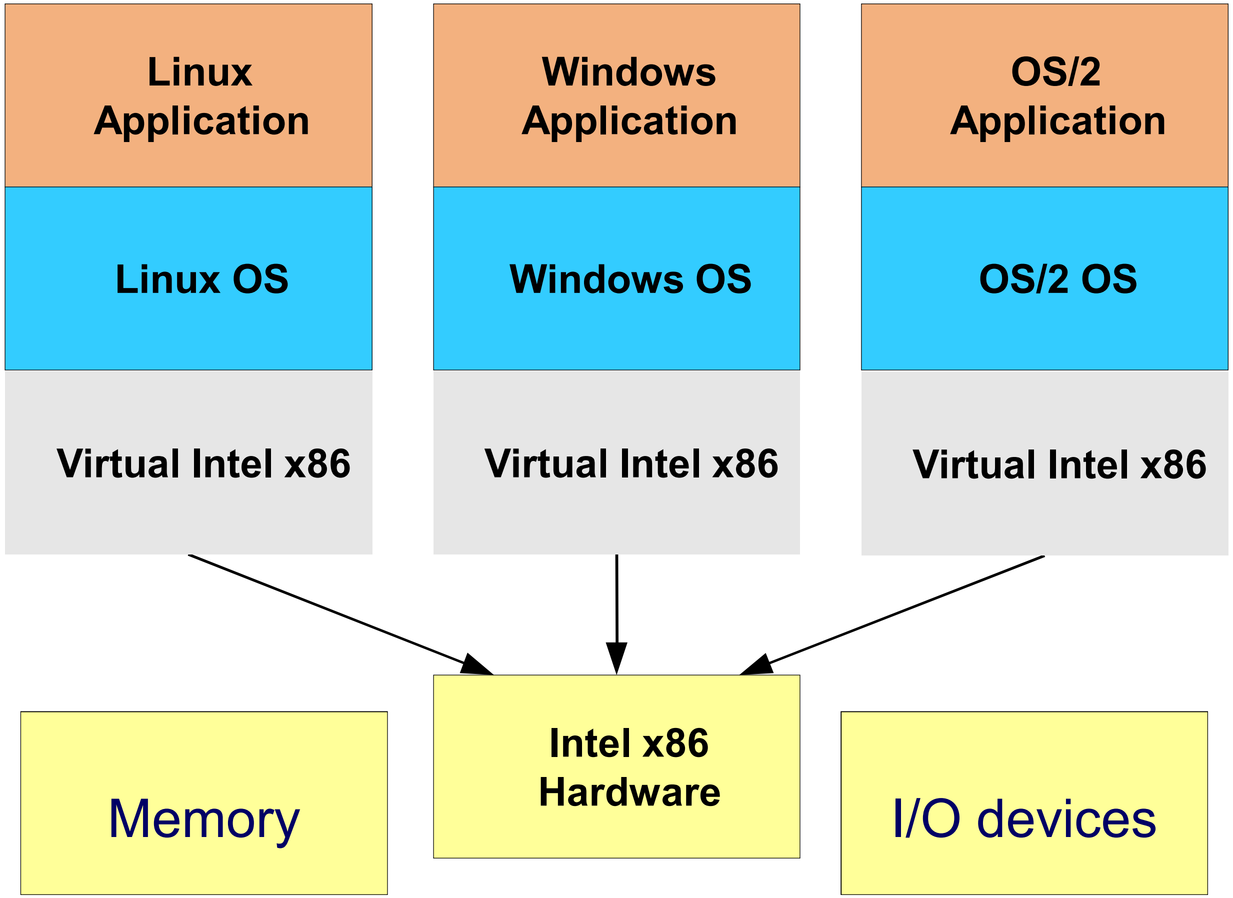
System VMs

Prof. Li-Pin Chang
ESSLab@NCTU

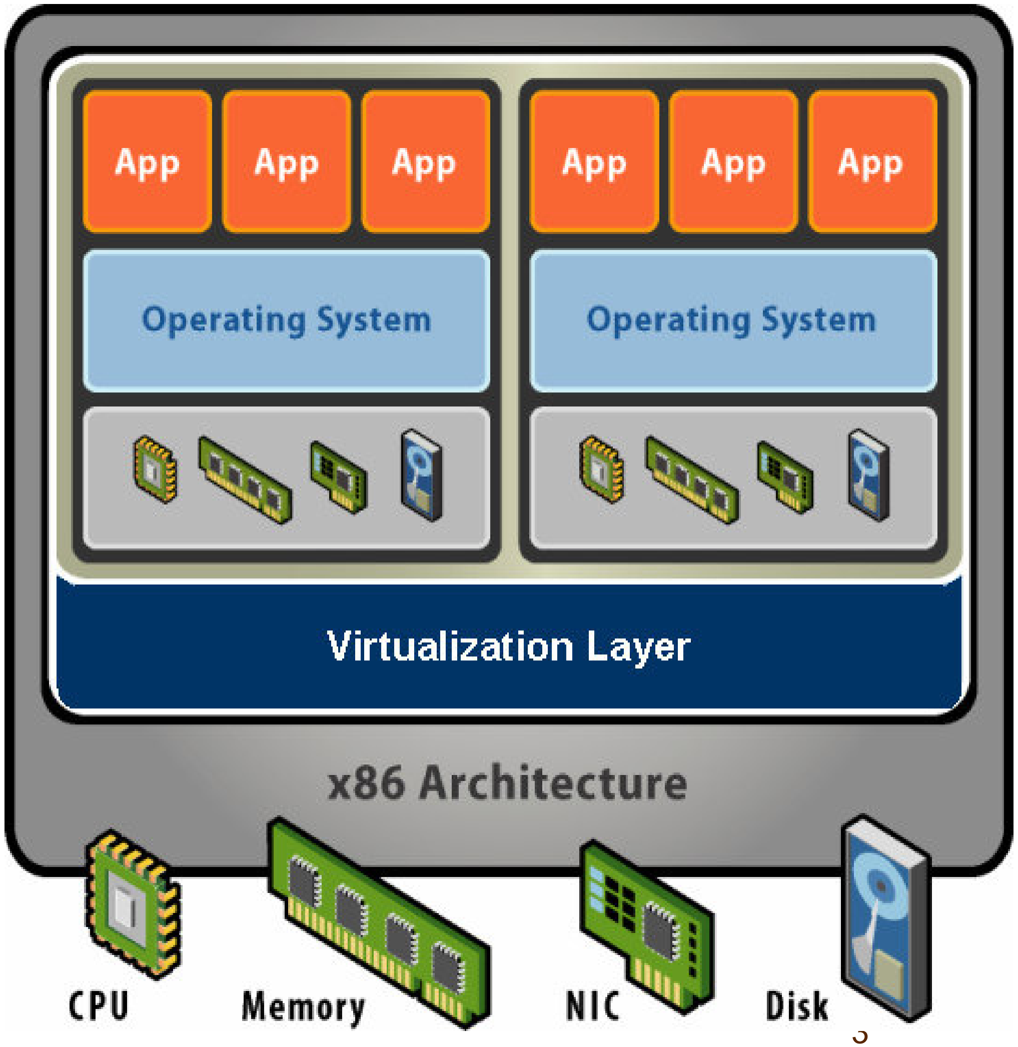
System VMs

- Support multiple guest OSes on single hardware platform; all running the same ISA



System VMs

- ❑ **Hardware-Level Abstraction**
- ❑ **Virtualization Software**
  - Extra level of indirection decouples HW and OS
  - Multiplexes physical HW Across multiple guest VM
  - Strong isolation between VMs
  - Improve utilization



Virtual to Physical mapping

- **Each virtual resource may or may not have a corresponding physical resource.**
  - When a physical resource is available
    - may be time shared by multiple virtual resources
    - may be partitioned for multiple virtual resources
  - When a physical resource is not available
    - may be emulated via software, or
    - emulated via a combination of software and other resources that are physically available on the host platform.

