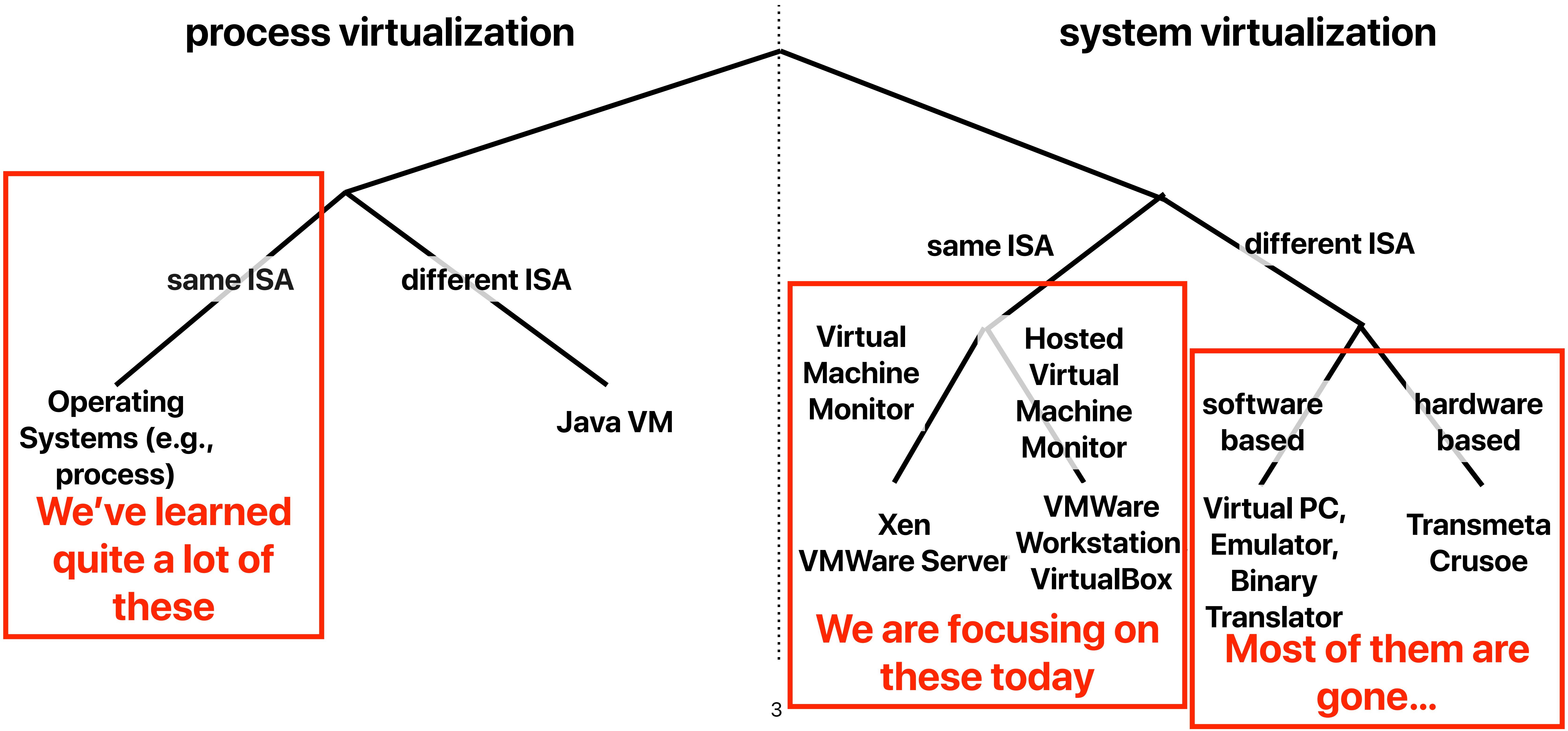
Virtual Machines & Reflections

Hung-Wei Tseng

Virtual Machines

Recap: Taxonomy of virtualization



Virtual machine architecture

Applications

Guest OS

Virtual Machine Monitor

The Machine

Hosted virtual machine

Applications | Applications | Applications

OS | OS | OS

device emulation, virtualization

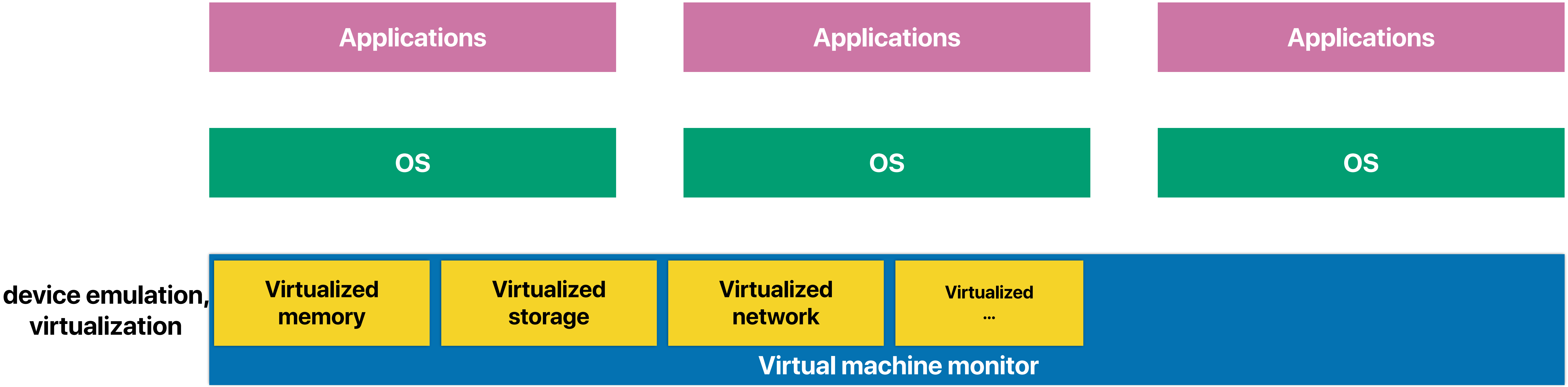
Virtualized memory | Virtualized storage | Virtualized network | Virtualized …

Hosted virtual machine monitor

device emulation, virtualization

Hosted operating system

Virtual machine monitors on bare machines



Back to 1974...

Formal Requirements for Virtualizable Third Generation Architectures

Gerald J. Popek
University of California, Los Angeles
and
Robert P. Goldberg
Honeywell Information Systems and
Harvard University

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.

Fidelity

Performance

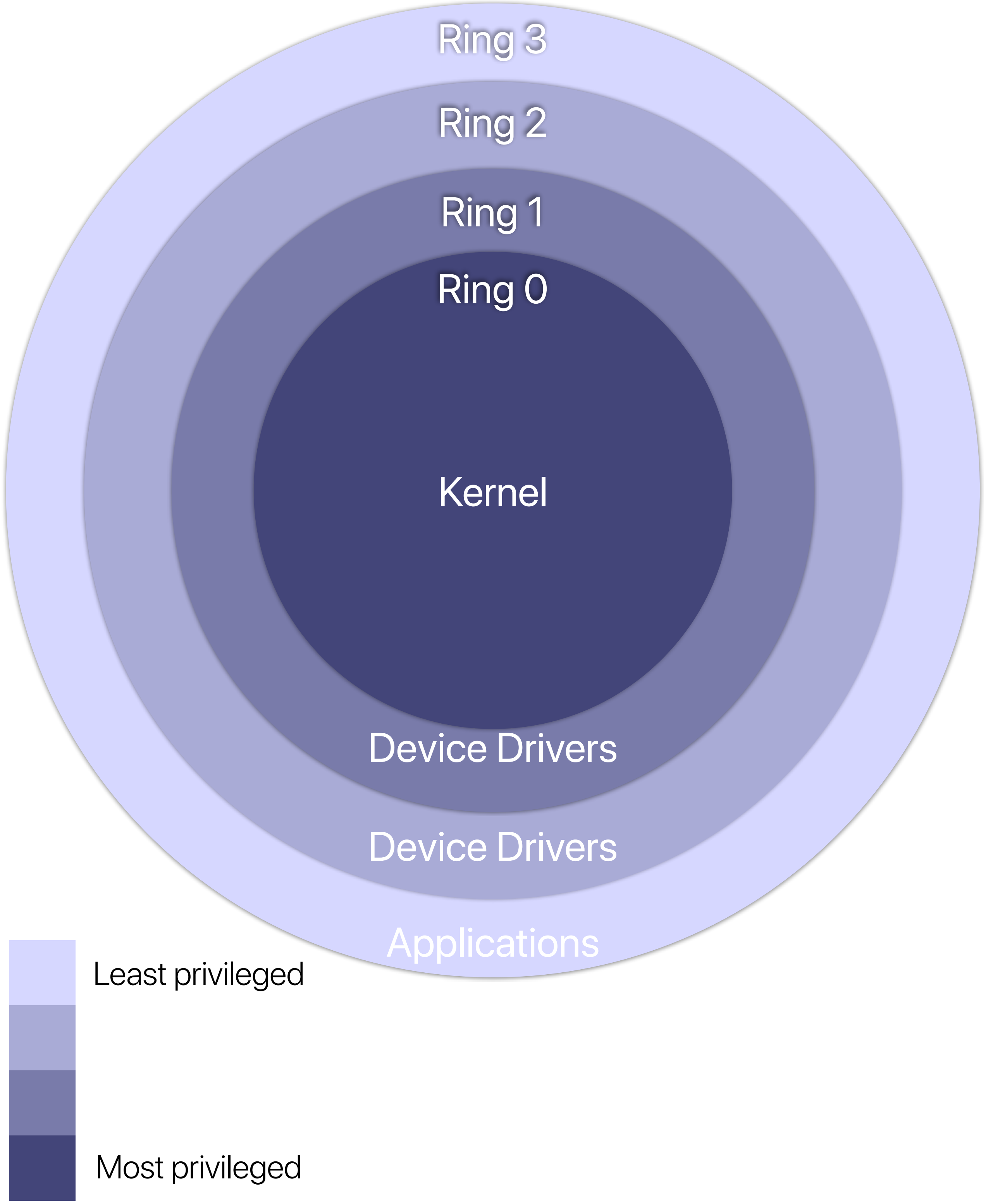
Safety and isolation

Three main ideas to classical VMs

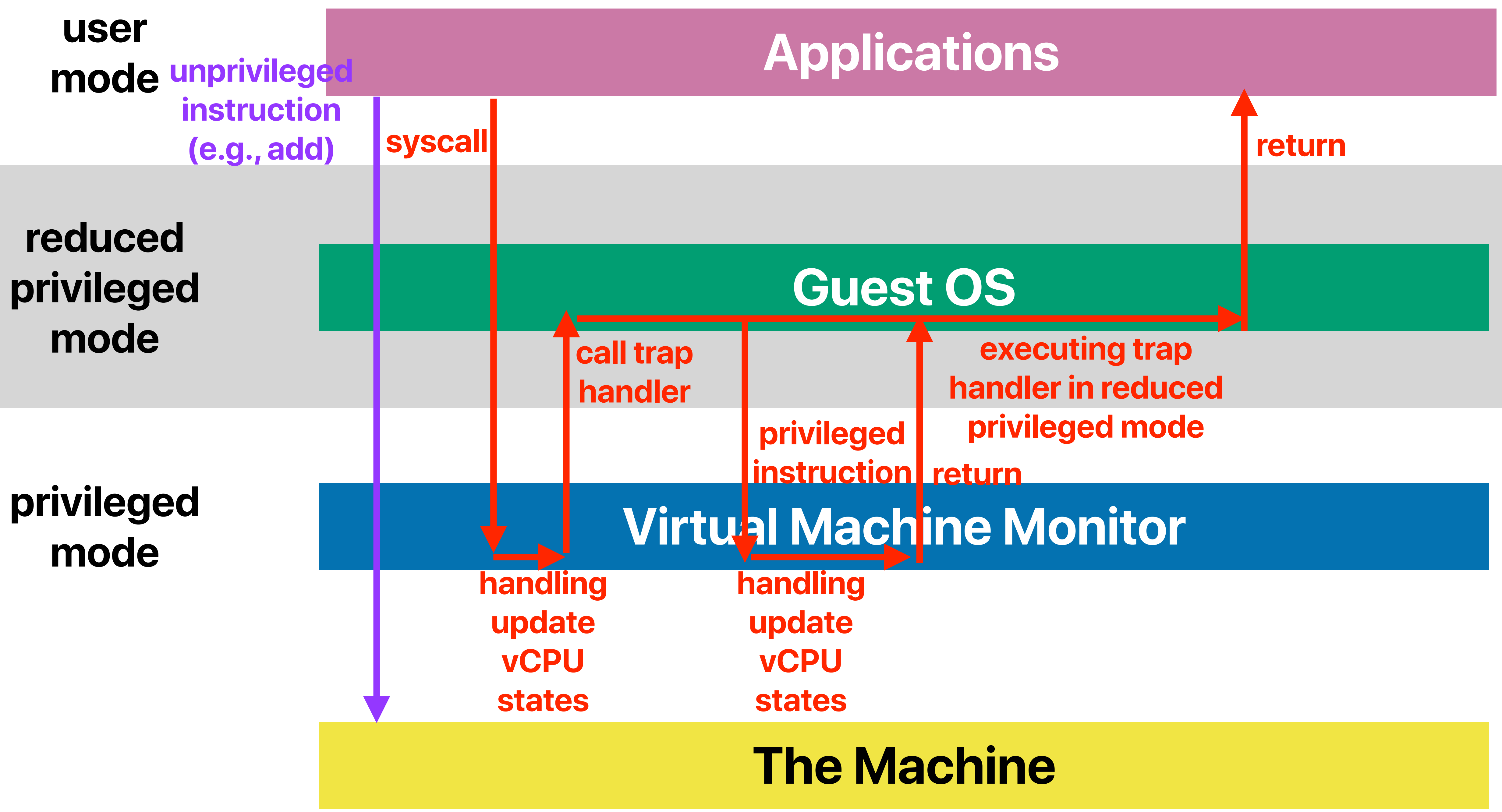
- De-privileging
- Primary and shadow structures
- Tracing

