virtually virtualised



Memory virtualisation

When virtual memory is virtually virtualised



Virtual Page	Real Page
---	---
1000	5000
---	---
2000	1500
---	---

Page Table for Program 1

Virtual Page	Real Page
---	---
1000	*Not mapped*
---	---
4000	3000
---	---

Page Table for Program 2

Virtual Page	Real Page
---	---
1000	500
---	---
4000	3000
---	---

Page Table for Program 3

VM1 Real Page	Physical Page
---	---
1500	500
---	---
3000	*Not mapped*
---	---
5000	1000
---	---

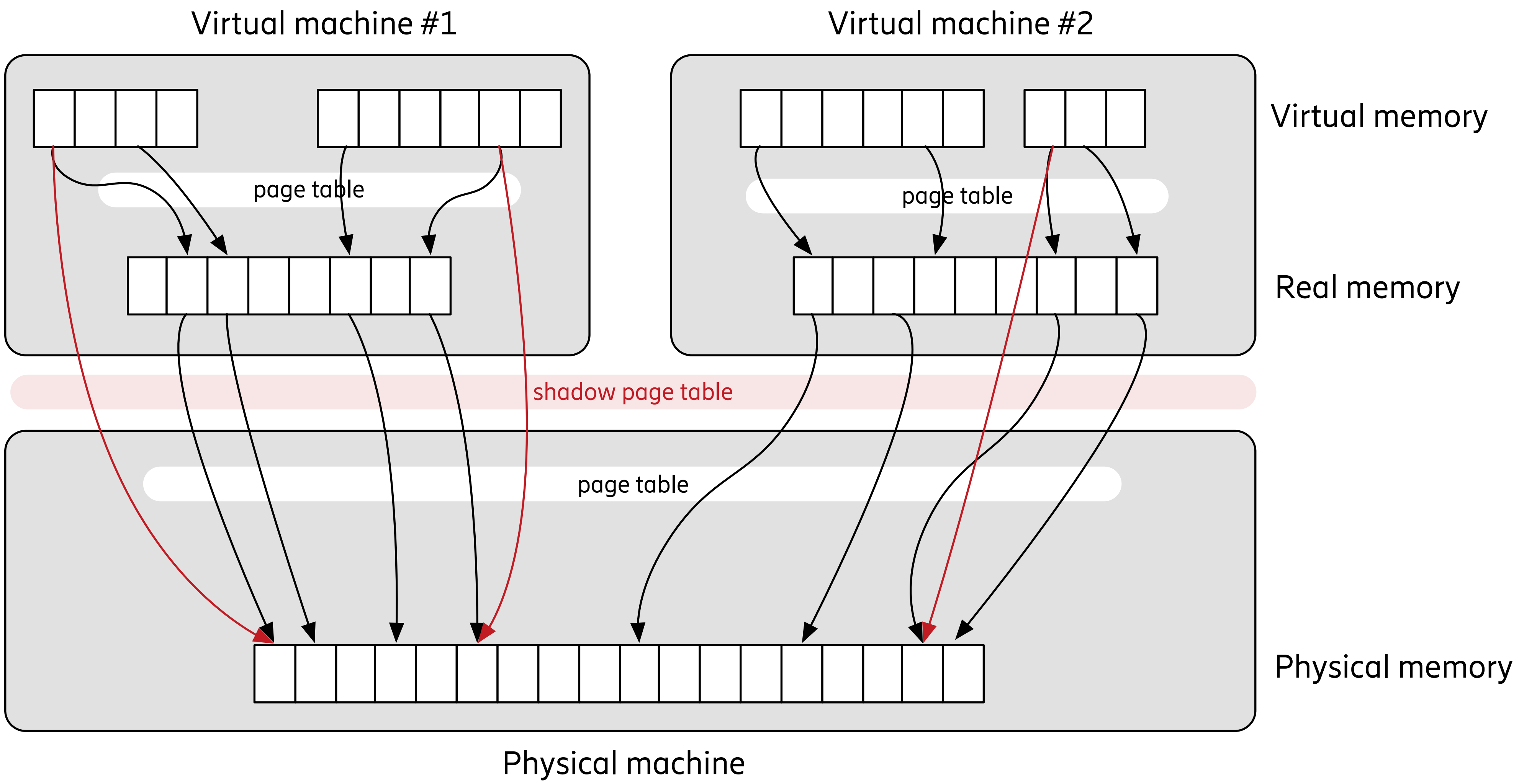
Real Map Table for VM1

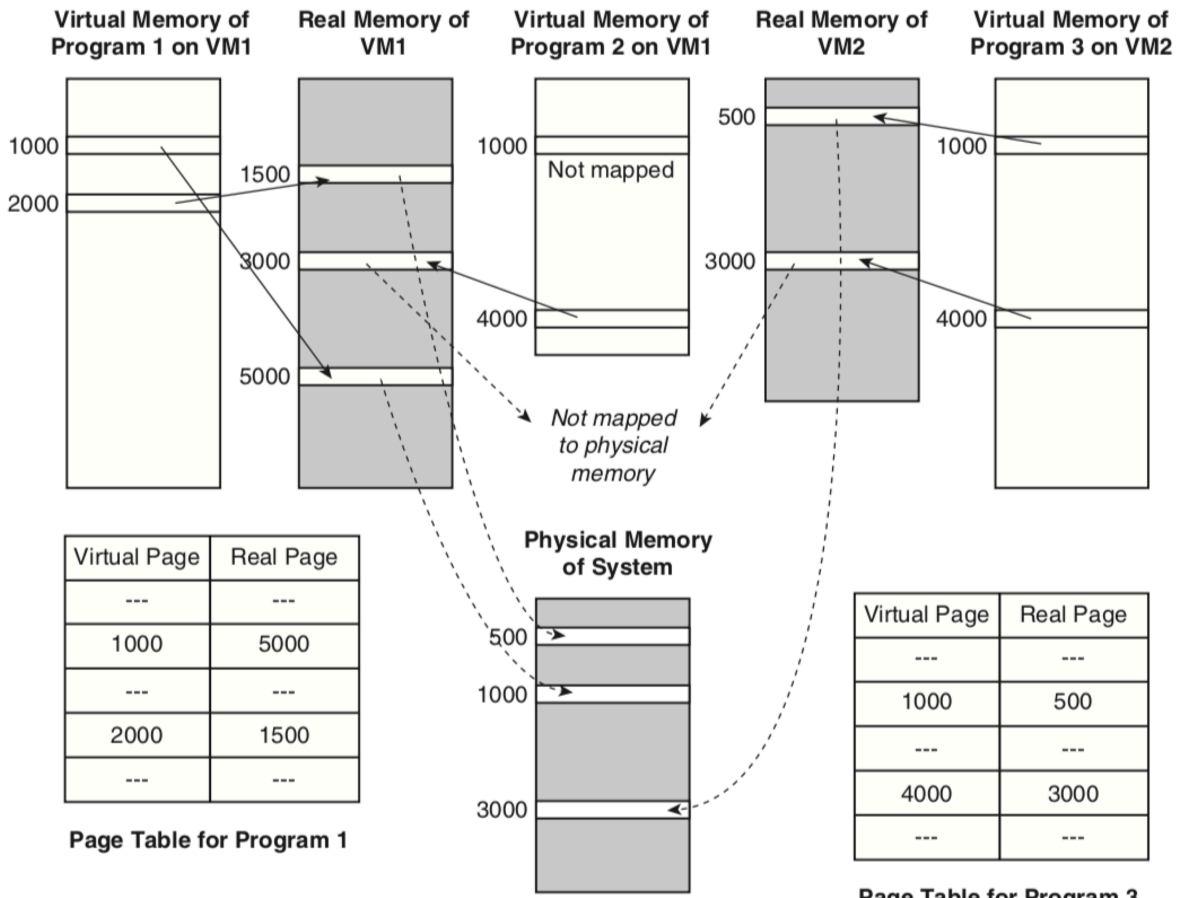
VM2 Real Page	Physical Page
---	---
500	3000
---	---
3000	*Not mapped*
---	---

Real Map Table for VM2

Memory virtualisation

When virtual memory is virtually virtualised





Virtual Page	Real Page
---	---
1000	5000
---	---
2000	1500
---	---

Page Table for Program 1

Virtual Page	Real Page
---	---
1000	*Not mapped*
---	---
4000	3000
---	---

Page Table for Program 2

Virtual Page	Real Page
---	---
1000	500
---	---
4000	3000
---	---

Page Table for Program 3

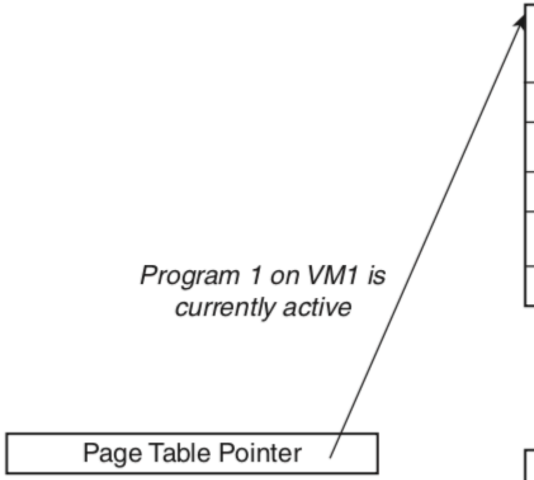
VM1 Real Page	Physical Page
---	---
1500	500
---	---
3000	*Not mapped*
---	---
5000	1000
---	---

Real Map Table for VM1

VM2 Real Page	Physical Page
---	---
500	3000
---	---
3000	*Not mapped*
---	---

Real Map Table for VM2

Shadow Page Tables Maintained by VMM



Virtual Page	Physical Page
---	---
1000	1000
---	---
2000	500
---	---

Shadow Page Table for Program 1 on VM1

Virtual Page	Physical Page
---	---
1000	*Not mapped*
---	---
4000	*Not mapped*
---	---

Shadow Page Table for Program 2 on VM1

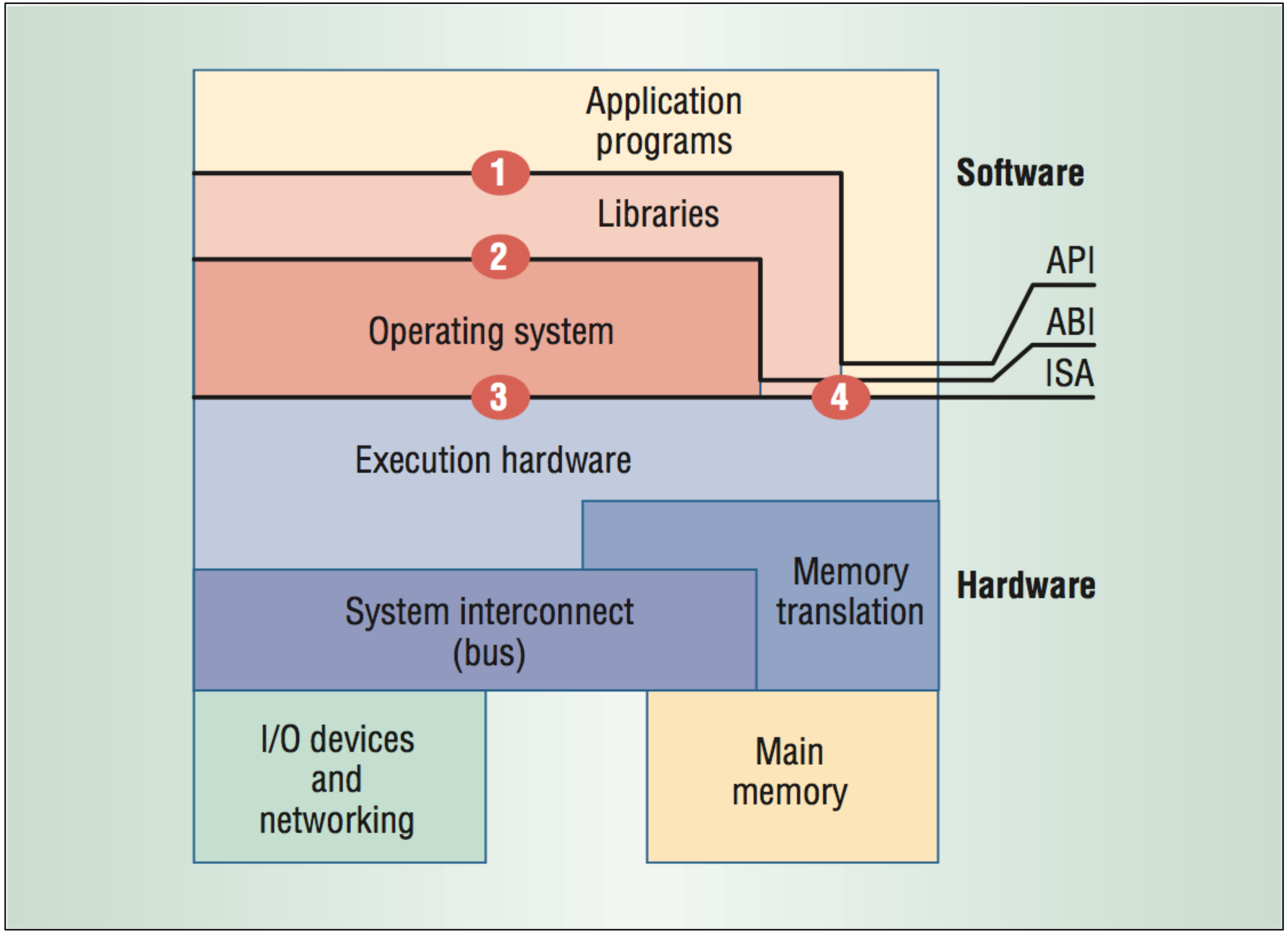
Virtual Page	Physical Page
---	---
1000	3000
---	---
4000	*Not mapped*
---	---

