OS Virtualization

CSC 456 Final Presentation

Brandon D. Shroyer

Introduction

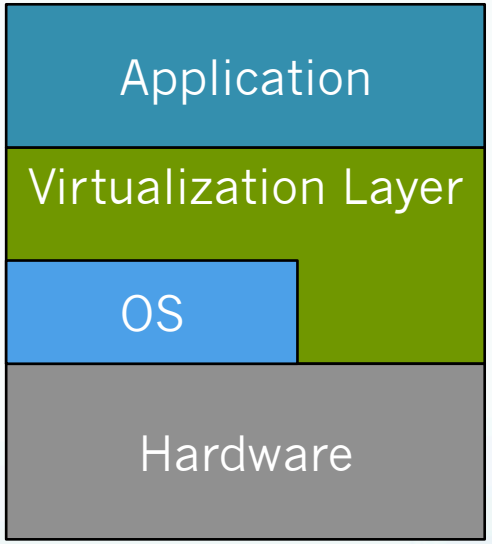
- *Virtualization*: Providing an interface to software that maps to some underlying system.
  - A one-to-one mapping between a guest and the host on which it runs [9, 10].
- Virtualized system should be an “efficient, isolated duplicate” [8] of the real one.
- *Process virtual machine* just supports a process; *system virtual machine* supports an entire system.

Why Virtualize?

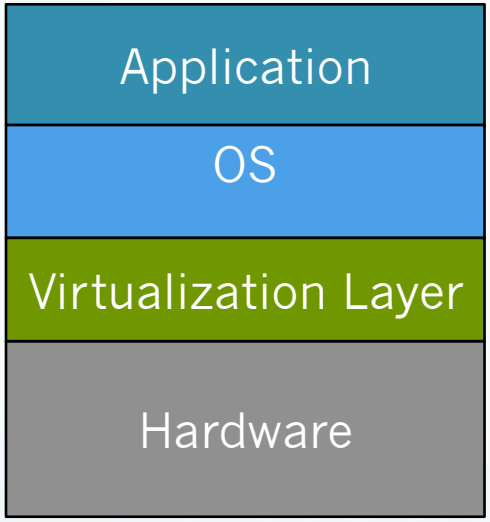
- Reasons for Virtualization
  - Hardware Economy
  - Versatility
  - Environment Specialization
  - Security
  - Safe Kernel Development
  - OS Research [12]

Process Virtualization

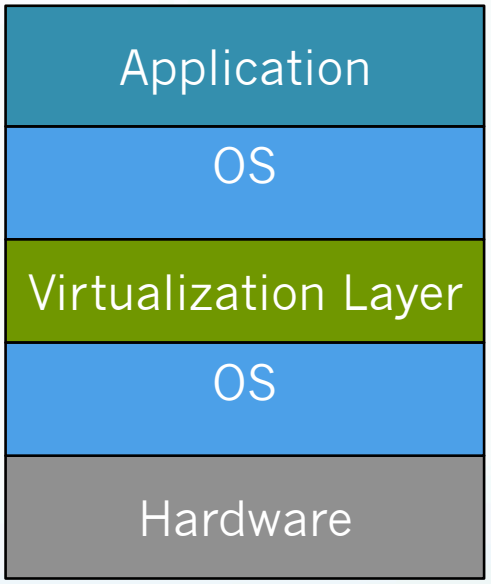
- VM interfaces with single process
- Application sees “virtual machine” as address space, registers, and instruction set [10].
- Examples:
  - Multiprogramming
  - Emulation for binaries
  - High-level language VMMs (e.g., JVM)



System Virtualization



Classical Virtualization



Hosted Virtualization/
Emulation

System Virtualization

- Interfaces with operating system
- OS sees VM as an actual machine—memory, I/O, CPU, etc [10].
- *Classic virtualization*: virtualization layer runs atop the hardware.
  - Usually found on servers (Xen, VMWare ESX)
- *Hosted or whole-system virtualization*: virtualization runs on an operating system
  - Popular for desktops (VMWare Workstation, Virtual PC)

Emulation

- Providing an interface to a system so that it can run on a system with a different interface [10].
  - Lets compiled binaries, OSes run on architectures with different ISA (binary translation)
  - Performance usually worse than classic virtualization.
- Example: QEMU [11]
  - Breaks CPU instructions into small ops, coded in C.
  - C code is compiled into small objects on native ISA.
  - dyngen utility runs code by dynamically stitching objects together (dynamic code generation).