Recap: privileged instructions

- The processor provides **normal** instructions and **privileged** instructions
  - Normal instructions: ADD, SUB, MUL, and etc …
  - Privileged instructions: HLT, CLTS, LIDT, LMSW, SIDT, ARPL, and etc…
- The processor provides different modes
  - User processes can use normal instructions
  - Privileged instruction can only be used if the processor is in proper mode

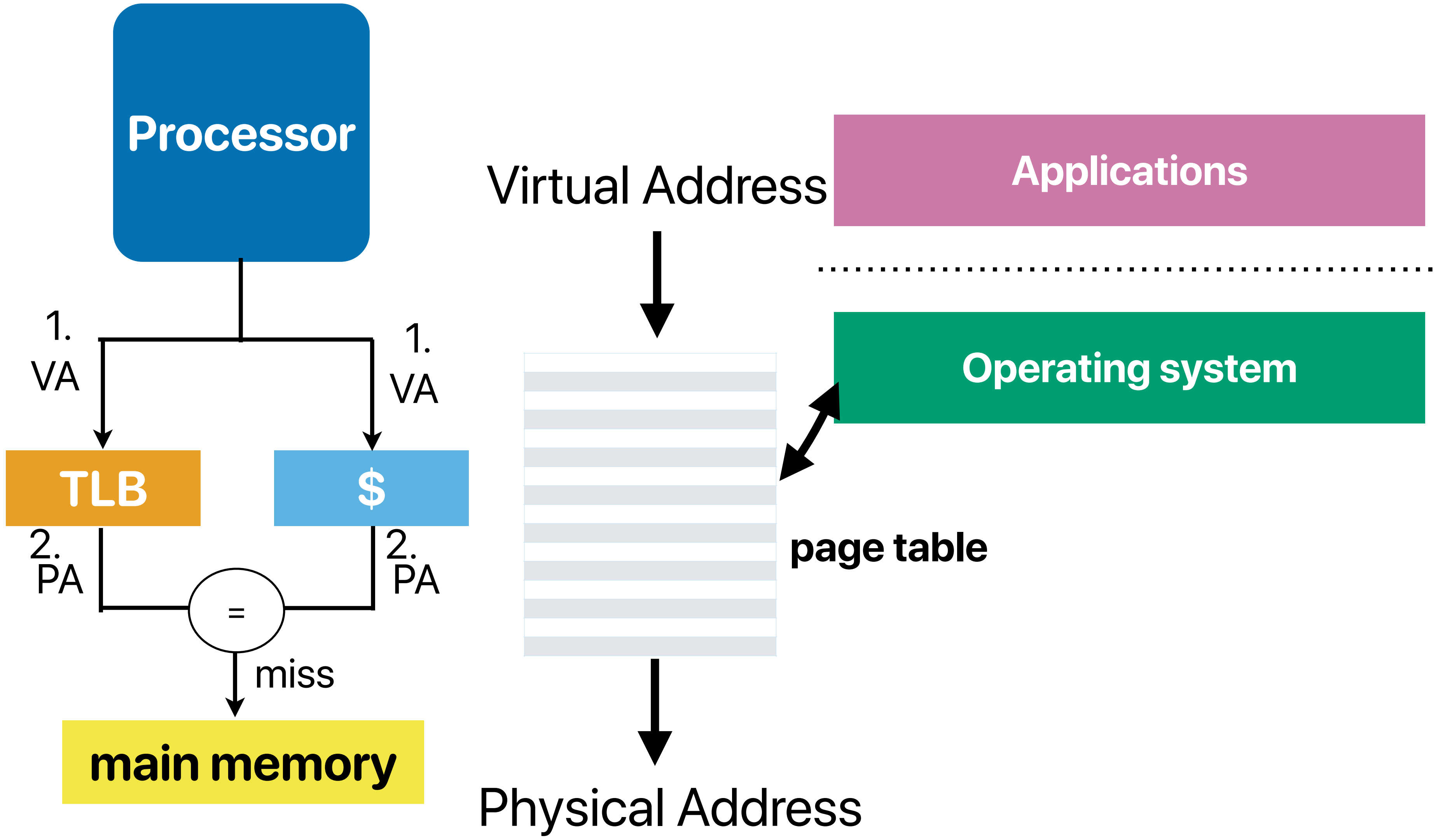


CPU Virtualization: Trap-and-emulate

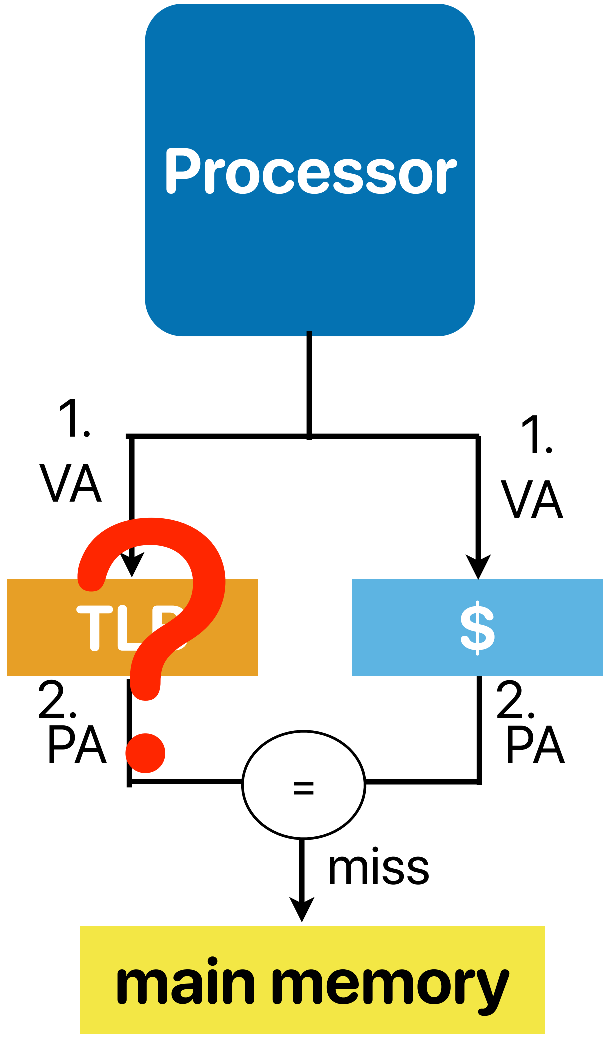


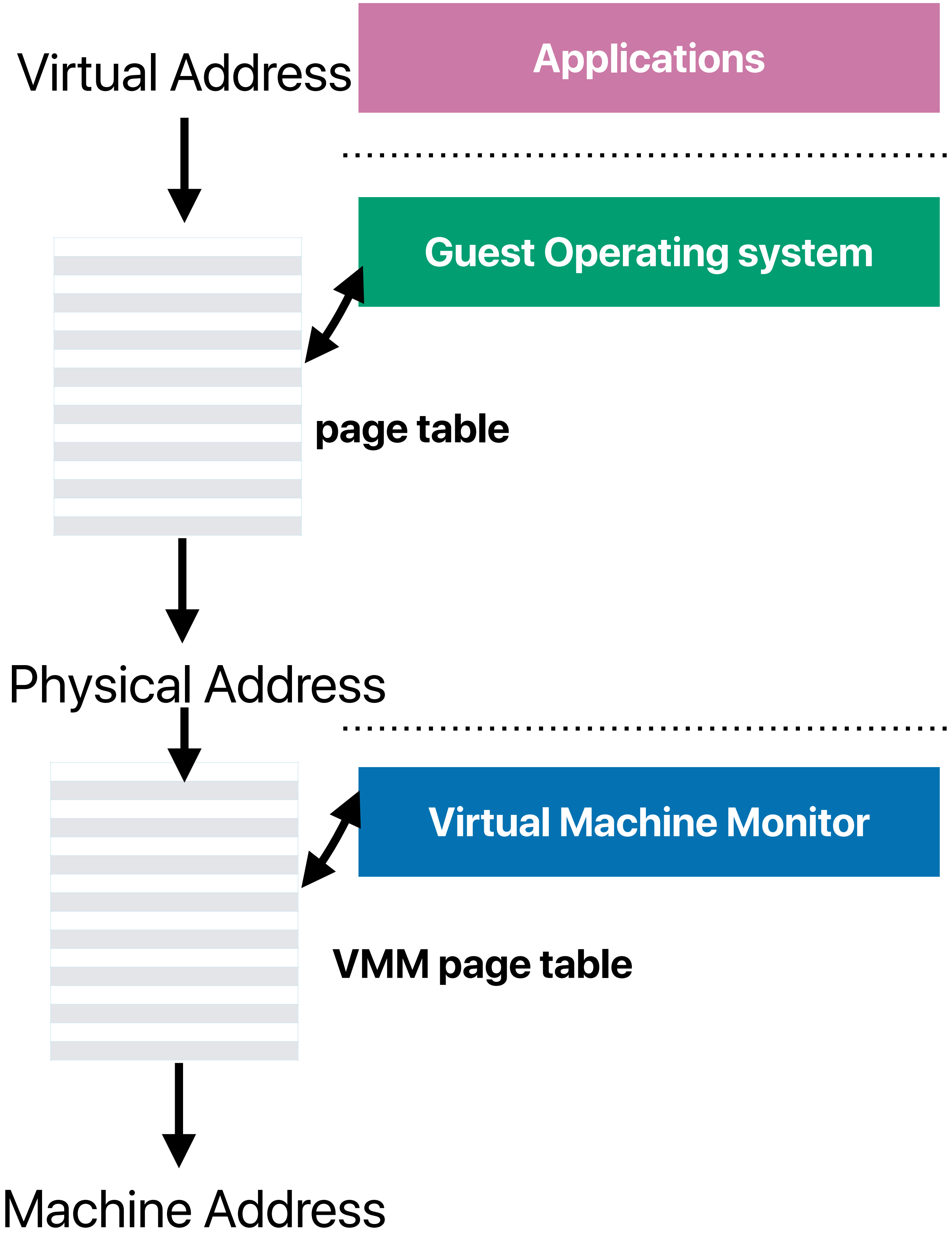
Recap: address translation with TLB

- This is called **virtually indexed, physically tagged cache**
- TLB hit: the translation is in the TLB, no penalty
- TLB miss: fetch the translation from the page table in main memory