Shadow Page Table for Program 3 on VM2

Virtualization Interfaces

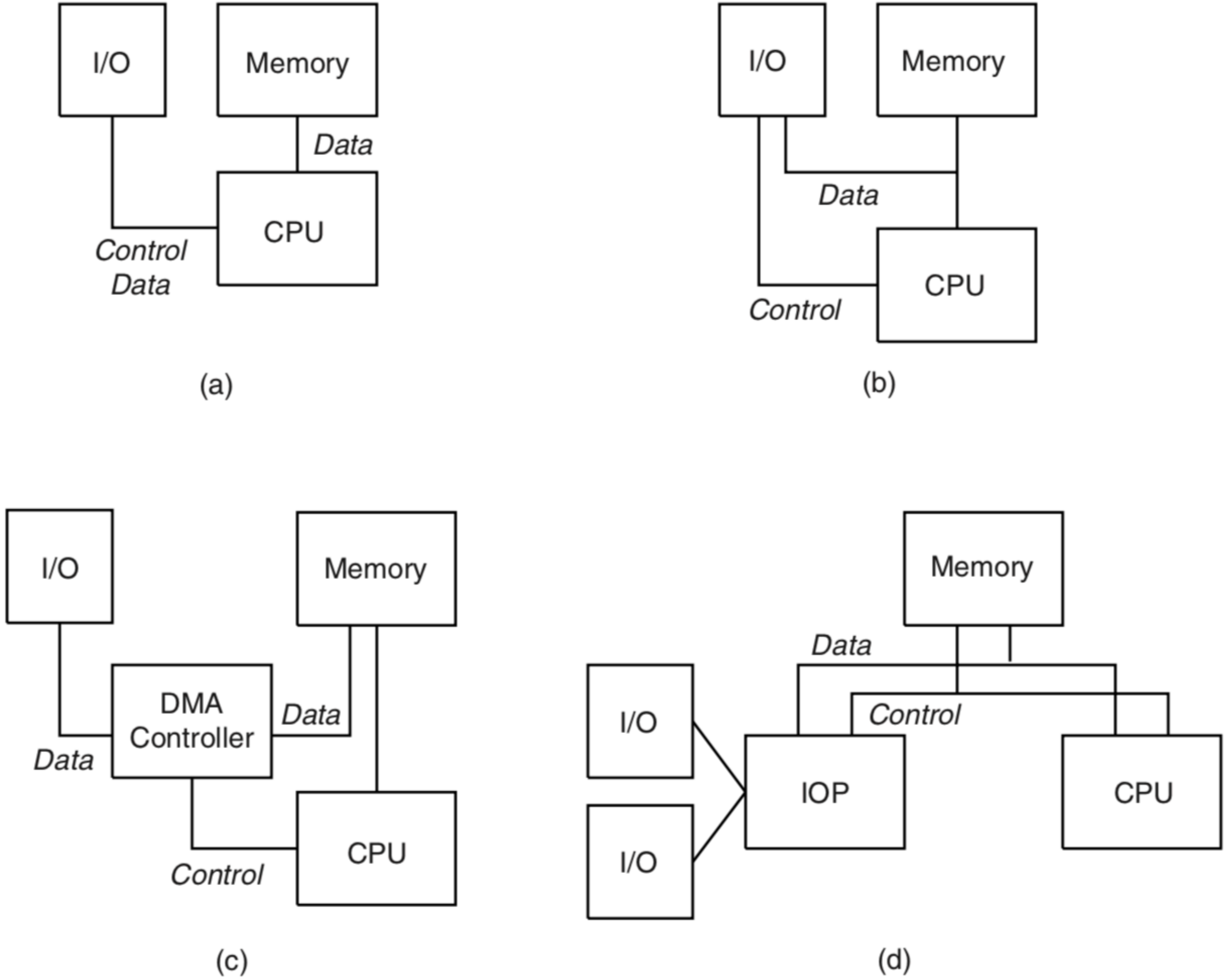
Until now we have looked at system level virtualisation, i.e. the whole machine is virtualised.

But that is not the only option!

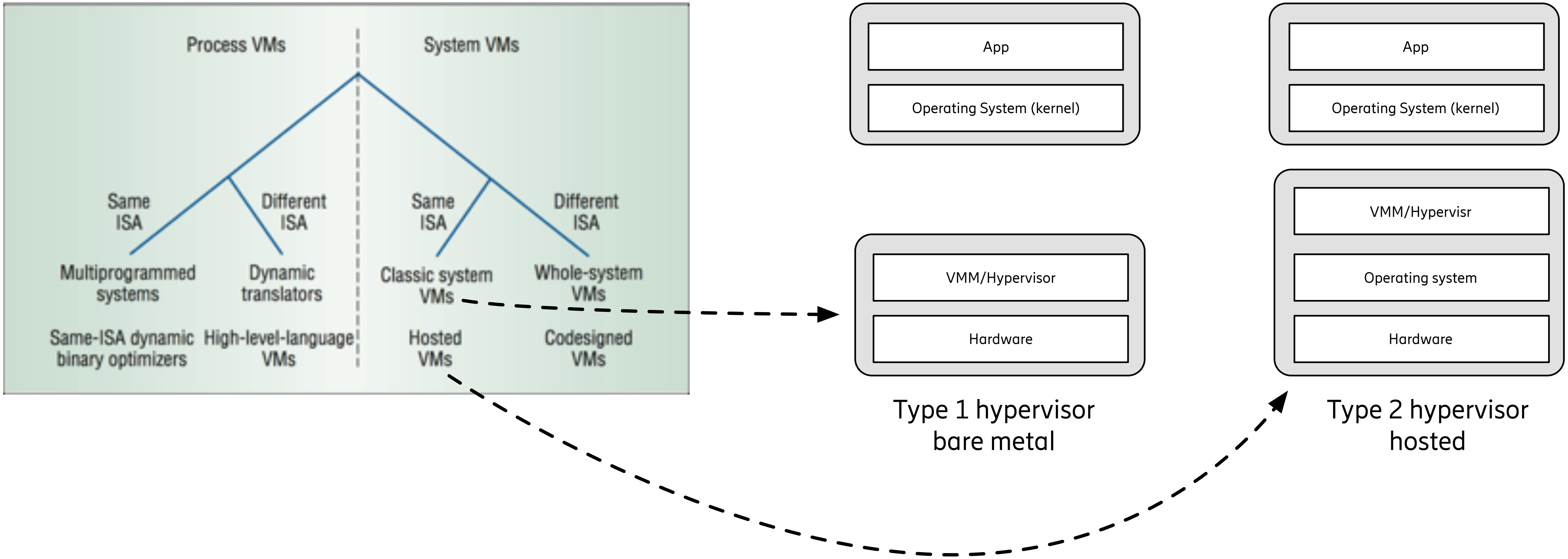


ISA = Instruction Set Architecture
3 = System ISA (Privileged calls)
4 = User ISA (User level calls)
ABI = Application Binary Interface

API = Application Programming Interface



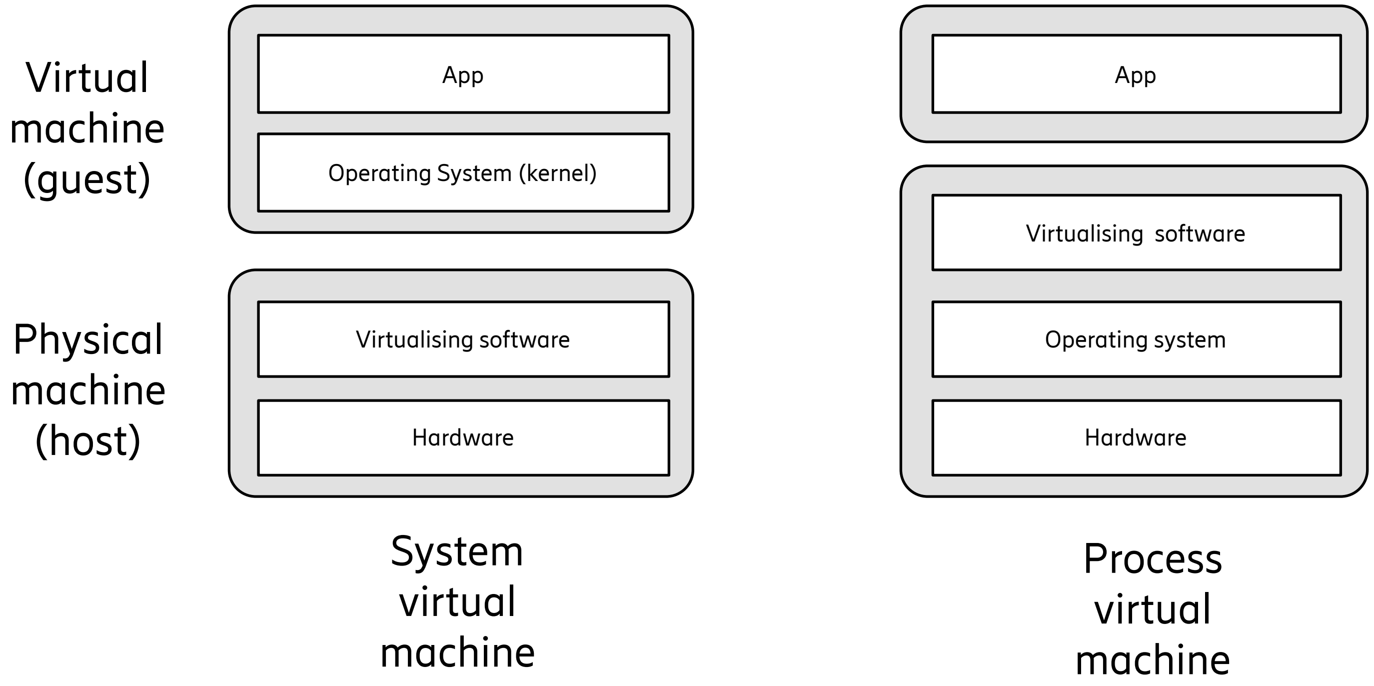
Different Types of Input/Output. *(a) Programmed I/O; (b) interrupt-driven I/O; (c) DMA-managed I/O; (d) IOP-based I/O.*


Process VMs
System VMs
Same ISA
Different ISA
Same ISA
Different ISA
Multiprogrammed systems
Dynamic translators
Classic system VMs
Whole-system VMs
Same-ISA dynamic binary optimizers
High-level-language VMs
Hosted VMs
Codesigned VMs
App
Operating System (kernel)
VMM/Hypervisor
Hardware
Type 1 hypervisor
bare metal
App
Operating System (kernel)
VMM/Hypervisr
Operating system
Hardware
Type 2 hypervisor
hosted

System VM vs Process VM

Until now we have looked at system level virtualisation, i.e. the whole machine is virtualised.

But that is not the only option!



LXC - Linux Containers

- Lightweight process level virtualization
- No VM (or VMM/hypervisor), just a Linux process
- A user space interface for the Linux kernel containment features:
  - Kernel namespaces, Apparmor/SELinux, Seccomp, Chroots, Kernel capabilities, cgroups
- Multiple containers share the same kernel
- A long story…
  - Chroot (1979) – change root directory for a running process, along with children → segregate and isolate processes, protecting global environment
  - Jails – additional process sandboxing features for isolating filesystems, users, networks (limiting apps in their functionality)
  - Solaris Zones – full application environments, with full user, process and filesystem space
  - Cgroups(2006) – process containers designed for isolating and limiting the resource usage of a process

Enter Docker Containers

- A user-space process (LXC)
  - Isolation based on Linux process mechanisms
  - Each container has its own network stack and file system
  - Share kernel with host
  - Containers can be stopped, paused, restarted

Name borrowed from the shipping industry, hence the aquatic theme.

Portability - can be used on any of supported types of ships

Wide variety of cargo that can be packed inside

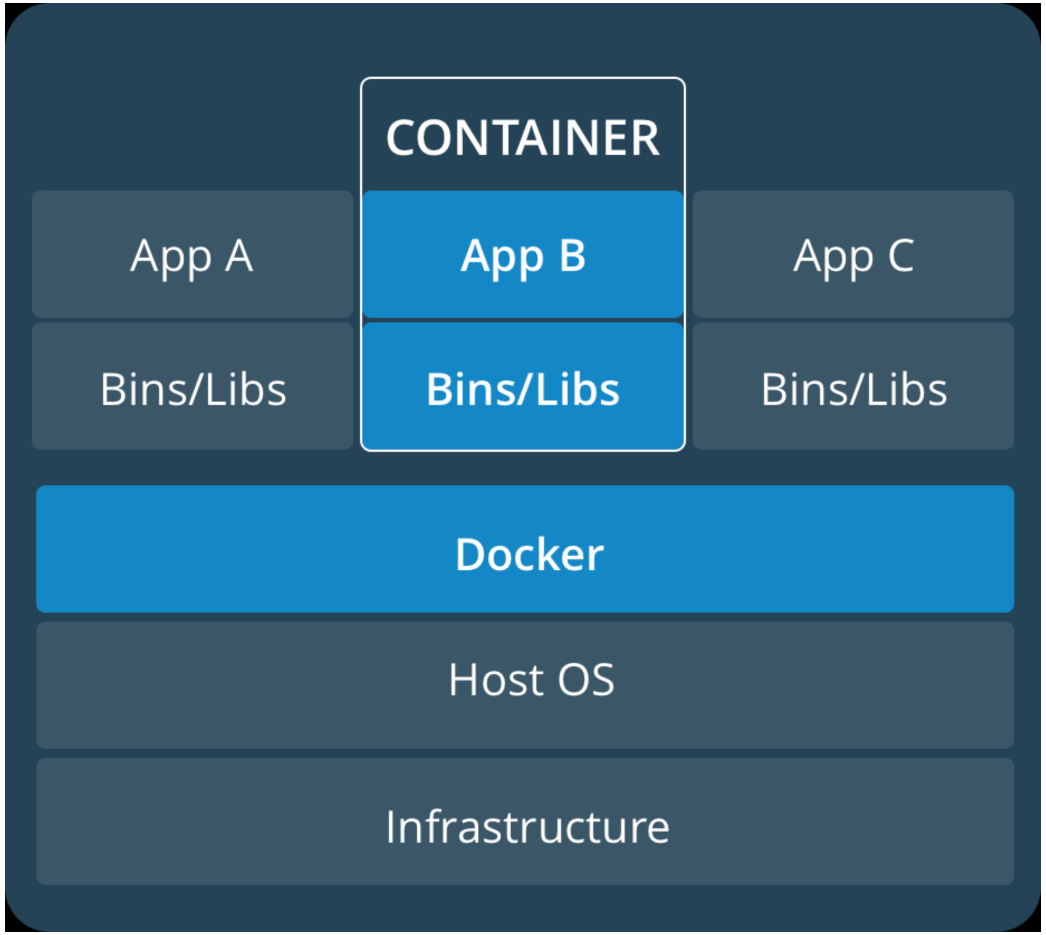
Standard sizes - standard fittings on ships