Some Important Terms

- *Virtual Machine (VM)*: An instance of of an operating system running on a virtualized system. Also known as a *virtual* or *guest OS.*

- *hypervisor:* The underlying virtualization system sitting between the guest OSes and the hardware. Also known as a *Virtual Machine Monitor (VMM).*

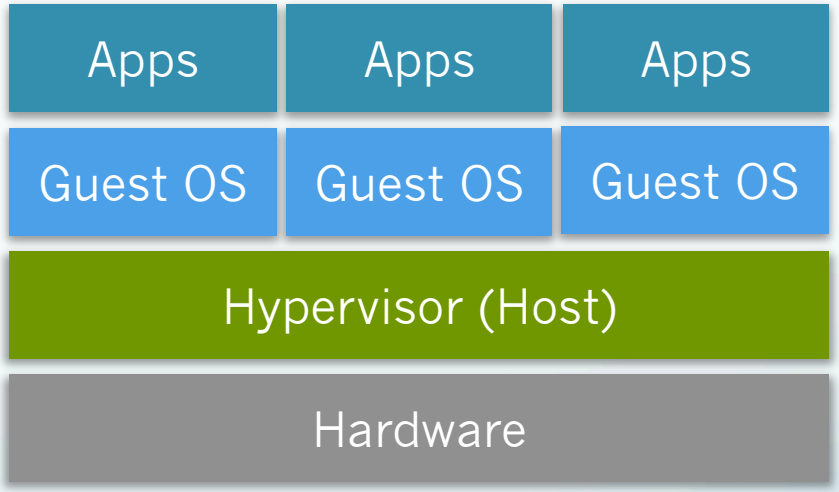
Requirements of a VMM

Developed by Popek & Goldberg in 1974 [8]:

1. Provides environment identical to underlying hardware.
2. Most of the instructions coming from the guest OS are executed by the hardware without being modified by the VMM.
3. Resource management is handled by the VMM (this all non-CPU hardware such as memory and peripherals).

Guest OS Model

- Hypervisor exists as a layer between the operating systems and the hardware.
- Performs memory management and scheduling required to coordinate multiple operating systems.
- May also have a separate controlling interface.



Virtualization Challenges

- Privileged Instructions
  - Handling architecture-imposed instruction privilege levels.
- Performance Requirements
  - Holding down the cost of VMM activities.
- Memory Management
  - Managing multiple address spaces efficiently.
- I/O Virtualization
  - Handling I/O requests from multiple operating systems.

Virtualizing Privileged Instructions

- x86 architecture has four privilege levels (rings).
- The OS assumes it will be executing in Ring 0.
- Many system calls require 0-level privileges to execute.
- Any virtualization strategy must find a way to circumvent this.

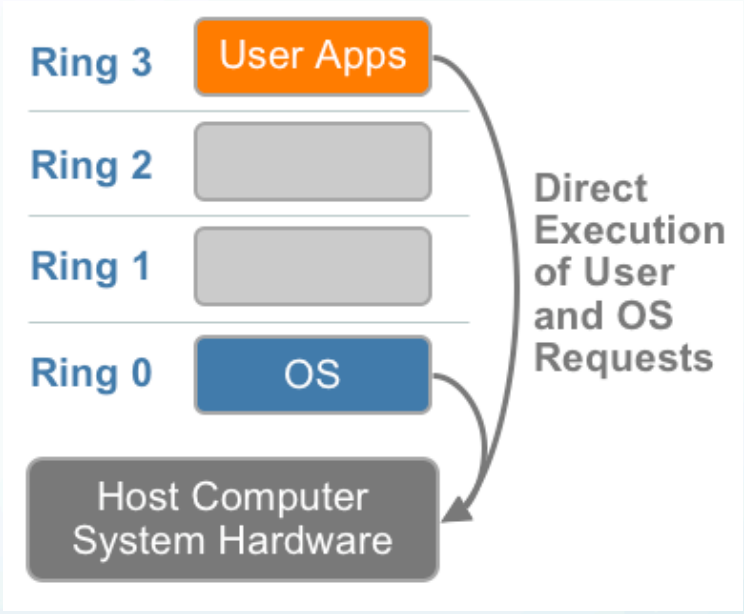


Image Source: VMWare White Paper, "Understanding Full Virtualization, Paravirtualization, and Hardware Assist", 2007.

Full Virtualization

- "Hardware is functionally identical to underlying architecture." [3]
- Typically accomplished through interpretation or binary translation.
- Advantage: Guest OS will run without any changes to source code.
- Disadvantage: Complex, usually slower than paravirtualization.

