Virtualization and Virtual Machines

CS522 – Principles of Computer Systems

Dr. Edouard Bugnion

Virtualization and Virtual Machines

This week

- Introduction, definitions,
- A short history of virtualization
- On VMM construction
- Case study – Resource management in virtualized systems
- Case study – VMM construction for non-virtualizeable architectures

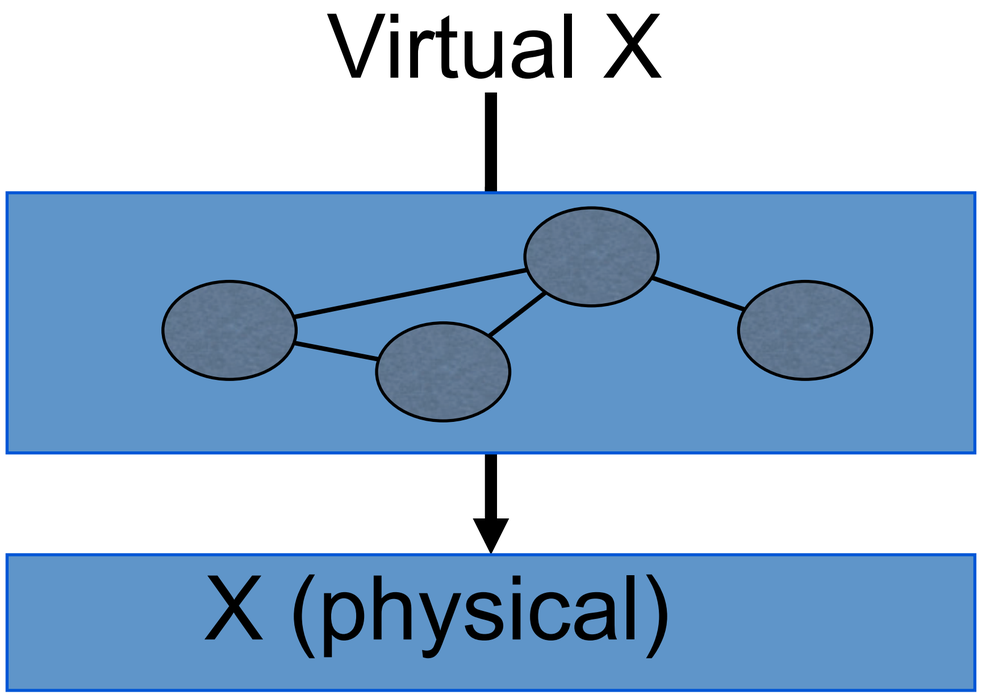
Virtualization

- Virtualization – application of fundamental principles of computer systems
  - Abstraction & Modularity
  - Layering & Indirection
  - Interpretation
- Motivation–
  - reduce complexity in computer systems
  - Enable automation
- Virtual machines are only one example of virtualization

Definition

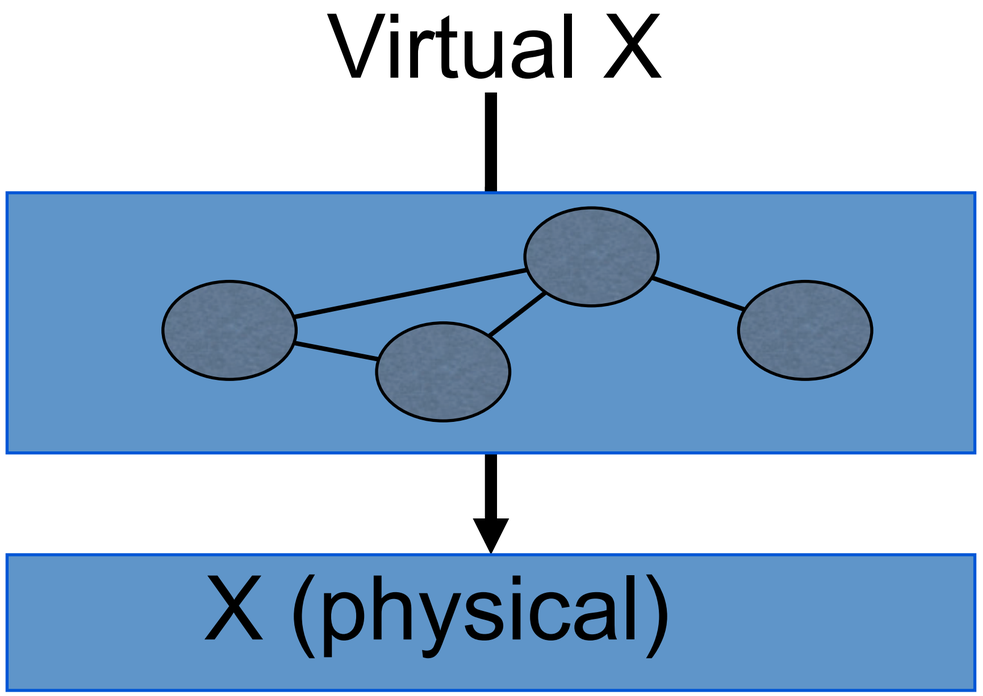
- **Layer** that exports the same **abstraction** as the layer that it relies upon

Definition



- **Layer** that exports the same **abstraction** as the layer that it relies upon
  - Hides the physical names from the abstraction (isolation)
- Virtualization enforces **modularity**
- Relies on combination of:
  - multiplexing
  - aggregation
  - emulation

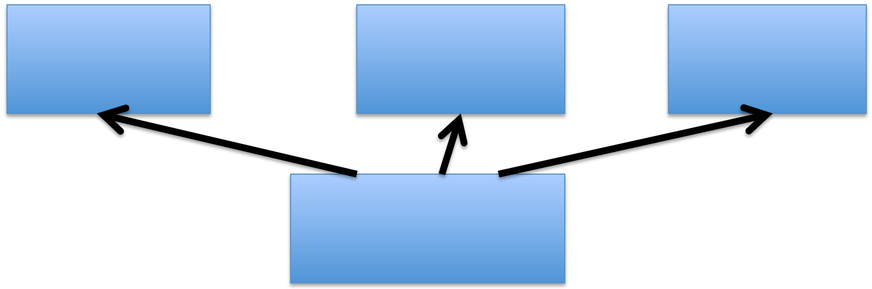
Virtualization within an OS [Salzer/Kasshoek ch 5.1]



- Threads
  - Virtual CPUs
- Virtual memory
  - Most significant innovation in computing since von Neumann architecture
- Sockets, pipes
  - Virtual links, ...
- Virtual disks
  - Volumes, RAID, ...

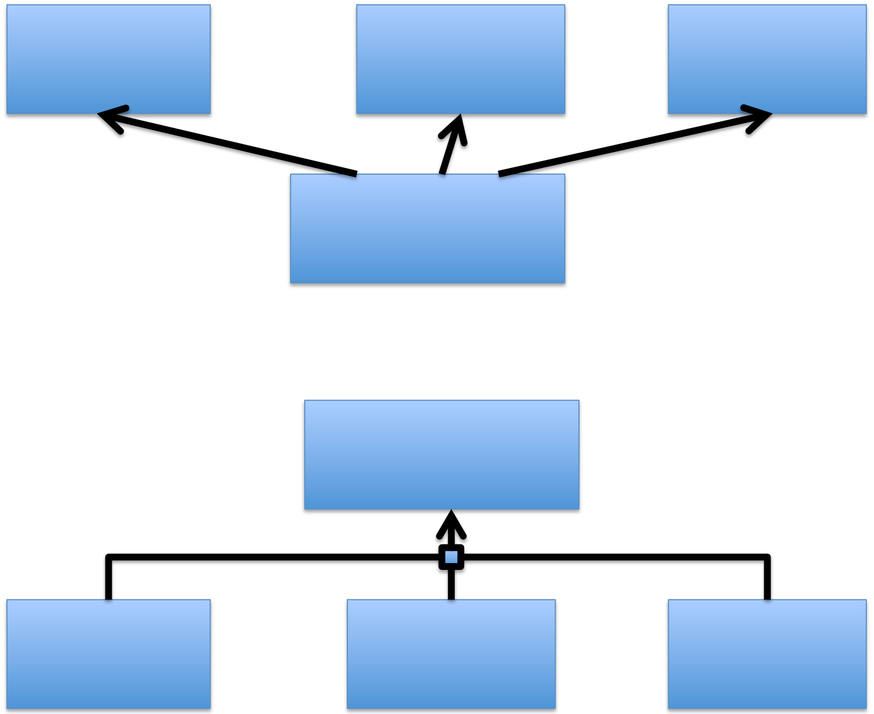
3 Virtualization Mechanisms

- Multiplexing
  - Expose one resource multiple times
  - Isolation through indirection
  - Often with hardware support
  - E.g. Virtual memory.



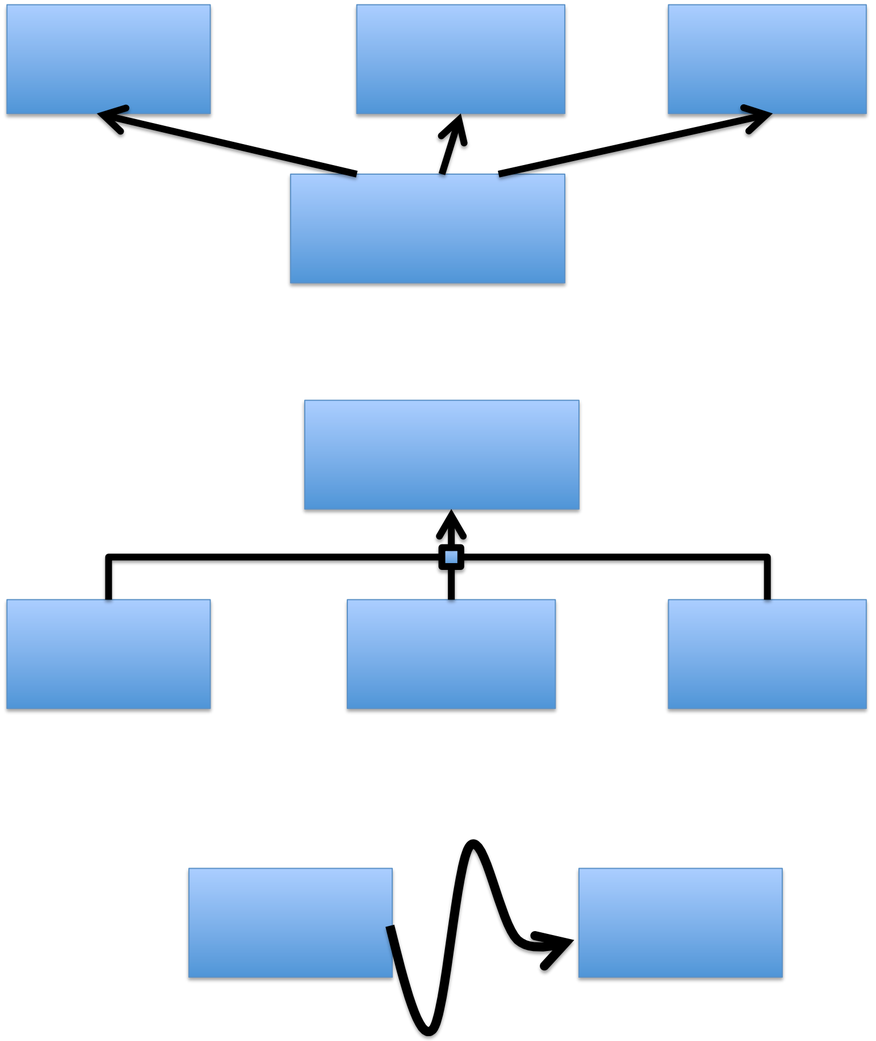
3 Virtualization Mechanisms

- Multiplexing
  - Expose one resource multiple times
  - Isolation through indirection
  - Often with hardware support
  - E.g. Virtual memory.
- Aggregation
  - Expose multiple resources as a single resource
  - Often with enhanced capabilities (availability, performance)
  - E.g. RAID, link aggregation, …



