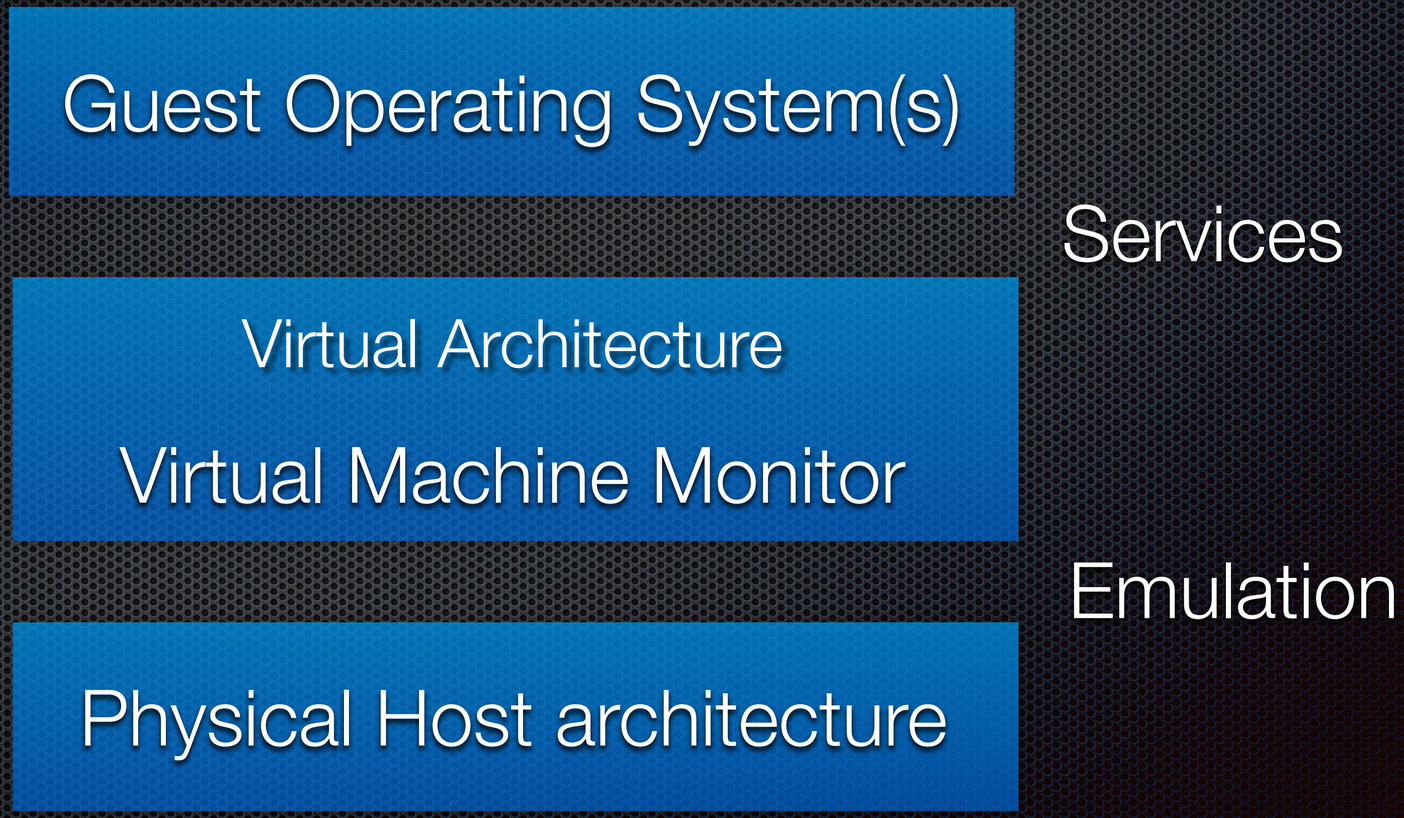
Virtualization

Clothing the Wolf in Wool

Virtual Machines

- Began in 1960s with IBM and MIT Project MAC
- Also called “open shop” operating systems
- Present user with the view of a bare machine
- Execute most instructions directly, but trap operations that would reveal the fiction, and emulate
- Hot topic until Unix, then architectural support drops
    - Example of M68010

Basic VM


Guest Operating System(s)
Services
Virtual Architecture
Virtual Machine Monitor
Emulation
Physical Host architecture

Advantages

- Greater security -- more isolation of tasks
- Ability to manage QoS for tasks
- Easy to profile tasks
- Can run multiple operating systems
- Can test and debug new OS code directly