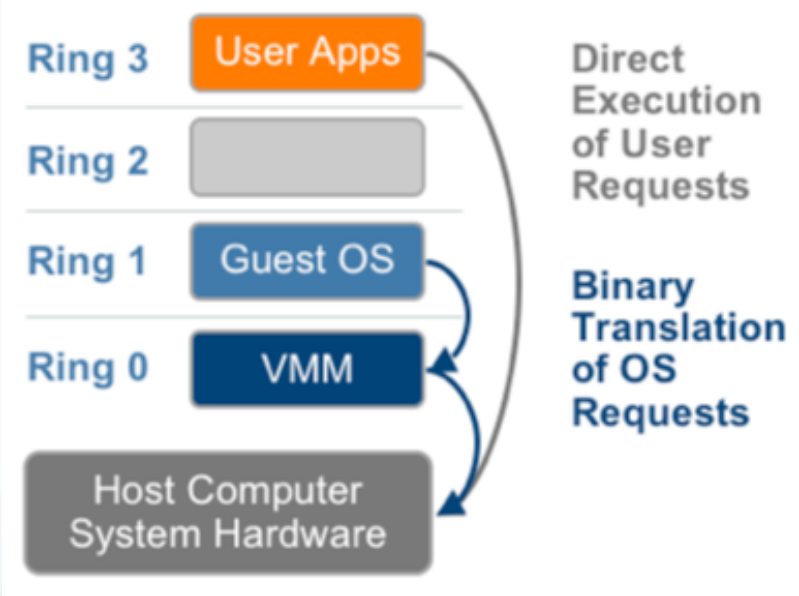


Image Source: VMWare White Paper, "Understanding Full Virtualization, Paravirtualization, and Hardware Assist", 2007.

Paravirtualization

- Replace certain unvirtualized sections of OS code with virtualization-friendly code.
- Virtual architecture "similar but not identical to the underlying architecture." [3]
- Advantages: easier, lower virtualization overhead
- Disadvantages: requires modifications to guest OS

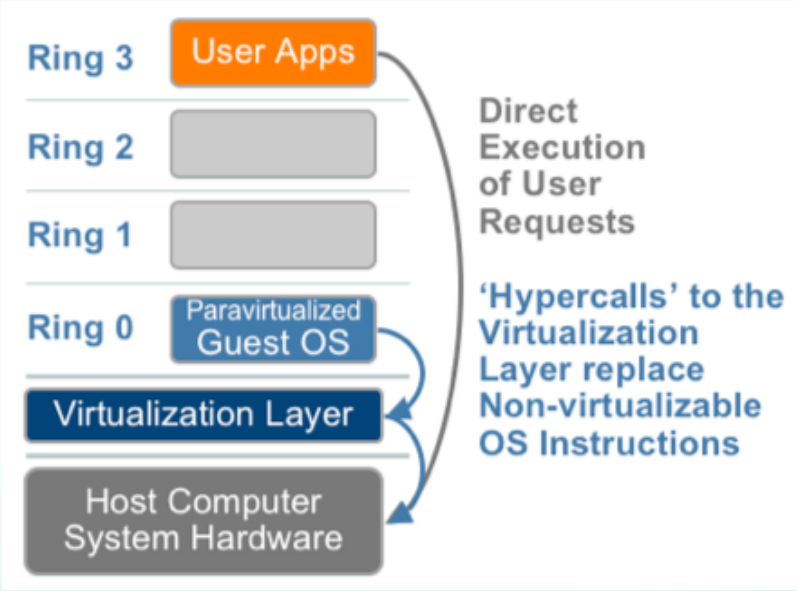
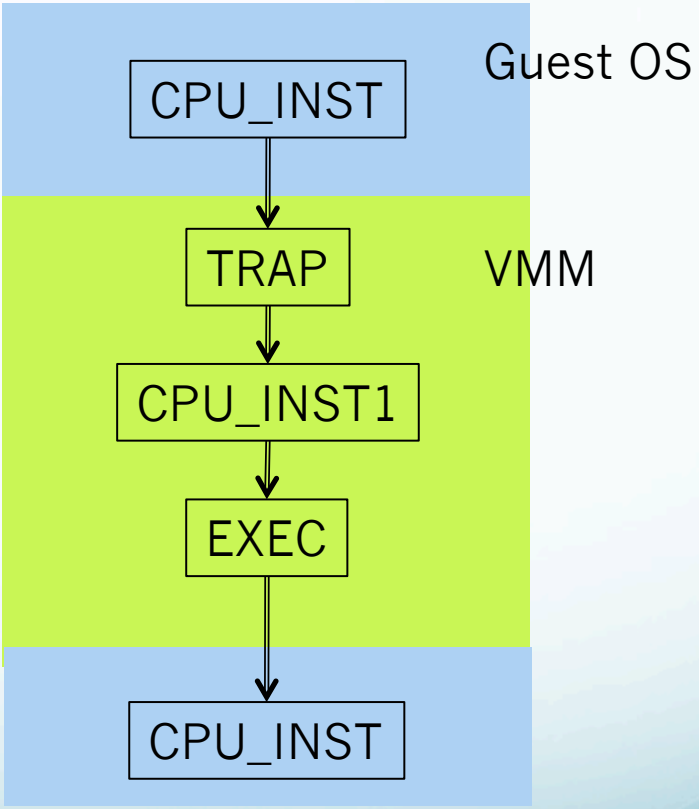


Image Source: VMWare White Paper, "Understanding Full Virtualization, Paravirtualization, and Hardware Assist", 2007.