Virtualization Properties

- **Isolation**
  - Fault isolation
  - Performance isolation
- **Encapsulation**
  - Cleanly capture all VM states
  - Enables VM snapshots, clones
- **Migration**
  - Decoupled from physical hardware
  - Enables live migration of VMs
- **Interposition**
  - Enables transparent resource overcommitment, encryption, compression, replication.

Applications

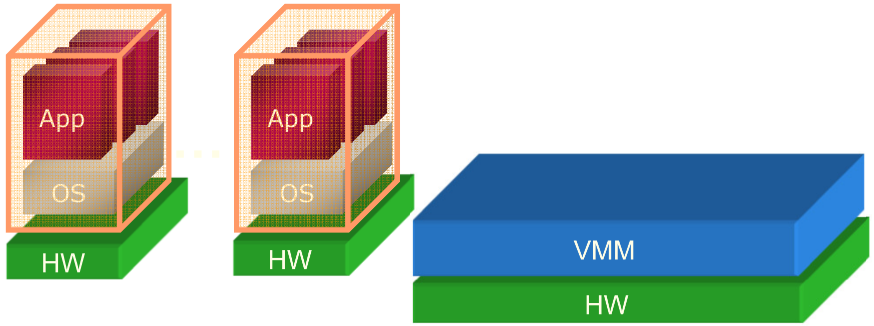
- **Resource consolidation**
  - Server consolidation
  - Client consolidation
- **Simultaneous support for multiple OSes/Apps**
  - Easy way to implement timesharing
- **Simultaneous support for different OSes/Apps**
  - E.g. Windows and Unix
- **Error containment**
  - If a VM crashes, the other VMs can continue to work
    Assumes VMM is correct (smaller/simpler)
- **Operating System debugging**
  - Can proceed while system is being used for normal work

Resource Consolidation

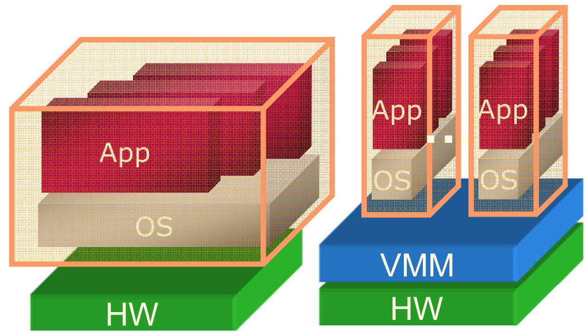
- **Server consolidation**
  - Reduce number of servers
  - Reduce space, power and cooling
  - 70-80% reduction numbers cited in industry
- **Client consolidation**
  - Developers: test multiple OS versions, distributed application configurations on a single machine
  - End user: Windows on Linux, Windows on Mac
  - Reduce physical desktop space, avoid managing multiple physical computers

Today’s Applications

Server Consolidation

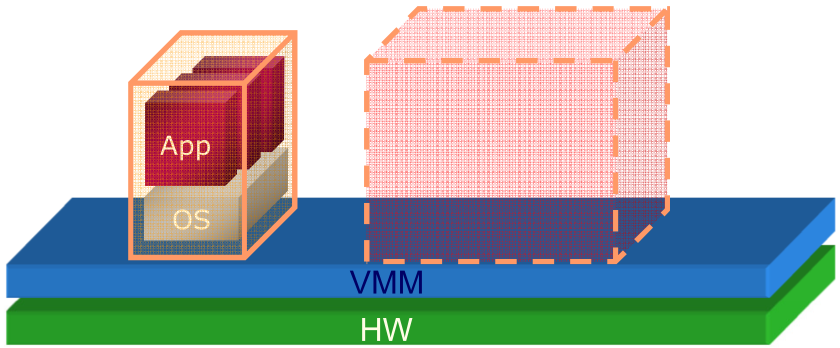


Benefit: Cost Savings



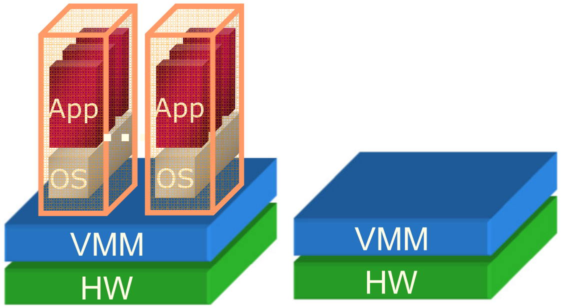
Work Isolation

R&D Production



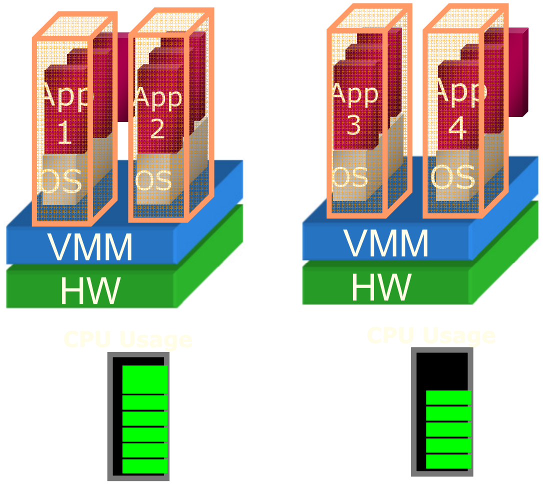
Benefit: Business Agility and Productivity