Many containers on a ship

Isolates cargo from each other

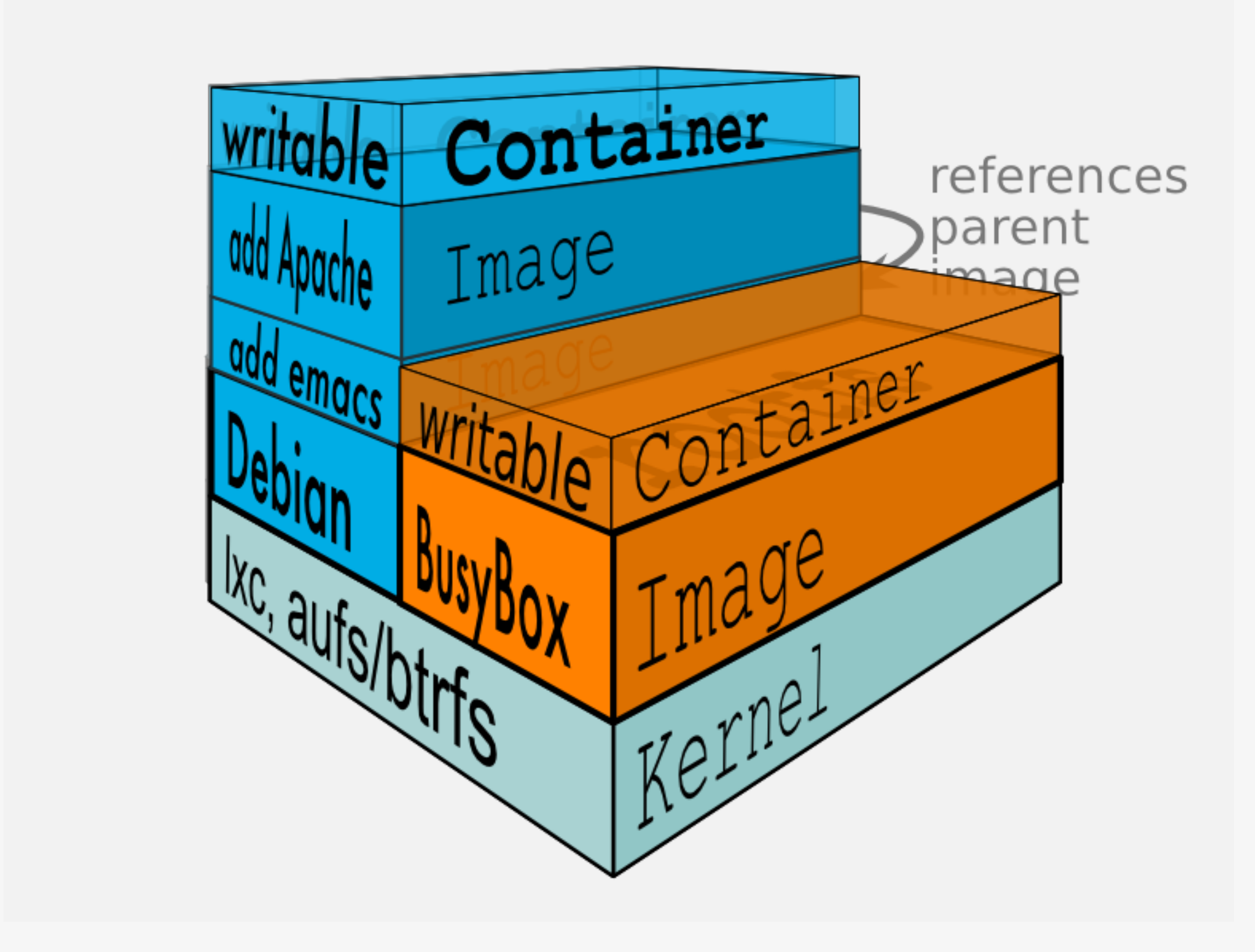
What does Docker offer?

- A simple way to pack code and dependencies together
- Apps that can run anywhere
- Low overhead
- A complete ecosystem for sharing images



Docker Containers

- Each container is built from a Docker image.
  - Images are read-only
  - Union mount merges the images together with a writable top layer
  - Copy-on-write
- Docker registries to store and publish images
  - DockerHub, etc.
  - Tons of applications ready for download
- Docker images are built in an hierarchical fashion, which facilitates collaboration and innovation
- Fast to start and stop
- Runs equally well on your laptop and in the cloud
- Solves the dependency mess



Docker Files

A recipe for building images

Dockerfile

```
FROM ubuntu:14.04
MAINTAINER Linus Karlsson <linus.karlsson@eit.lth.se>

RUN apt-get update && apt-get install -y python-pip
RUN pip install Flask
ADD server.py /srv/server.py

EXPOSE 5000
CMD python /srv/server.py
```

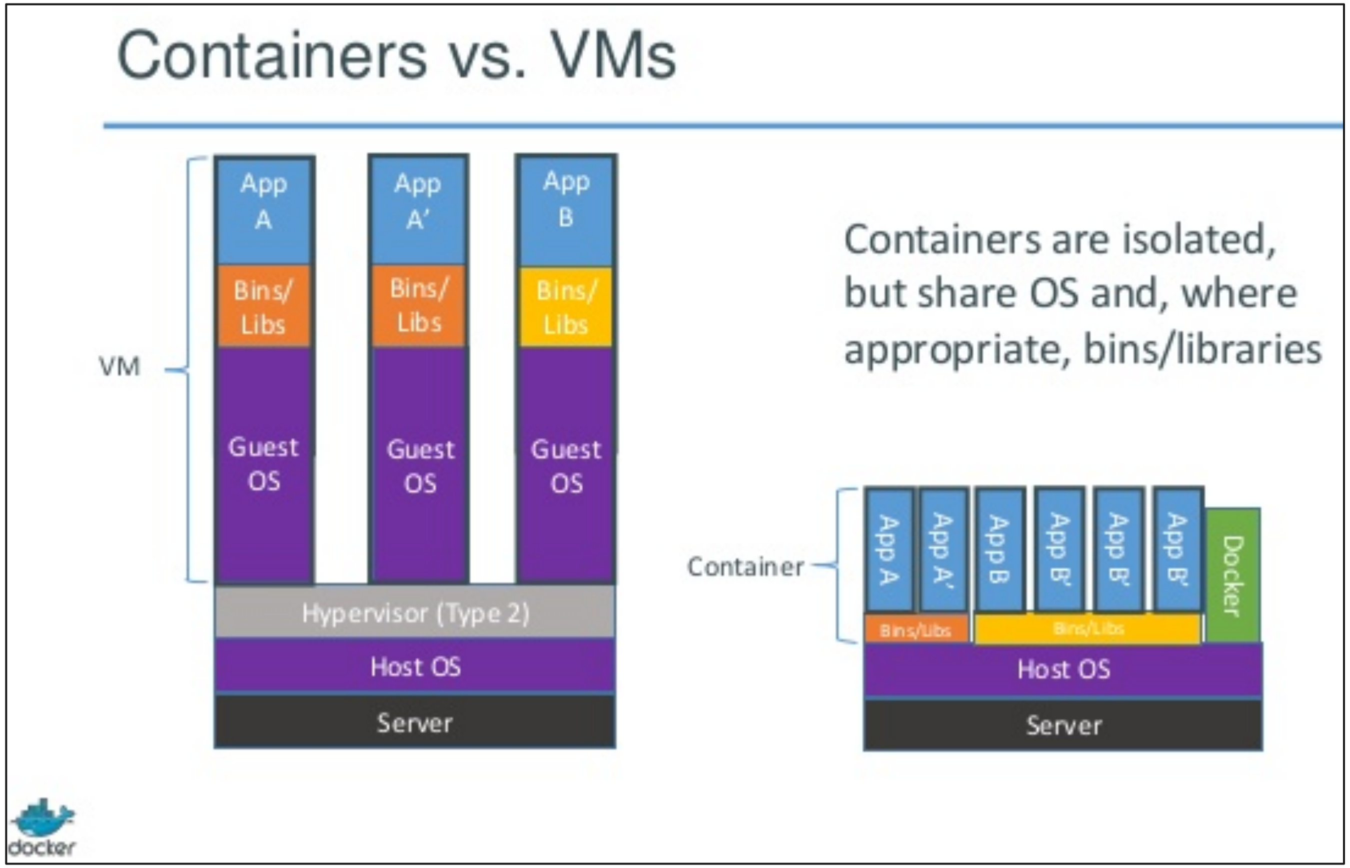
Easy to create repeatable environments
Fits well into the automation workflow

Using Docker

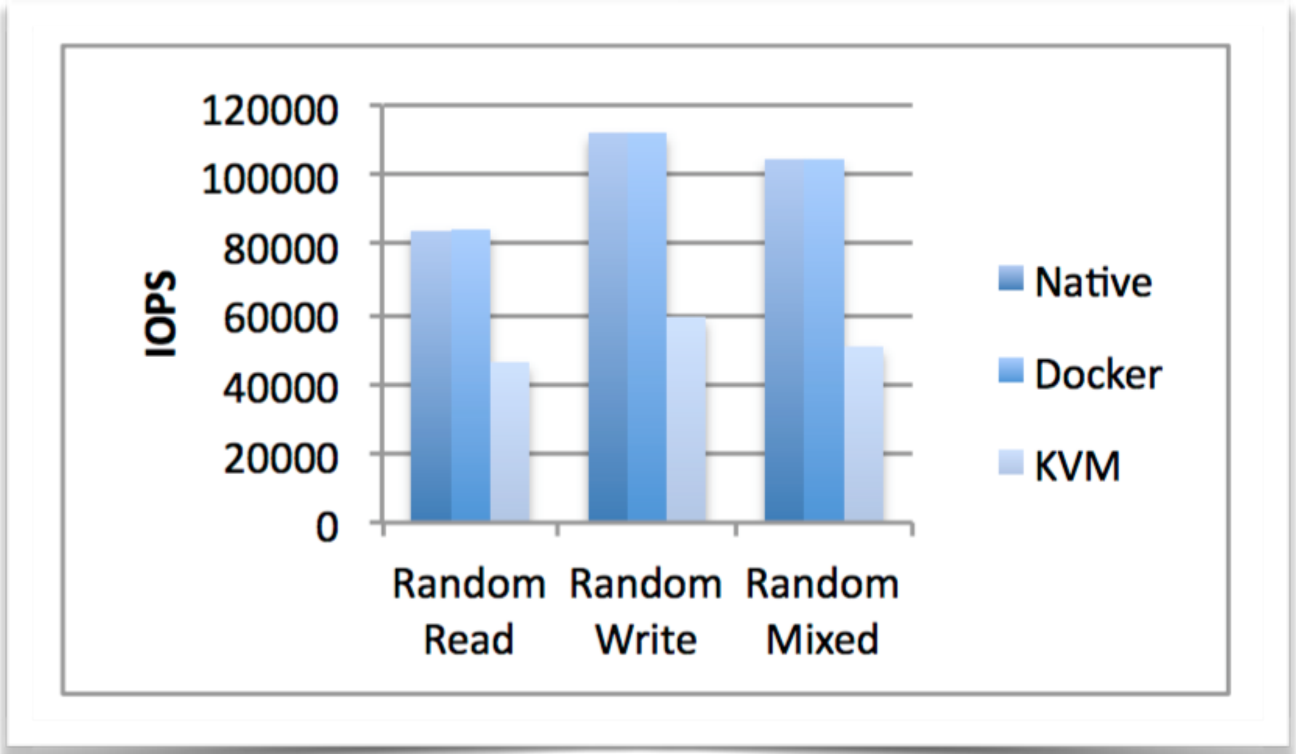
```
$ docker run -it ubuntu /bin/bash
$ docker create -t -i fedora bash
  6d8af538ec541dd581ebc2a24153a28329acb5268abe5ef868c1f1a261221752

$ docker start -a -i 6d8af538ec5
bash-4.2#

$ docker stop <container>
$ docker pause <container>
$ docker restart <container>
$ docker rm <container>

$ docker run -v /host/directory:/container/directory -it ubuntu /bin/bash
$ docker run -v /Users/ejoheke/:/my-host -it ubuntu /bin/bash

$ docker ps
$ docker ps --all
$ docker images
$ docker rmi $(docker images -q)
$ docker stop $(docker ps -q)
$ docker rm $(docker ps --all -q)
```

Docker vs VMs

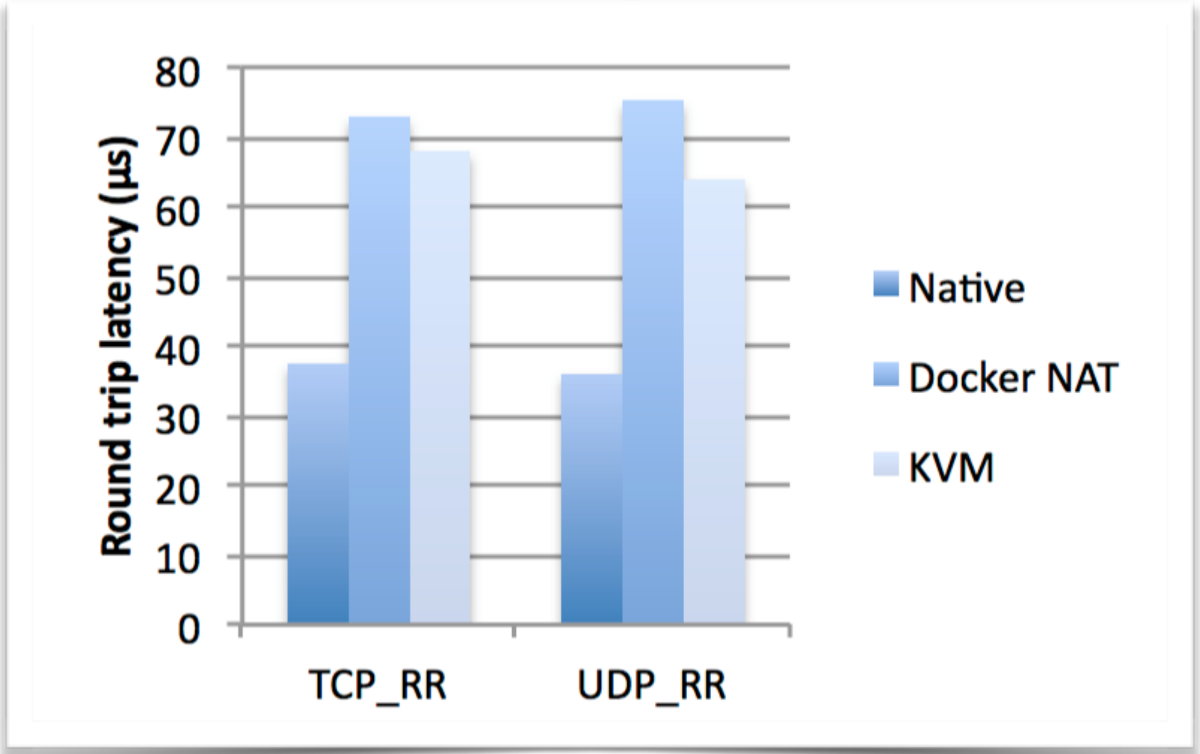
- Virtual machines have their own complete guest OS.
  - Separate kernels. Takes time to boot.
  - A small application we want to run quickly adds up to much data.
  - Consumes host resources
  - Thorough isolation
- Docker
  - Shares kernel with host OS.
  - Runs as a process inside the host.
  - Only applications and its dependencies.
  - Efficiency, better reuse of host OS resources
  - Docker contains OS, but runs natively
  - Less isolation



Performance



Storage



Networking

IBM Research, An Updated Performance Comparison of Virtual Machines and Linux Containers, 2014-07-21, http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/ $File/rc25482.pdf