Performance

- Modern VMMs based around trap-and-emulate [8].
- When a guest OS executes a privileged instruction, control is passed to VMM (VMM “traps” on instruction), which decides how to handle instruction [8].
- VMM generates instructions to handle trapped instruction (emulation).
- Non-privileged instructions do not trap (system stays in guest context).



Trap-and-Emulate Problems

- Trap-and-emulate is expensive
  - Requires context-switch from guest OS mode to VMM.
- x86 is not trap-friendly
  - Guest's CPL privilege level is visible in hardware registers; cannot change it in a way that the guest OS cannot detect [5].
  - Some instructions are not privileged, but access privileged systems (page tables, for example) [5].

VMWare Virtualization

- Full virtualization implemented through dynamic binary translation [5].
  - Translated code is grouped and stored in translation caches (TCs).
  - Callout method replaces traps with stored emulation functions.
  - In-TC emulation blocks are even more efficient.
  - Adaptive binary translation rewrites translated blocks to minimize PTE traps [5].
- Direct execution of user-space code further reduces overhead [5].

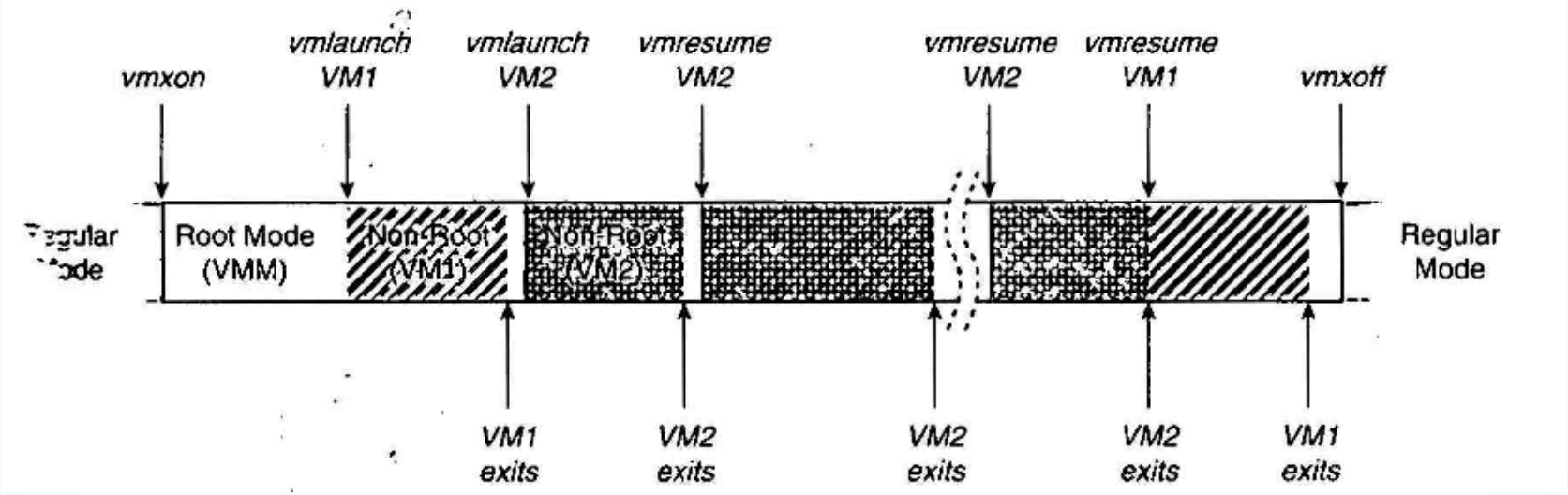
Xen Virtualization

- Xen occupies privilege level 0; guest OS occupies privilege level 1.
- OS code is modified so that high-privilege calls (hypercalls) are made to and trapped by Xen [3].
- Xen traps guest OS instructions using table of exception handlers.
  - Frequently used handlers (e.g., system calls) have special handlers that allow guest OS to bypass privilege level 0 [3].
  - Approach does not work with page faults.
- Handlers are vetted by Xen before being stored.

Hardware-Assisted Virtualization

- Hardware virtualization-assist released in 2006 [5].
  - Intel, AMD both have technologies of this type.
- Introduces new VMX runtime mode.
  - Two modes: guest (for OS) and root (for VMM).
  - Each mode has all four CPL privilege levels available [8].
  - Switching from guest to VMM does not require changes in privilege level.
  - Root mode supports special VMX instructions.
  - Virtual machine control block [5] contains control flags and state information for active guest OS.
  - New CPU instructions for entering and exiting VMM mode.
- Does not support I/O virtualization.