Emerging Applications



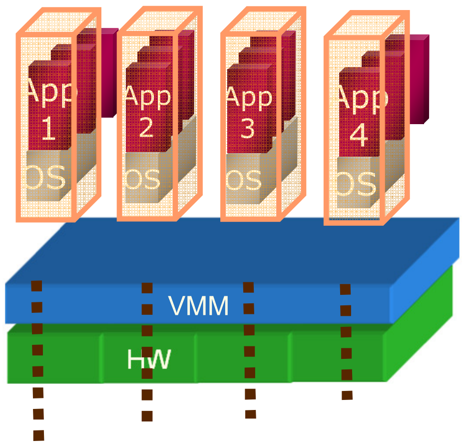
Benefit: Business Continuity

Disaster Recovery

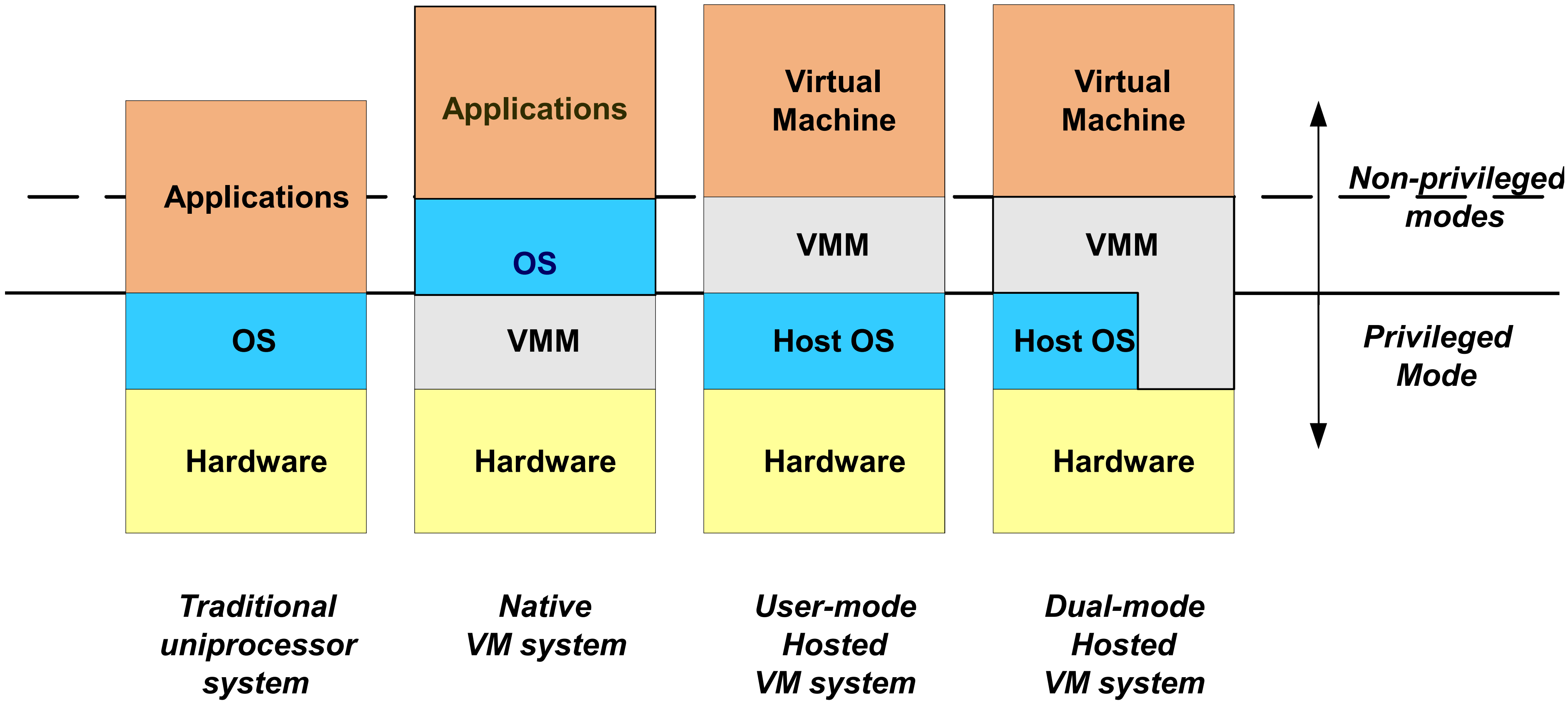


Partitioning

Dynamic load balancing

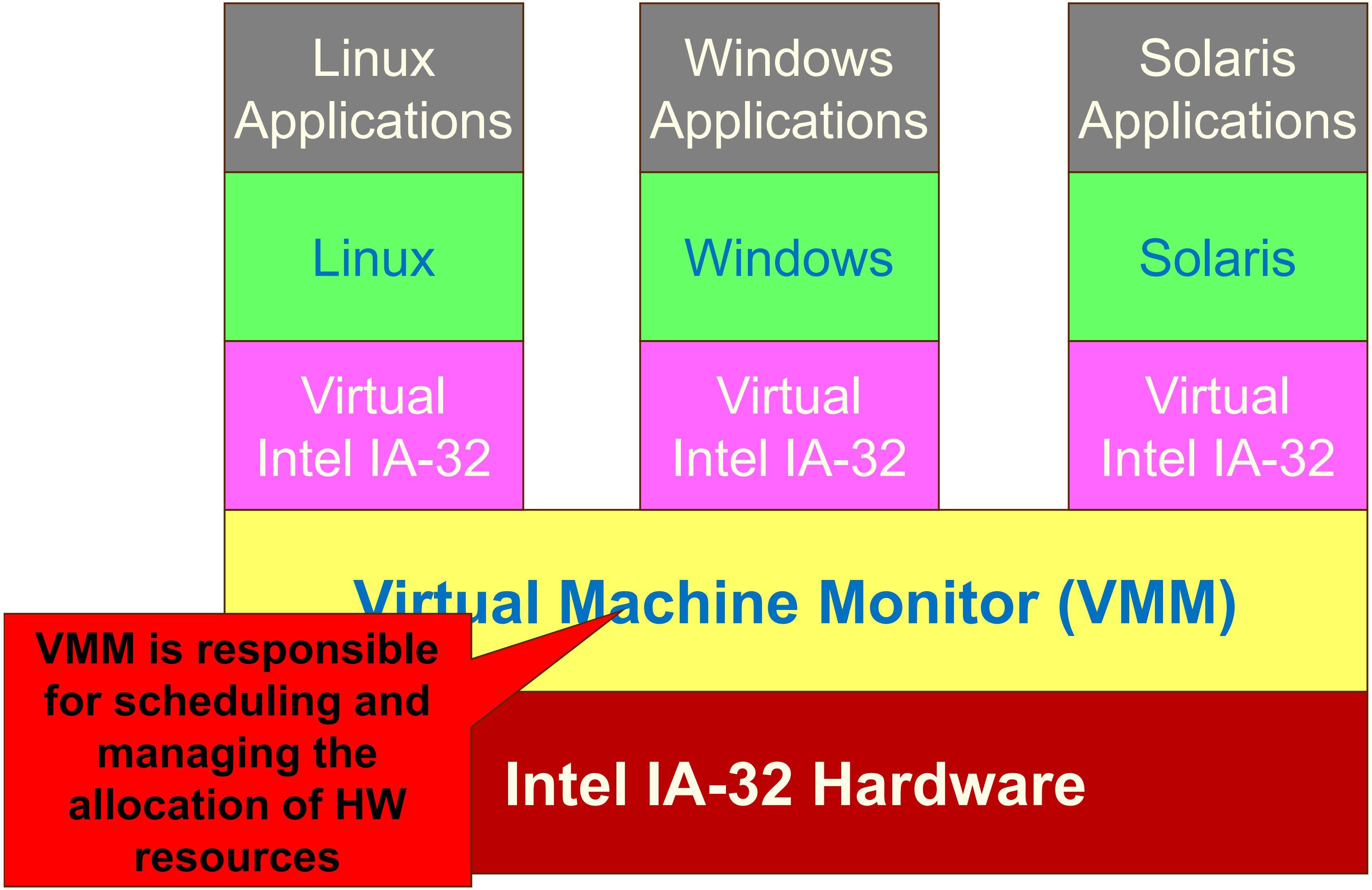


Native and Hosted VMs

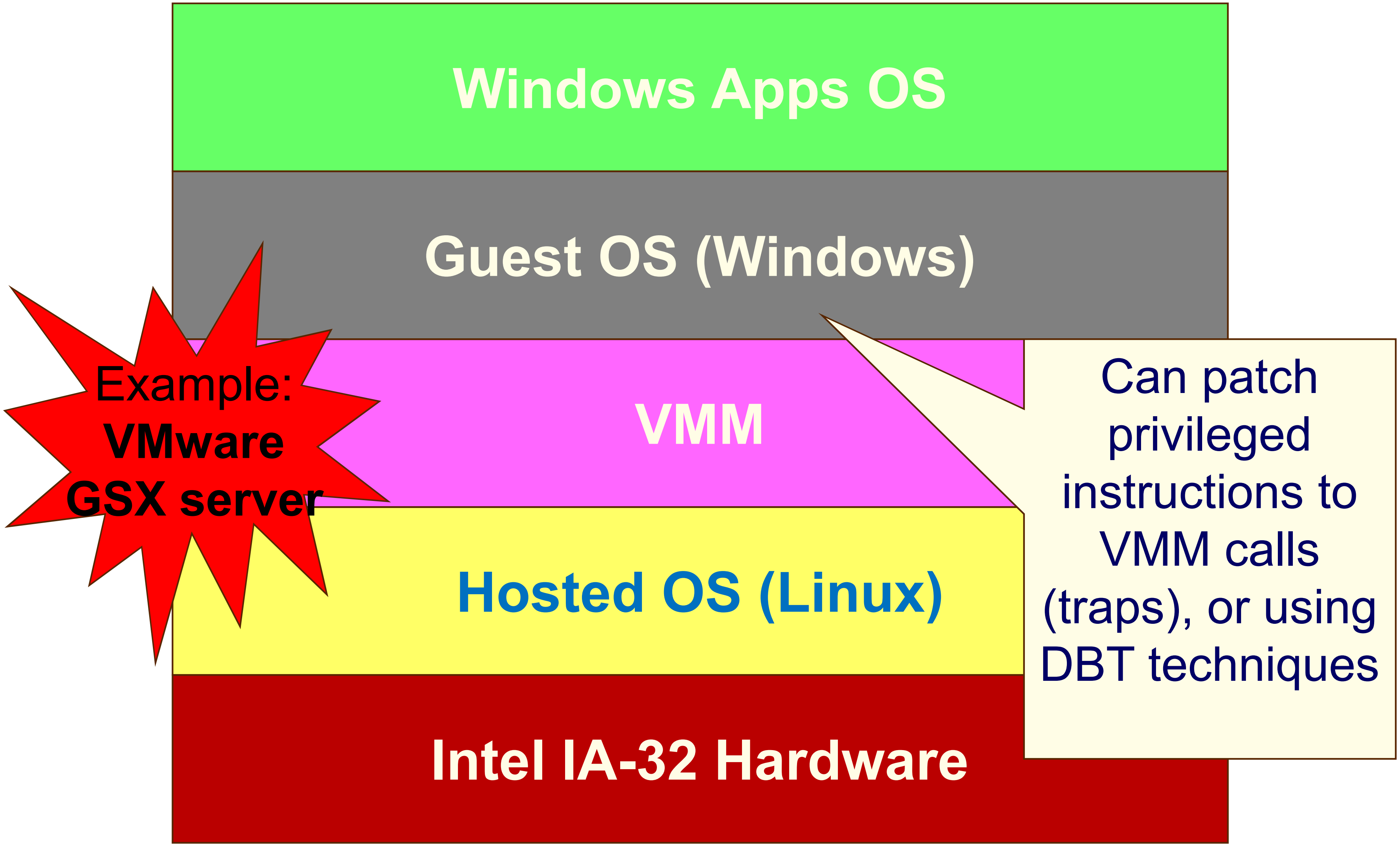


User convenience and Implementation simplicity

Native System VM Environment

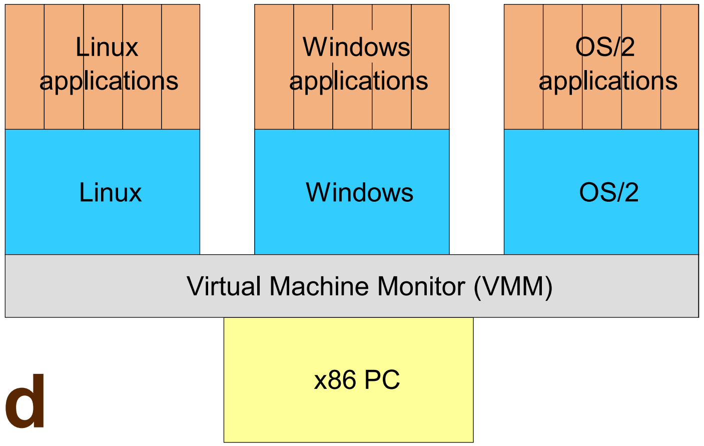


User Mode Hosted VM



System VMs