3 Virtualization Mechanisms

- Multiplexing
  - Expose one resource multiple times
  - Isolation through indirection
  - Often with hardware support
  - E.g. Virtual memory.
- Aggregation
  - Expose multiple resources as a single resource
  - Often with enhanced capabilities (availability, performance)
  - E.g. RAID, link aggregation, …
- Emulation
  - Use software to emulate a virtual resource different than the physical resource
  - E.g. RAM disk, virtual tape, byte code, …



Architectural support for virtualization

- When the resource is architected (i.e. designed) to enable virtualization in a straightforward manner
  - without having to rely on generalized emulation (big hammer)
  - Attribute of the underlying hardware architecture
- Applies to various elements of computer system
  - processors (Intel VT-x, AMD-v, ARM v7)
  - memory management unit (e.g. TLB, EPT)
  - i/o devices, ...

Closing thought

- Two forms of layering
  - Create a new abstraction
  - Export the same abstraction (=virtualization)
- Many forms of virtualization: of a computer system, a disk, a database, a broadcast domain, an application server, ….
- When should you virtualize (rather than create a new abstraction)?
- What should you virtualize?

Next: Virtualization and Virtual Machines

delete

Virtualization by multiplexing (1/3)

- Expose a resource among multiple virtual entities
  - E.g., CPU, memory, machine, link
- Isolation generally ensured through indirection (which hides physical names)
- Examples:
  - OS abstractions of thread (virtual CPU), address space (virtual memory), pipes (virtual links), ...
  - Storage abstractions of volumes / LUNs
  - Virtual Machine Monitor: abstraction of virtual machine (virtual computer)

Virtualization by aggregation (2/3)

- Aggregate multiple physical resources into a single virtual resource
  - Typically with enhanced availability and scalability properties
- Examples:
  - Storage (RAID 0, RAID 1, RAID 1+0, RAID 5)
  - Networking : Link aggregation, L4-L7 load balancing, proxies, ...
  - Applications (active/passive clusters)

Virtualization by emulation (3/3)

- Use software to emulate a virtual resource different than the underlying physical resource
- Example:
  - RAM disk, virtual tapes, ...
  - binary translation, byte-code runtimes, ...
  - virtual devices

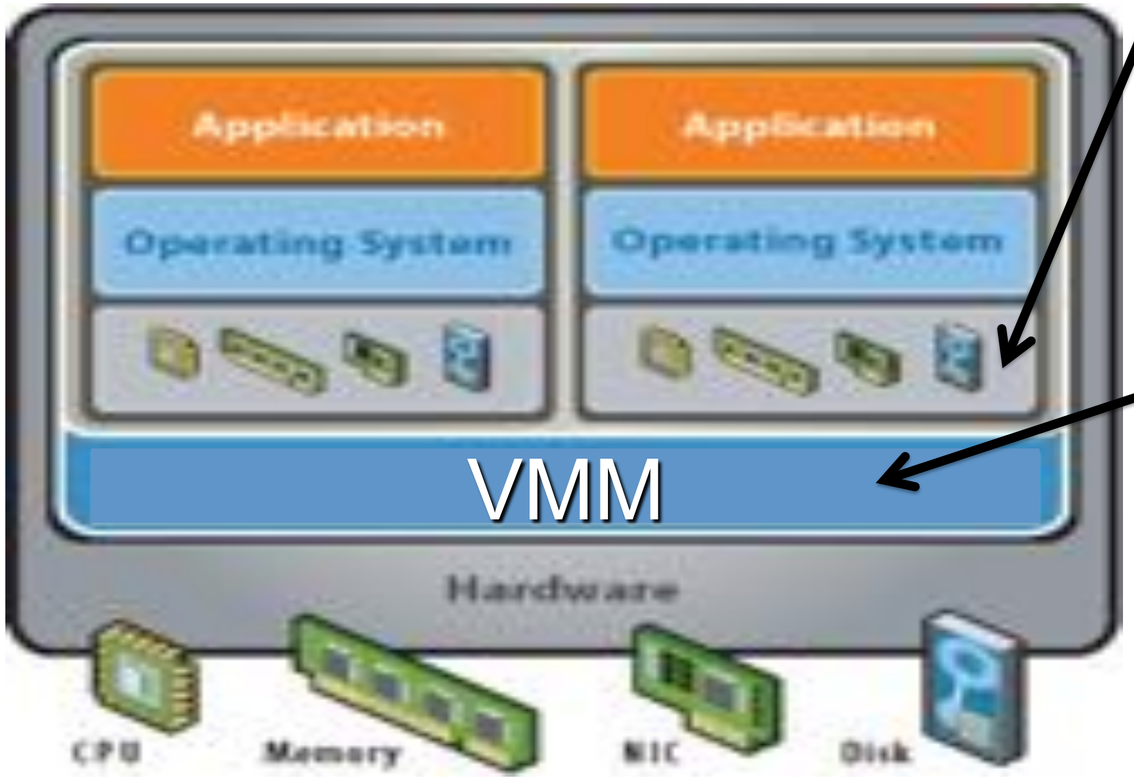
Virtual Machines – a historical perspective

CS522 – Principles of Computer Systems

Dr. Edouard Bugnion

Virtualization and Virtual Machines

Popek/Goldberg [1974]



- A virtual machine is an efficient, isolated duplicate of the real machine

- The Virtual Machine Monitor
- is in complete control of the system
- uses the hardware so that programs show at worst minor decreases in speed

Early Virtual Machines

- Early mainframe era
- Few platforms → allowed sharing of resources
- Limited OS functionality → enabled rapid innovation
- Hardware support for virtualization
- Solid set of principles and requirements [Popek/Goldberg 1974]

Early Virtual Machines

- Early mainframe era
- Few platforms → allowed sharing of resources
- Limited OS functionality → enabled rapid innovation
- Hardware support for virtualization
- Solid set of principles and requirements [Popek/Goldberg 1974]

- Rejected in favor of operating systems
- Innovation shifted to deliver new capabilities within OS (e.g., IBM MVS)
- Loss of architectural support for virtualization
- By mid-1990's: virtual machines were largely viewed as a relic of early computing era

The Return of Virtual Machines

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

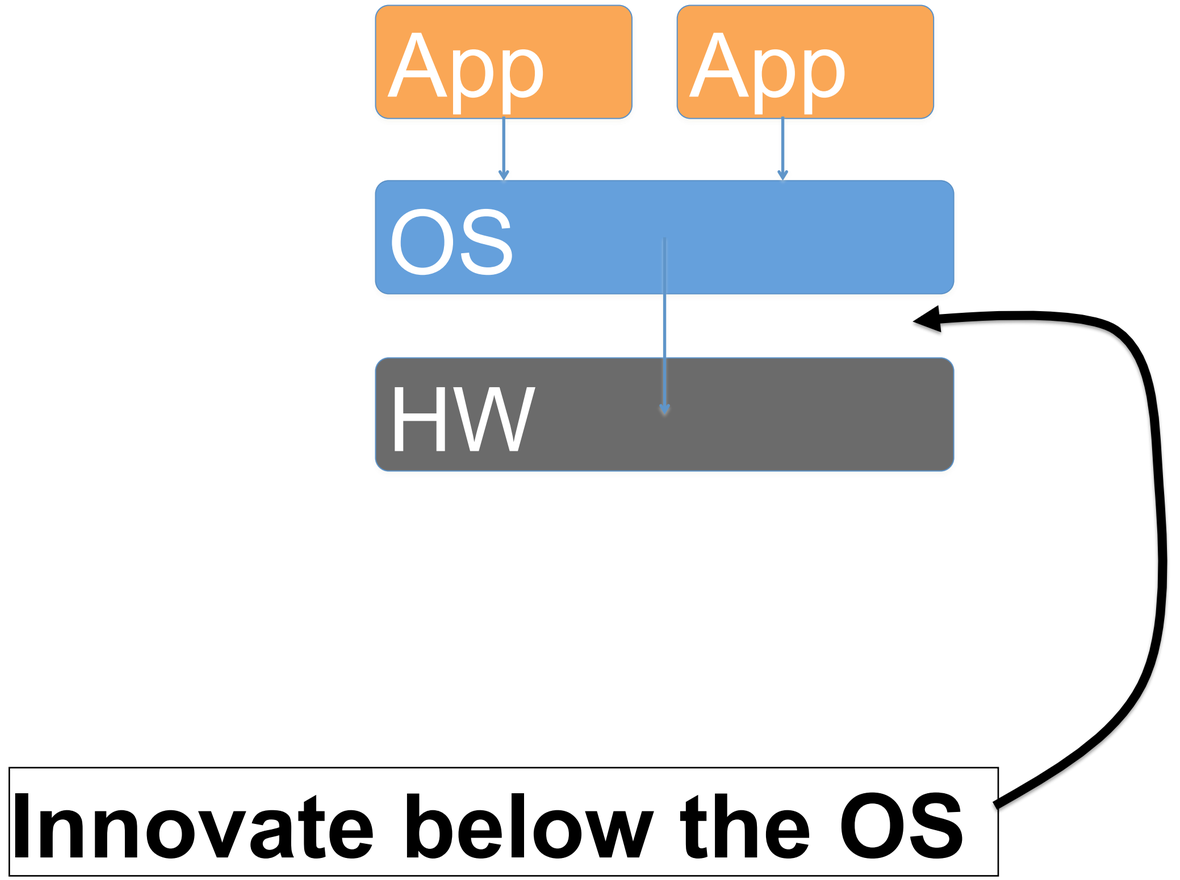
- **Why consider bringing them back?**

- When the OS fails
  - Missing functionality
  - Poorly adapted to hardware
  - Too complex and slow to innovate
- Because the OS is special
  - There can only be one (!)
  - Near monopolies

The Return of Virtual Machines

- **Why consider bringing them back?**

- When the OS fails
  - Missing functionality
  - Poorly adapted to hardware
  - Too complex and slow to innovate
- Because the OS is special
  - There can only be one (!)
  - Near monopolies