Intel VT-X



- Both modes have no restrictions on privilege
  - No need for software-based deprivileging

Image Source: Smith, J. and Nair, R. *Virtual Machines*, Morgan Kaufmann, 2005.

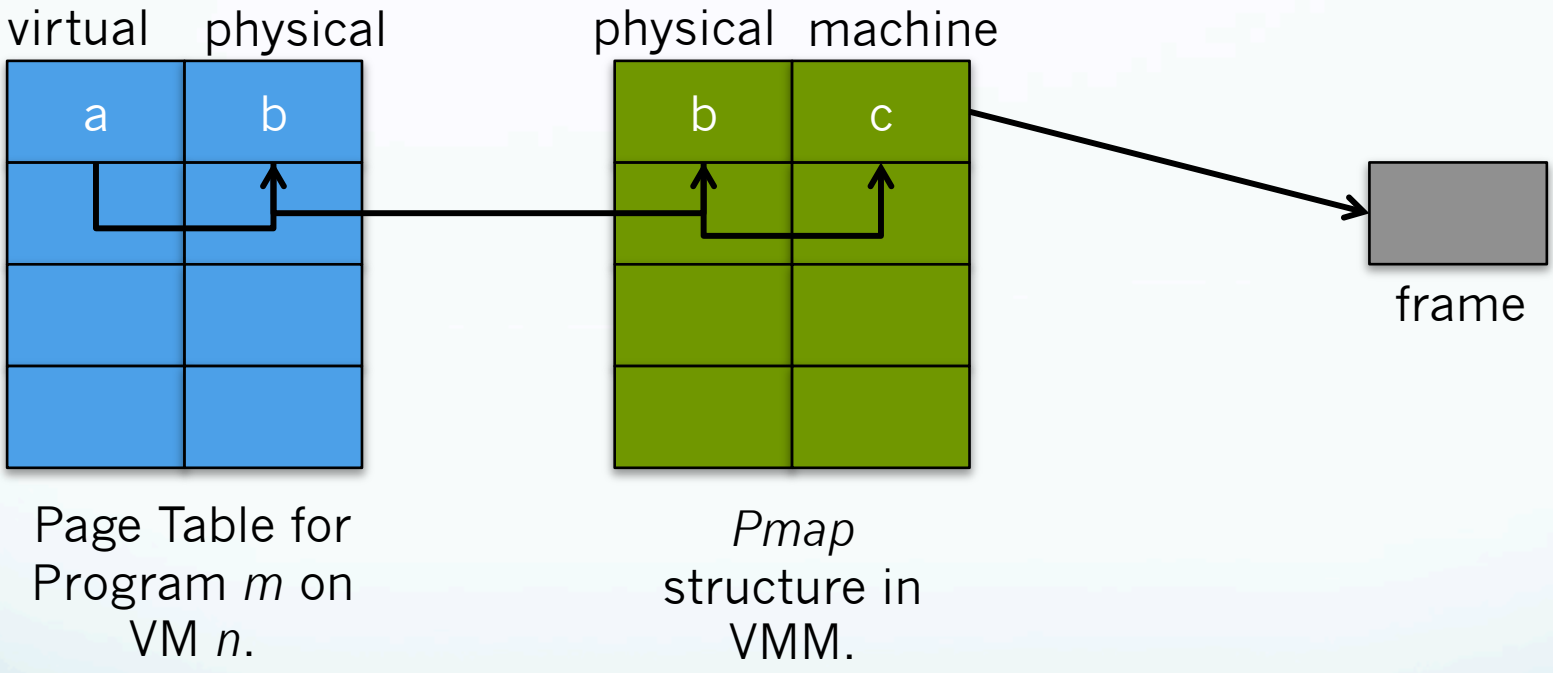
Applications of VT-X

- Xen uses Intel VT-x to host fully-virtualized guests alongside paravirtualized guests [6].
  - System has root (VMM) and non-root (guest) modes, each with privilege levels 0-3.
  - QEMU/Bochs projects provide emulations
- VMWare does not make use of VT technology [5].
  - VMWare's software-based VMMs significantly outperformed VT-X-based VMMs [5].
  - VT-X virtualization is trap-based, and DBT tries to eliminate traps wherever possible.