Containers empowering microservices

Quicker start times simplified both prototyping and auto-scaling

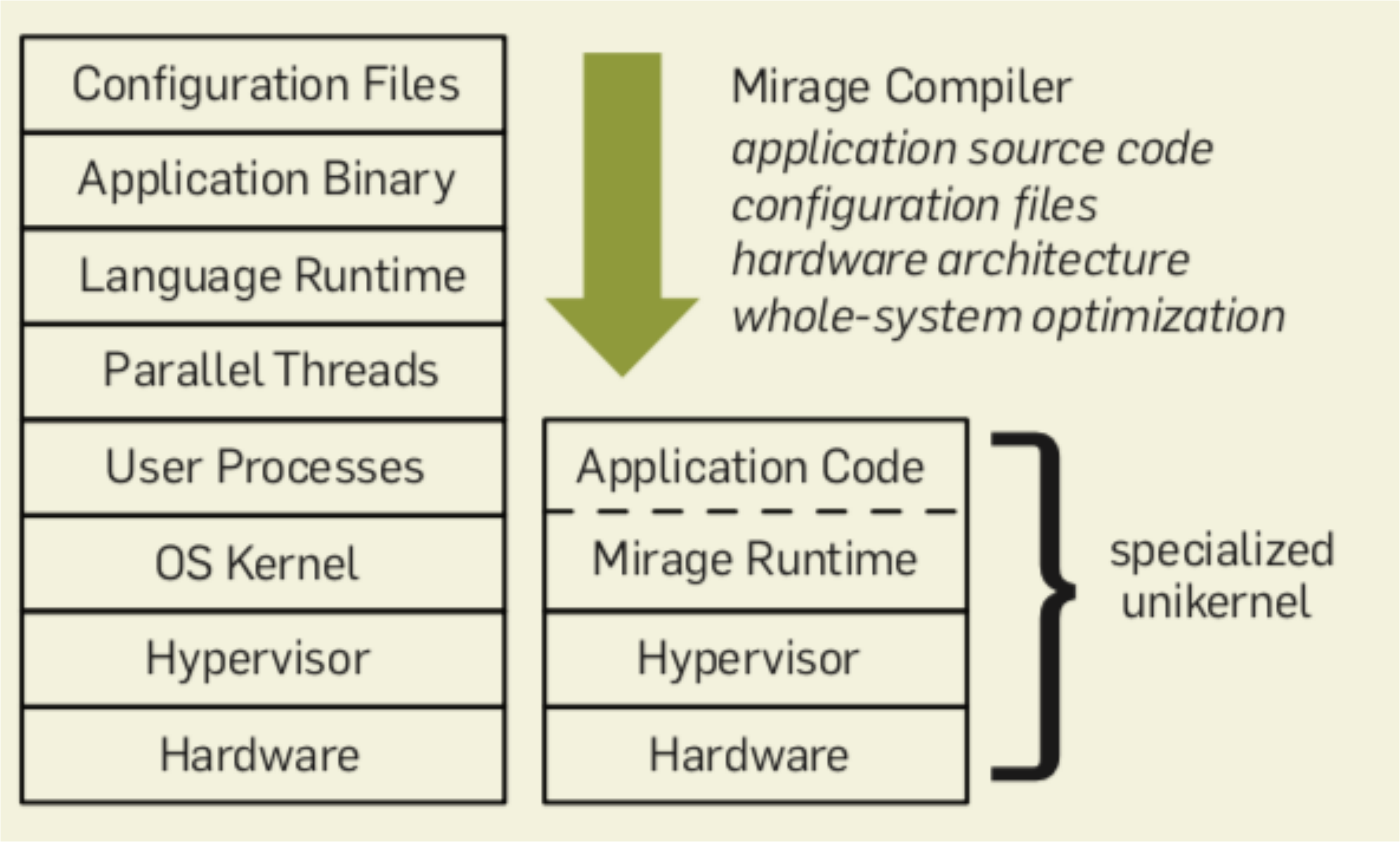
Allow work to be done independently on modules and facilitates independent releases for components

Isolated and abstracted runtime environments, that can be tailored for each module

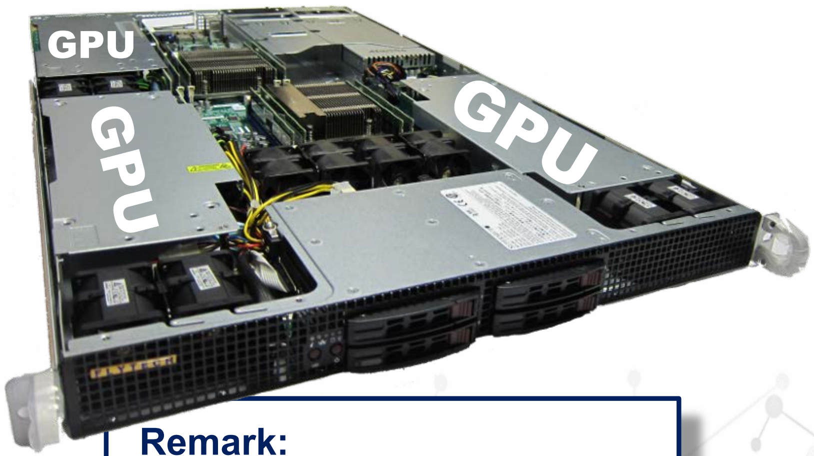
Shared runtime environment, for heterogenous applications

Unikernels

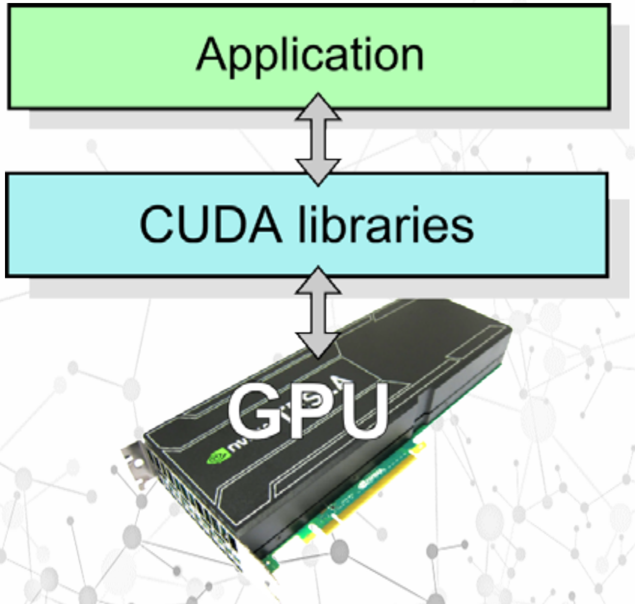
The goal of mirageoS is to restructure entire Vms—including
all kernel and user-space code— into more modular components that areflexible,secure, and reusable in
the style of a library operatingsystem.



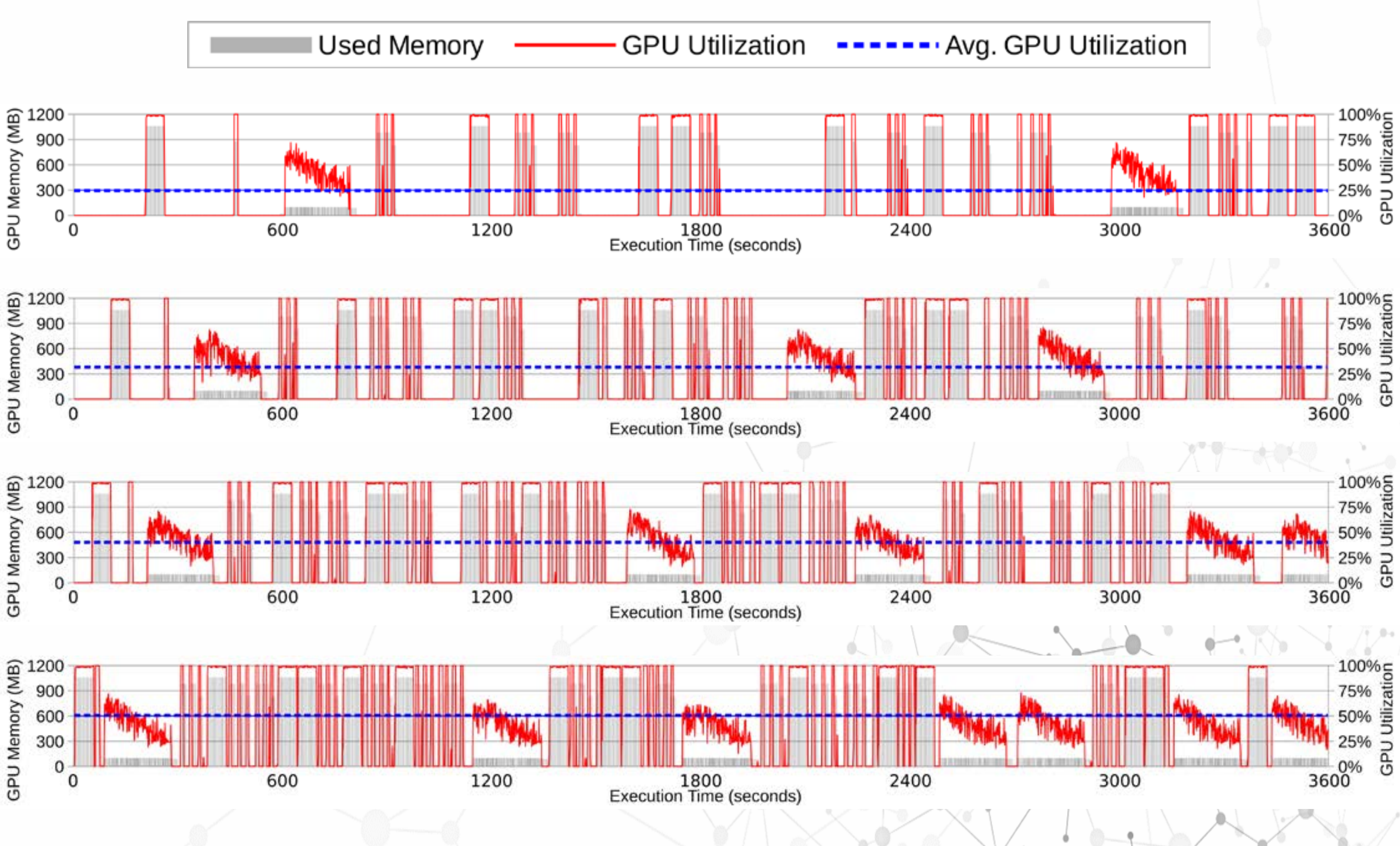
GPU virtualisation



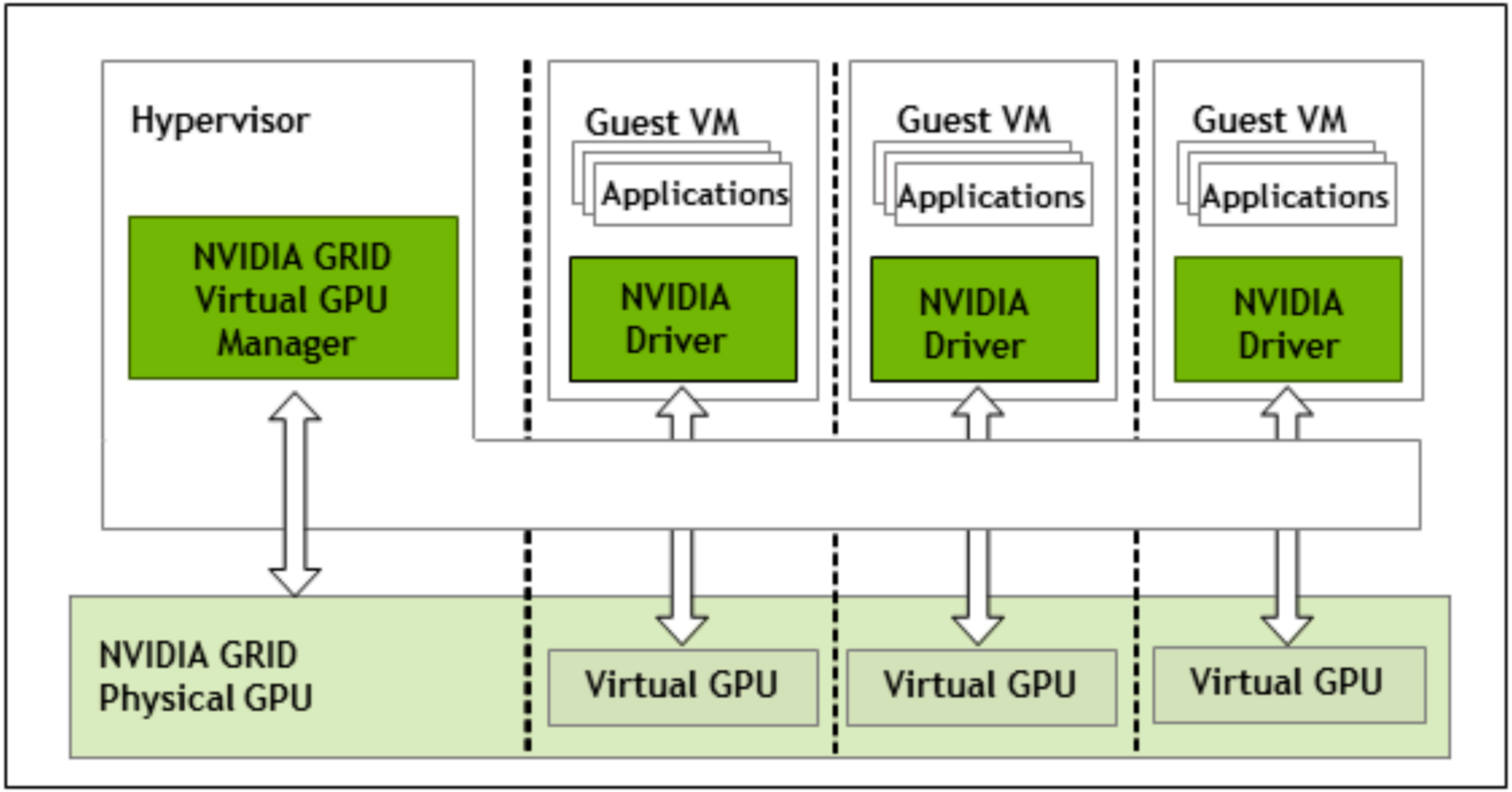
Remark:
GPUs can only be used within the node they are attached to