Security

- Each guest OS has its own virtual machine
  - Even if guest OS is compromised, it’s in a sandbox
- Different users can be run in separate VMs
- VMM can watch for bad behavior and shut down VM
  - Manages denial of service attacks

Quality of Service (QoS)

- Each VM can have its own resource allocation
- VMM scheduler manages access to actual resources
- Different users see different machine configurations
- Performance can be adjusted dynamically
  - Respond to need
  - Throttle excess usage

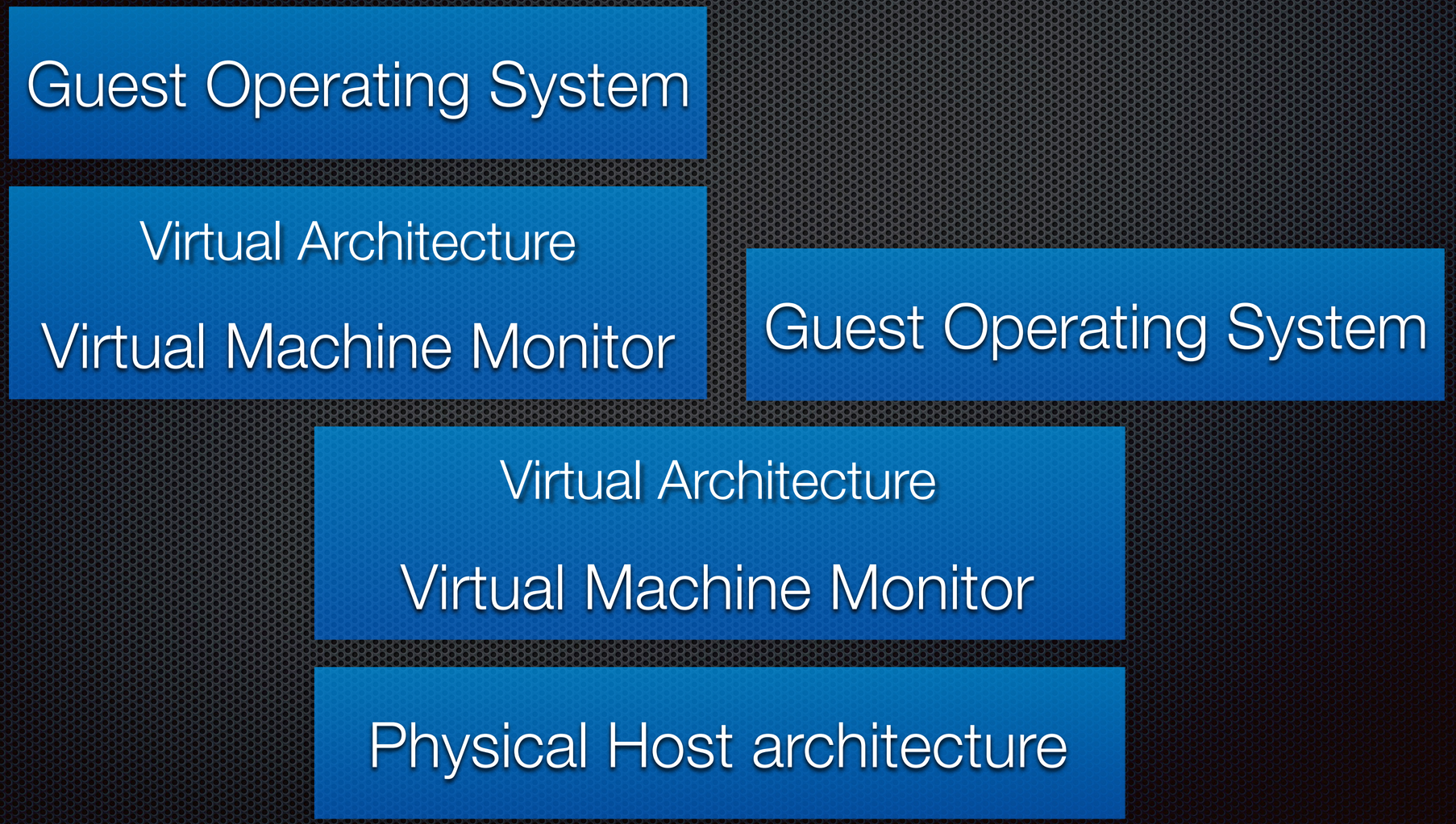
Ease of Profiling

- Every OS service request goes through the VMM
- Can keep statistics on many different factors
- Useful for QoS management, security monitoring, provisioning