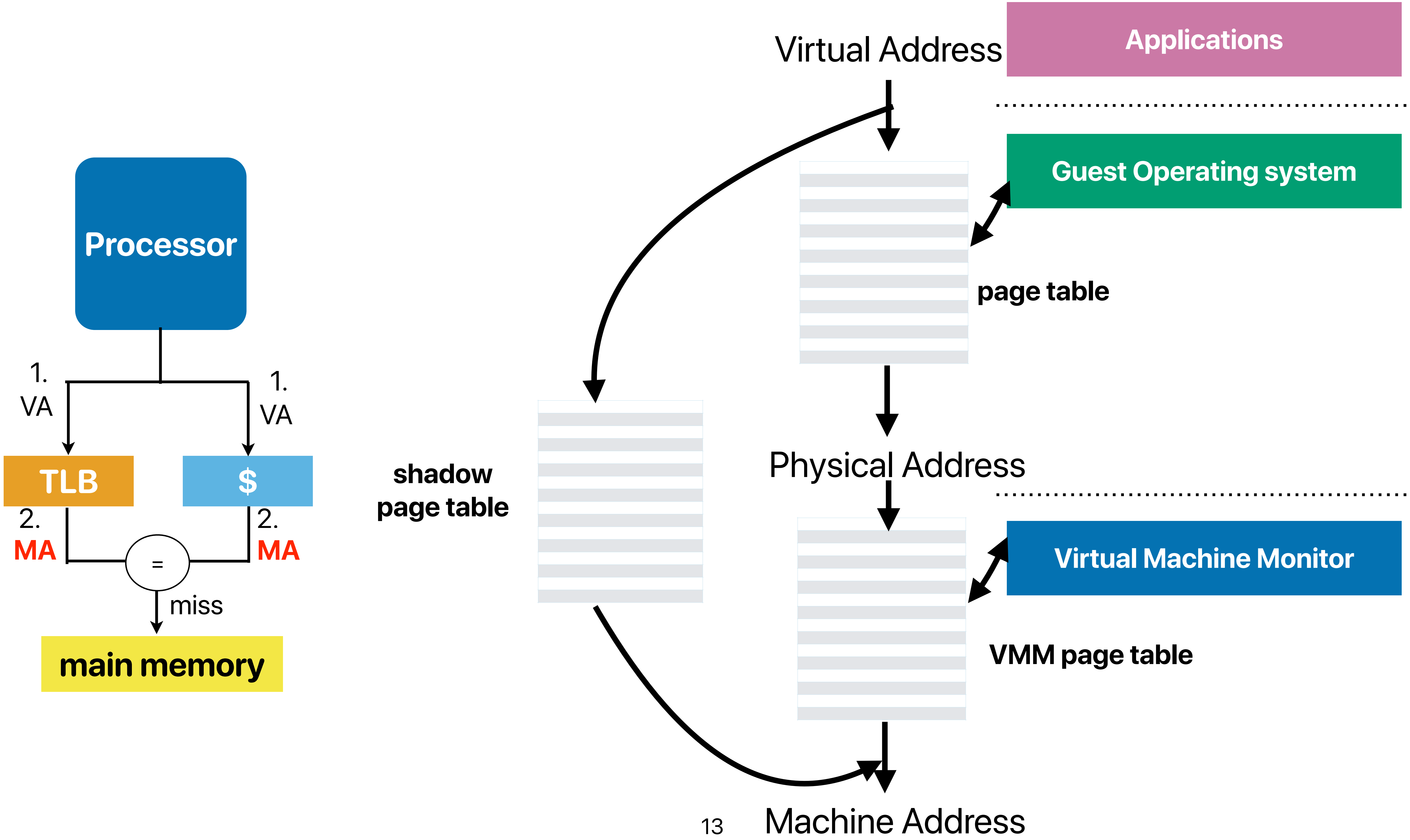


Address translation in VM





Address translation in VM



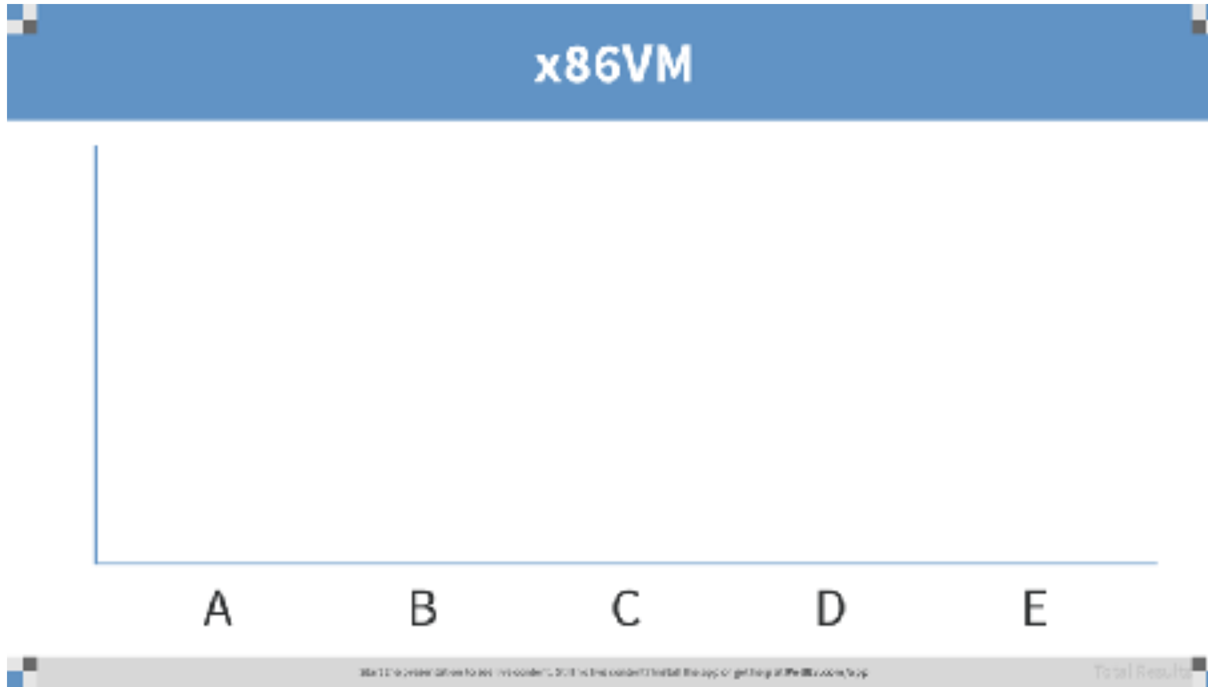
Tracing

- Trace accesses to important system data structures
- Memory tracing: You need to make the shadow page table consistent with guest OS page table
  - Protect these structures with write-protected
  - If anyone tries to modify the protected PTE — trigger a segfault handler
  - The segfault handler will deal with these write-protected locations and consistency issues for both tables

https://www.pollev.com/hungweitseng close in 1:00

Why this doesn't work with x86

- The classical x86 architectures cannot allow the VMM to use the classical trap-and-emulation for virtualizing guest operating systems. How many of the following best describes the reasons?
  ① The guest OS can be aware that it's not running in a privileged mode
  ② A privileged instruction in the guest OS may not trigger a trap
  ③ x86 does not provide a mechanism to set write-protected pages and handlers for tracing
  ④ x86's hardware-walk hierarchical page table structure prevents the use of shadow page tables.
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4



Why this doesn't work with x86

- The classical x86 architectures cannot allow the VMM to use the classical trap-and-emulation for virtualizing guest operating systems. How many of the following best describes the reasons?
  - ① The guest OS can be aware that it's not running in a privileged mode
  - ② A privileged instruction in the guest OS may not trigger a trap
  - ③ x86 does not provide a mechanism to set write-protected pages and handlers for tracing
  - ④ x86's hardware-walk hierarchical page table structure prevents the use of shadow page tables.

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

- *Visibility of privileged state*. The guest can observe that it has been deprivileged when it reads its code segment selector (`%cs`) since the current privilege level (*CPL*) is stored in the low two bits of `%cs`.

- *Lack of traps when privileged instructions run at user-level*. For example, in privileged code `popf` ("pop flags") may change both ALU flags (e.g., ZF) and system flags (e.g., IF, which controls interrupt delivery). For a deprivileged guest, we need kernel mode `popf` to trap so that the VMM can emulate it against the virtual IF. Unfortunately, a deprivileged `popf`, like any user-mode `popf`, simply suppresses attempts to modify IF; no trap happens.

Obstacles of virtual machines on x86

- Visibility of privileged state
  - Guest OS can see it's de-privileged
  - x86 stores the current privileged level in %cs register
- Dual-mode instructions
  - Lack of traps when privileged instructions run at user-level
    - popf instruction can change IF and ZF flags
    - deprivileged popf instruction can only change ZF, but not IF — popf doesn't trap

A Comparison of Software and Hardware Techniques for x86 Virtualization

Keith Adams and Ole Agesen
VMware

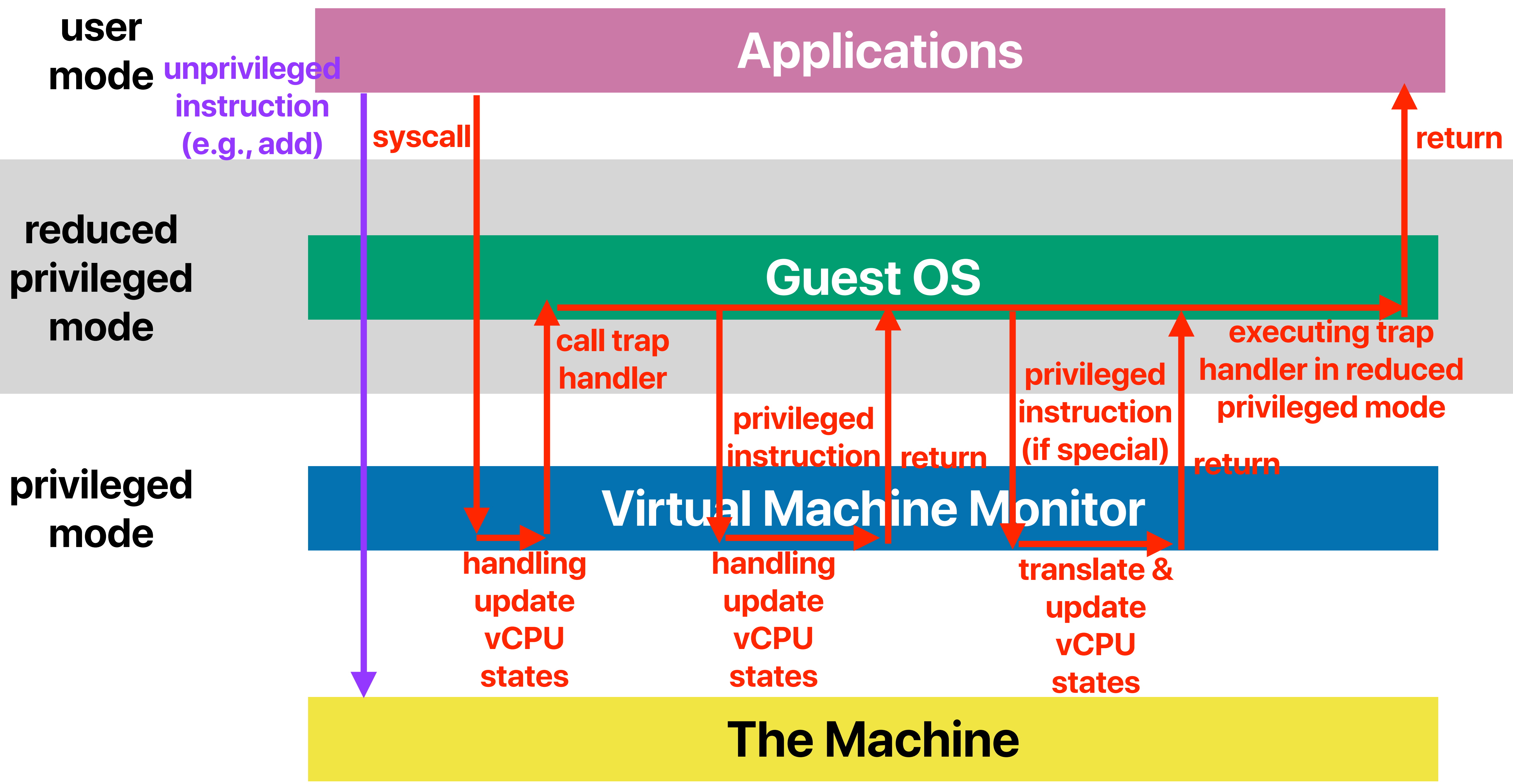
Binary translator

- Binary
- Dynamic
- On demand
- System level
- Subsetting
- Adaptive

Binary translation on x86

- If the virtualized CPU is in user mode
  - Instructions execute directly
- If the virtualized CPU is in kernel mode
  - VMM examines every instruction that the guest OS is about to execute in the near future by prefetching and reading instructions from the current program counter
  - Non-special instructions run natively
  - Special instructions (those instruction may have missing flags set) are "translated" into equivalent instructions with flags set