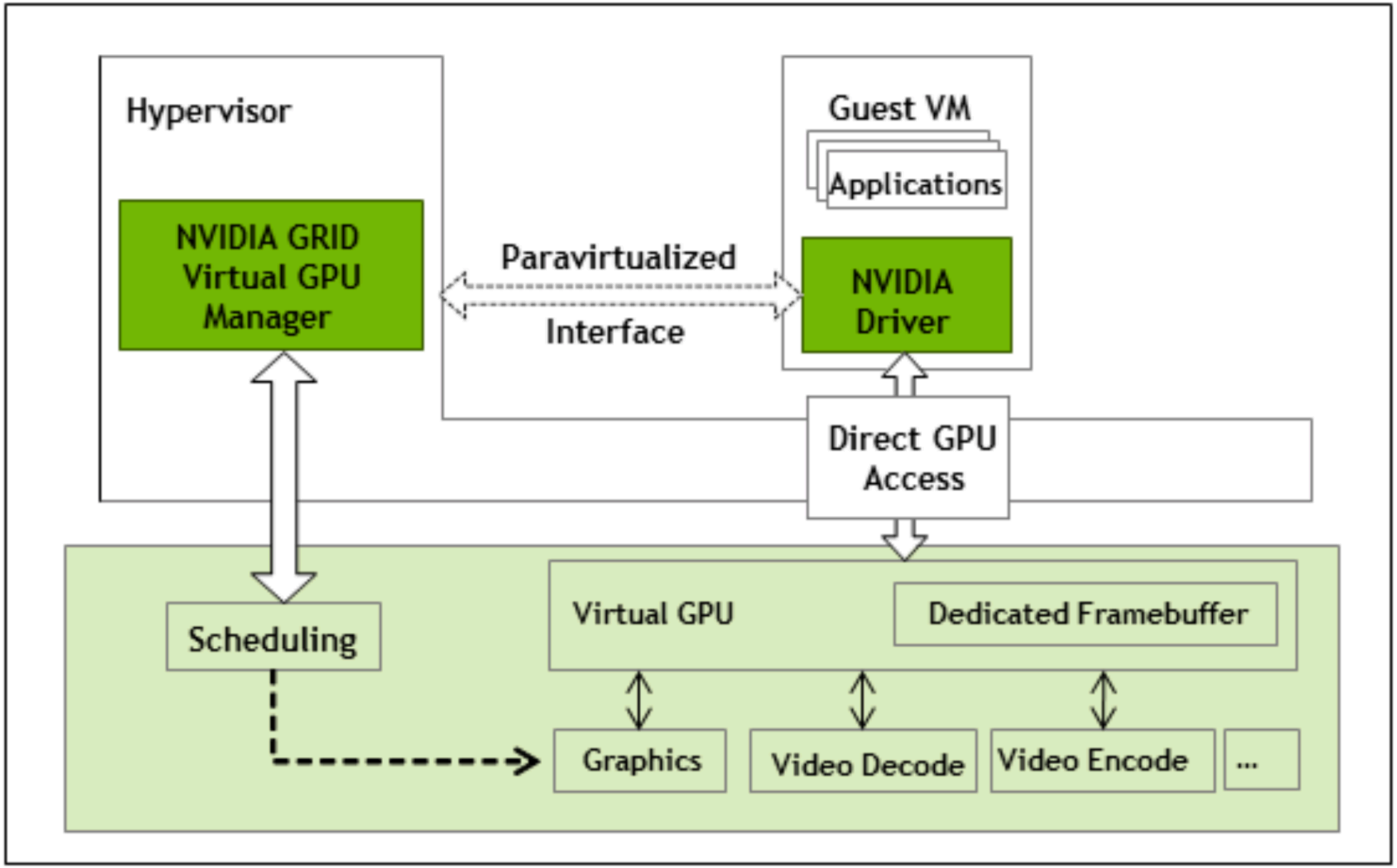
GPU utilisation often becomes an issue



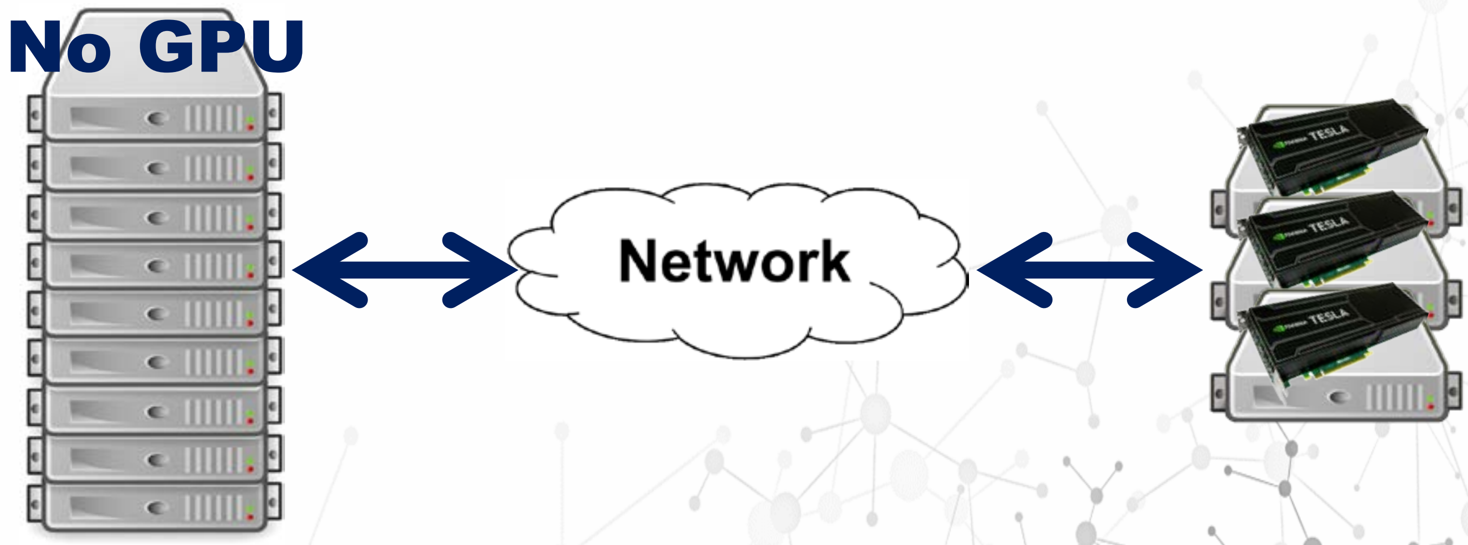
GPU virtualisation



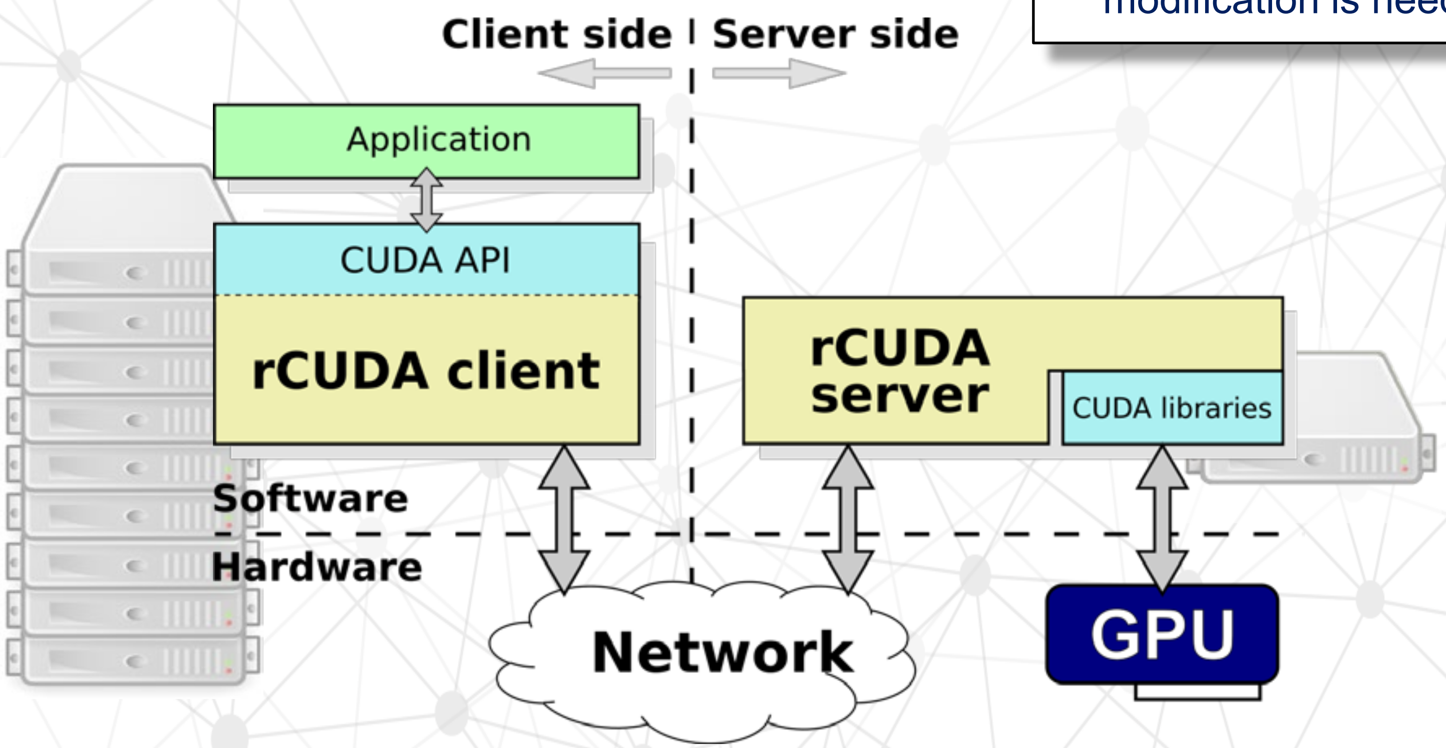
GPU virtualisation



Remote GPU virtualisation



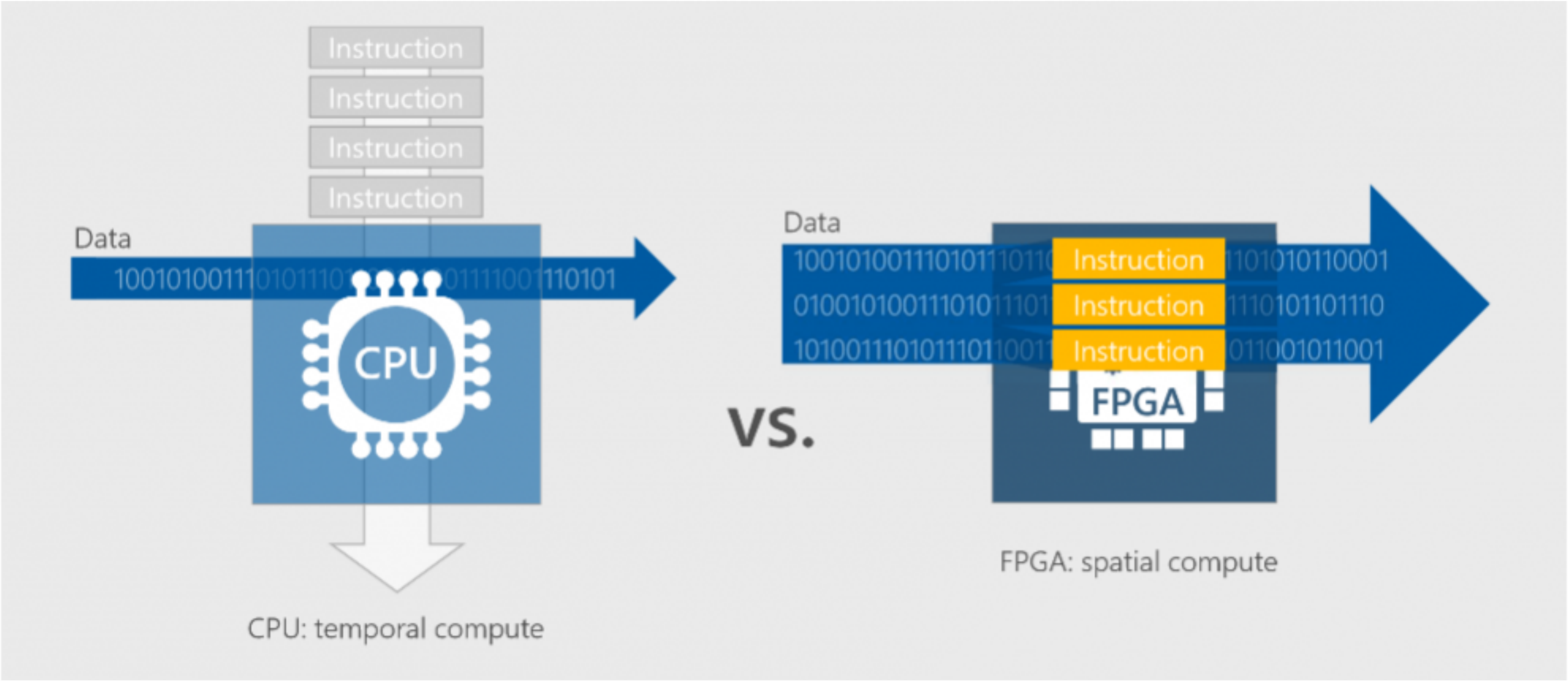
Access to remote GPU is transparent to applications: no source code modification is needed



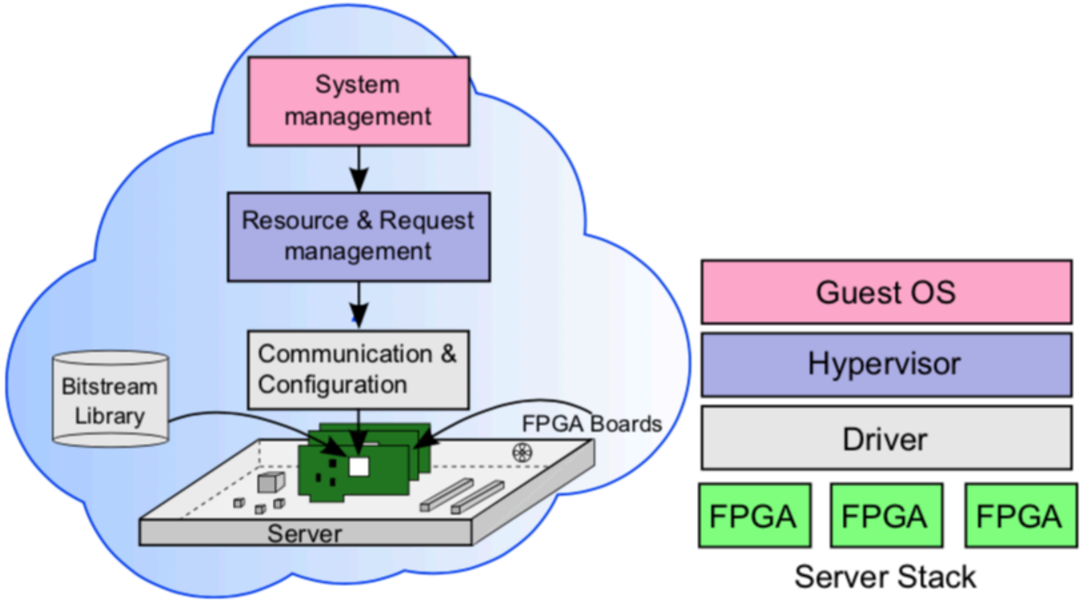
rCUDA is a development by Universitat Politècnica de València