Virtualizing Memory

- Virtualization software must find a way to handle paging requests of operating systems, keeping each set of pages separate.
- Memory virtualization must not impose too much overhead, or performance and scalability will be impaired.
- Guest OS must each have an address space, be convinced that it has access to the entire address space.
- SOLUTION: most modern VMMs add an additional layer of abstraction in address space [4].
  - *Machine Address*—bare hardware address.
  - *Physical Address*—VMM abstraction of machine address, used by guest Oses.
  - Guest maintains virtual-to-physical page tables.
  - VMM maintains pmap structure containing physical-to-machine page mappings.

Memory Problem

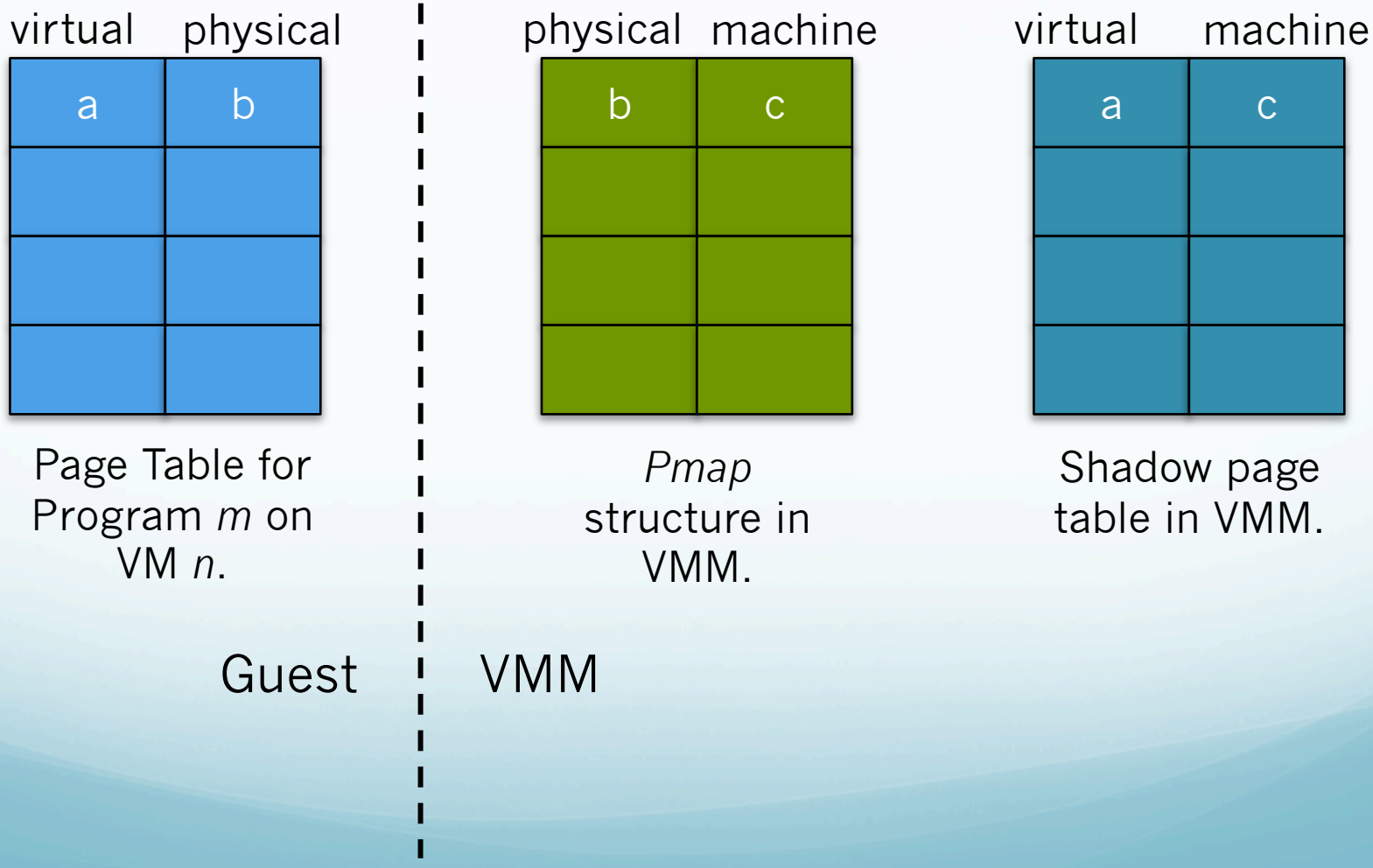


That's a lot of lookups!

Shadow Page Tables

- *Shadow page tables* map virtual memory to machine memory [4].
  - One page table maintained per guest OS.
  - TLB caches results from shadow page tables.
- Shadow page tables must be kept consistent with guest pages.
  - VMM updates shadow page tables when pmap (physical-to-machine) records are updated.
- VMM now has access to virtual addresses, eliminating two page table lookups.