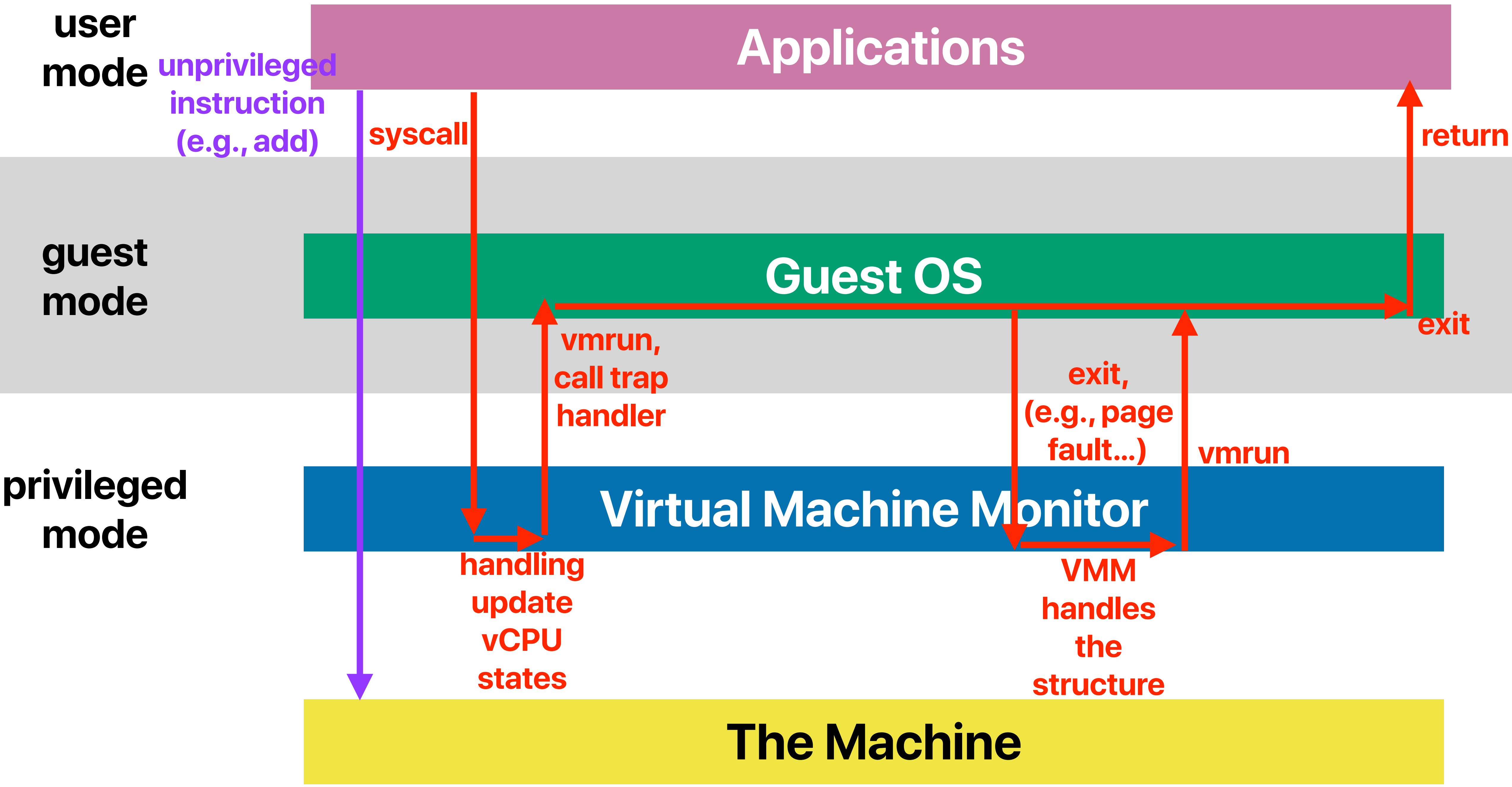
Trap-and-emulate with Binary Translation



Hardware virtualization in modern x86

- VMCB (Virtual machine control block)
  - Settings that determine what actions cause the guest to exit to host
  - All CPU state for a guest is located in VMCB data-structure
- A new, less privileged execution mode, guest mode
  - `vmrun` instruction to enter VMX mode
  - Many instructions and events cause VMX exits
  - Control fields in VMCB can change VMX exit behavior

Trap-and-emulate with hardware virtualization



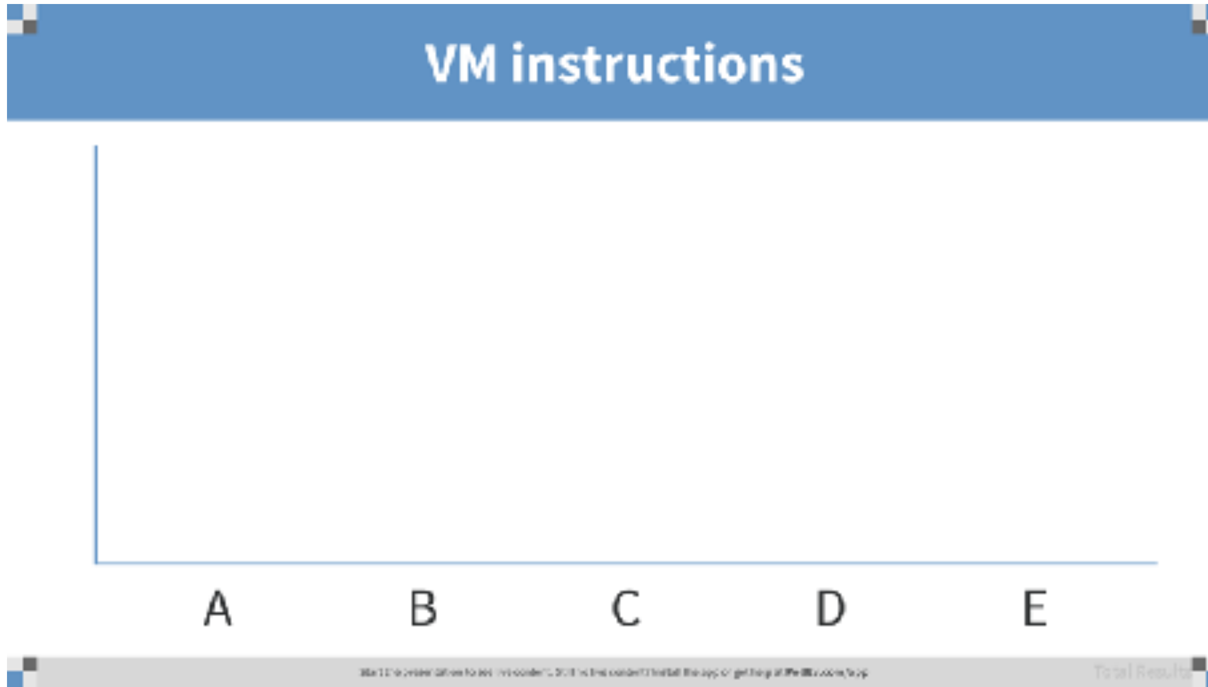
How hardware VM works

- VMM fills in VMCB exception table for Guest OS
  - Sets bit in VMCB not exit on syscall exception
- VMM executes vmrun
- Application invokes syscall
- CPU —> CPL #0, does not trap, vectors to VMCB exception table

https://www.pollev.com/hungweitseng close in 1:00

When to use hardware support for VM

- How many of the following situations can x86 VMX/VT-X instruction set extensions help improve the performance of VMM?
  ① Executing system calls
  ② Handling page faults
  ③ Modifying a page table entry
  ④ Calling a function
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4



When to use hardware support for VM

- How many of the following situations can x86 VMX/VT-X instruction set extensions help improve the performance of VMM?
  ① Executing system calls
  ② Handling page faults
  ③ Modifying a page table entry
  ④ Calling a function
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Virtualization overhead

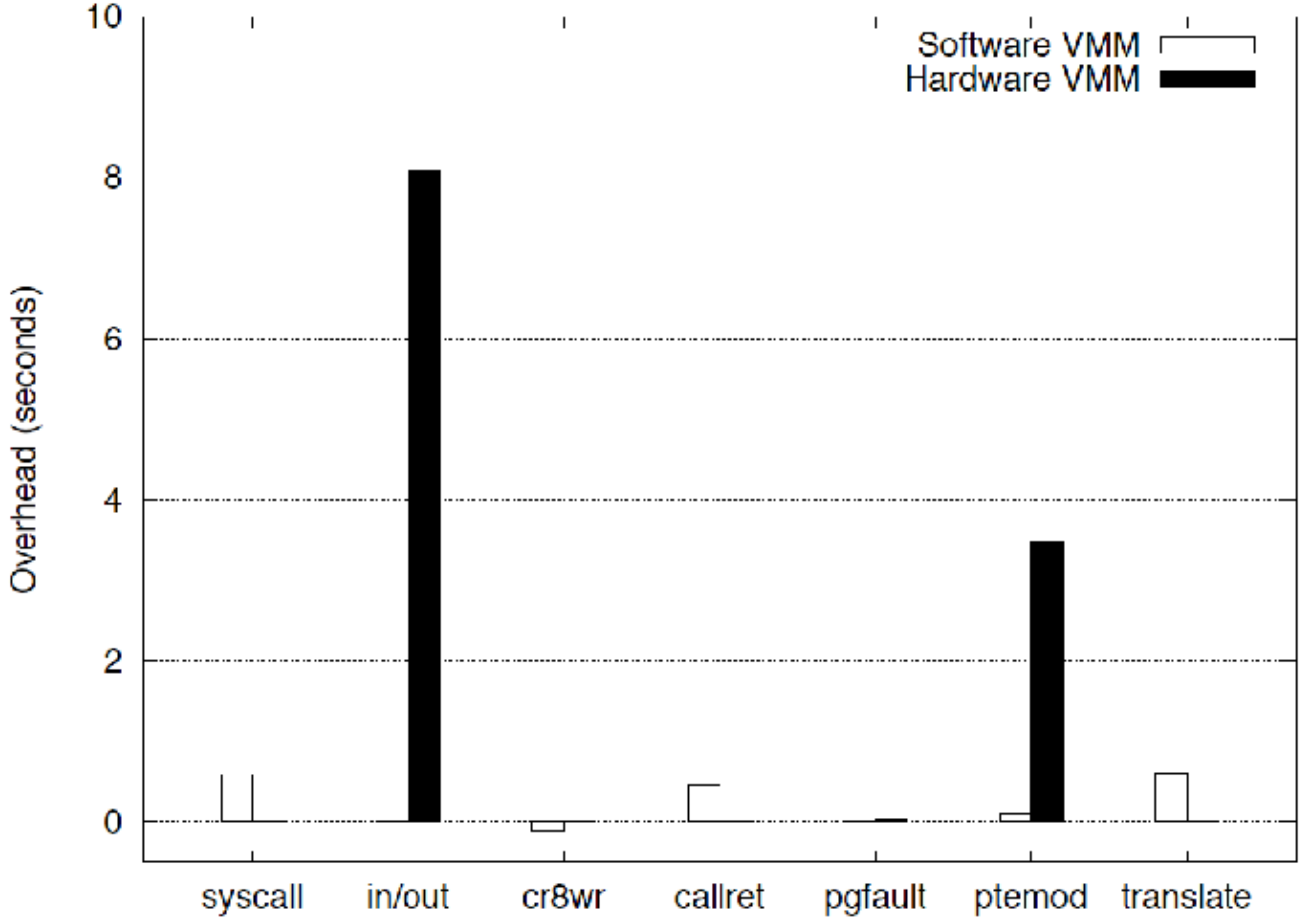


Figure 5. Sources of virtualization overhead in an XP boot/halt.

Nanobenchmarks

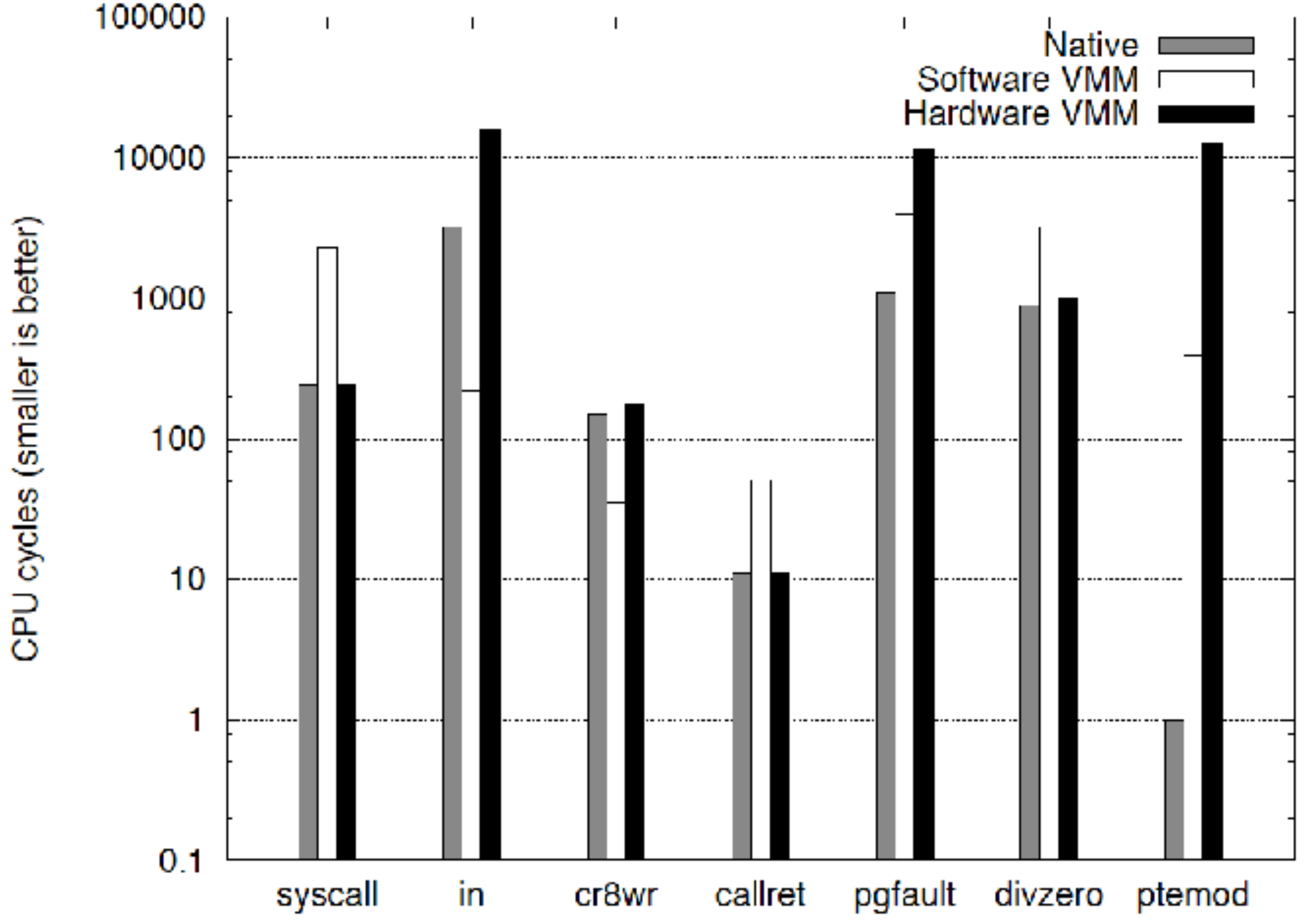


Figure 4. Virtualization nanobenchmarks.