- Virtual Machine Monitor (VMM) manages real hardware resources
- All Guest systems must be given logical hardware resources
- All resources are *virtualized*
  - By partitioning real resources
  - By sharing real resources
- Guest state must be managed
  - By using indirection
  - By copying



System VMs: Processor Mgmt/Protection

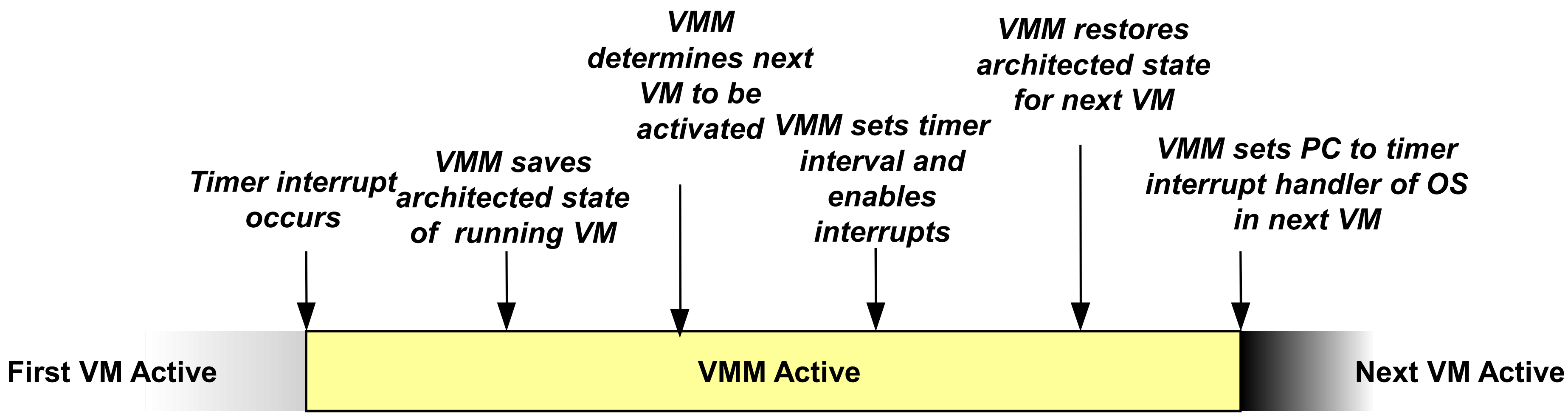
- **VMM runs in *system* mode**
  - VMM manages/protects processor through conventional mechanisms
- **Guest OSes run in *user* mode**