Next: On VMM construction

Virtualization – VMM construction

CS522-Principles of Computer Systems

Dr. Edouard Bugnion

VMM multiplex and emulate

- Multiplex CPU and memory
  - Each VM has its own **virtual CPU/core**
    - P=core; V=core
  - Each VM has its own **virtual physical memory**
    - P=physical memory; V=physical memory
- Emulate
  - Sensitive instructions executed by the VM (“trap-and-emulate”)
  - I/O devices: virtual disk, virtual keyboard, virtual screen, virtual NIC, …

Review – OS/App layering

- Key observation – application executes (unprivileged) instructions directly on the CPU

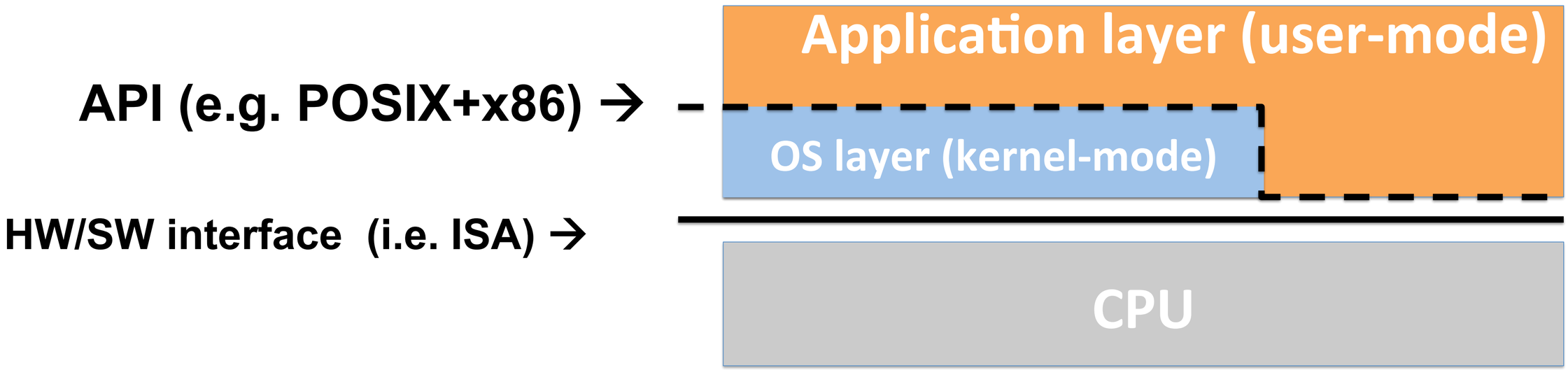
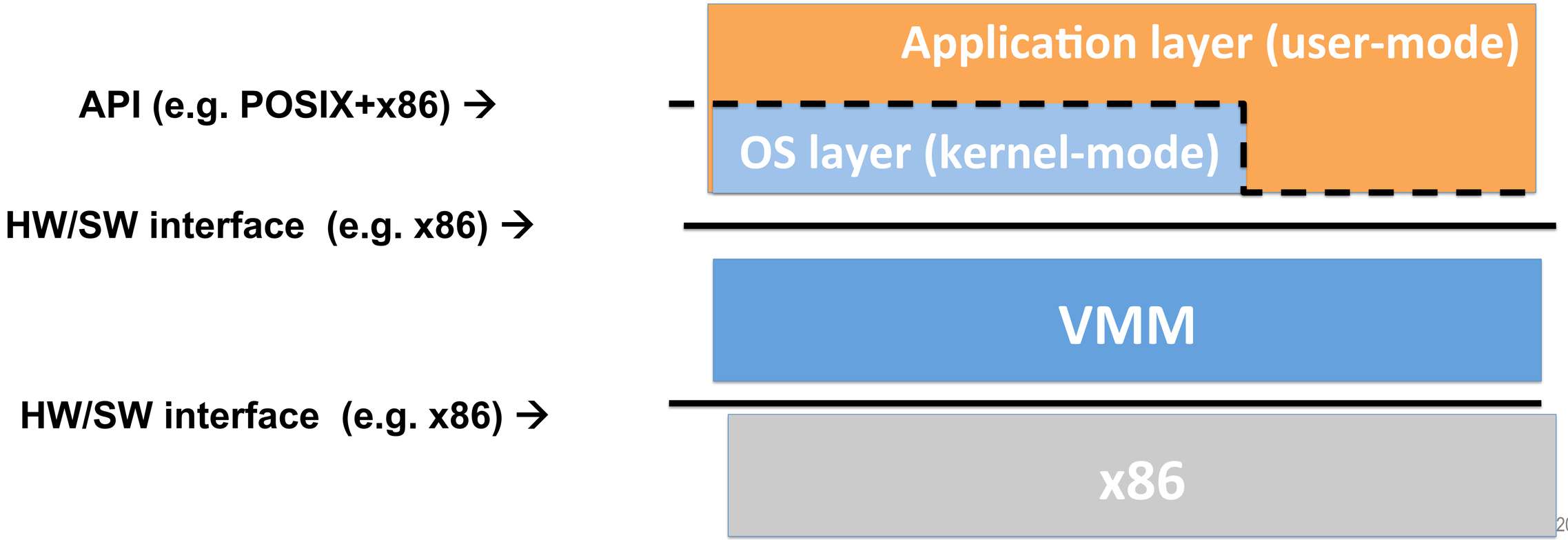


Fig: see Saltzer/Kaashoek 2.16.

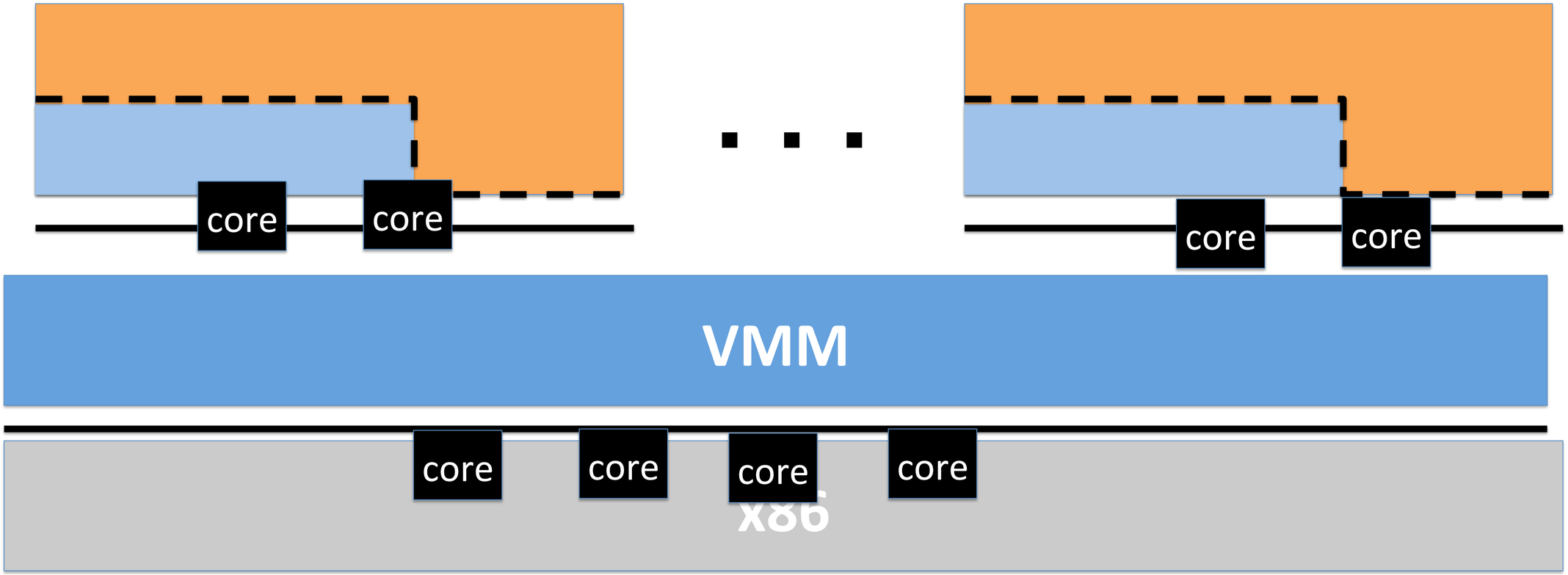
Adding a layer of indirection

- Virtual CPU == virtualize the HW/SW interface (in particular ISA)
- Standard virtualization trick – expose the same abstraction V as the underlying resource P and everything else will “still work” ™



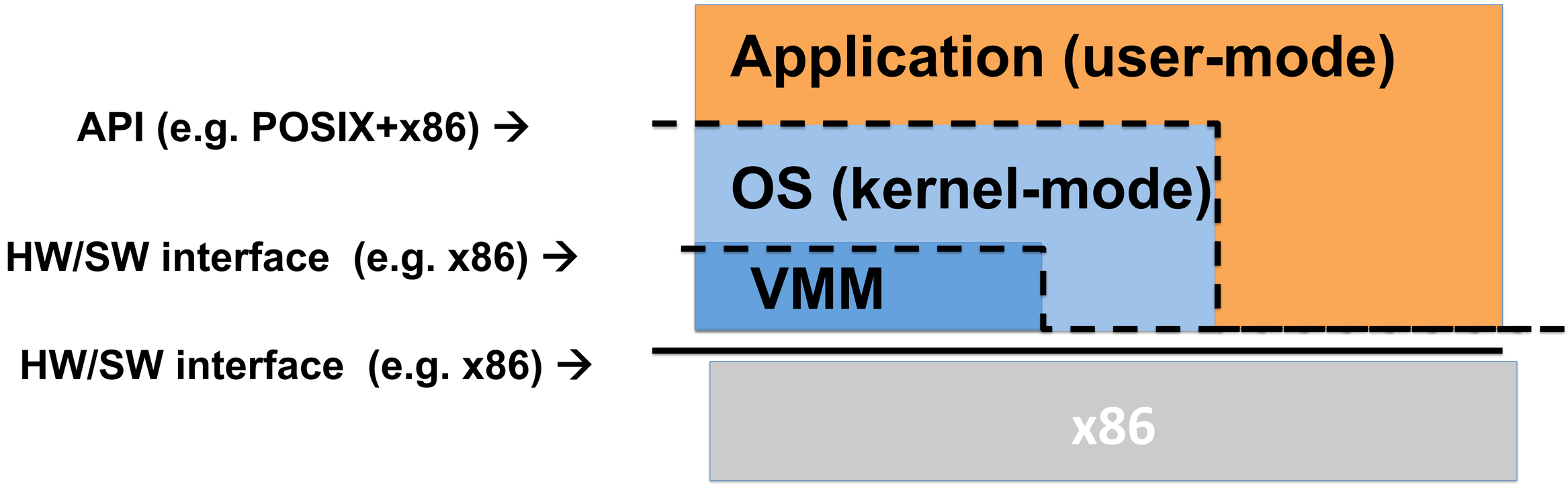
Multiple VMs– space/time multiplexing

- Multiplexing in space and in time
  - N VMs with n vCPU each on M hardware core
  - N * n can be greater than M → scheduling similar to OS
- VM are isolated



Efficiency through direct execution

- When P=x86 && V=x86; use x86 whenever possible
- How / when ?