Multiple/Test systems

- Since VMM presents a bare machine, each VM can run a different OS
  - If VMM is recursively virtualizable, can run itself as a guest, enabling OS debugging on a live machine
- Traditionally, the virtual architecture is a copy of the host architecture, but smaller
  - Can be different, although emulation makes it slow

Recursively Virtualizable



Disadvantages

- More complex OS/VMM
- Takes more resources than kernel OS
- Requires hardware support
- Can be slow if hardware support is weak
- Gets in the way of performance research

Gerald Popek CACM 1974

Formal Requirements for Virtualizable Third Generation Architectures

Requirements

- Equivalence: Virtual machine must look like bare HW (but smaller)
- Resource control: All resources are virtualized and managed by the VMM
- Efficiency: Has to be nearly as fast as running natively

Instruction Types

- Privileged: Implies supervisor state with special instr.
- Control sensitive: Changes processor mode, or memory map
- Behavior sensitive: Behavior depends on mode or on location (anything that can reveal state of other tasks)
- A VMM requires that the sensitive instructions be a subset of the privileged instructions

Recursive Virtualizability

- If there is proper HW support, a VM can run recursively within itself
- Allows nested operating systems, layers of control
- Rarely supported

Paravirtualization

- Can fake virtualization with JIT or direct binary translation (DBT) of sensitive, non-privileged instructions
- XEN approach when no HW virtualization mode
- Can’t be completely hidden from adversary
  - Code can check whether a sensitive, non-privileged instruction has been rewritten

Two Types

- Type 1 (Xen) runs as separate privilege layer
  - Has a separate OS process for virtualizing I/O
  - All apps run on guest operating systems
- Type 2 (KVM) modifies existing OS (e.g., Linux)
  - Uses existing device drivers in base OS
  - Runs some apps directly

Examples of x86 Non-virtualizable Instructions

Sensitive, Non-privileged x86 Instructions

- SGDT – Store Global Descriptor Table register
- SIDT – Store Interrupt Descriptor Table register
- SLDT – Store Local Descriptor Table register
- SMSW – Store Machine Status Word
- PUSHF(D) – Push EFLAGS register on stack (16 and 32-bit versions)
- POPF(D) – Pop EFLAGS register from stack, with some privilege levels (16 and 32-bit versions)

Sensitive, Non-privileged x86 Instructions

- LAR – Load access rights into GP register
- LSL – Load segment address limit into GP register
- VERR – Verify if code/data segment is readable based on current protection level
- VERW – Verify is code/data segment is writeable based on current protection level
- POP – Can raise general protection exception depending on target register and protection level
- PUSH – Can push protection status onto the stack

Sensitive, Non-privileged x86 Instructions

- CALL – Can call to same or a different privilege level, saving return info
- JMP – Like CALL, but without saving return info
- INT n – Like a far CALL to a different level, but also pushes EFLAGS on stack
- RET – Can return between privilege levels
- STR – Store segment selector (including privilege bits)
- MOV – Can be used to load or store control register set

Virtualization Extensions

- VT-x introduced 2005 by Intel
- AMD-V introduced 2006 by AMD
- AMD Rapid Virtualization Indexing (nested page tables) adds hardware to MMU
- Intel adds Extended Page Tables
- VT-D provides virtualization of directed I/O, trapping DMA, etc.
- Typically not enabled in BIOS
- VirtualBox claims to be faster without VT-x

Itanium Example of VM Subtlety

- Trap to a higher level, setting violation status
- Can only return to a lower level resetting status
- Can’t forward the violation to a guest OS to handle

Beyond the ISA

- Areas that are hard to virtualize
  - Complex virtual memory
  - I/O and network devices
  - Graphics, GPUs
  - Multithreading
  - Cache coherence
  - Multicore
  - TLB, branch predictor -- clever optimizations can backfire when they are virtual

User Level Virtual Machine

- Java virtual machine
  - Executes bytecode on stack architecture
  - Some bytecodes much more complex that ISA
- Simulators (MARSSx86, PTLsim, QEMU)
  - QEMU emulates (a bit like paravirtualization)
  - PTLsim simulates each instruction
  - MARSSx86 uses QEMU to fast forward, then simulates with fairly accurate timing

Cloud Computing

- Clusters of virtual nodes
- Less intensive tasks share physical node
- More intensive tasks share less
- Facilitates image migration for load balancing