  $\Rightarrow$ Guest OSes do not have direct control over hardware resources

  All attempts to interact w/ hardware resources are intercepted by VMM (trap)
- **VMM manages shadow copies of Guest System state (incl. control registers)**
- **VMM schedules and runs Guest Systems**

VM Timesharing

- **VMM Timeshares resources among guests**
  - Similar to OS timesharing applications

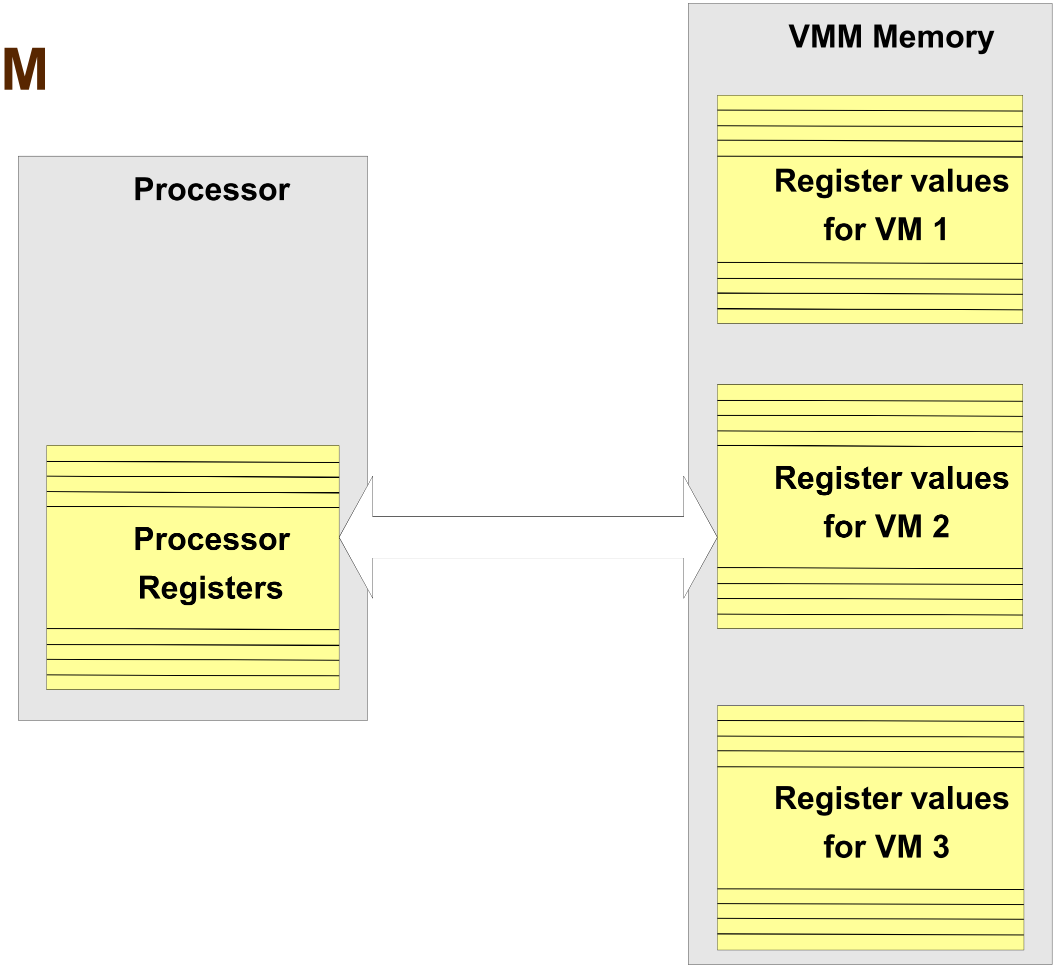


Privileged Resource

- **To prevent VMs from taking over the host, some resources must not be directly available to VMs**
- **E.g., system timer generate interrupts for the VMM to get resources from VMs**
  - Such as CPU
  - Thus timer is never directly accessible to VMs

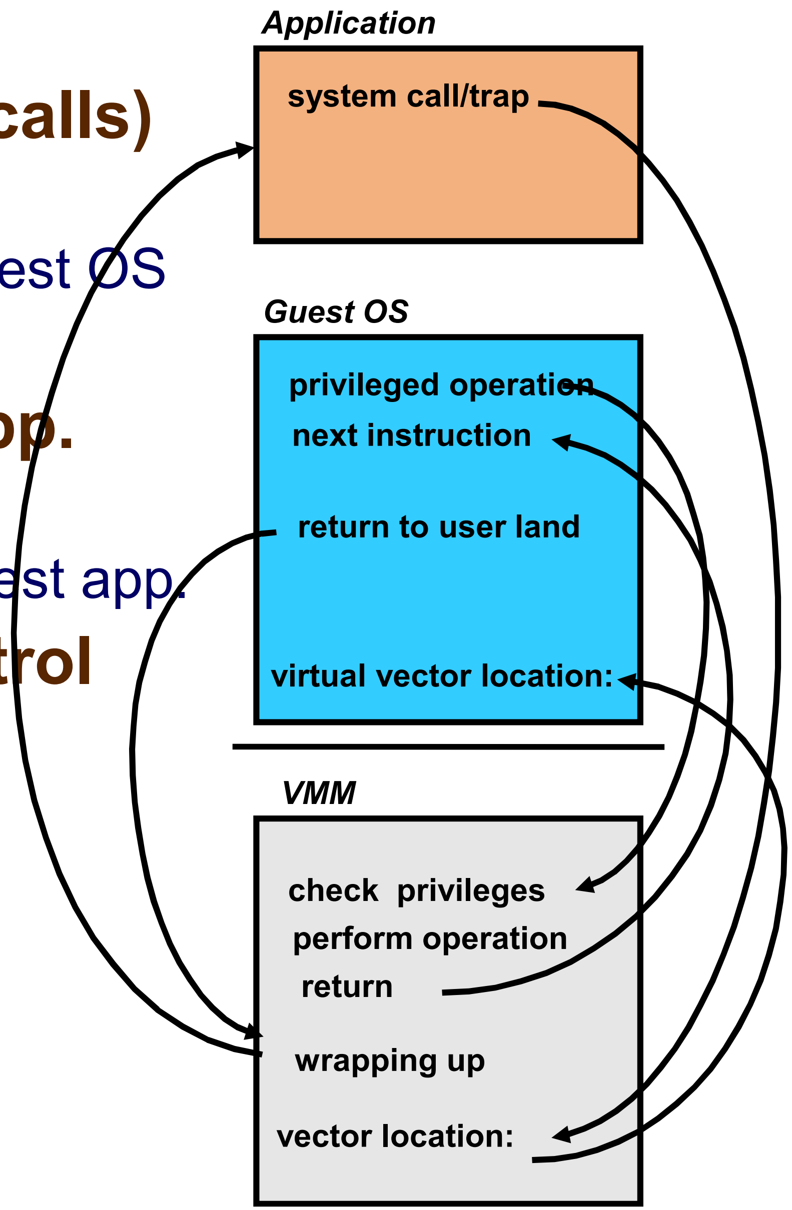
Virtualizing State

- Copying
- Hold guest state in VMM Memory
- Copy state on guest switch



Processor Management/Protection

- **Traps and interrupts (& sys calls)**
  - Transfer to VMM
  - VMM determines appropriate Guest OS
  - VMM transfers to Guest OS
- **Guest OS "return" to user app.**
  - Transfer to VMM
  - VMM bounces return back to Guest app.
- **Read/Write of protected control registers**
  - Trap to VMM
  - VMM reads/modifies guest copy
  - May modify shadow copy
  - Returns to Guest