Shadow Page Tables



Shadow Page Table Drawbacks

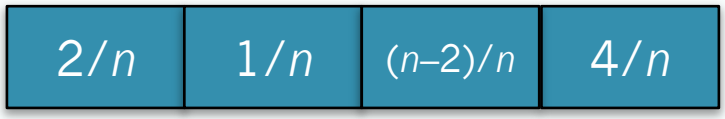
- Updates are expensive
  - On a write, the VMM must update the VM and the shadow page table.
  - TLB must be flushed on world switch.
    - TLB from other guest will be full of machine addresses that would be invalid in the new context.

Direct Access

- Direct access to hardware is not permitted by the Popek and Goldberg model [8].
- VMWare and Xen both bend this rule, allow guests to access hardware directly in certain cases.
- Xen uses validated access model [3].
  - Fine-grained control over direct access.
- VMWare allows user-mode instructions to bypass BT, go straight to CPU [5].
- Memory accesses are sometimes batched to minimize context switches.

Load Balancing Problem

- Assume VMM divides address space evenly among guests.
- If guest workload is not balanced, one guest could be routinely starved for memory.
- Other guests have way more than they need.
- Solution: memory overcommitment



Memory Overcommitment

- *Overcommitment*: committing more total memory to guest OSes than actually exists on the system [4].
  - Guest memory can be adjusted according to workload.
  - Higher-workload servers get better performance than with a simple even allocation.
- Requires some mechanism to reclaim memory from other guests [4].
- Poor page replacement schemes can result in *double paging* [4].
  - VMM marks page for reclamation, OS immediately moves reclaimed page out of memory
  - Most common in high memory-usage situations.

Ballooning

- Mechanism for page reclamation.
  - Technique to induce page-ins, page-outs in a guest OS.
- "Balloon module" [4] loaded on guest OS reserves physical pages; can be expanded or contracted.
- Balloon inflates, guest starts releasing memory
- Balloon deflates, guest may start allocating pages.
- VMWare and Xen both support ballooning.

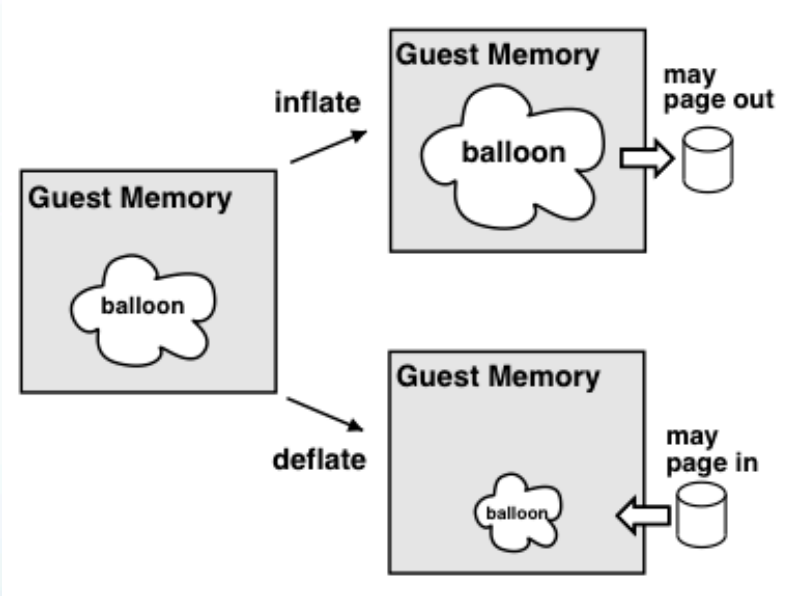
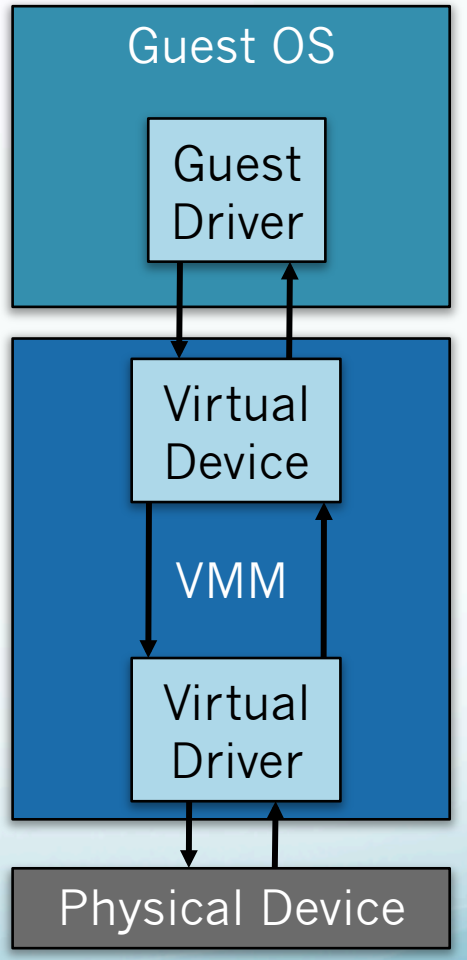


Image Source: Waldspurger, C. "Memory Resource Management in VMware ESX Server", OSDI 2002.