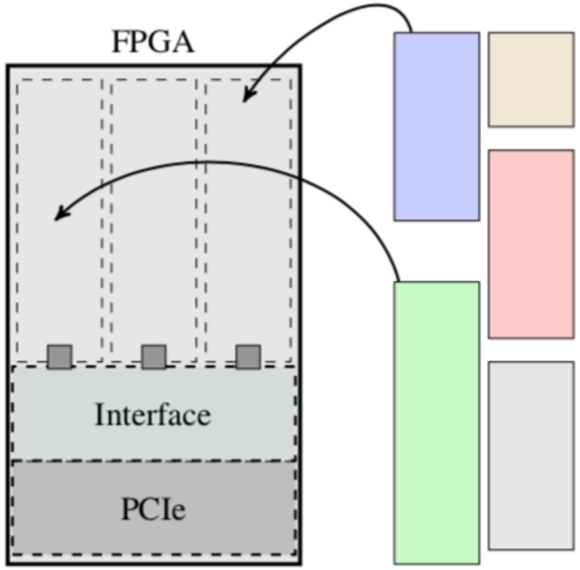
rCUDA remote CUDA

FPGA virtualisation



FPGA virtualisation

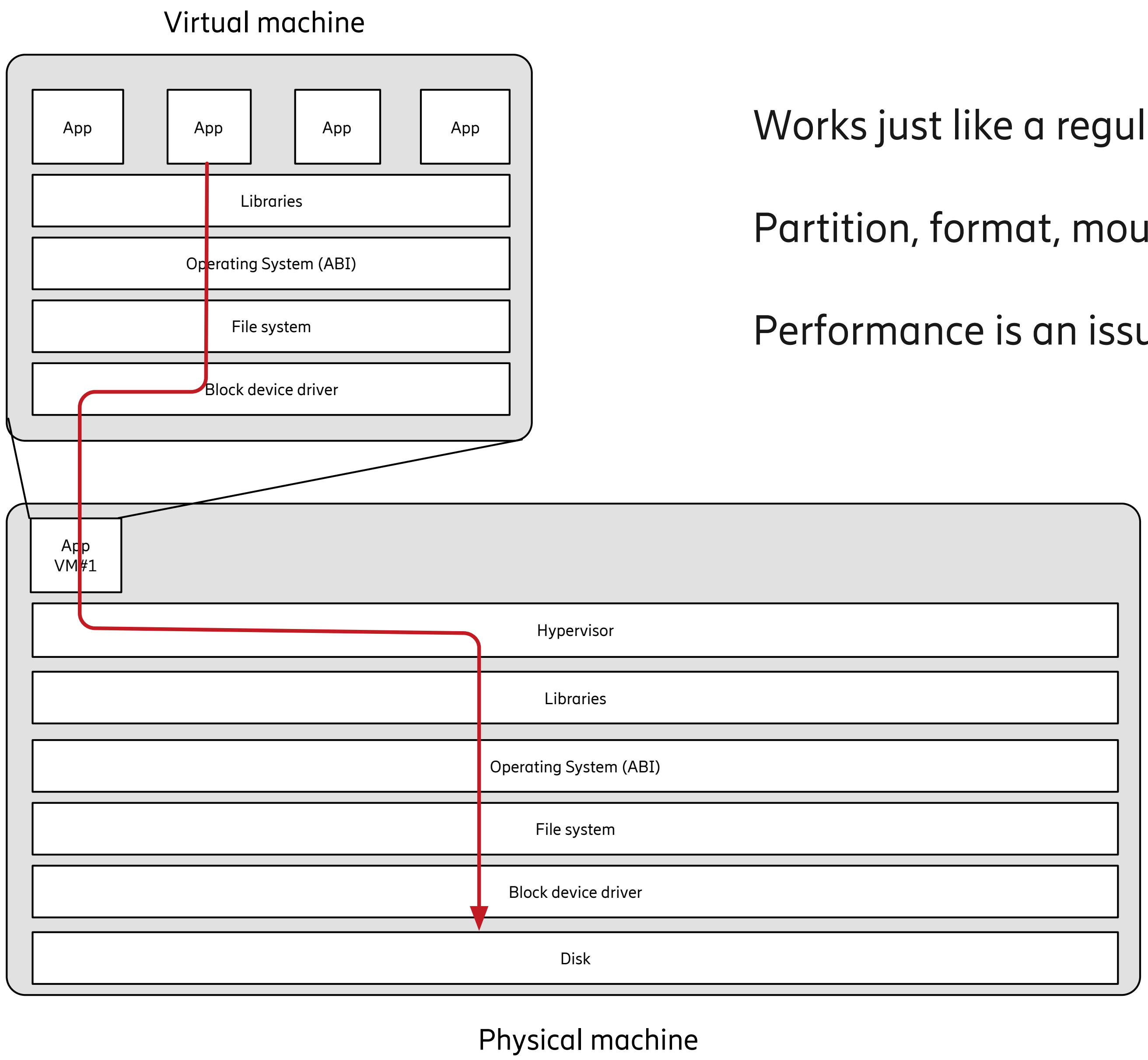




Fahmy et al., "Virtualized FPGA Accelerators for Efficient Cloud Computing", IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom), 2015

Storage virtualisation

Block storage (virtual hard disk)



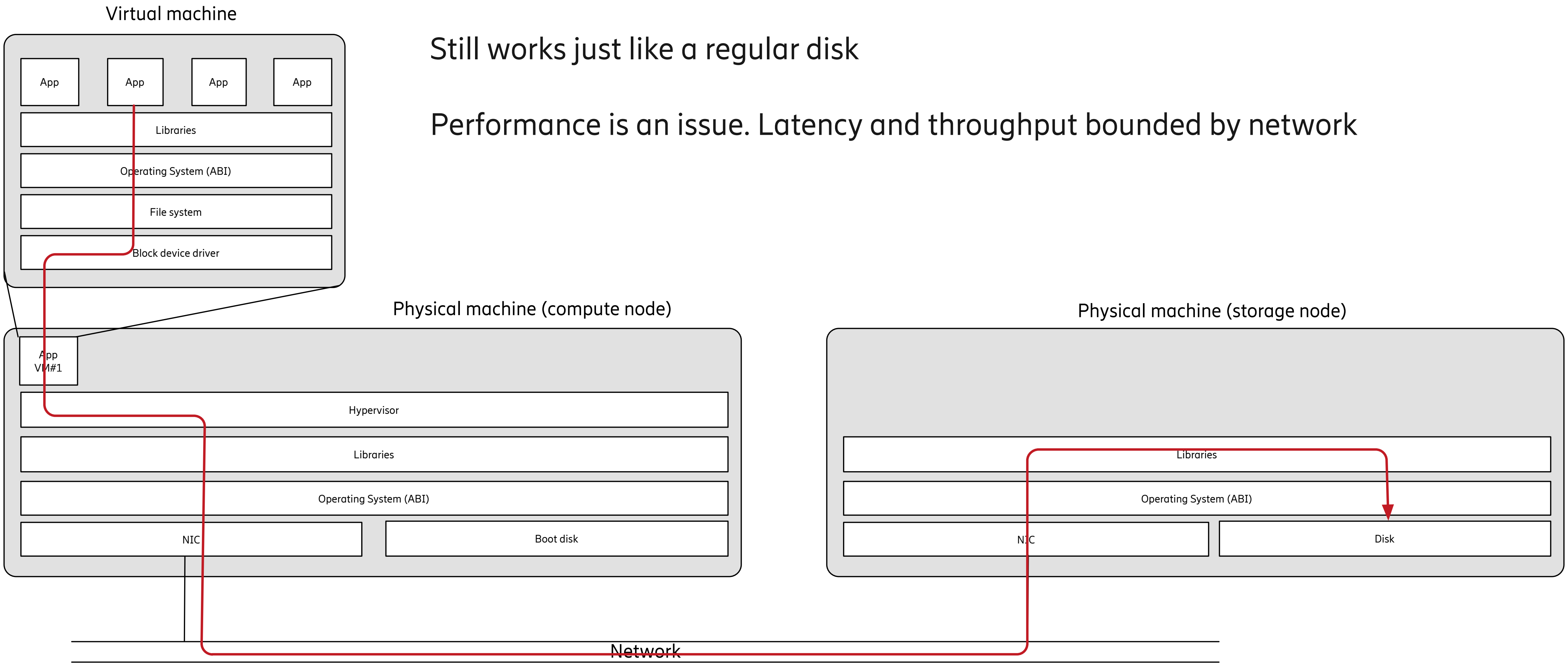
Works just like a regular disk.

Partition, format, mount

Performance is an issue

Storage virtualisation

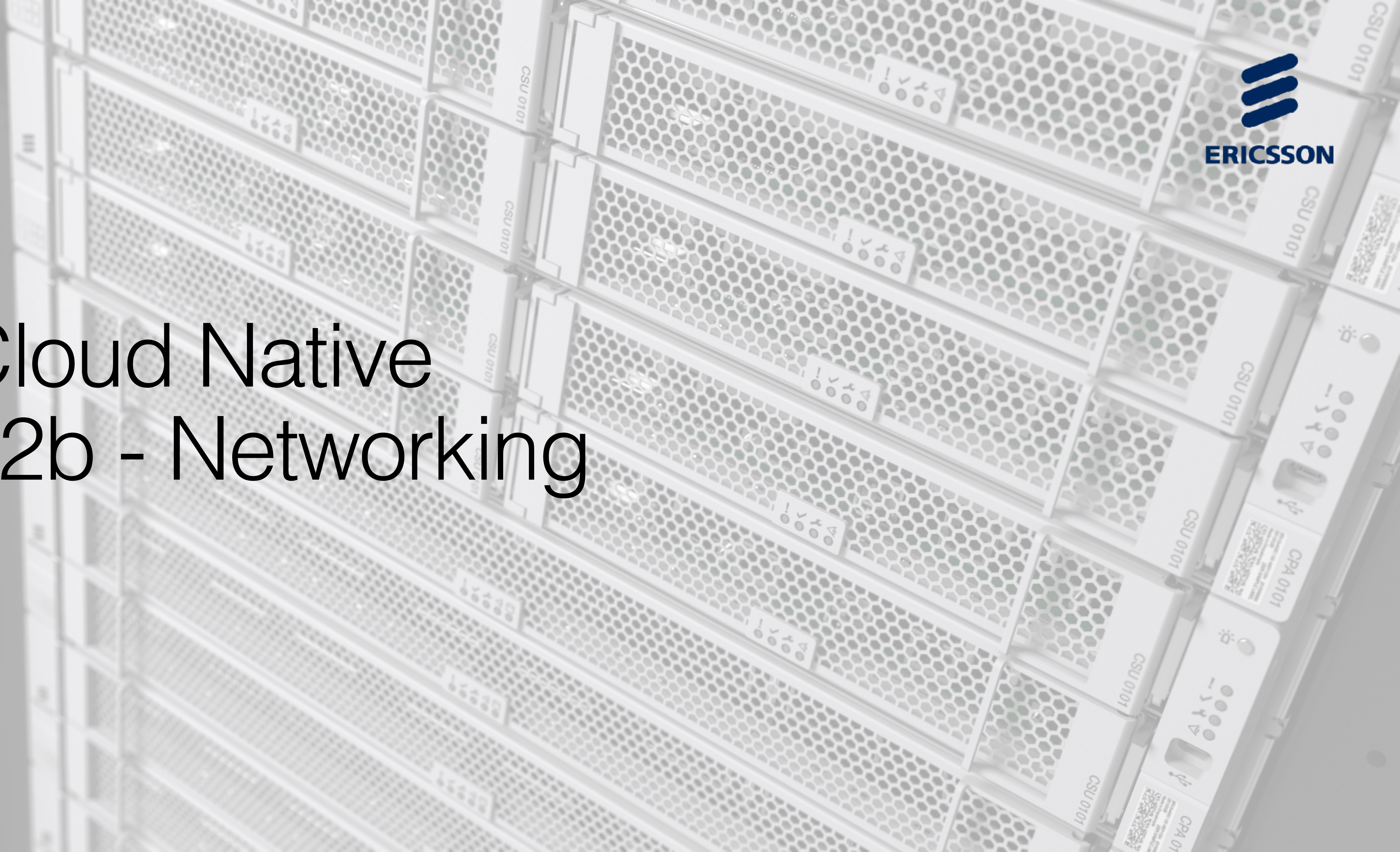
Remote block storage (virtual hard disk)



Still works just like a regular disk

Performance is an issue. Latency and throughput bounded by network

Cloud Native
#2b - Networking



Networking 101

The stack

7	Application Layer
6	Presentation Layer
5	Session Layer
4	Transport Layer
3	Network Layer
2	Data Link Layer
1	Physical Layer