OS VMs: Key Issue – ISA Virtualizability

- **What if privileged instruction no-ops in user mode? (rather than trapping)**
  - Then… VMM can't intercept when Guest OS attempts the privileged instruction

- **What if user can access memory with real address?**
  - Then… a guest OS may see that the real memory it *really* has is different from the memory it *thinks* it has

- **What if user can read system control registers?**
  - Then… guest OS may not read the same state value that it *thinks* it wrote

Virtualizability (Popek, Goldberg, 74)

- Classic work in formalizing OS VM concepts
- Defines basic VM properties
- Defines properties of instruction sets
- **Proves that VMM can be constructed if instruction set properties hold**
- Extends to recursive VMs
- Reduces to hybrid VMs

VM Properties

- **Efficiency**
  - Non-sensitive (non-privileged, or innocuous) instructions must be executed natively on the hardware
- **Resource control**
  - Guest software should never directly change the configuration of hardware resource
- **Equivalence**
  - Any program execute on a virtual machine must behave exactly the same as it does on the native hardware

Privileged Instructions, Definition:

- ***Trap if executed in user mode; not in supervisor mode***
- **Privileged instructions are *required* to trap**
  - No-op in user mode is *not* enough
- **Ex: in/out instructions in x86**

Control Sensitive instructions:

1. **All instructions that change the amount of (memory) resources (or the mapping)**
   - base/limit register in simplified paper version
   - page table in general
2. **All instructions that change the processor mode**

- **Instructions that provide control of resources**
- **Examples:**
  - Load TLB (if TLB is architected)
  - Load control register
  - Return to user mode

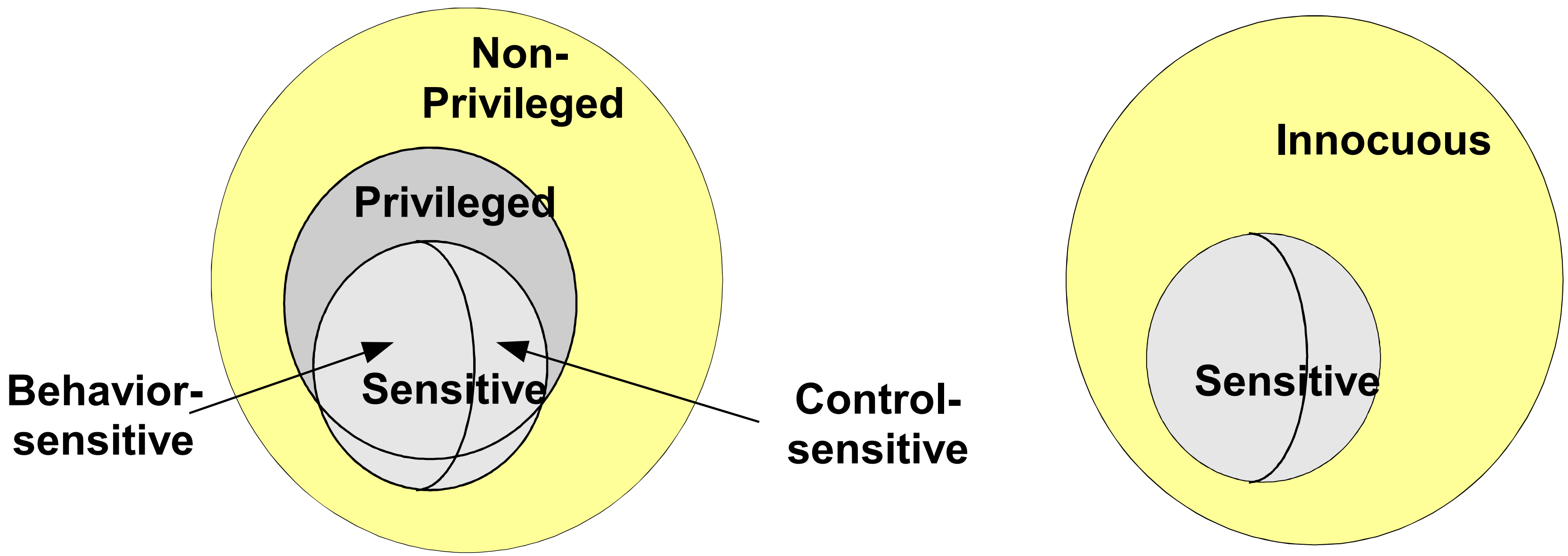
Behavior Sensitive instructions:

1. **All instructions whose results depend on the mapping of physical memory**
2. **All instructions whose behavior depends on the mode of the CPU**

- **Instructions whose behavior depends on configuration of specific resources (and who owns them)**
- **Examples:**
  - Load physical address
  - POPF (Intel x86): Interrupt-enable flag remains unaffected in user mode

- **However, the problem is that in some ISAs, not all sensitive instructions trap**
- **E.g., POPF instruction in x86**
  - Modification to the interrupt enable flag results in a no-op operation, rather than trap