Macrobenchmarks

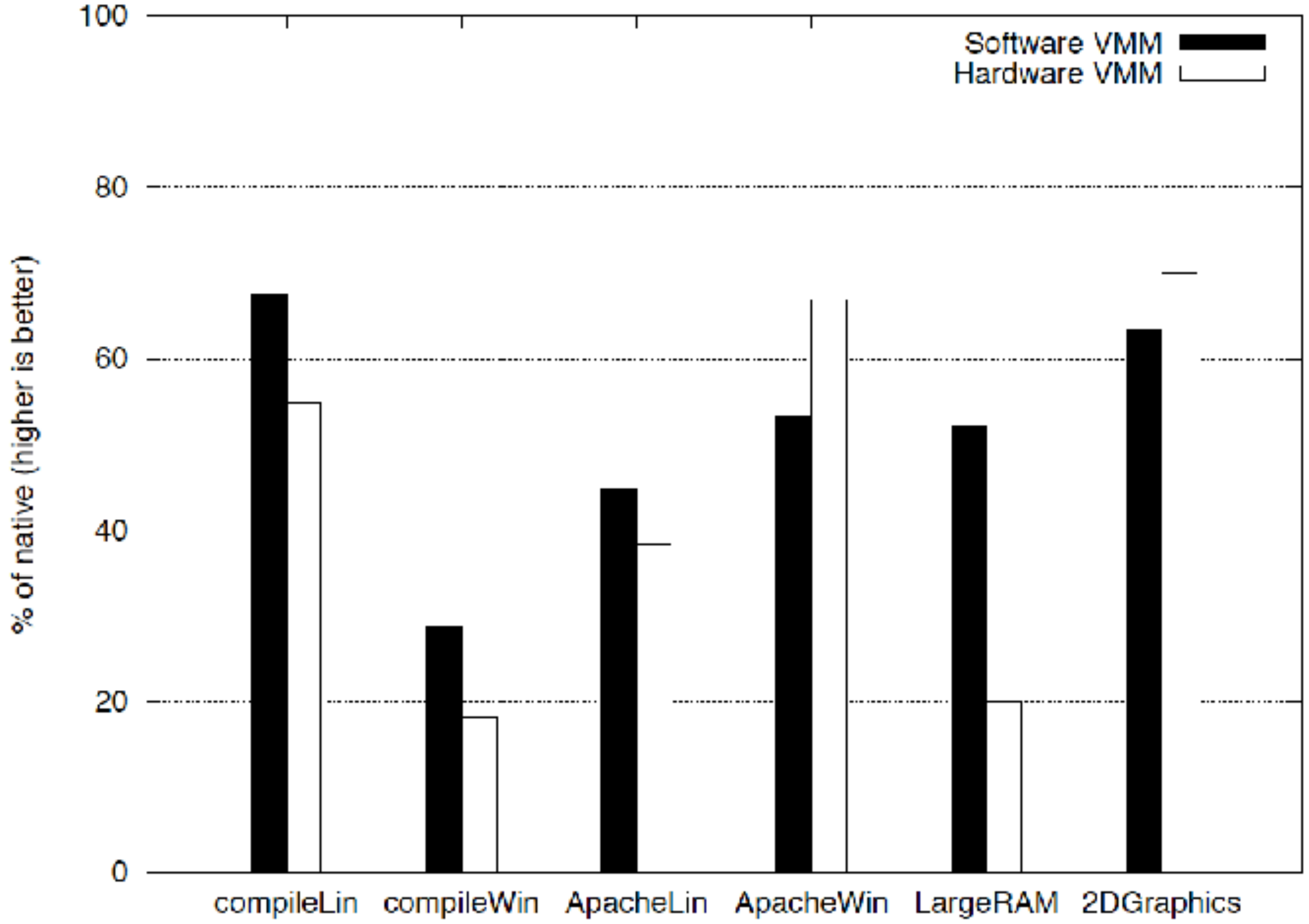


Figure 3. Macrobenchmarks.

When to use hardware support for VM

- How many of the following situations can x86 VMX/VT-X instruction set extensions help improve the performance of VMM?
  ① Executing system calls — guest OS runs in VM mode, no VMM intervention
  ② Handling page faults — software VMM doesn't need to use vmrun and exit
  ③ Modifying a page table entry
  ④ Calling a function — hardware VMM doesn't need BT
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

	3.8GHz P4 672	2.66GHz Core 2 Duo
VM entry	2409	937
Page fault VM exit	1931	1186
VMCB read	178	52
VMCB write	171	44

Table 1. Micro-architectural improvements (cycles).

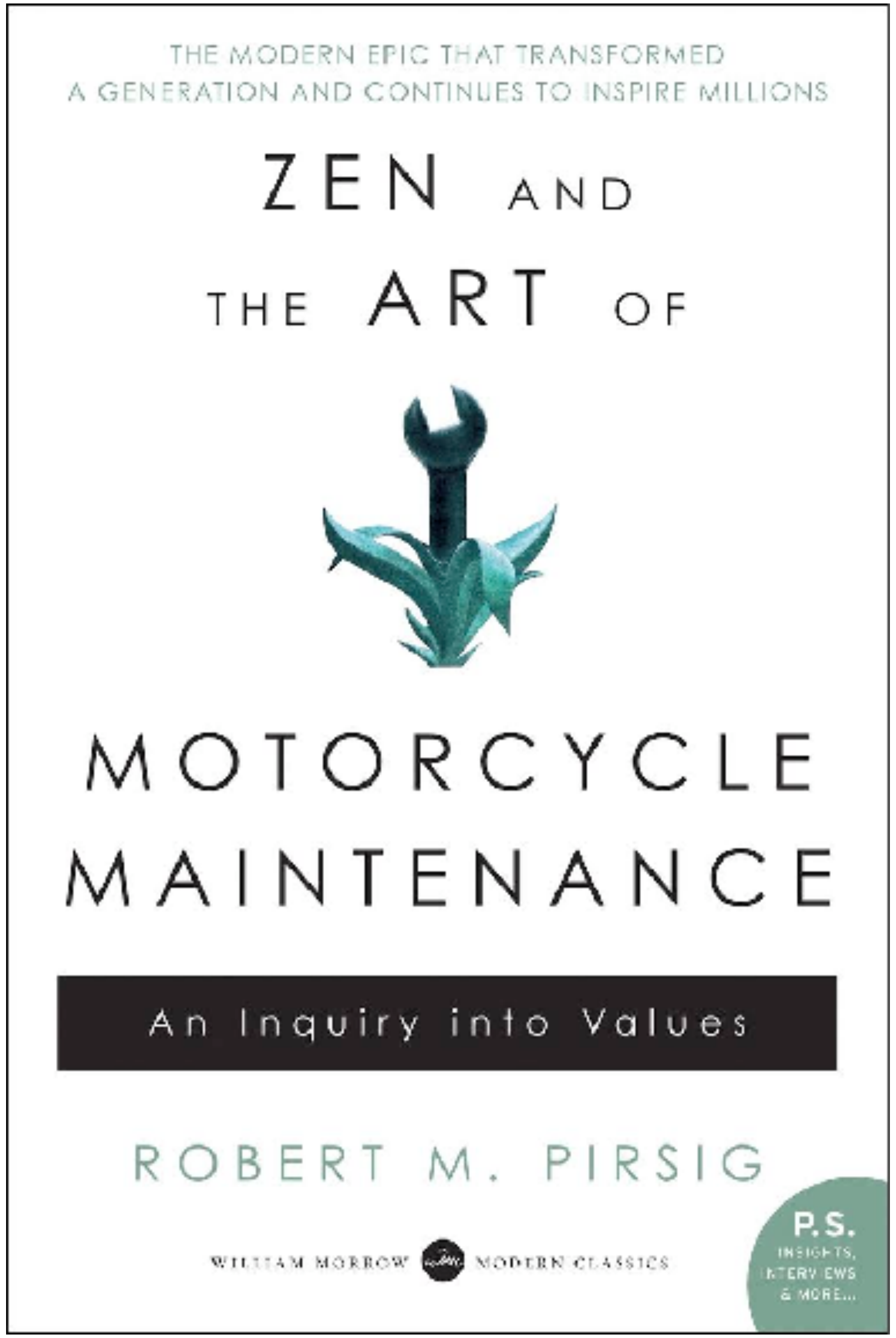
Side-by-side comparison

- Binary Translation VMM:
  - Converts traps to callouts
    - Callouts faster than trapping
  - Faster emulation routine
    - VMM does not need to reconstruct state
  - Avoids callouts entirely
- Hardware VMM:
  - Preserves code density
  - No precise exception overhead
  - Faster system calls

Xen and the Art of Virtualization

Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield
University of Cambridge Computer Laboratory

Why "Xen and the Art of Virtualization"?



Why Xen?

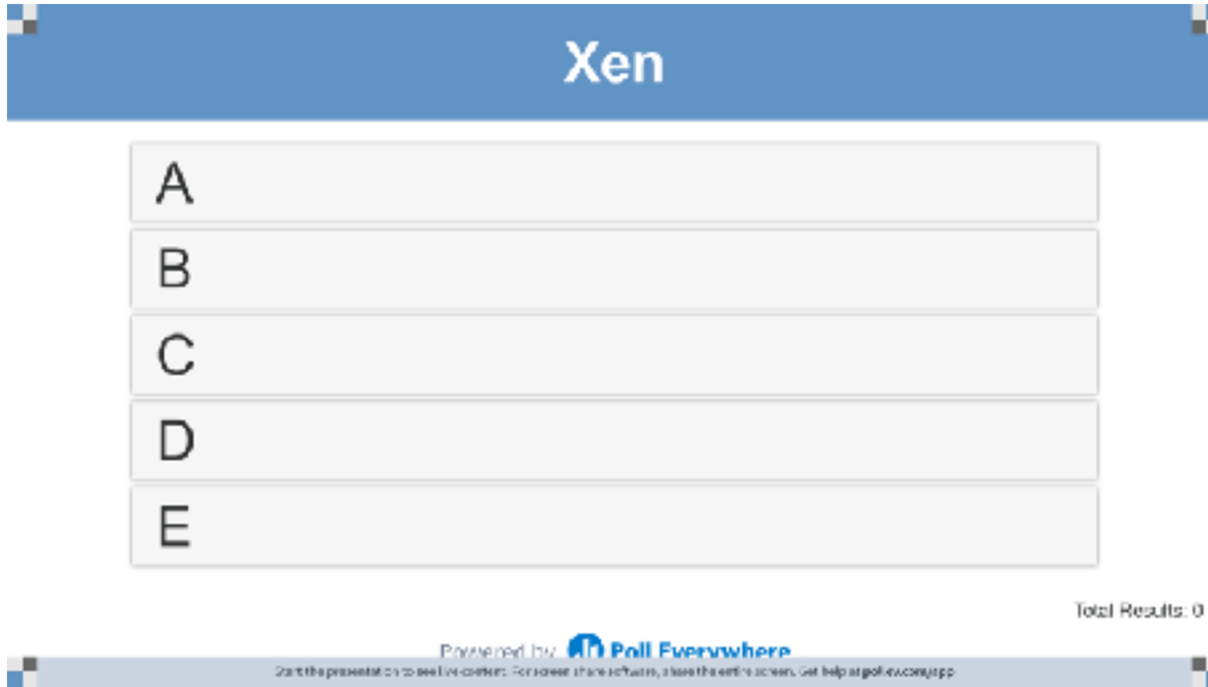
- Server consolidation: improve the server utilization
- Server co-location
- Secure distributed computing
- We want to host many full OS instances efficiently
  - The overhead of full virtualization/resource container is large
  - Hard to achieve Quality of Service guarantee because a VM is treated as a process in the host operating system

https://www.pollev.com/hungweitseng close in 1:00

What Xen proposed?