Popek / Goldberg Theorem [1974]

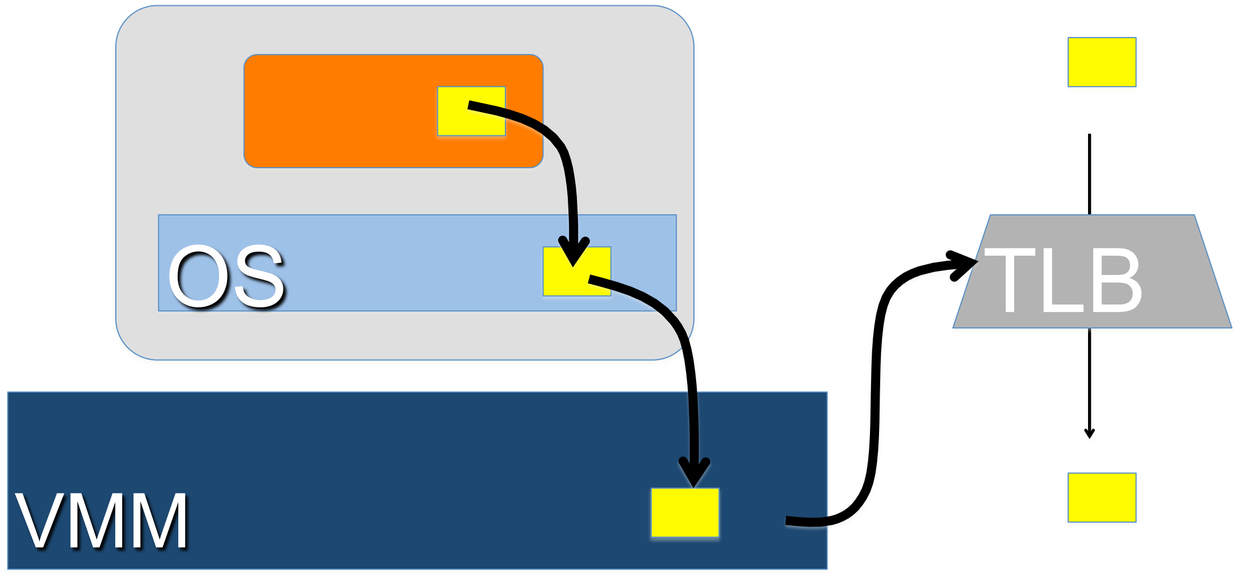
- Defines a property of an ISA
- Classification of instructions
  - Privileged – kernel mode only
  - Sensitive -- semantics vary (other than trap) based on privilege level
- Theorem -- A VMM using only direct execution is possible only if all sensitive instructions are privileged.
- Intuition: construst a VMM that:
  - Runs guest kernel in user mode
  - “trap-and-emulate” of all sensitive instructions
- Details in the paper

Virtualizing physical memory

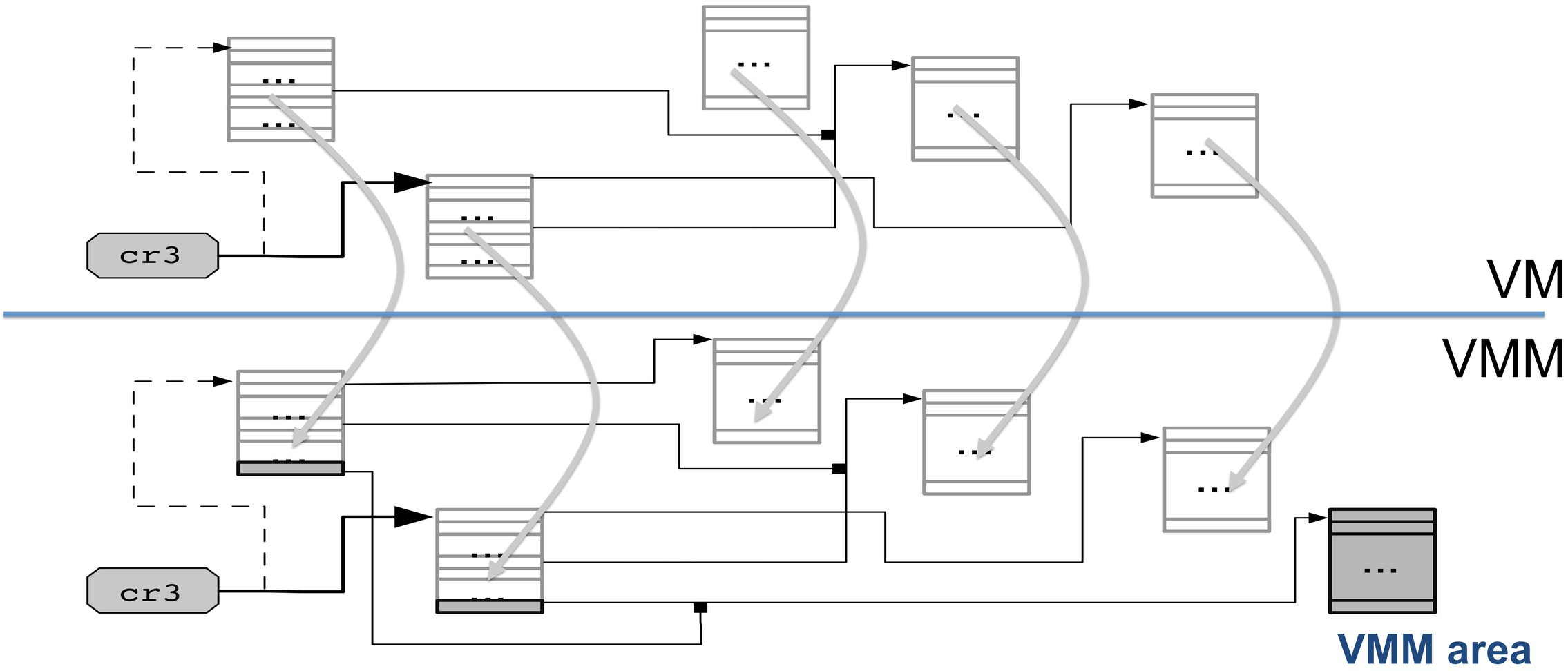
- Challenge – efficient multiplexing of physical memory

Virtualizing physical memory

- Challenge – efficient multiplexing of physical memory
- Two levels of indirection:
  - VA → gPA (managed by OS)
  - gPA → hPA (managed by VMM)
- TLB maps virtual to (host-)physical address
- Two possible answers
  - Shadow page tables.
  - Nested page tables.

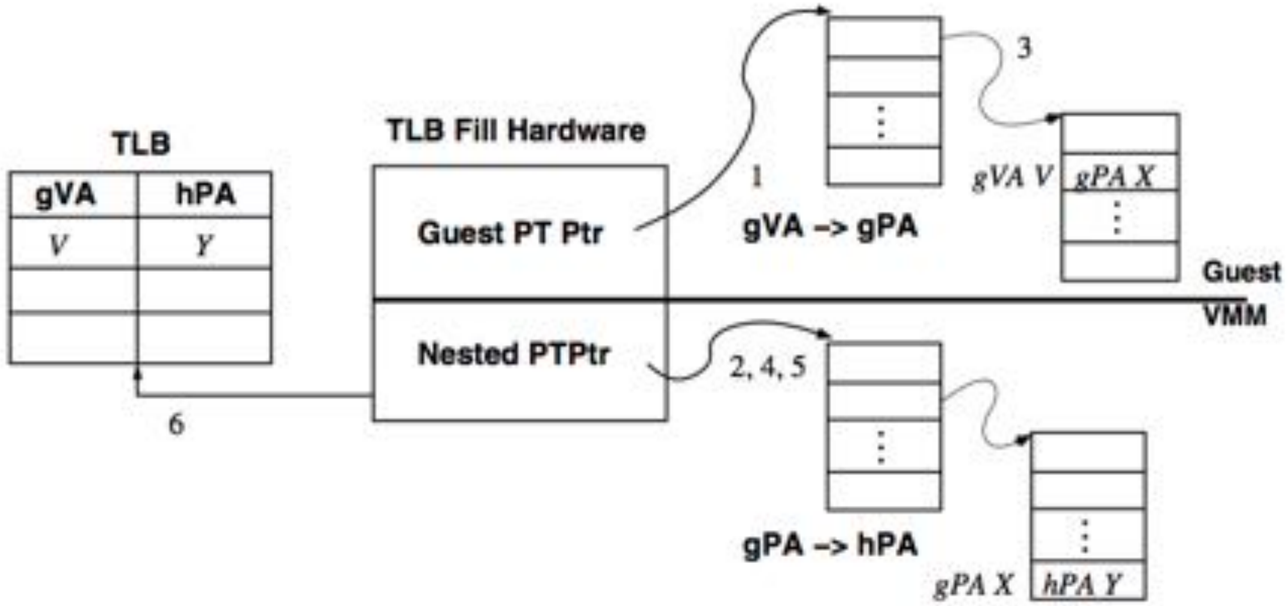


Shadow Page Tables (no architectural support)



Nested page tables (aka architectural support)

- Hardware implementation that simultaneously walks through two page table structures
  - 2 hardware registers (P-cr3 and DP-cr3) determine roots
- Available on all current Intel and AMD processors



type I and type II VMMs

- type I VMMs allocate and schedule physical resources (cpu, mem, io) among the virtual machines.
- type II VMMs rely on a separate host operating system for resource scheduling.
- Efficient resource scheduling is fundamental to any layered design
  - Specific complexities with virtual machines

type I and type II VMMs

- type I VMMs allocate and schedule physical resources (cpu, mem, io) among the virtual machines.
- type II VMMs rely on a separate host operating system for resource scheduling.
- Efficient resource scheduling is fundamental to any layered design
  - Specific complexities with virtual machines

- Examples of current tools
- “type I” architectures –
  - Xen (open-source)
  - VMware vSphere
  - Microsoft Hyper-V
- “type II” architectures –
  - KVM (Linux host) (open-source)
  - VMware Workstation (Windows host) and Fusion (OX X host)
  - Parallels (Windows and OS X hosts)