OSI Model

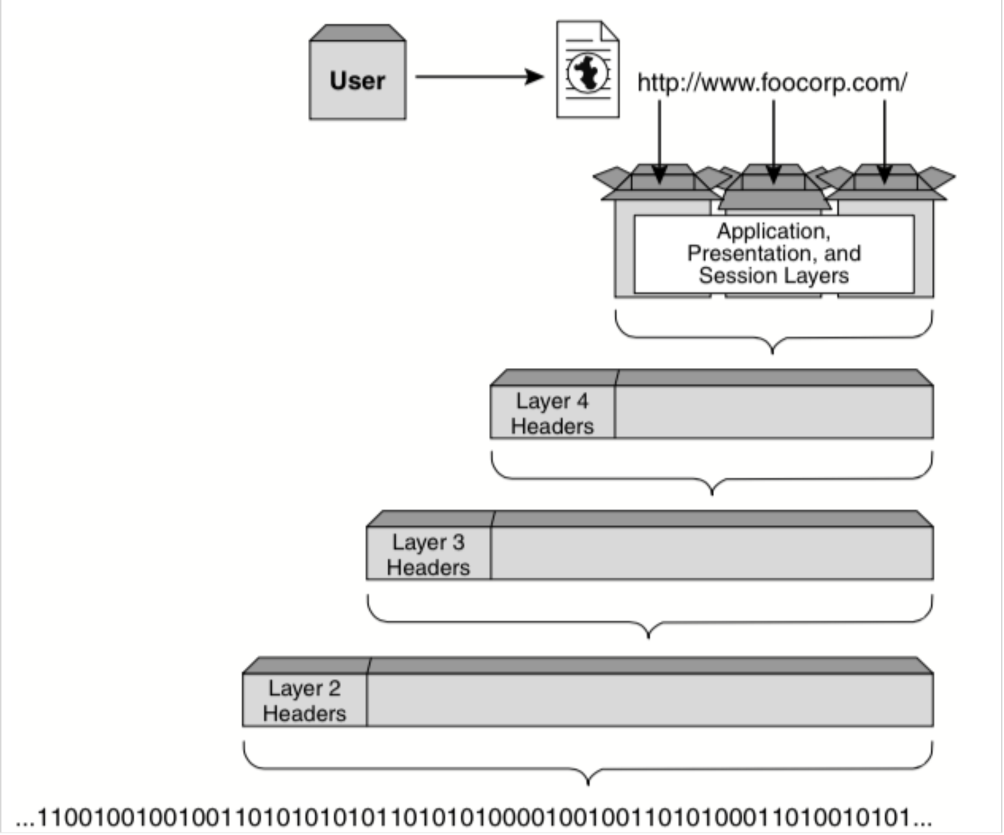
HTTP, FTP

TCP, UDP

IP

Ethernet, token ring

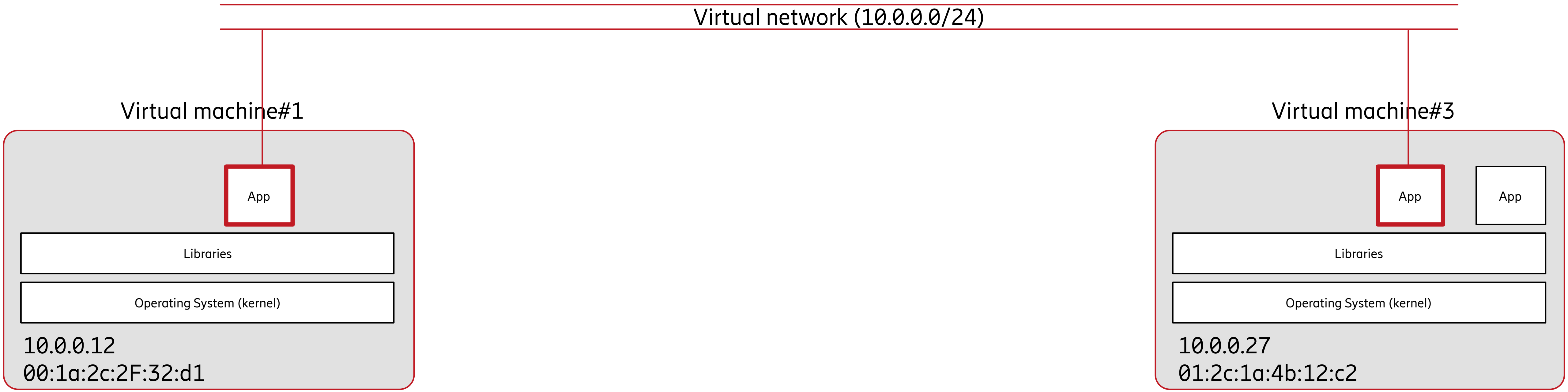
1000BaseT



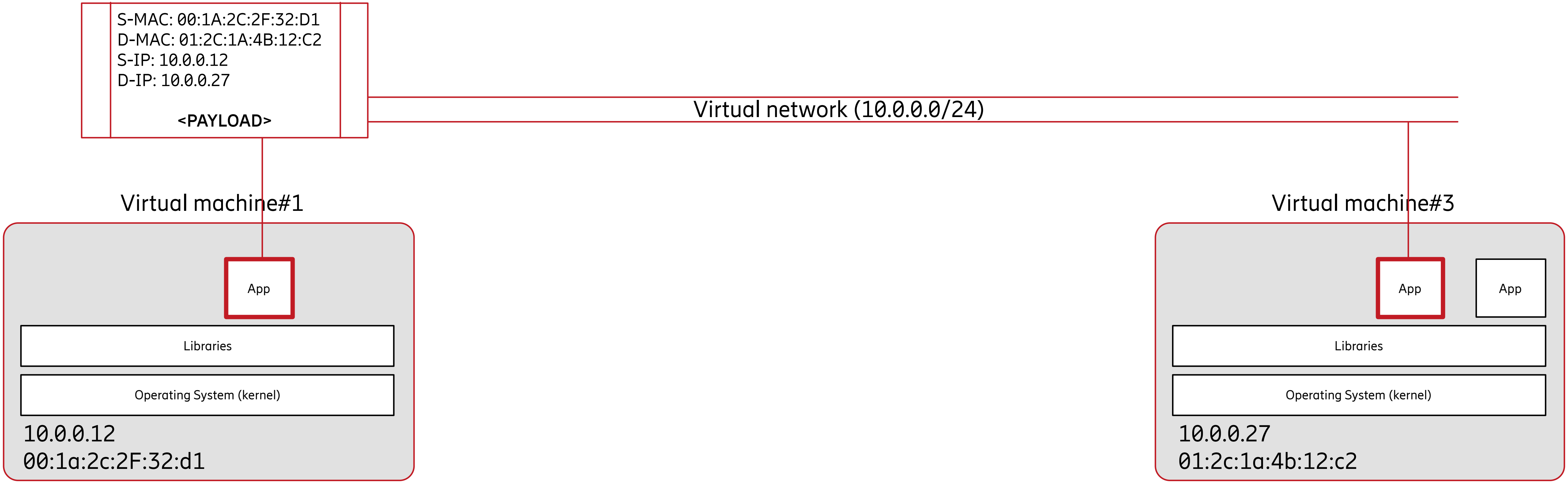
True definition of a layer n protocol:
Anything designed by a committee whose charter is to design a layer n protocol

Source: Optimizing Network Performance with Content Switching: Server, Firewall, and By Matthew Syn