- Xen uses "para-virtualization" against "full-virtualization". Regarding para-virtualization, please identify how many of the following statements is/are correct.
  ① Para-Virtualization requires guest OSes and applications to be modified
  ② Para-virtualization allows the guest OS to be correctly virtualized without binary translations in VMM
  ③ Para-virtualization allows the guest OS to better support time-sensitive tasks
  ④ Para-virtualization allows a guest OS to manage physical to machine address mapping directly

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4



What Xen proposed?

- Xen uses "para-virtualization" against "full-virtualization". Regarding para-virtualization, please identify how many of the following statements is/are correct.
  ① Para-Virtualization requires guest OSes and ~~applications~~ to be modified
  ② Para-virtualization allows the guest OS to be correctly virtualized without binary translations in VMM
  ③ Para-virtualization allows the guest OS to better support time-sensitive tasks
  ④ Para-virtualization allows a guest OS to manage physical to machine address mapping directly
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Xen hypervisor

user mode

Applications	Applications	Applications

reduced privileged mode (ring 1)

Modified OS	Modified OS	Modified OS

device emulation, virtualization

Para-Virtualized CPU	Para-Virtualized memory	Para-Virtualized storage	Para-Virtualized network	Para-Virtualized ...

Xen

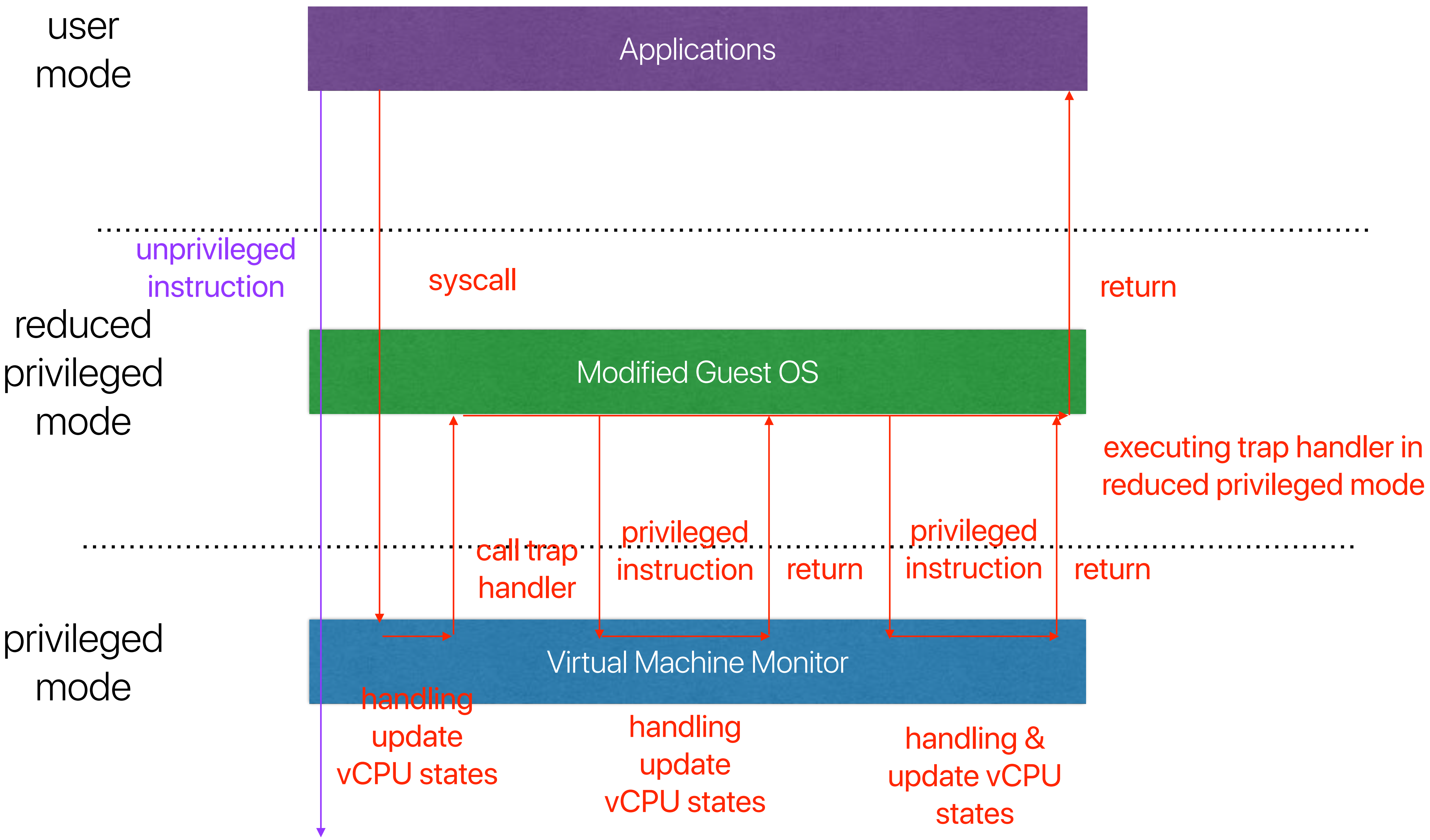
privileged mode

Paravirtualization

- Solution to issues with x86 instruction set
  - Don’t allow guest OS to issue sensitive instructions
  - Replace those sensitive instructions that don’t trap to ones that will trap
- Guest OS makes “hypercalls” (like system calls) to interact with system resources
  - Allows hypervisor to provide protection between VMs
- Exceptions handled by registering handler table with Xen
  - Fast handler for OS system calls invoked directly
  - Page fault handler modified to read address from replica location
- Guest OS changes largely confined to arch-specific code
  - Compile for ARCH=xen instead of ARCH=i686
  - Original port of Linux required only 1.36% of OS to be modified

Trap-and-emulate

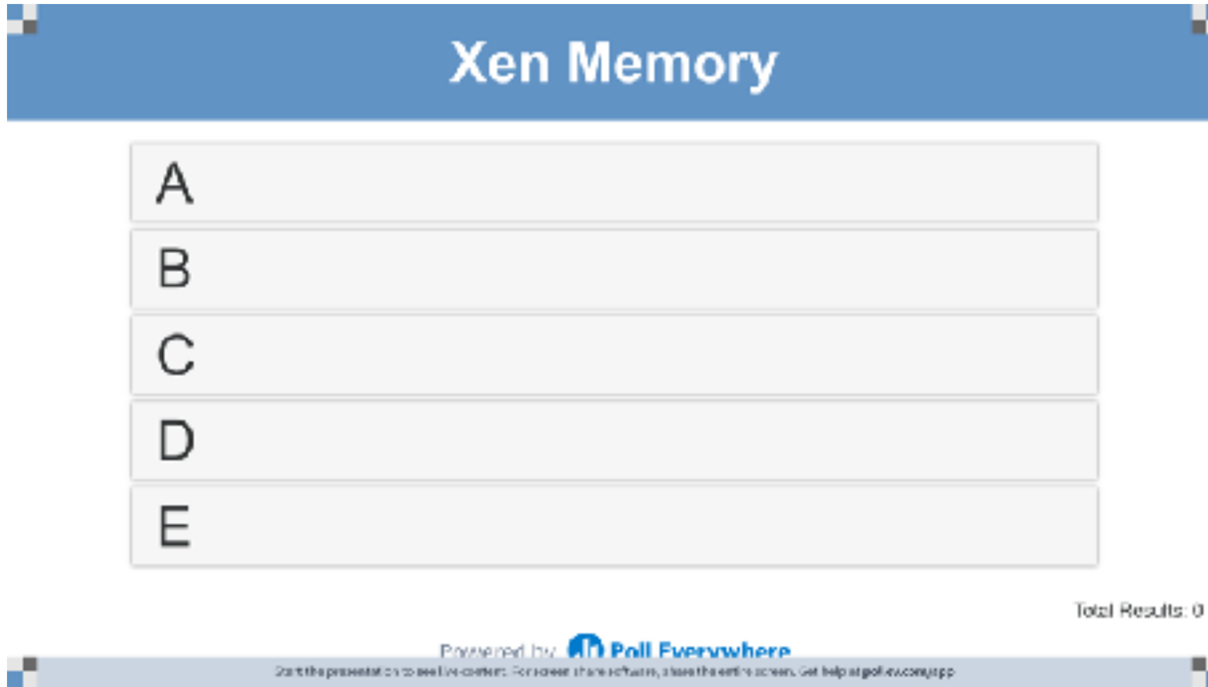
As we modified the OS code, no binary translation is necessary



https://www.pollev.com/hungweitseng close in 1:00

How para-virtualization work for memory allocation in Xen

- Regarding Xen's memory para-virtualization strategy, please identify how many of the following statements is/are correct
  - ① Switching guest OSes will trigger TLB flush
  - ② Xen is involved in and validated every page table update
  - ③ Xen must maintain a shadow table during page table updates
  - ④ x86 processors can directly access the page table in a guest OS of Xen
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4



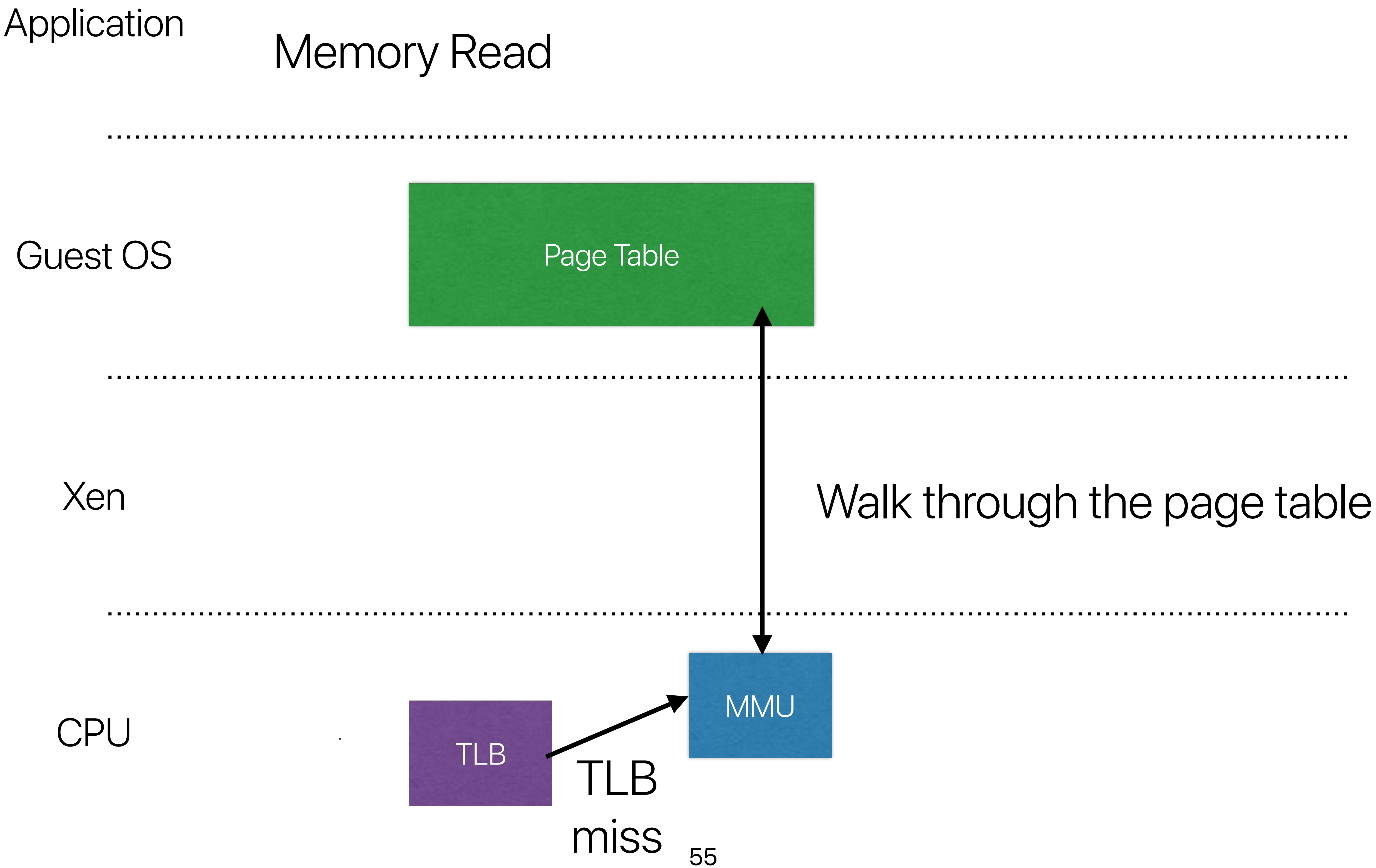
How para-virtualization work for memory allocation in Xen

- Regarding Xen's memory para-virtualization strategy, please identify how many of the following statements is/are correct
  - ① Switching guest OSes will trigger TLB flush
    Because x86 TLBs are not tagged (you don't have PIDs)
  - ② Xen is involved in and validated every page table update
    use hypercalls to achieve this — code modification is necessary
  - ③ Xen must ~~maintain a shadow table during page table updates~~
  - ④ x86 processors can directly access the page table in a guest OS of Xen
    avoid the usage of shadow page table, reducing the overhead
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4