Wrap-up

Virtualization Case study – Memory resource management in Disco

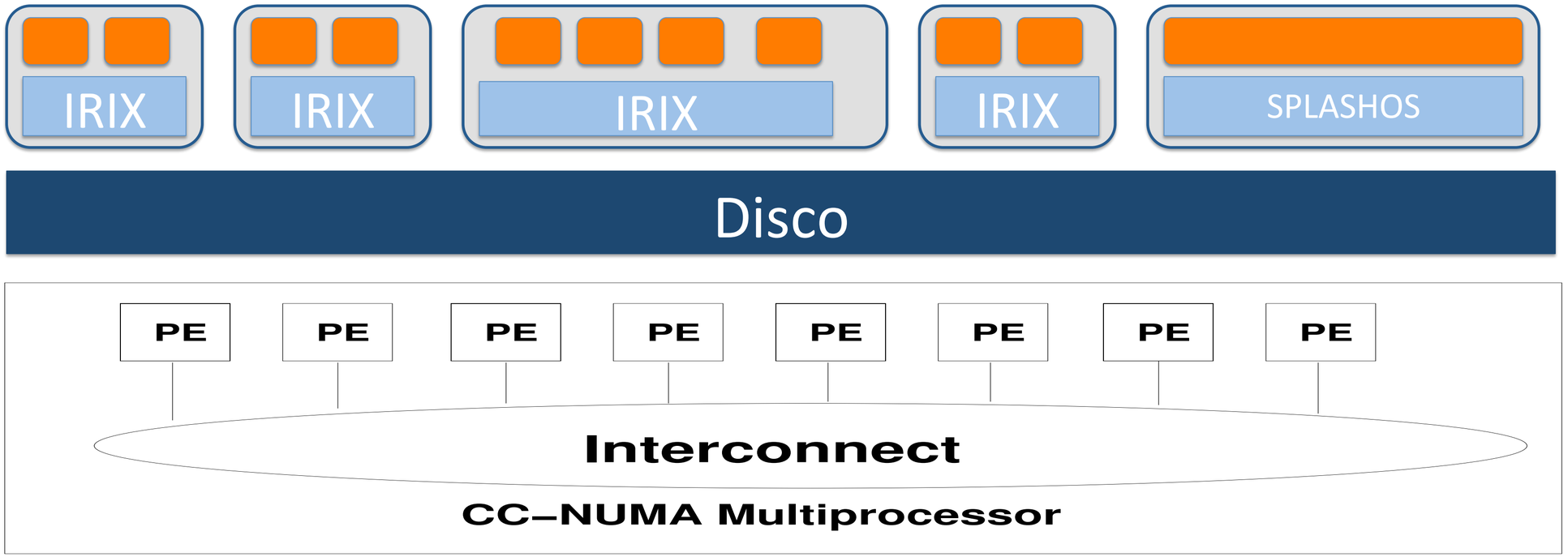
CS522 – Principles of Computer Systems

Dr. Edouard Bugnion

Memory Resource Management in Disco

- Disco: Running Commodity Operating Systems on Scalable Multiprocessors
  - [SOSP ‘97, TOCS ’97, SIGOPS HOF 08]
  - Early paper that returned interest in VMM
- Technical contributions
  - Memory resource management
  - (ccNUMA optimizations)
- Approach influenced solutions found in today’s commercial products

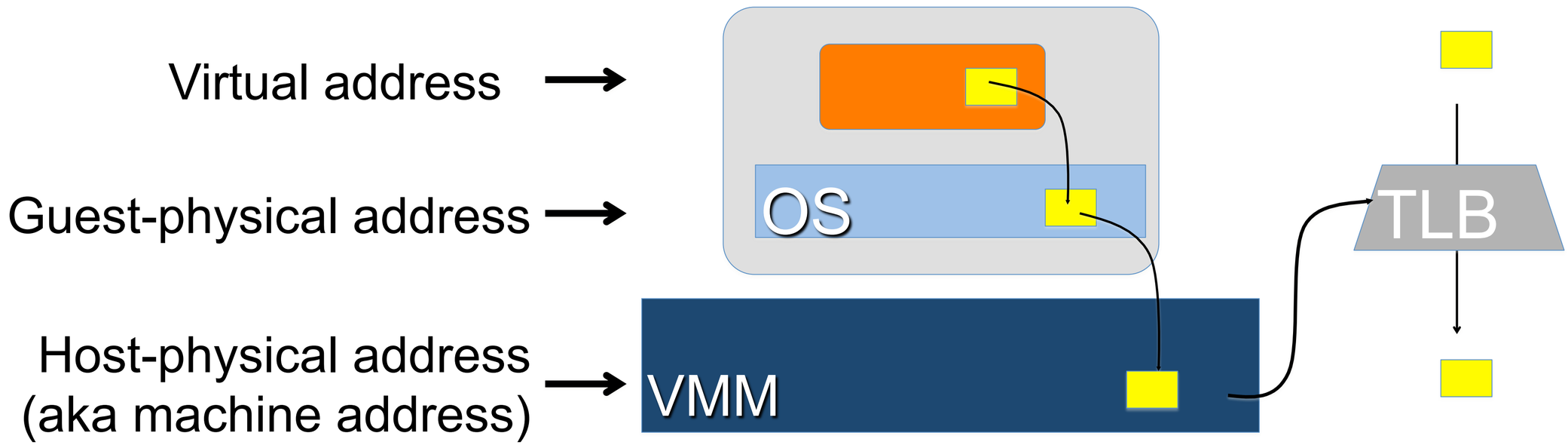
Stanford FLASH + Disco



Disco – highly-scalable Virtual Machine Monitor for a scalable ccNUMA machine

Memory Resource Management goals: (1) minimize memory overheads; (2) optimize locality

Memory virtualization using shadow page tables



Key building block of VMMs:

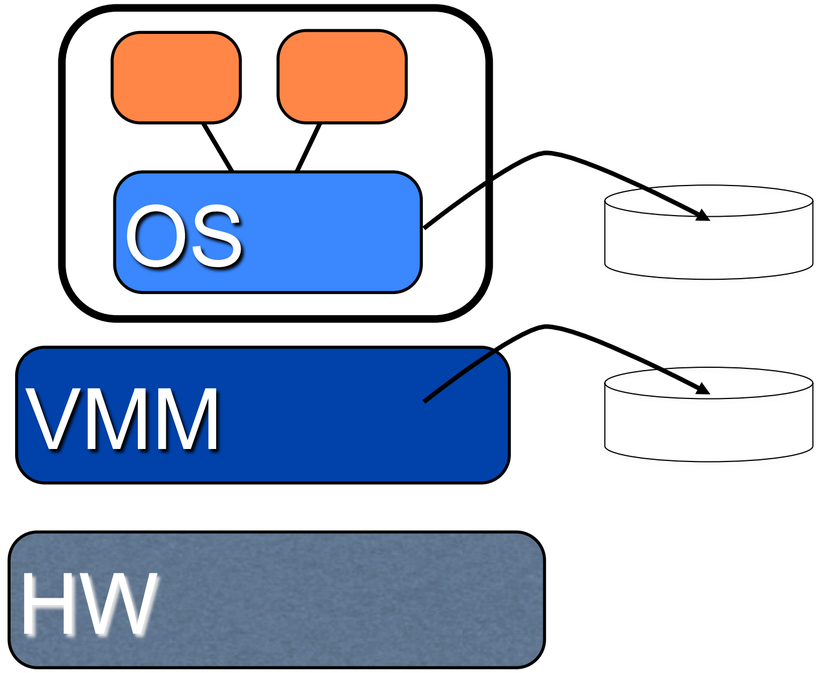
- Multiplexing : of physical memory among virtual machines
- Indirection: inserting virtual --> host-physical mappings in TLB
- Resource management - allocation of guest-physical to host-physical mappings

Memory resource management

- Known challenge when memory is over-committed
- VMM controls additional level of indirection
  - the guest-physical --> machine mappings
  - can change mappings at any time
  - can force them to be read-only

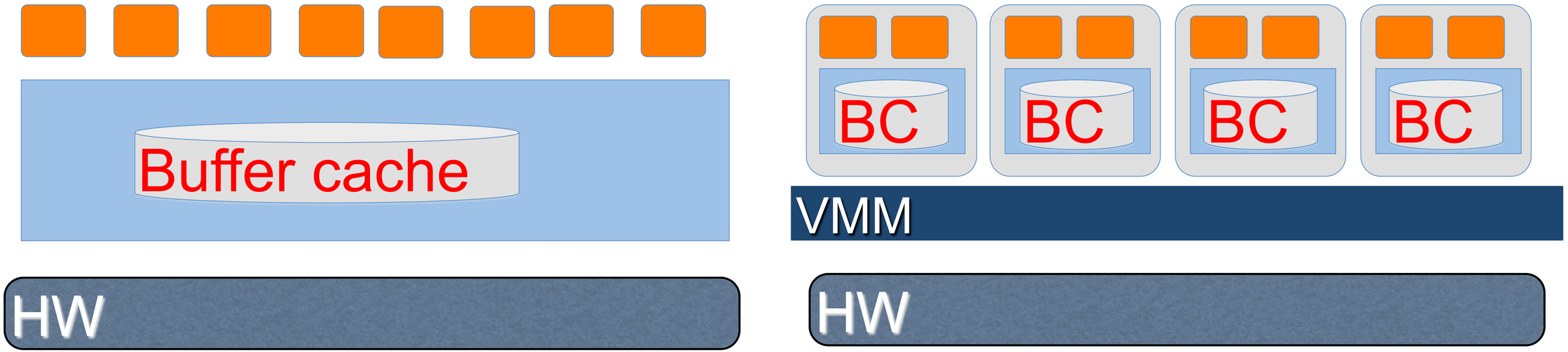
Double swapping problem

- VMM is under memory pressure, swaps page X
- Later, OS is under memory pressure, decides to swap page X
  - VMM must first (i) swap out another page, and (ii) swap page X back in
  - Only then can OS swap X out.