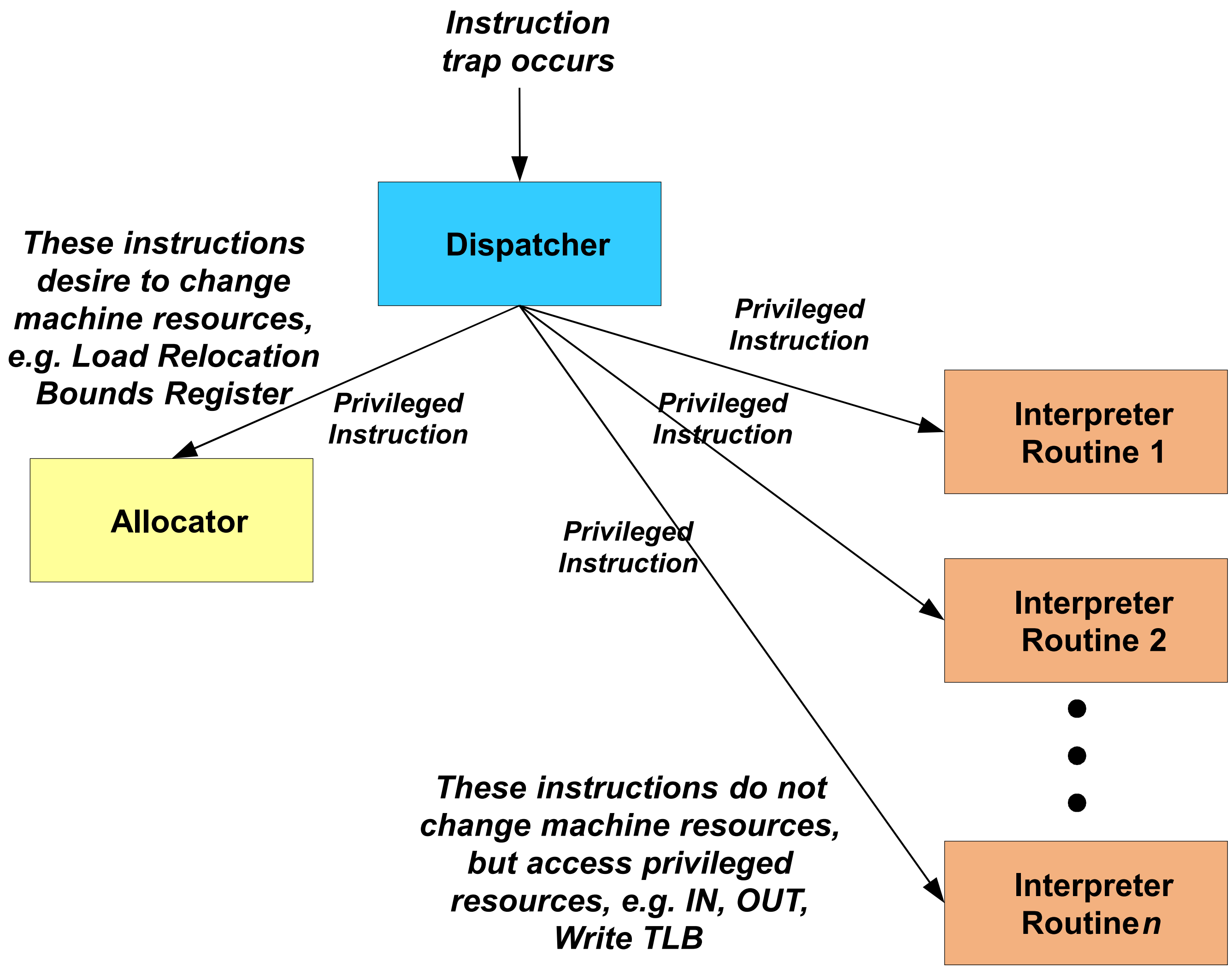
Instruction Types -- Summary



- **Innocuous Instructions:** Those that are not control or behavior sensitive
- For efficient ISA virtualization, the set of sensitive instructions must be the set of privileged instruction

- **Instruction patching**
  - For ISAs whose sensitive instructions may not trap
  - The VMM scans the executable image before execution
  - Sensitive instructions are patched with trap instructions
  - E.g., VMware for x86

VMM components



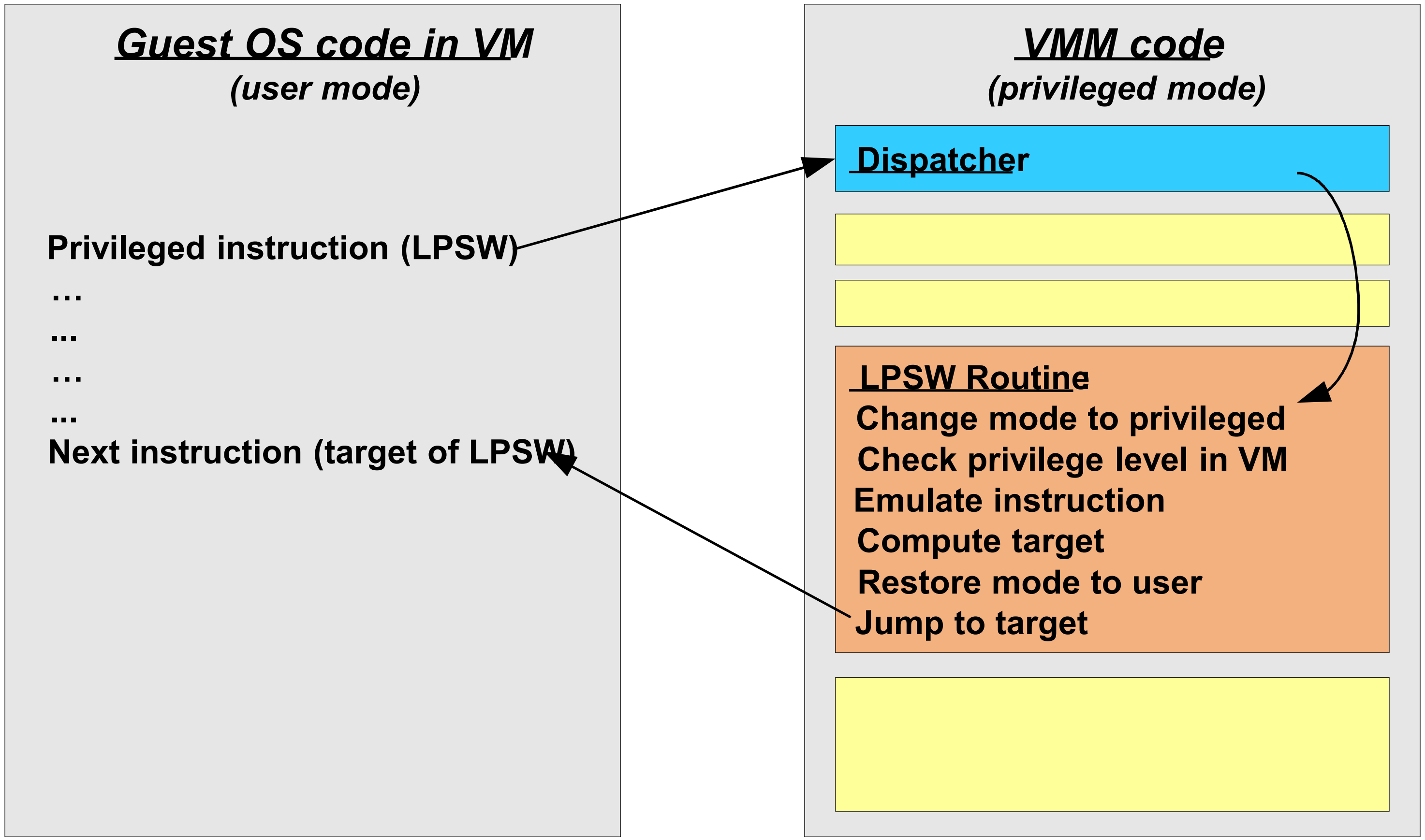
VMM components

- **Dispatcher**
  - Target of vectored traps – entry point for VMM
  - Decides which of other components to call
- **Allocator**
  - Decides which system resources should be provided and to manage shared resources among VMs
- **Interpreters**
  - Emulate the effects of privileged instructions
- **VMM runs in supervisor mode; all other software in user mode**

Privileged Instruction Handling

LPSW: Load Program Status Word

Includes Mode Bit and PC (among other things)



Virtual Machines: Main Theorem

A virtual machine monitor can be constructed if the set of sensitive instructions is a subset of the set of privileged instructions

Proof shows

Equivalence by interpreting privileged instructions and executing remaining instructions natively

Resource control by having all instructions that change resources trap to the VMM