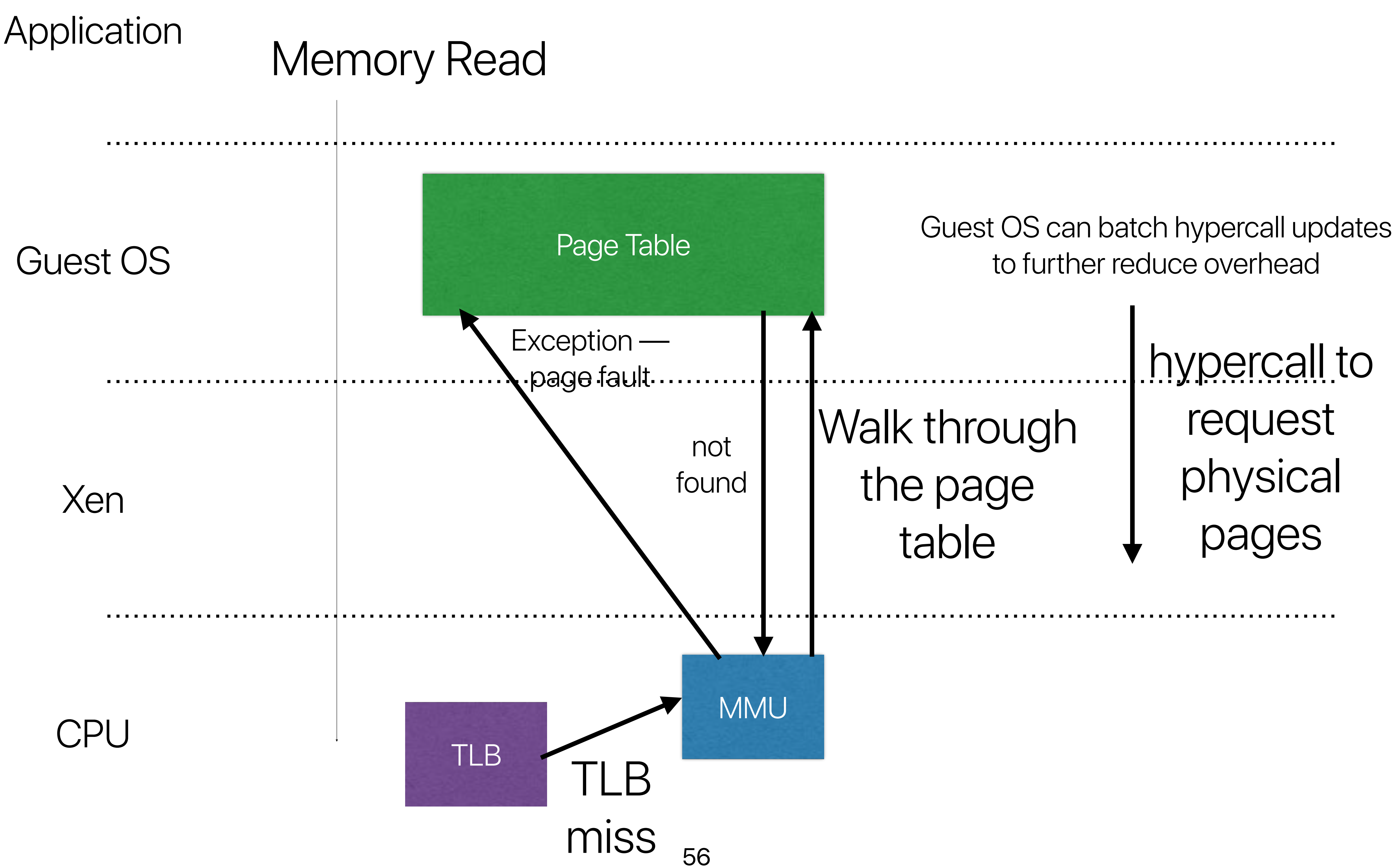
MMU Virtualization: Direct mode

- Modifying the guest OS to be involved only for page table updates
- Restricting the guest OS to have only read access
- Writing to page tables is protected and must use a hypercall — Xen can verify and allocate pages

Accessing a page — TLB miss



Accessing a page — page fault



Balloon driver

- Mechanism that forces guest OS to give up memory
- Balloon driver consumes physical memory allocated in the guest OS
- The memory consumed by Balloon is given to Xen
- The guest OS uses hypercalls to see and change the state

I/O virtualization

- Exposes I/O devices as asynchronous I/O rings to guest OS
- Exposes the device abstraction to minimize the change in device drivers
- Xen pins a few physical memory as DMA buffers and exposes to the guest OS to avoid copying overhead
- Use an up call to notify the guest OS as opposed to interrupts

Network virtualization

- Virtual firewall for each physical network interface
- Virtual interface for each physical network interface in each guest OS
- Circular Queue — Mechanism supporting I/O between Xen and guest OSes
  - Ring buffers for exchanging requests
  - Producer-consumer problem
    - Producers: guest OSes
    - Consumer: Xen

Performance

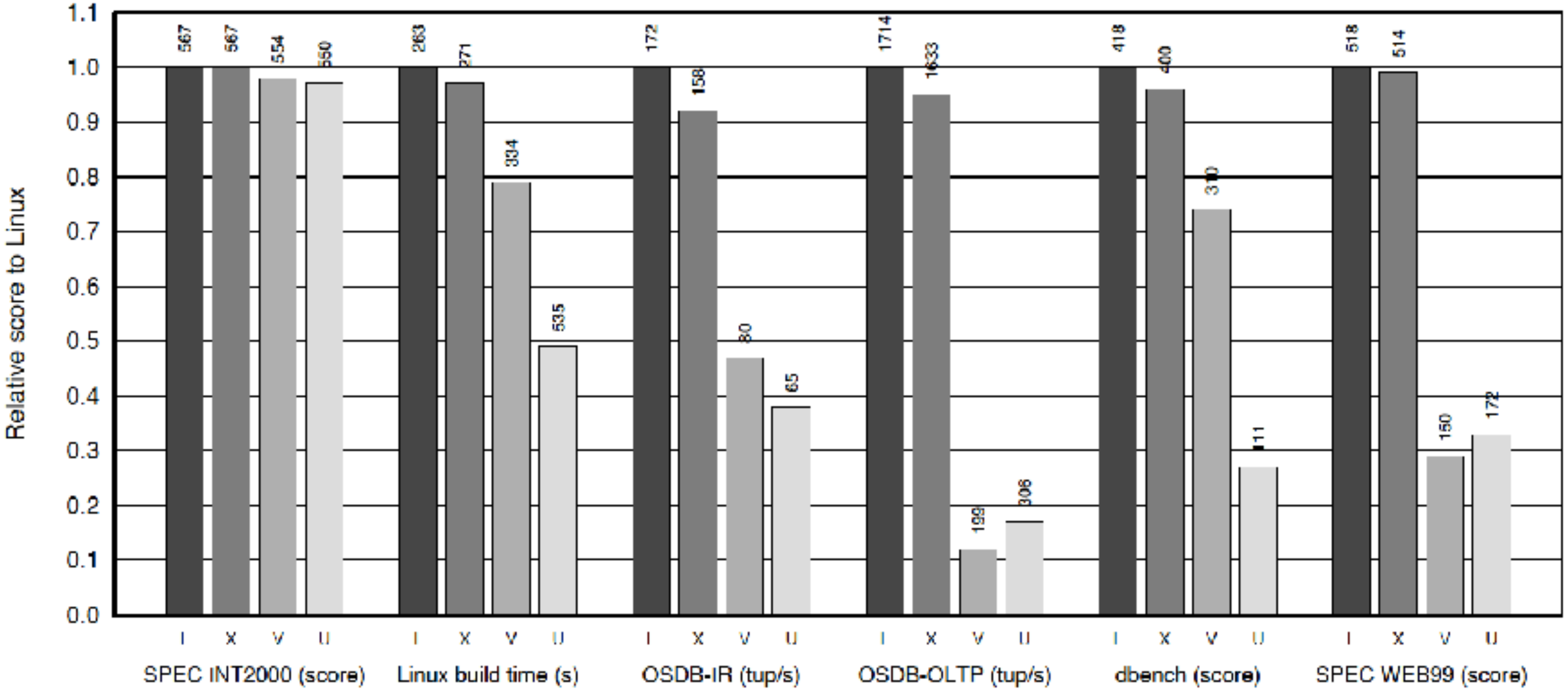


Figure 3: Relative performance of native Linux (L), XenoLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).

Overhead

Config	null call	null I/O	stat	open close	slct TCP	sig inst	sig hndl	fork proc	exec proc	sh proc
L-SMP	0.53	0.81	2.10	3.51	23.2	0.83	2.94	143	601	4k2
L-UP	0.45	0.50	1.28	1.92	5.70	0.68	2.49	110	530	4k0
Xen	0.46	0.50	1.22	1.88	5.69	0.69	1.75	**198**	**768**	**4k8**
VMW	0.73	0.83	1.88	2.99	11.1	1.02	4.63	874	2k3	10k
UML	24.7	25.1	36.1	62.8	39.9	26.0	46.0	21k	33k	58k

Table 3: **lmbench**: Processes - times in μs

Config	2p 0K	2p 16K	2p 64K	8p 16K	8p 64K	16p 16K	16p 64K
L-SMP	1.69	1.88	2.03	2.36	26.8	4.79	38.4
L-UP	0.77	0.91	1.06	1.03	24.3	3.61	37.6
Xen	**1.97**	**2.22**	**2.67**	**3.07**	**28.7**	**7.08**	39.4
VMW	18.1	17.6	21.3	22.4	51.6	41.7	72.2
UML	15.5	14.6	14.4	16.3	36.8	23.6	52.0

Table 4: **lmbench**: Context switching times in μs

Config	0K File create	0K File delete	10K File create	10K File delete	Mmap lat	Prot fault	Page fault
L-SMP	44.9	24.2	123	45.2	99.0	1.33	1.88
L-UP	32.1	6.08	66.0	12.5	68.0	1.06	1.42
Xen	32.5	5.86	68.2	13.6	**139**	1.40	**2.73**
VMW	35.3	9.3	85.6	21.4	620	7.53	12.4
UML	130	65.7	250	113	1k4	21.8	26.3

Table 5: **lmbench**: File & VM system latencies in μs

	TCP MTU 1500 TX	TCP MTU 1500 RX	TCP MTU 500 TX	TCP MTU 500 RX
Linux	897	897	602	544
Xen	897 (-0%)	897 (-0%)	516 (-14%)	467 (-14%)
VMW	291 (-68%)	615 (-31%)	101 (-83%)	137 (-75%)
UML	165 (-82%)	203 (-77%)	61.1(-90%)	91.4(-83%)

Table 6: **ttcp**: Bandwidth in Mb/s

Effort of porting

- Do you buy this?

OS subsection	# lines	
	Linux	**XP**
Architecture-independent	78	1299
Virtual network driver	484	–
Virtual block-device driver	1070	–
Xen-specific (non-driver)	1363	3321
Total	**2995**	**4620**
(Portion of total x86 code base	**1.36%**	**0.04%)**

Later evolution of Xen

- x86-64 removes ring 1, 2
  - Both applications and guest OSes in ring 3
  - Using guest mode in Intel VT-X/AMD VMX when necessary
- Higher performance NIC through segment offload
- Enhanced support for unmodified guest OSes using hardware virtualization
- Secure isolation between VMs