Global memory efficiency

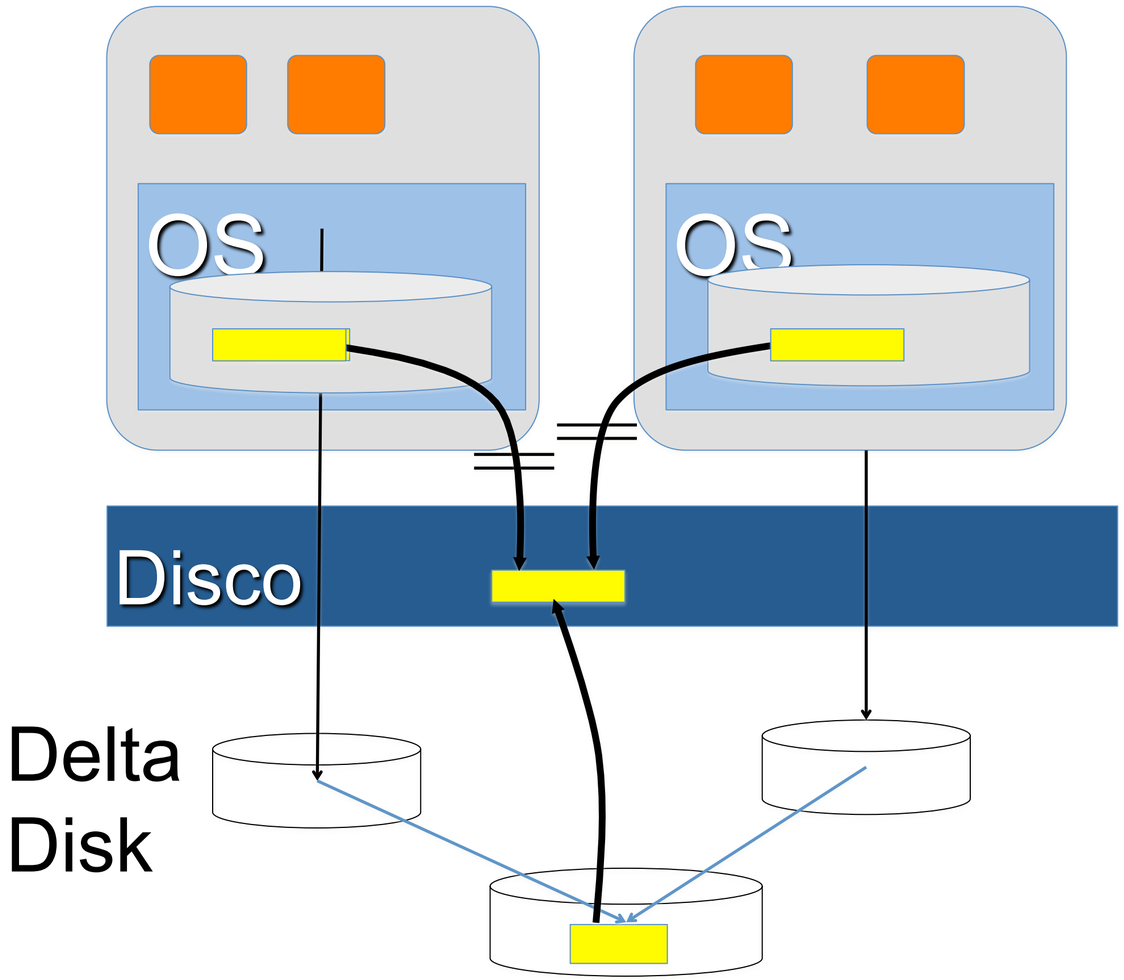
- Workload split in N virtual machines of a virtual cluster
  - Processes can be partitioned
  - File system buffer cache is replicated, with mostly identical content
- How to transparently share the buffer caches without involving the OS ?



Transparent memory sharing

- Solution: transparently share physical memory between virtual machines
- Identify know identical pages by:
  - Interposing on DMA
  - Interposing on network traffic

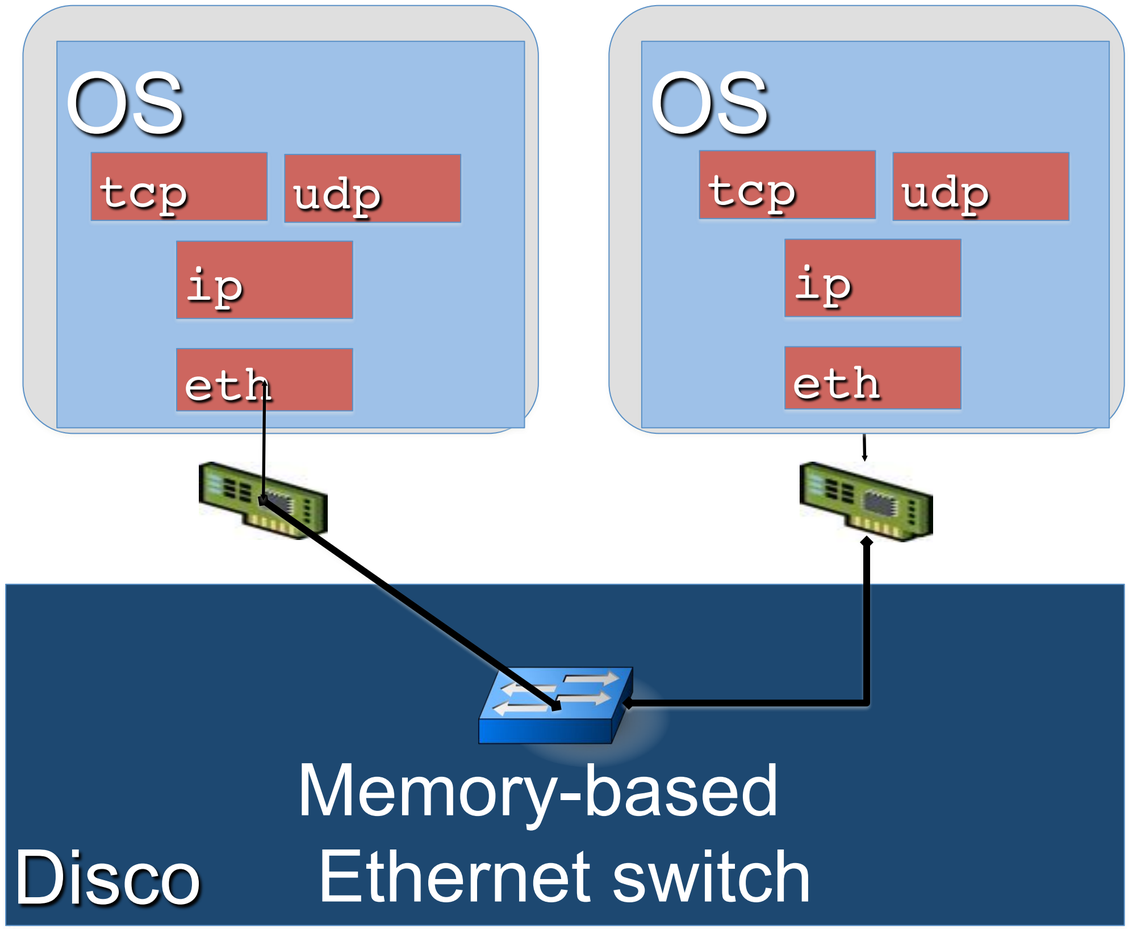
Interpose on DMA – shared "base" disk



Shared base disk

- N copies of nearly identical filesystems
- On SCSI READ to the base disk:
  - Associate page with disk offset
  - Protect the page; share it read-only
  - Avoid subsequent SCSI READs
- Copy-on-write (memory)
  - Triggered by page fault
- Copy-on-write (disk)
  - Triggered by SCSI write

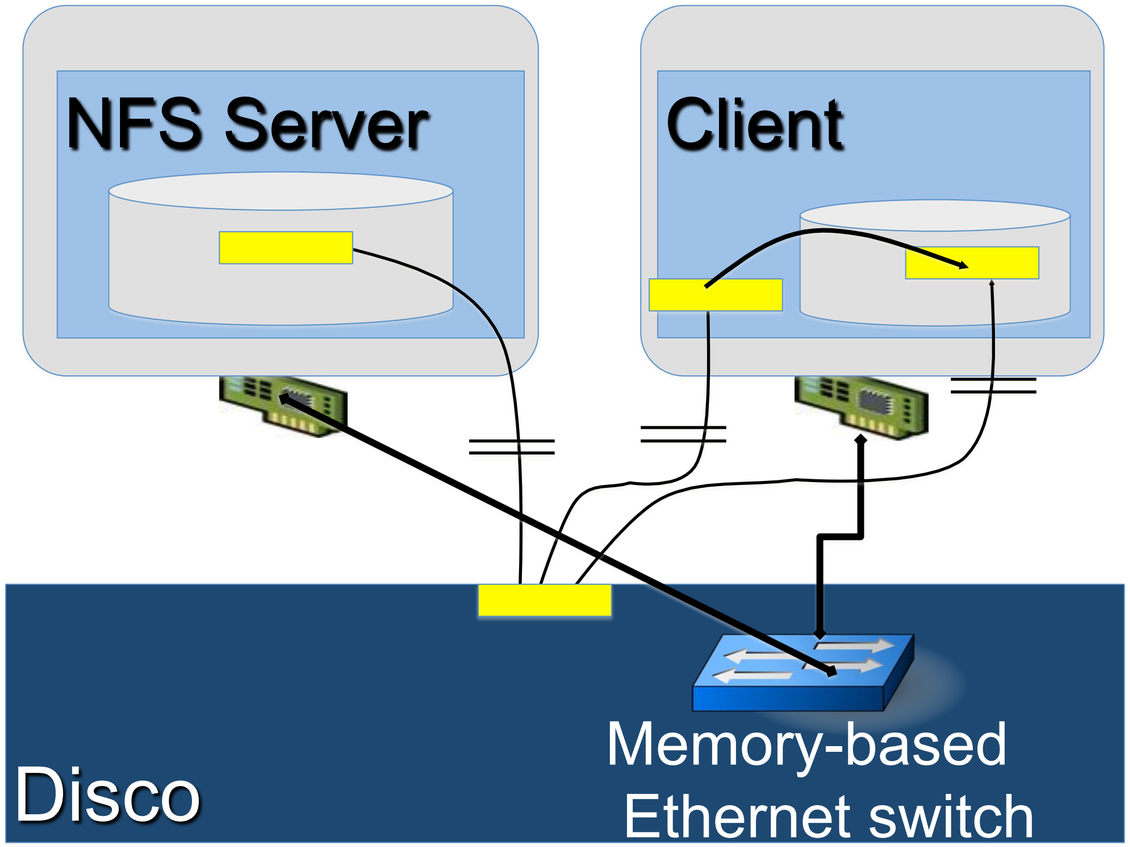
Background – VM networking



- Modern OS have built-in networking stacks
- Disco
  - Virtual Ethernet NIC per VM
  - Software Ethernet switch between VMs
- Virtual machines communicate like real machines

Interpose on network traffic

N NFS clients + 1 NFS sever



- Disco's in-memory Ethernet switch
  - Copies small fragments
  - Remaps page-size fragments
- Operating system modification
  - IRIX's NFS code – copy using "hypercall
- → Transparently share filesystem buffer caches of NFS clients and servers.