I/O Virtualization

- Performance is critical for virtualized I/O
  - Many I/O devices are time-sensitive or require low latency [7].
- Most common method: device emulation
  - VMM presents guest OS with a virtual device [7].
  - Preserves security, handles concurrency, but imposes more overhead.



I/O Virtualization Problems

- Multiplexing
  - How to share hardware access among multiple OSes.
- Switching Expense
  - Low-level I/O functionality happens at the VMM level, requiring a context switch.

Packet Queuing

- Both major VMMs use an asynchronous ring buffer to store I/O descriptors.
- Batches I/O operations to minimize cost of world switches [7].
- Sends and receives exist in same buffer.
- If buffer fills up, an exit is triggered [7].

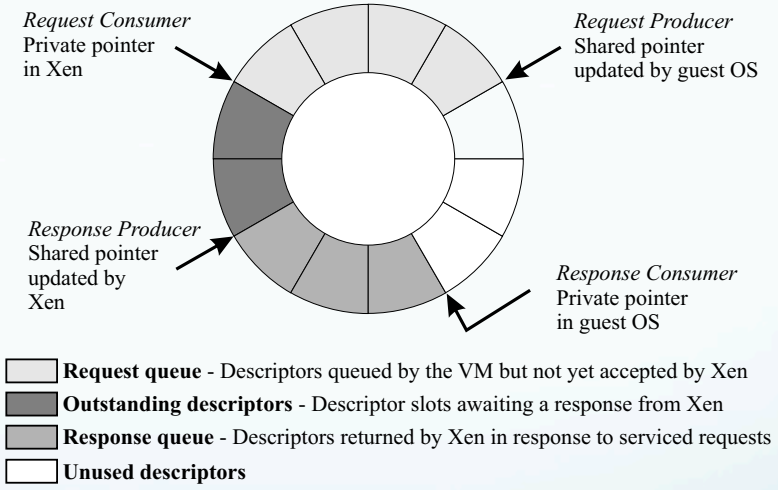


Image Source: Barham, P. et al. "Xen and the Art of Virtualization", SOSP 2003.

I/O Rings, continued

Xen

- Rings contain memory descriptors pointing to I/O buffer regions declared in guest address space.
- Guest and VMM deposit and remove messages using a producer-consumer model [2].
- Xen 3.0 places device drivers on their own virtual domains, minimizing the effect of driver crashes.

VMWare

- Ring buffer is constructed in and managed by VMM.
- If VMM detects a great deal of entries and exits, it starts queuing I/O requests in ring buffer [7].
- Next interrupt triggers transmission of accumulated messages.

Summary

- Current VMM implementations provide safe, relatively efficient virtualization, albeit often at the expense of theoretical soundness [8].
- The x86 architecture requires a) binary translation, b) paravirtualization, or c) hardware support to virtualize.
- Binary translation and instruction trapping costs are currently the largest drains on efficiency [5].
- Management of memory and other resources remains a complex and expensive task in modern virtualization implementations.

References

1. Singh, A. "An Introduction To Virtualization", www.kernelthread.com, 2004.
2. VMWare White Paper, "Understanding Full Virtualization, Paravirtualization, and Hardware Assist", 2007.
3. Barham, P. et al. "Xen and the Art of Virtualization", SOSP 2003.
4. Waldspurger, C. "Memory Resource Management in VMware ESX Server", OSDI 2002.
5. Adams, K. and Agesen, O. "A Comparison of Software and Hardware Techniques for x86 Virtualization", ASPLOS 2006.
6. Pratt, I. et al. "Xen 3.0 and the Art of Virtualization", Linux Symposium 2005.
7. Sugerman, J. et al. "Virtualizing I/O Devices on Vmware Workstation's Hosted Virtual Machine Monitor", Usenix, 2001.
8. Popek, G. and Kgoldberg, R. "Formal Requirements for Virtualizable Third-Generation Architectures", Communications of the ACM, 1974.
9. Mahalingam, M. "I/O Architectures for Virtualization", VMWorld, 2006.
10. Smith, J. and Nair, R. *Virtual Machines*, Morgan Kaufmann, 2005.
11. Bellard, F. "QEMU, a Fast and Portable Translator", USENIX 2005.
12. Silberschatz, A., Galvin, P., Gagne, G. *Operating System Concepts*, Eighth Edition. Wiley & Sons, 2009.