Efficiency by executing all non-privileged instructions directly on hardware

Privilege instructions are naturally sensitive instructions, so p=s

Quick Review

- **What are the major applications of system virtual machines?**
  1) Migrating apps from one ISA to another
  2) Supporting multiple (same or different) OSes on the same machine
  3) Resource consolidation
  4) Improve application availability

  2,3,4

- **In VM systems, which of the following statements are true?**
  1) The Guest OS always runs in user mode
  2) The VMM always run in privileged mode
  3) A native VM system is easier to implement than a hosted VM system

  1

- **Based on Popek and Goldberg definition, which of the following ISA are virtualizable?**
  1) The privileged IS > the sensitive IS
  2) The privileged IS = the sensitive IS
  3) The privileged IS < the sensitive IS
  4) A RISC

2

- **In a VM system, when a system call is made by an application, how many times the VMM will be entered?**
  1) Zero times
  2) Once
  3) Two times
  4) At least two times

4

- **Based on Popek and Goldberg definition, which of the following ISA are virtualizable?**
  1) X86
  2) ARM
  3) IBM System 370
  4) Itanium

3

- **What resources in a machine must be virtualized ?**
  1) Processor or processors
  2) Memory and storage systems
  3) I/O devices

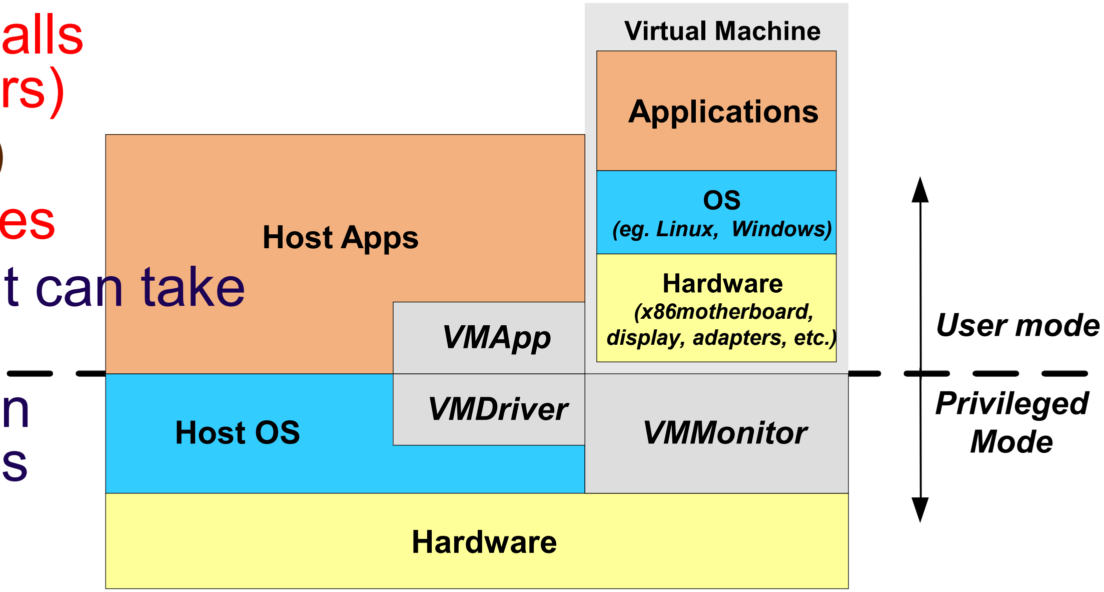
1,2,3

VMware: an x86 System Virtual Machine

- **Applying Conventional VMs to PCs – Problems:**
  - Installing the VMM on bare hardware, then booting Guests onto VMM.
  - Need to support many device types, many more drivers
- **VMware solves both problems**
- **Uses Host OS/Guest OS model**
  - “Hosted VM”
  - Uses Host OS for some VMM functions
    - Including I/O

Vmware GSX: Three Main components

- Begin with already-loaded Host OS
- VMMonitor (System-level VMM)
  - Slipped under installed OS via Pseudo-Driver
- VMApp (User-level VMM)
  - Appears as ordinary application to installed OS
  - Can make normal I/O calls (and use installed drivers)
- VMDriver (Pseudo-Driver)
  - Host OS-specific routines
  - Installed as a driver, but can take over the machine
  - Acts as conduit between System and User VMMs

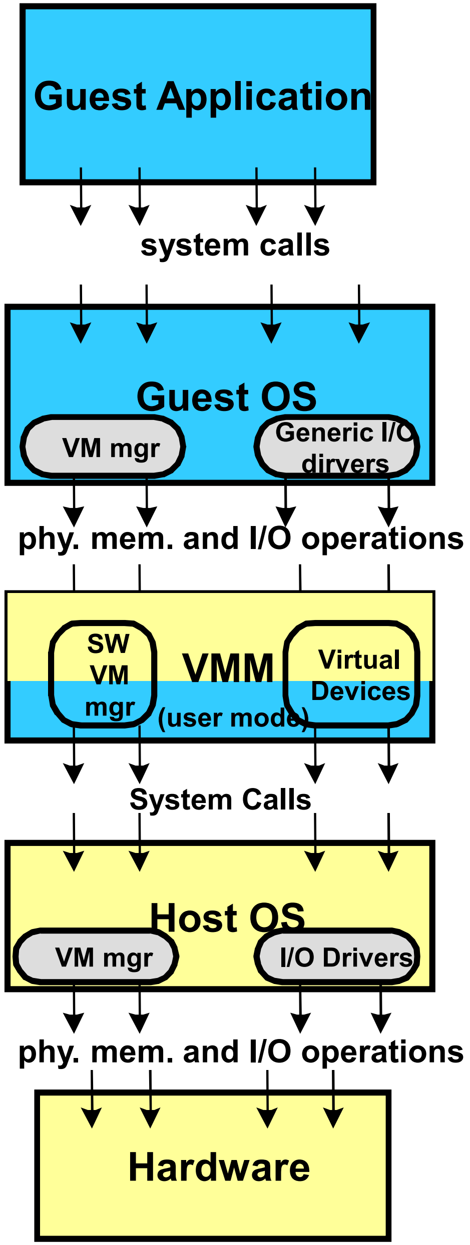


Resource Management

- **Host OS schedules processor resource**
  - User-level VMM is just another application
- **Host OS manages memory**
  - VM memory is allocated as address space of User-level VMM
  - User level VMM "mallocs"; whole VM uses it

VMware I/O

- Guest OS contains generic drivers
- Generic drivers operate on virtual devices managed by user mode portion of VMM
- User mode portion of VMM makes normal system calls
- System calls cause Host OS to use real drivers and devices



Example of I/O Virtualization (network packet send) in VMware

Guest OS

↓ OUT to I/O port

VMM

↓ Context switch

VMDriver

↓ Return to VMApp

VMApp

↓ Syscall

VMNet driver

↓ Bridge code

Host Ethernet driver

↓ OUT to I/O port

HW packet launch

I/O Sequence

- Guest application makes system call
- Intercepted by System-level VMM, reflected to Guest OS (↓↑)
- Guest OS performs I/O operations specified in generic drivers
- System-level VMM captures I/O operations, and interprets them (↓)
- Passes operation back up to User-level VMM (↑)
- User-level VMM performs I/O call to Host OS (↓↑)
- Guest OS returns to Guest Application via system-level VMM (↓↑)