Example: NFS Read reply

Memory resource management today

- As important as ever
  - VM memory size increases
  - VM density increases
  - VM migration increases
- Limitation of Disco's mechanisms:
  - Applies only to pages that are a-priori identical because of I/O interposition
  - Does not solve the double buffering problem
- Key improvements :
  - Carl Waldspurger, Memory Resource Management in ESX Server, OSDI 2002

Thank you

Virtualization Case study – Bringing virtualization to x86

CS522 – Principles of Computer Systems

Dr. Edouard Bugnion

Welcome back to 1998-99

- Bringing Virtualization to the x86 architecture with VMware Workstation 1.0
  - ACM System Software Award 2009
  - [TOCS 2012]

Challenges

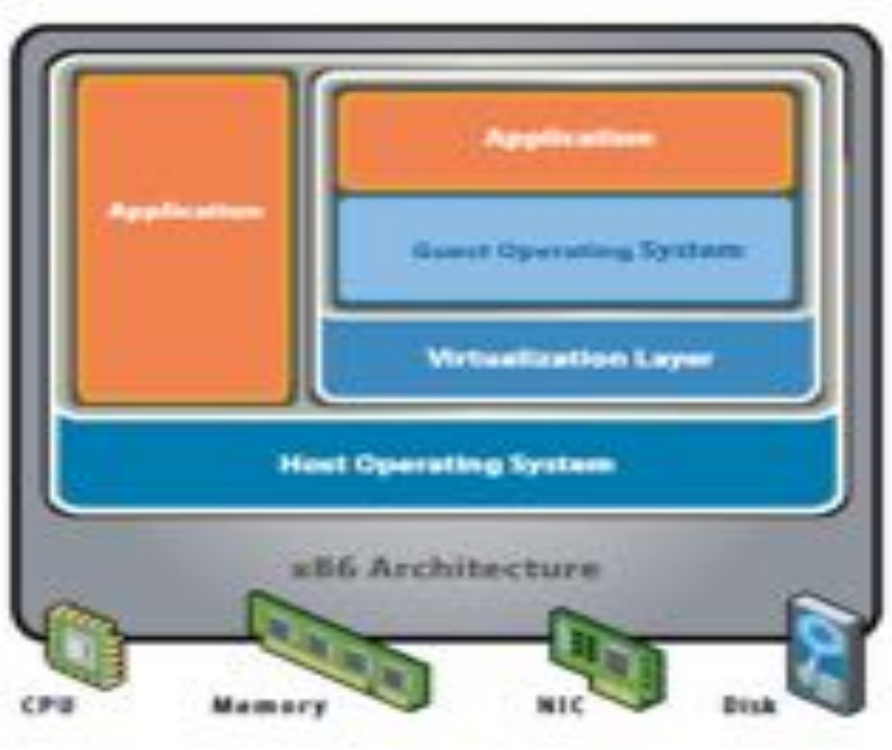
- x86 architecture is not virtualizable
  - 17 instructions violate Popek/Goldberg criteria
  - Need to run unmodified OS (Windows!)
- x86 architecture is of daunting complexity
  - Many protection mechanisms and execution modes
  - None virtualizable (save for v8066)
- Diversity of peripherals
  - With vendor-specific drivers
  - Little documentation or source
- Need for a simple user experience
  - No reinstall

Two contributions

- How to create a type II VMM running on top of a protected commodity host operating systems
  - Addresses the device diversity and user experience challenges
- How to virtualize the x86 architecture (which fails the Popek/ Goldberg criteria)
  - Addresses pragmatically the x86 challenges
  - Principle: focus on requirements of supported guest operating systems

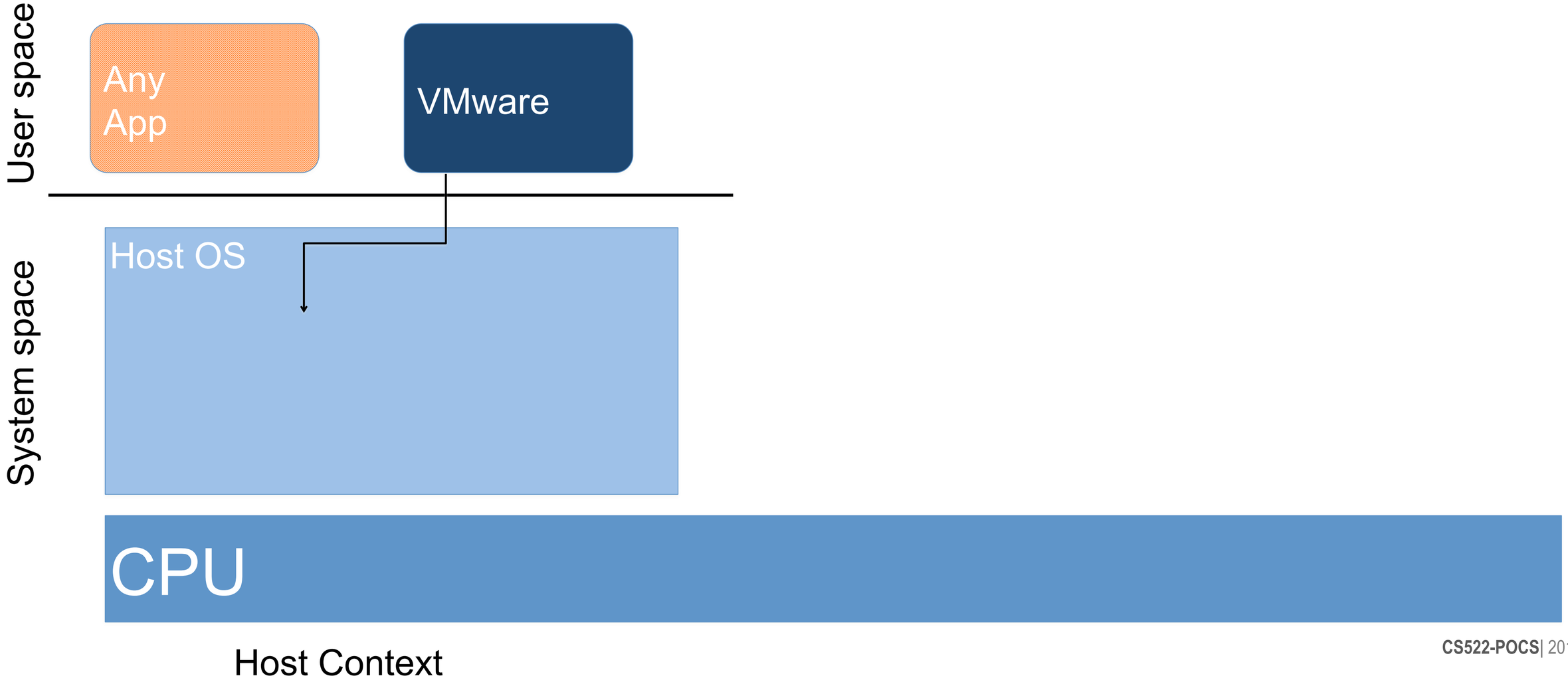
#1: Simplicity and diversity

- Solution: Operate VMM as an type II hypervisor
  - extension of an existing (host) operating system
- Challenge: constraints from the host operating system
  - Address space
  - Exception handlers, …