Network virtualisation



Network virtualisation

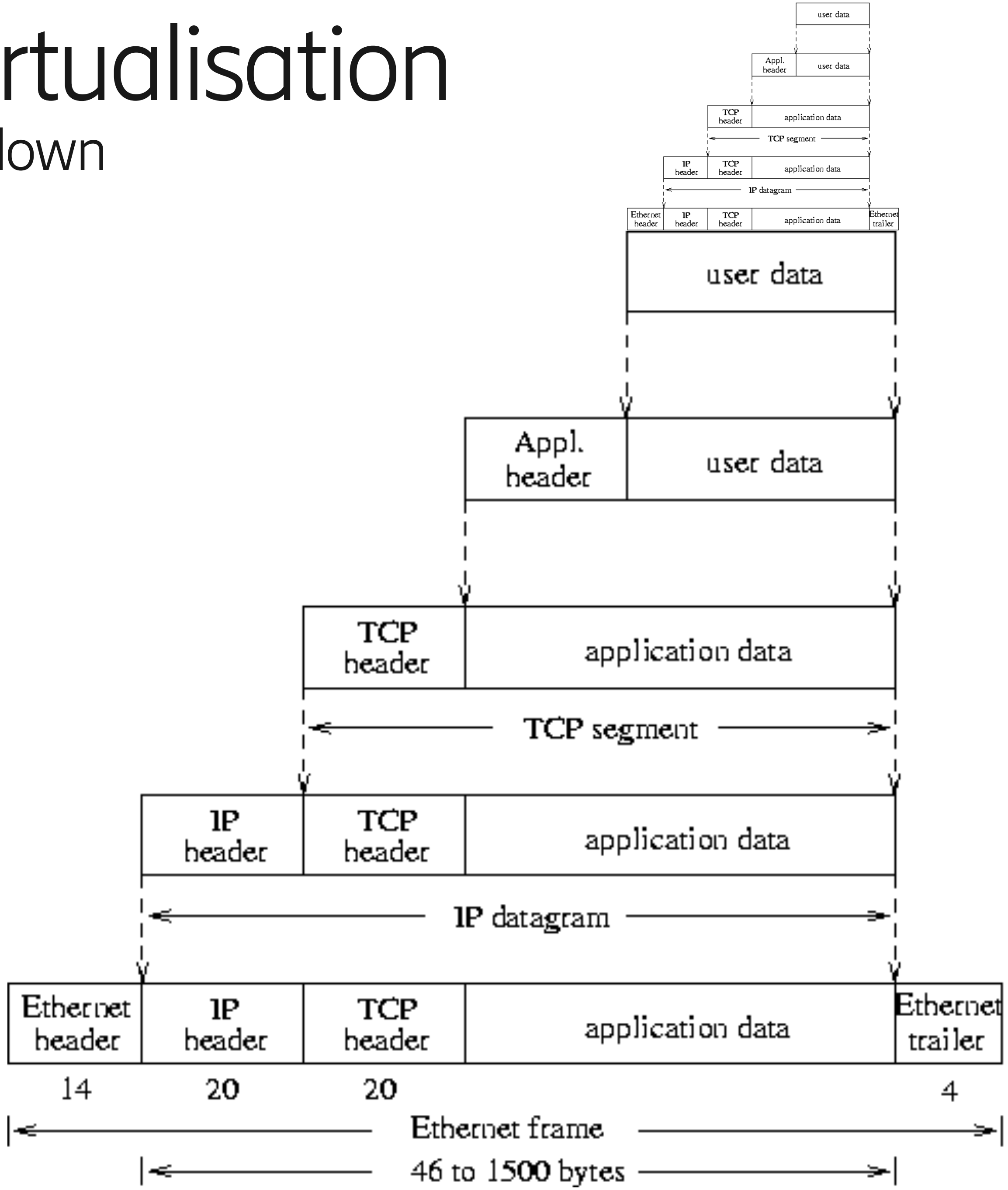


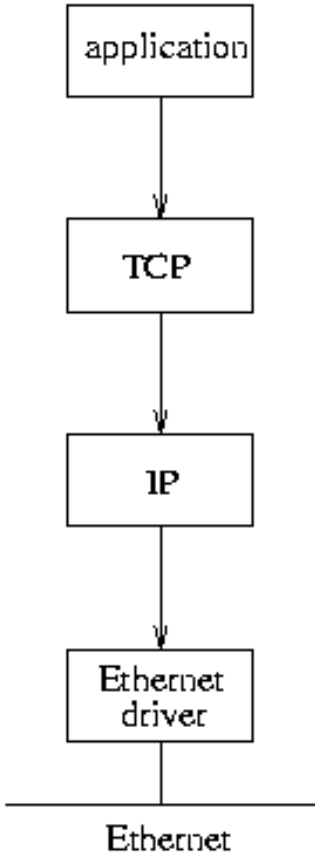
Network virtualisation

Turtles all the way down

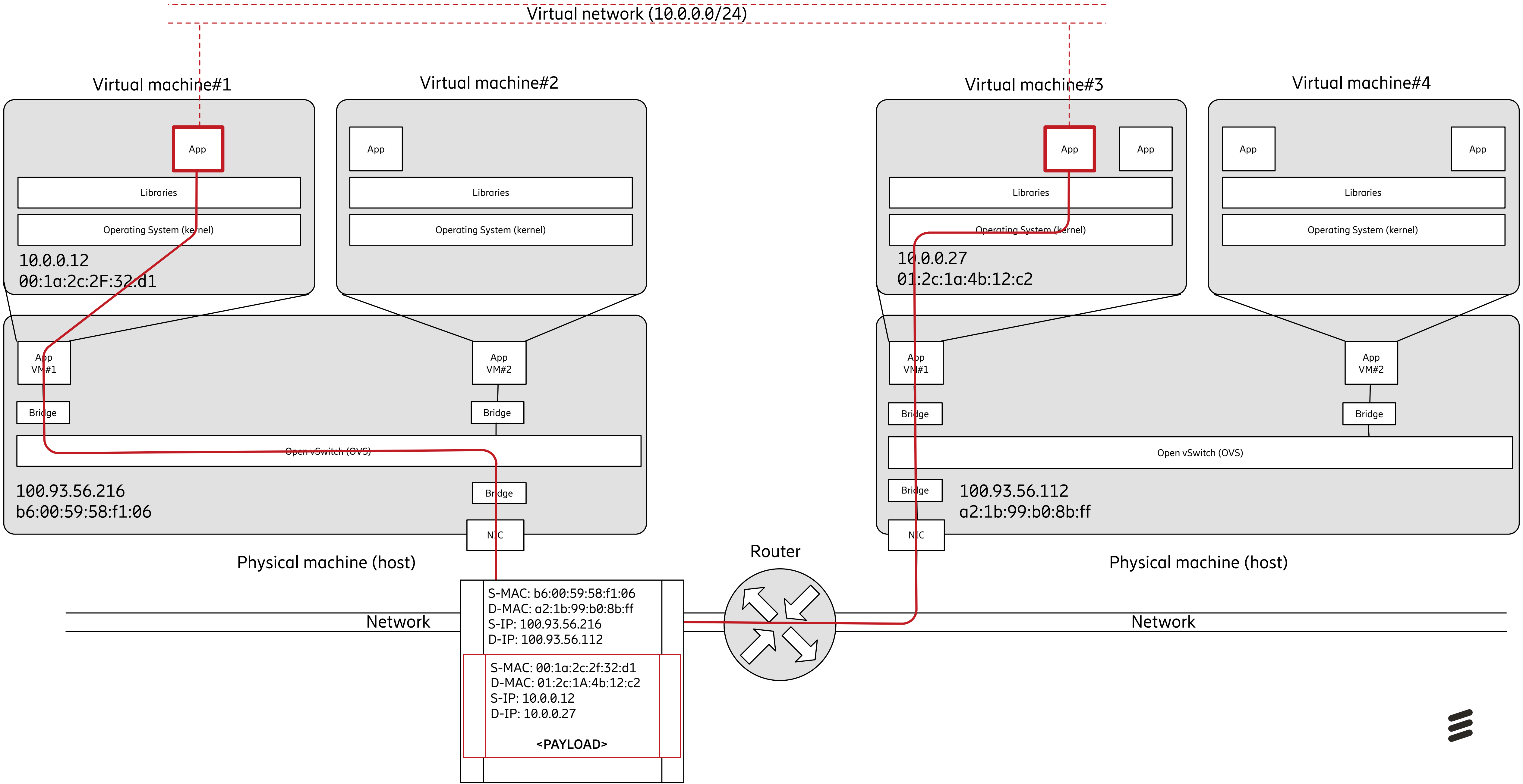
Guest

Host





Network virtualisation

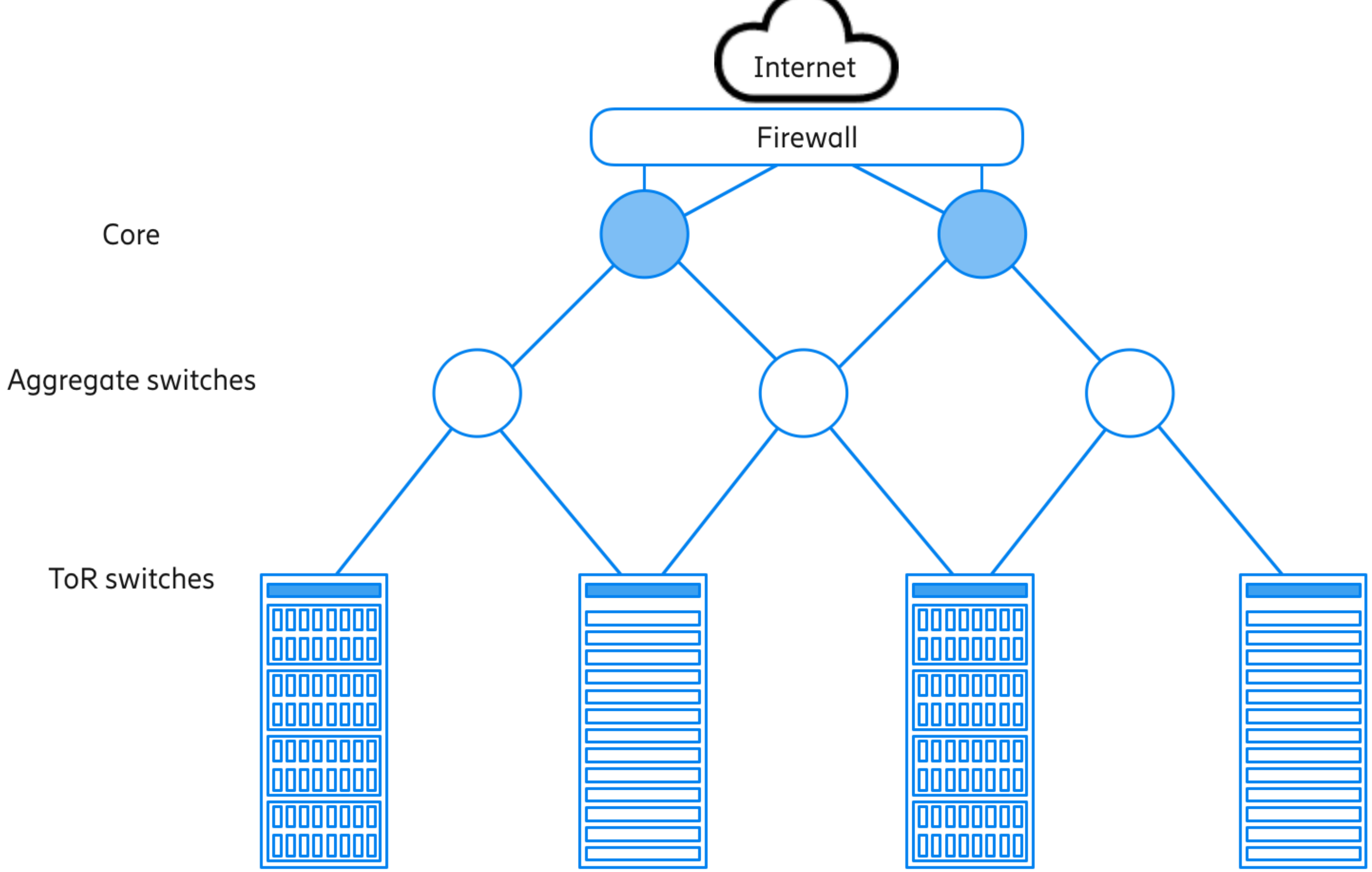


Tunneling

- Provides a network service that the underlying network cannot provide.
  - IPv6 over IPv4
  - VPN - Virtual Private Network, provide secure access to a network using non-secure networks. Uses IPSec "encrypt an IP datagram and put it in an IP datagram"
- Usually violates the OSI model, i.e., the layer m payload contains layer n<m protocol data.
- Communication between data centers typically over tunnels.

- VXLAN
  - VLAN on steroids.
  - Addresses scalability problem of layer-2 networks.
  - Allows 2^24 logical networks. Identified by VXLAN Network Identifier (VNI).
  - Encapsulates layer-2 frame in UDP datagram. Layer 2 on top of layer 3!
  - Connect separate layer-2 domains to create one domain.
  - Machines are identified uniquely by the combination of their MAC address and VNI.
  - VXLAN Tunnel End Points (VTEP) encapsulate/decapsulate layer-2 frames.

Cloud Networking



- Dynamics
  - mobility, migration of VMs
  - short lived services
  - on demand scaling
- Scaling
  - many VMs on many hosts
- Isolation
  - tenants sharing the same physical resource
- Traffic
  - North-south/East-west
  - Not always on physical links
- Make DNS a bit more complicated (and important)

The Two Networking "Planes"

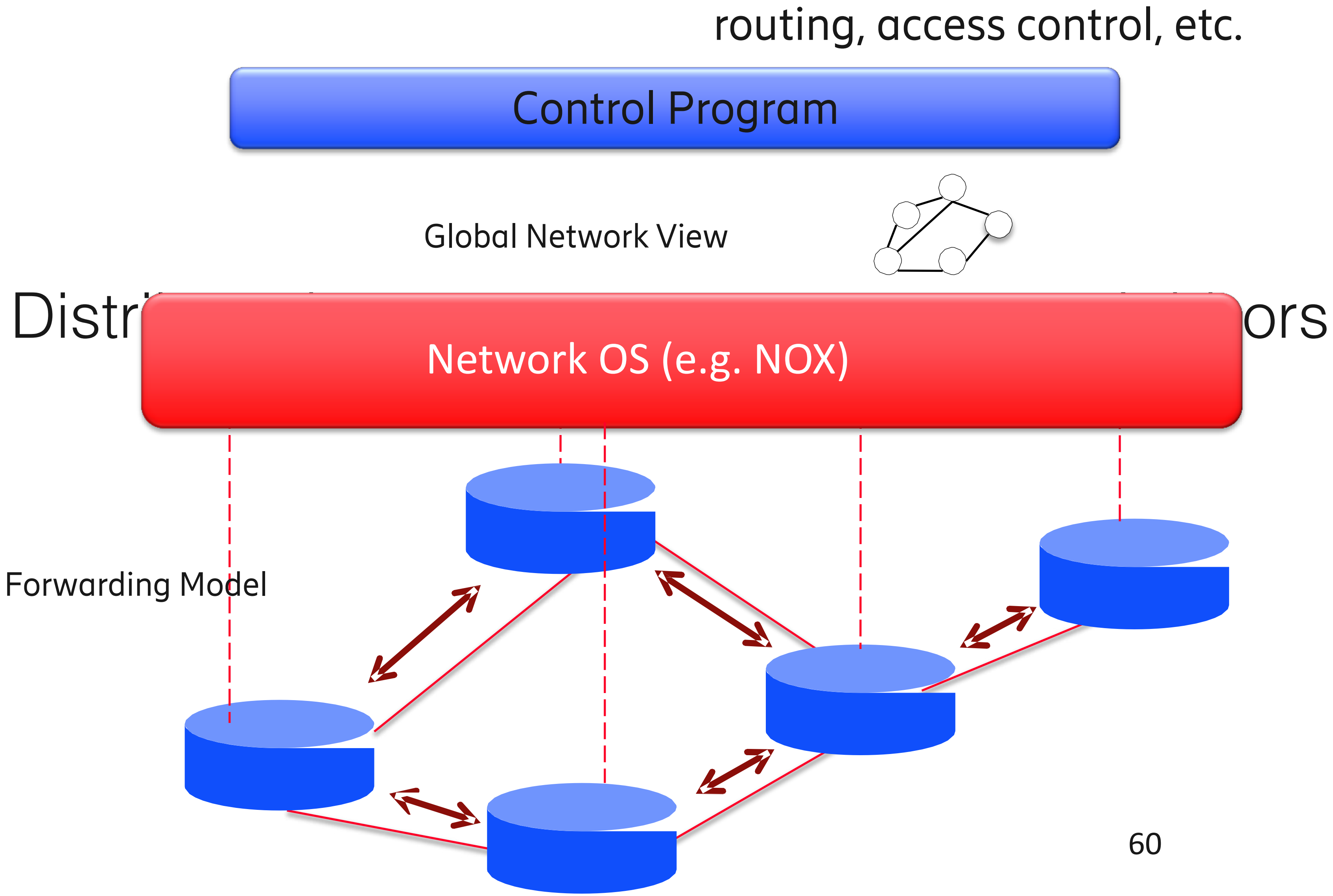
Data plane: processing and delivery of packets with local forwarding state

- Forwarding state + packet header -> forwarding decision

Control plane: compute the forwarding state in switches/routers

- Determines how and where packets are forwarded

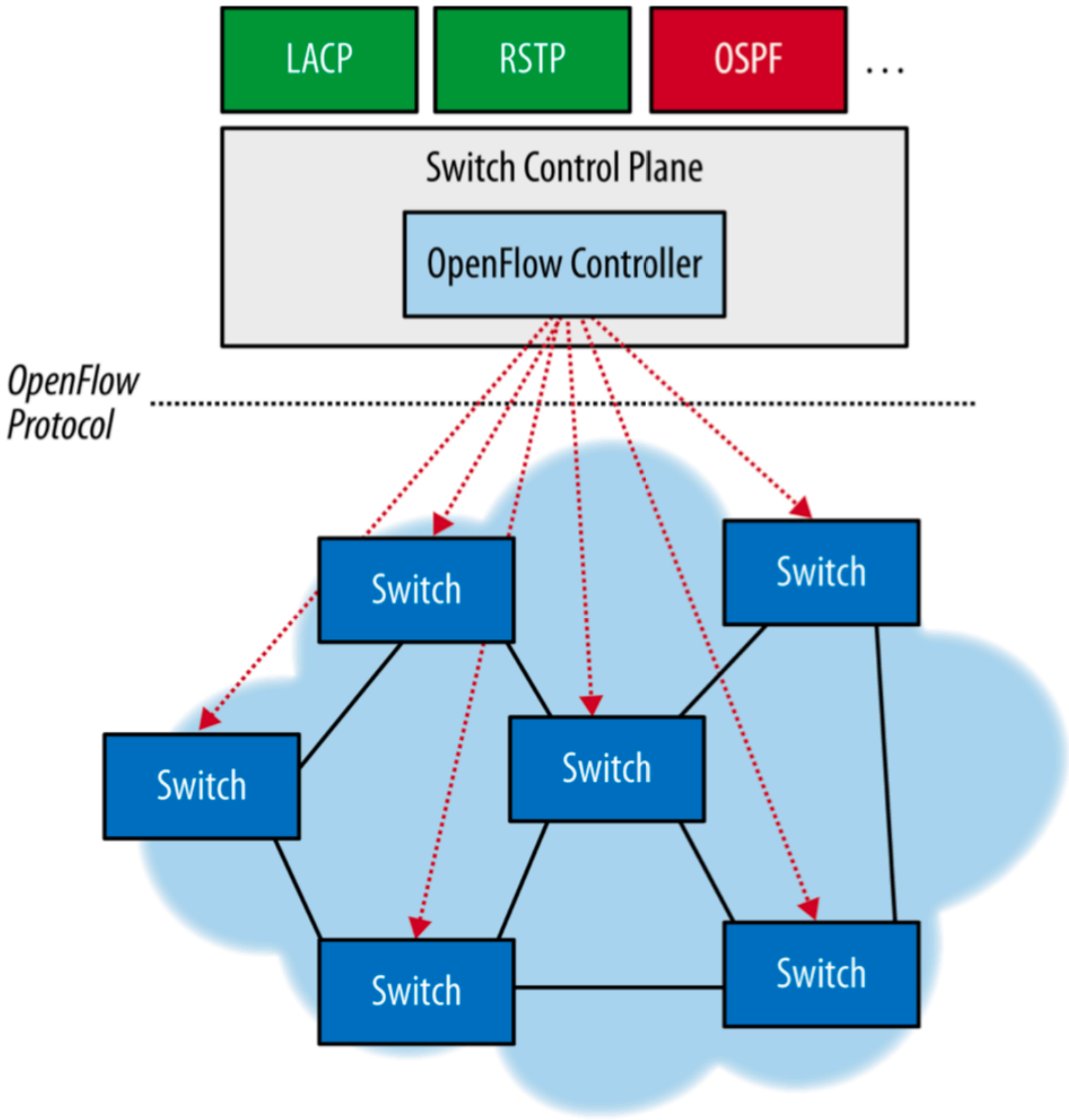
Traditional Switches/Routers



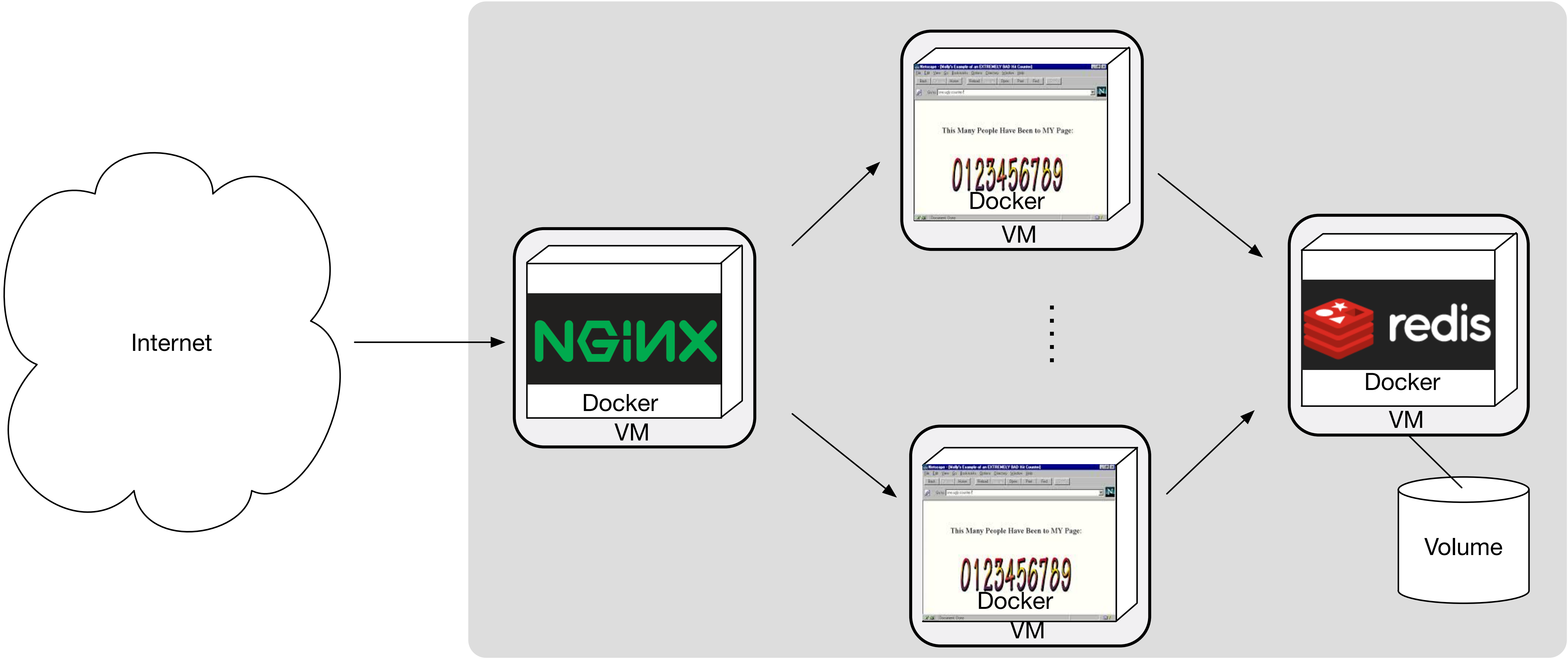
SDN

Software Defined Networking

- Introduces a centralized control plance
- Networks are hard to manage (=>expensive)
- Computation and storage have been virtualized
- Networks are hard to evolve
- Simplify the hardware nodes



Assignment #2



Bonus assignment on SDN for the brave…

fin