Hints for computer system design

Butler W. Lampson

Computer Science Laboratory Xerox Palo Alto Research Center

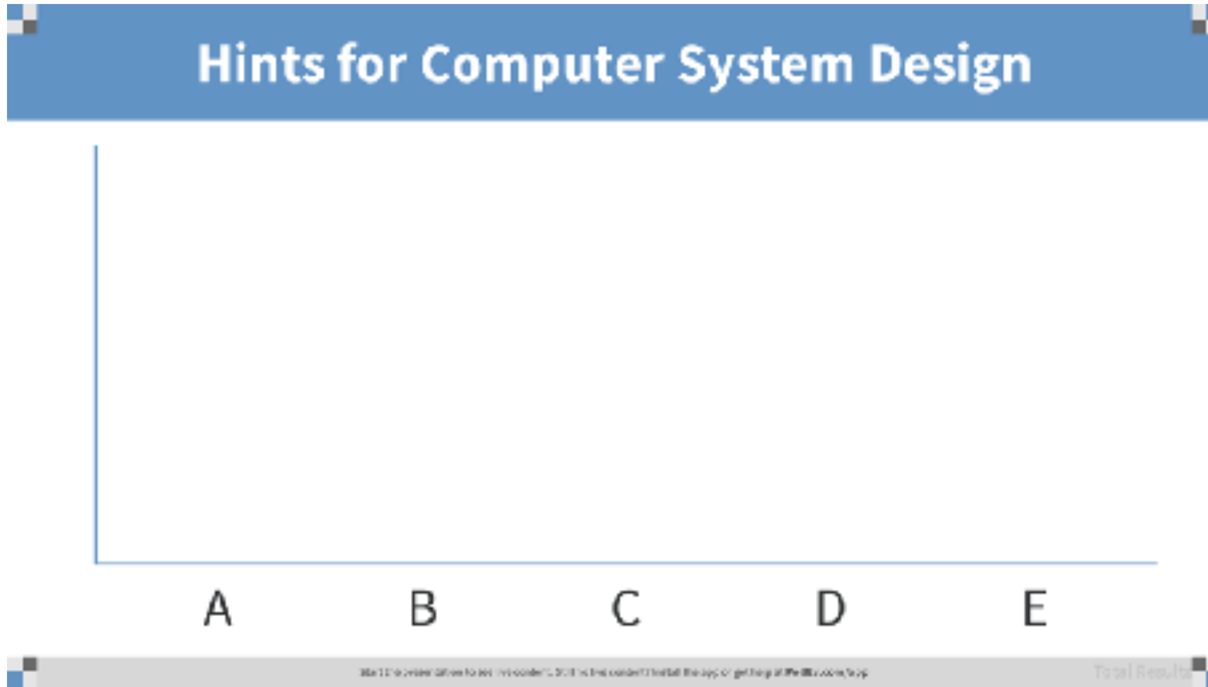
Hints for computer system design

Why?	*Functionality* Does it work?	*Speed* Is it fast enough?	*Fault-tolerance* Does it keep working?
Where?			
Completeness	Separate normal and worst case	Shed load	
		End to end	End to end
		Safety first	
Interface	Do one thing well:	Make it fast	End-to-end
	Don't generalize	Split resources	Log updates
	Get it right	Static analysis	Make actions atomic
	Don't hide power	Dynamic translation	
	Use procedure arguments		
	Leave it to the client		
	Keep basic interfaces stable		
	Keep a place to stand		
Implementation	Plan to throw one away	Cache answers	Make actions atomic
	Keep secrets	Use hints	Use hints
	Use a good idea again	Use brute force	
	Divide and conquer	Compute in background	
		Batch processing	

https://www.pollev.com/hungweitseng close in 1:00

Cloud storage and Lampson's paper

- How many of the following cloud storage system represents the idea of "Separate normal and worst case"
  ① Facebook's f4
  ② Google's GFS
  ③ Microsoft's Window Azure Storage
  ④ NetApp's NFS
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4



Cloud storage and Lampson's paper

- How many of the following cloud storage system represents the idea of "Separate normal and worst case"
  ① Facebook's f4
  ② Google's GFS
  ③ Microsoft's Window Azure Storage
  ④ NetApp's NFS
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Completeness

- Separate normal and worst case
- Make normal case fast
- The worst case must make progress
  - Saturation
  - Thrashing

Interface — Keep it simple, stupid

- Do one thing at a time or do it well
  - Don’t generalize
  - Example
    - Interlisp-D stores each virtual page on a dedicated disk page
      - 900 lines of code for files, 500 lines of code for paging
      - fast — page fault needs one disk access, constant computing cost
    - Pilot system allows virtual pages to be mapped to file pages
      - 11000 lines of code
      - Slower — two disk accesses in handling a page fault, under utilize the disk speed
- Get it right

More on Interfaces

- Make it fast, rather than general or powerful
  - CISC v.s. RISC
- Don’t hide power
  - Are we doing all right with FTL?
- Use procedure arguments to provide flexibility in an interface
  - Thinking about SQL v.s. function calls
- Leave it to the client
  - Monitors’ scheduling
  - Unix’s I/O streams

Implementation

- Keep basic interfaces stable
  - What happen if you changed something in the header file?
- Keep a place to stand if you do have to change interfaces
  - Mach/Sprite are both compatible with existing UNIX even though they completely rewrote the kernel
- Plan to throw one away
- Keep secrets of the implementation — make no assumption other system components
  - Don’t assume you will definitely have less than 16K objects!
- Use a good idea again
  - Caching!
  - Replicas
- Divide and conquer

Speed

- Split resources in a fixed way if in doubt, rather than sharing them
  - Processes
  - VMM: Multiplexing resources Guest OSs aren’t even aware that they’re sharing
- Use static analysis — compilers
- Dynamic translation from a convenient (compact, easily modified or easily displayed) representation to one that can be quickly interpreted is an important variation on the old idea of compiling
  - Java byte-code
  - LLVM
- Cache answers to expensive computations, rather than doing them over
- Use hints to speed up normal execution
  - The Ethernet: carrier sensing, exponential backoff

Speed

- When in doubt, use brute force
- Compute in background when possible
  - Free list instead of swapping out on demand
  - Cleanup in log structured file systems: segment cleaning could be scheduled at nighttime.
- Use batch processing if possible
  - Soft timers: uses trigger states to batch process handling events to avoid trashing the cache more often than necessary
  - Write buffers
- Safety first
- Shed load to control demand, rather than allowing the system to become overloaded
  - Thread pool
  - MLQ scheduling
  - Working set algorithm
  - Xen v.s. VMWare

Fault-tolerance

- End-to-end
  - Network protocols
- Log updates
  - Logs can be reliably written/read
  - Logs can be cheaply forced out to disk, which can survive a crash
    - Log structured file systems
    - RAID5 in Elephant
- Make actions atomic or restartable
  - NFS
  - atomic instructions for locks

Final Exam

Logistics

- Part 1 — any 80 hours you pick (starting from 3/11 12am — 3/17 11:59:00pm)
  - Two of the questions are considered as comprehensive exam
- Part 2 — time unlimited, starting from 8pm — 3/17 11:59:00pm
- Final is cumulative
- If you help others, you're hurting yourself
  - since grades are given according to your relative rank in the class.
  - I will directly send all skeptical cases to misconduct office

Part 1

- Free answer questions (2)
  - One about process, the other about virtual memory
  - Count as comprehensive exam questions as well
- Multiple choices (10) — like your midterm
- Multiple answer (5)

Sample Final Part 1

Latency Numbers Every Programmer Should Know (2020 Version)

Operations	Latency (ns)	Latency (us)	Latency (ms)	
L1 cache reference	0.5 ns			~ 1 CPU cycle
Branch mispredict	3 ns			
L2 cache reference	4 ns			14x L1 cache
Mutex lock/unlock	17 ns			
Send 2K bytes over network	44 ns			
Main memory reference	100 ns			20x L2 cache, 200x L1 cache
Compress 1K bytes with Zippy	2,000 ns	2 us		
Read 1 MB sequentially from memory	3,000 ns	3 us		
Read 4K randomly from SSD*	16,000 ns	16 us		
Read 1 MB sequentially from SSD*	49,000 ns	49 us		
Round trip within same datacenter	500,000 ns	500 us		
Read 1 MB sequentially from disk	825,000 ns	825 us		
Disk seek	2,000,000 ns	2,000 us	2 ms	4x datacenter roundtrip
Send packet CA-Netherlands-CA	150,000,000 ns	150,000 us	150 ms	

https://colin-scott.github.io/personal_website/research/interactive_latency.html

Overhead

- The latency of ?
  - A TLB miss?
  - A page fault?
  - A kernel switch?
  - A context switch?
- Under what condition will:
  - Saturation occur?
  - Thrashing occur?

Cylinder groups

- Which of the following factor of disk access can cylinder groups help to improve when manage files?
  A. Seek time
  B. Rotational delay
  C. Data transfer latency
  D. A and B
  E. A and C

Why do we need LFS?

- How many of the following problems will Log-structured file systems solve?
  ① The performance of small random writes
  ② The efficiency of large file accesses
  ③ The space overhead of metadata in the file system
  ④ Reduce the main memory space used by the file system
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Polling v.s. Interrupt

- Comparing polling and interrupt, how many of the following statements are true
  ① When interacting with high-speed device, using polling can achieve better performance
  ② Interrupt can improve CPU utilization if the device only needs service from the processor occasionally
  ③ Interrupt allows asynchronous I/O in programs
  ④ The overhead of handling an event after polling is higher than handling the same event after receiving an interrupt

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Large Chunks

- How many of the following statements can large chunks help address?
  1. Storage based on inexpensive disks that fail frequently
  2. Many large files in contrast to small files for personal data
  3. Primarily reading streams of data
  4. Sequential writes appending to the end of existing files
  5. Must support multiple concurrent operations
  6. Bandwidth is more critical than latency

  A. 1
  B. 2
  C. 3
  D. 4
  E. 5

flat structure in GFS

- How many of the following statements can flat file system structure help address in GFS?
  - ① Storage based on inexpensive disks that fail frequently
  - ② Many large files in contrast to small files for personal data
  - ③ Primarily reading streams of data
  - ④ Sequential writes appending to the end of existing files
  - ⑤ Must support multiple concurrent operations
  - ⑥ Bandwidth is more critical than latency
  - A. 1
  - B. 2
  - C. 3
  - D. 4
  - E. 5

What is a stream?

- Regarding a stream in WAS, please identify how many of the following statements is/are true
  ① A stream is a list of extents, in which an extent consists of consecutive blocks
  ② Each block in the stream contains a checksum to ensure the data integrity
  ③ An update to a stream can only be appended to the end of the stream
  ④ Two streams can share the same set of extents
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Google search architecture

- How many of the following fulfill the design agenda of the Google search architecture described in this paper?
  - ① Reduce the hardware cost by using commodity-class and unreliable PCs
  - ② Use RAID to provide efficiency and reliability
  - ③ Use replication for better request throughput and availability
  - ④ Optimize for the peak performance
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4

The role of the OS in virtual memory management

- How many of the following tasks in virtual memory management always requires the assistance of operating system?
  - ① Address translation
  - ② Growth of process address space
  - ③ Tracking free physical memory locations
  - ④ Maintaining mapping tables
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4

Why not microkernels?

- Although Mach's design strongly influenced modern operating systems, why most modern operating systems do not adopt the design of microkernels?
  A. Microkernels are more difficult to extend than monolithic kernels
  B. Microkernels are more difficult to maintain than monolithic kernels
  C. Microkernels are less stable than monolithic kernels
  D. Microkernels are not as competitive as monolithic kernels in terms of application performance
  E. Microkernels are less flexible than monolithic kernels

Protection

- Regarding the protection in UNIX, how many of the followings is/are correct?
  ① The same file may have different permissions for different user-id
  ② The owner of the file may not have the permission of writing a file
  ③ If the user does not have a permission to access a device, set-user-id will guarantee that the user will not be able to access that device
  ④ In the UNIX system described in this paper, if the file owner is "foo", then the user "bar" will have the same permission as another user (e.g. "xyz").

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Multiple answers

- Which paper(s) is(are) designing FS for read-intensive data access?
- Which paper(s) is(are) designing FS for write-intensive data access?
- Which paper(s) is(are) designing FS for MapReduce?
- What is saturation? What paper talks about it?
- **Can you relate papers with Butler Lampson's "Hints for Computer System Design"?**
  - Caching?
  - Batch processing?
  - Atomic operations?
  - Logs?
  - Separate normal and worst case

	Title
A	The Structure of the 'THE'-Multiprogramming System
B	The Nucleus of a Multiprogramming System
C	The UNIX Time-Sharing System
D	Mach: A New Kernel Foundation For UNIX Development
E	An experimental time-sharing system
F	The Linux Scheduler: a Decade of Wasted Cores
G	Virtual Memory Management in VAX/VMS
H	Machine-Independent Virtual Memory Management for Paged Uniprocessor and Multiprocessor Architectures
I	Converting a Swap-Based System to do Paging in an Architecture Lacking Page-Reference Bits
J	WSCLOCK-A Simple and Effective Algorithm for Virtual Memory Management
K	A Fast File System for Unix
L	The Design and Implementation of a Log-Structured File System
M	eNVy: a non-volatile, main memory storage system
N	Don't stack your log on my log
O	The Google File System
P	MapReduce: Simplified Data Processing on Large Clusters
Q	Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency
R	f4: Facebook's Warm BLOB Storage System
S	Web Search for a Planet: The Google Cluster Architecture
T	A comparison of software and hardware techniques for x86 virtualization
U	Wait-Free Synchronization
V	RCU Usage In the Linux Kernel: Eighteen Years Later

Part 2

- Brainstorming questions *6 problem sets— research questions, design decisions. Not actually having a standard answer
  - Keep it short but hit the point
  - You may get negative feedback if you provide irrelevant information
  - If you're asked to make a design decision, make sure you cover the following aspects
    - Why your choice makes sense to the problem asked/needs to be addressed
    - Why **other listed options are not competitive** as your choice

Announcement

- iEVAL
  - We highly value your opinions
  - Submit your screenshot of confirmation, equivalent to a full-credit reading quiz
- Revision
  - All slots are taken
  - If you cannot make a reservation, e-mail me and the TA
  - You still have to submit your code on time by tomorrow
- Check your grades on eLearn as soon as possible
  - We drop 6 of your lowest reading quizzes
  - Participation will show up as 4 reading quizzes

Thank you all for this great quarter!