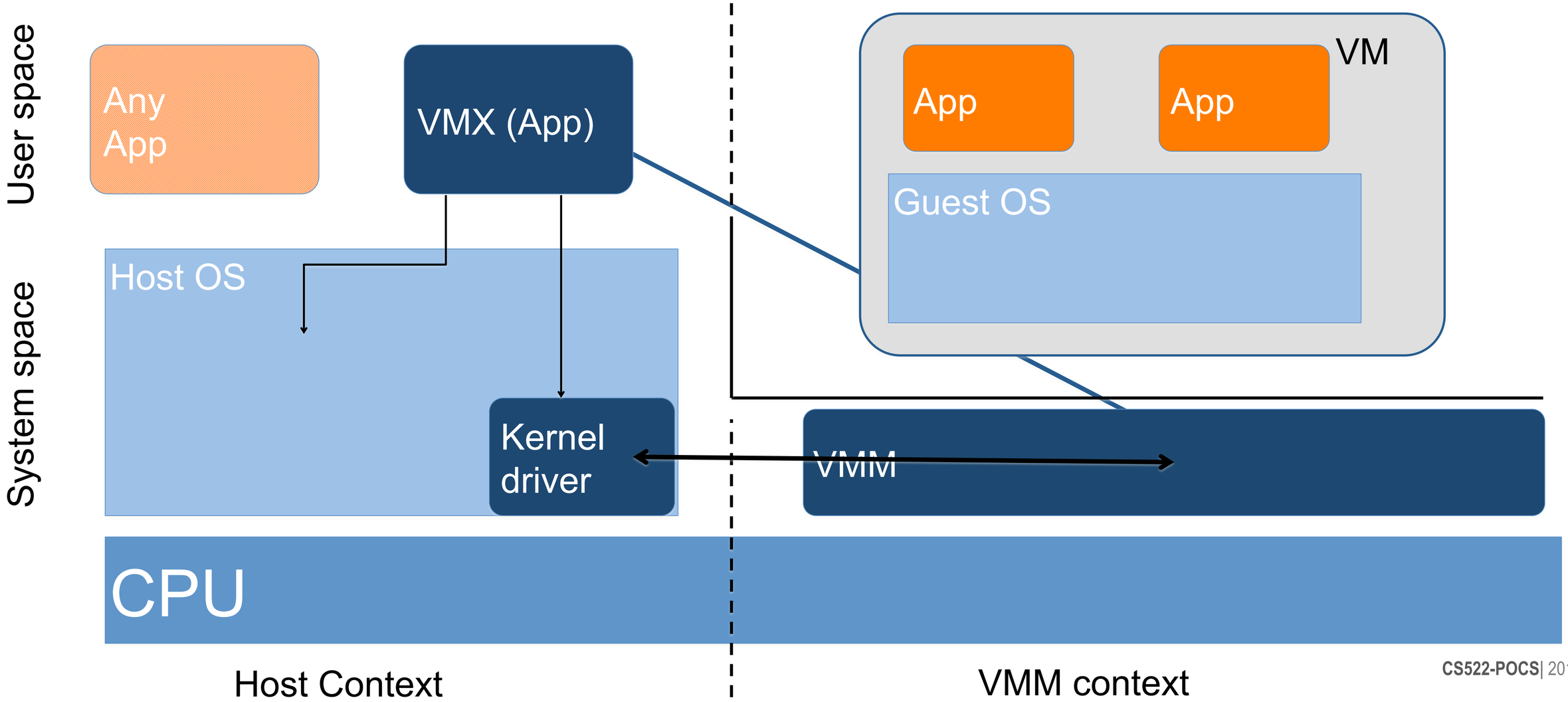


VMware Hosted Architecture

VMware Hosted Architecture



VMware Hosted Architecture



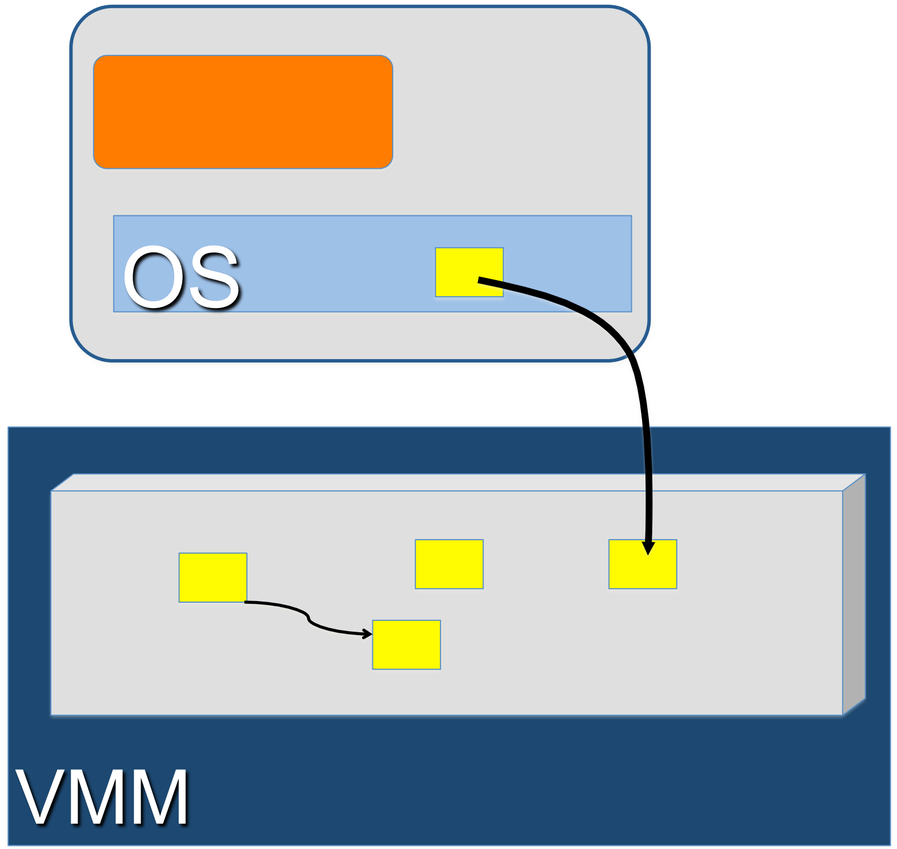
#2: Virtualizing the x86 architecture

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- Challenge – can't use classic technique
  - "trap-and-emulate" (direct execution) is not sufficient
  - 17 sensitive, non-privileged instructions
  - Need to support binary compatibility
- Observation
  - most x86 sensitive instructions are only sensitive when executing system software
- Solution
  - Use trap-and-emulate when possible, i.e. when there are no sensitive instructions
  - Use binary translation when necessary and faithfully emulate the semantics of all instructions (including the sensitive ones)

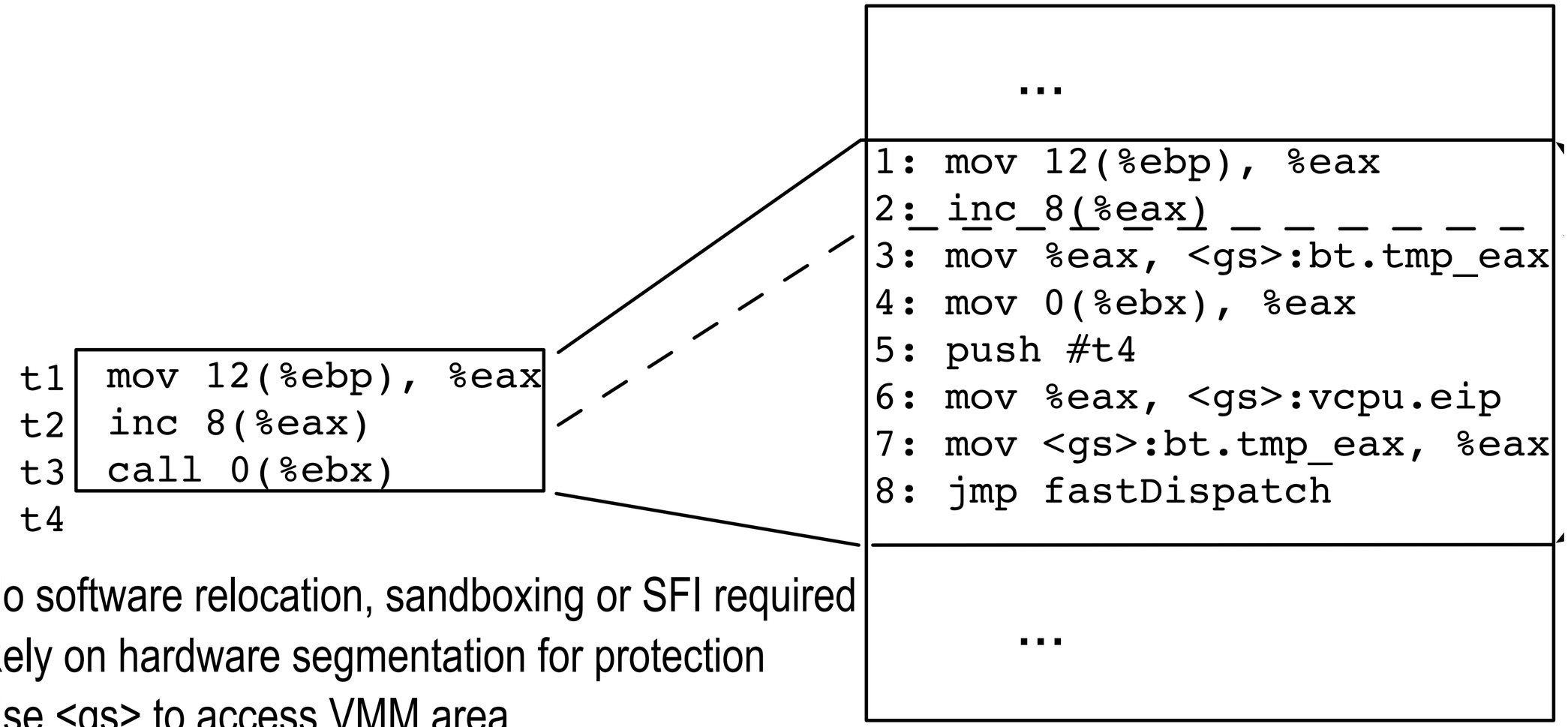
Basics of Dynamic Binary Translation

- Translator – convert basic block of VM instructions into a executable sequence
  - Convert sensitive instructions
- Translation cache – buffer holding many cached translations
- Dispatch function –jumps into translation cache



Challenge – performance suitable for VMM

Most instructions are translated 1:1

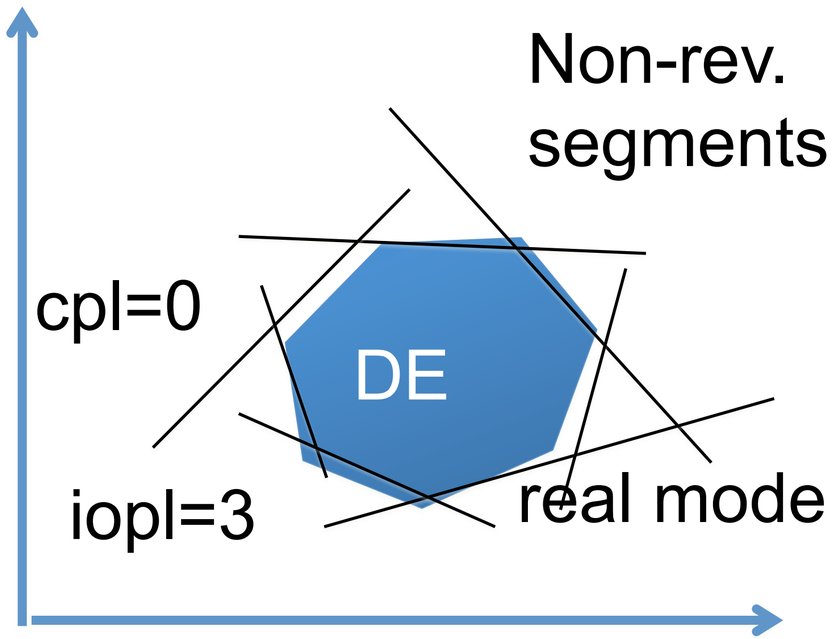


- No software relocation, sandboxing or SFI required
- Rely on hardware segmentation for protection
- Use <gs> to access VMM area

Decision algorithm

- Direct execution only possible in constrained situations
  - Running untrusted code that cannot enable/disable interrupts (or has interrupts enabled)
  - Running either protected mode or v8086 mode (not real mode)
  - All segments are reversible
    - CPU version of segment matches memory copy

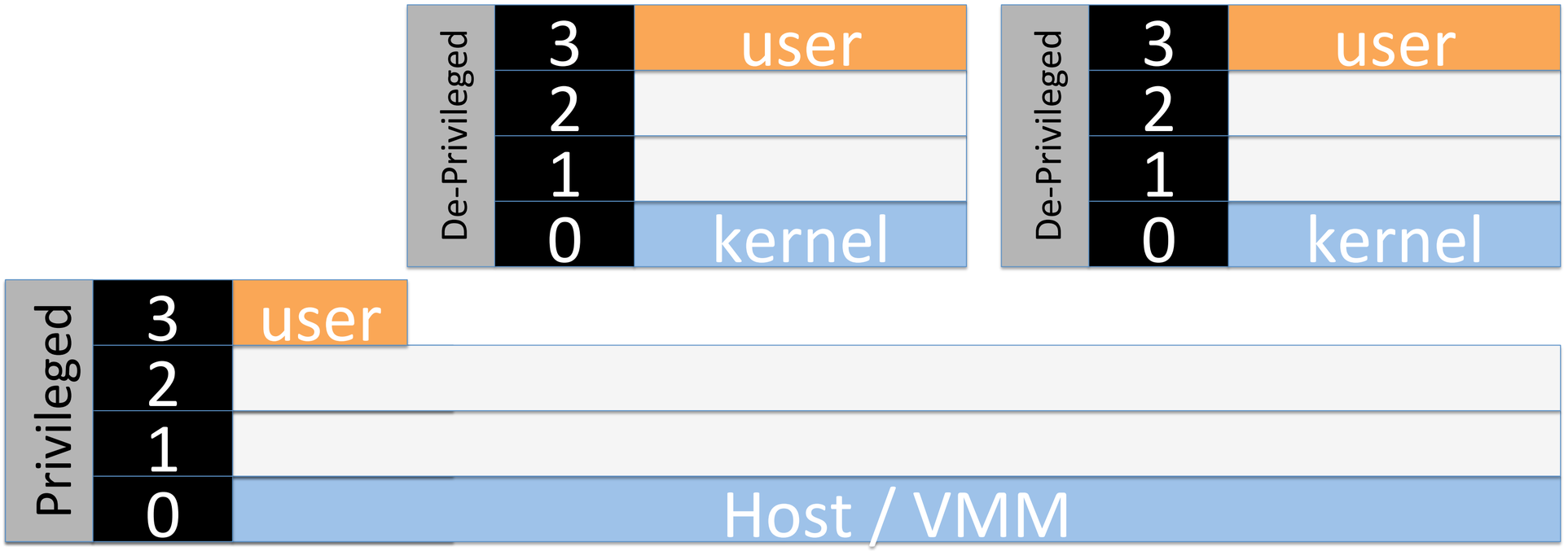
Key decision algorithm – what is “system” ?



- System = when the possibly sensitive instructions are actually sensitive
- Direct execution only possible outside of “system” modes, i.e when:
  - Running untrusted, user-level code that cannot enable/disable interrupts
  - Running in protected mode or in v8086 mode
  - Running with all segments reversible
- Details in the paper

Virtualization since 2005: Intel VT-x and AMD-v

- Available on all current 64-bit processors
  - Duplicate the 4 protection rings
- Meets Popek/Goldberg criteria
- Used by all virtualization solutions today



Thank you.