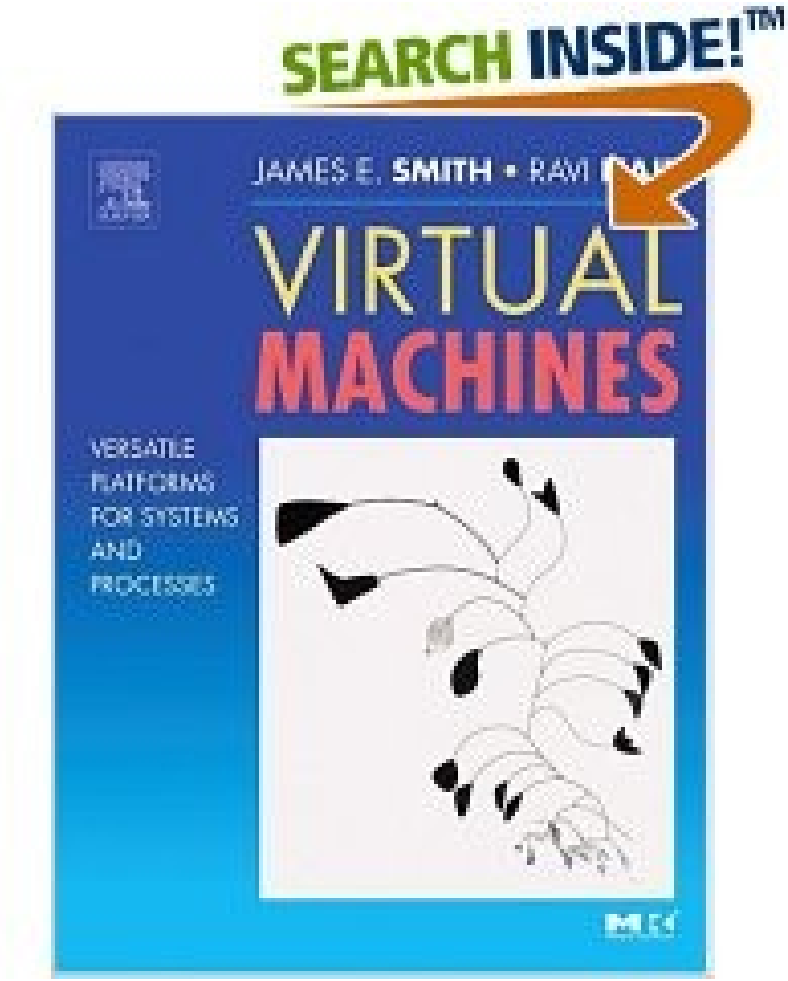
Introduction to Virtualization

Reference



Virtual Machines: Versatile Platforms for Systems and Processes

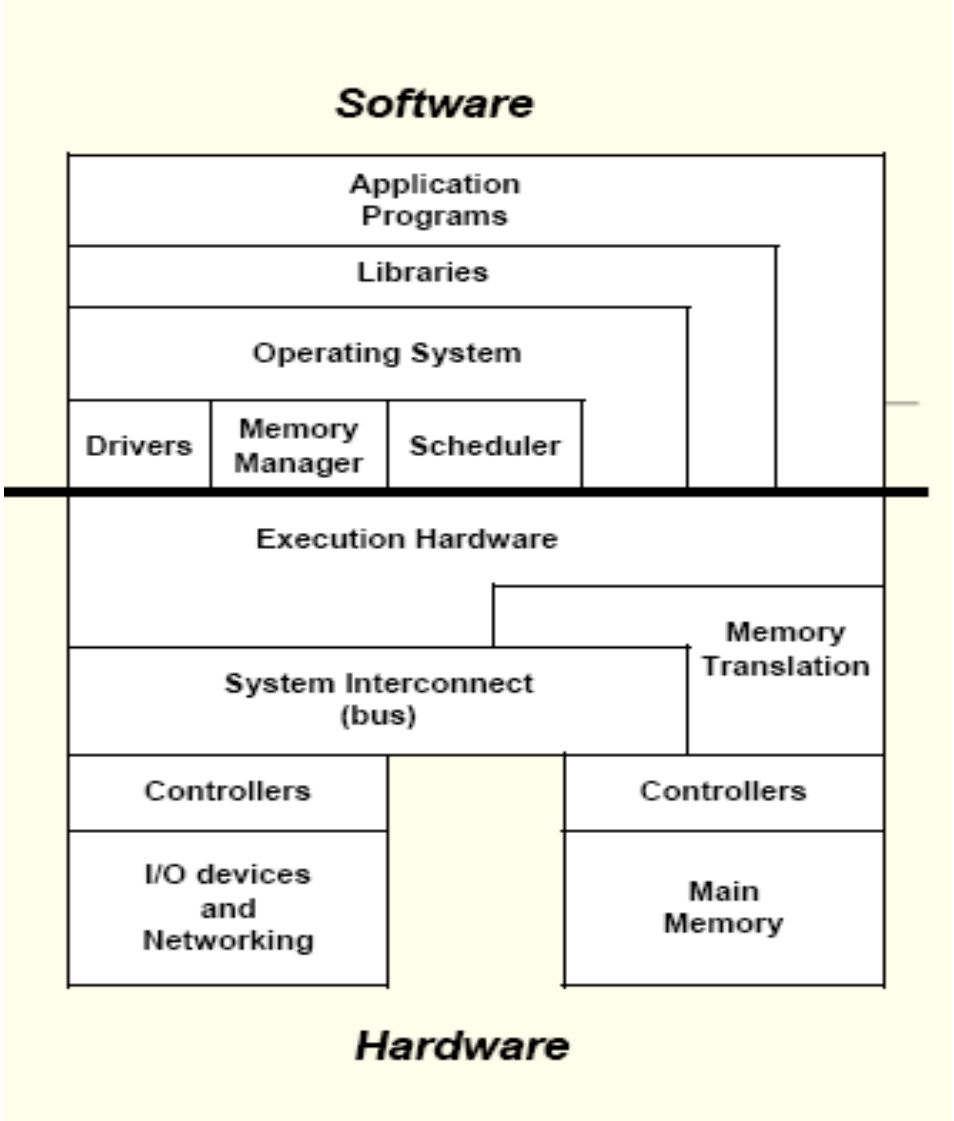
by Jim Smith, Ravi Nair

Publisher: Morgan Kaufmann (2005)

ISBN-10: 1558609105

Abstraction

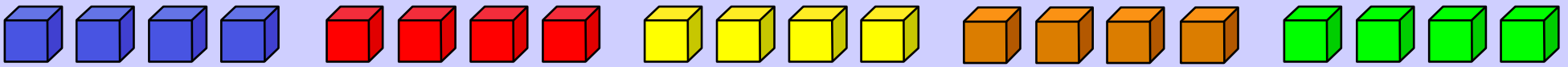
- Computer systems are built on levels of abstraction
- Higher level of abstraction hide details at lower levels
- Example: files are an abstraction of a disk



Definition

A concept that places an abstraction layer between resources and the user of the resources
so that the logical view is different from the physical view.

Virtualization Concept



Virtual Resources

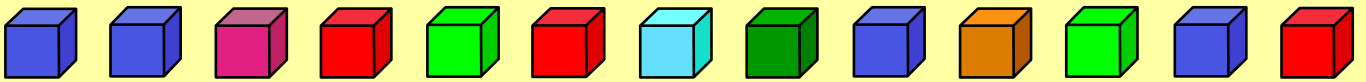
- ☒Substitutes for real resources: same interfaces/functions, different attributes.
- ☒Often of part of the underlying resource, but may span multiple resources.

Virtualization – a substitution process

- ☒Creates virtual resources from real resources.
- ☒Primarily accomplished with software and/or firmware.

Resources

- ☒Components with architected interfaces/functions.
- ☒Usually physical. May be centralized or distributed.
- ☒Examples: memory, disk drives, networks, servers.

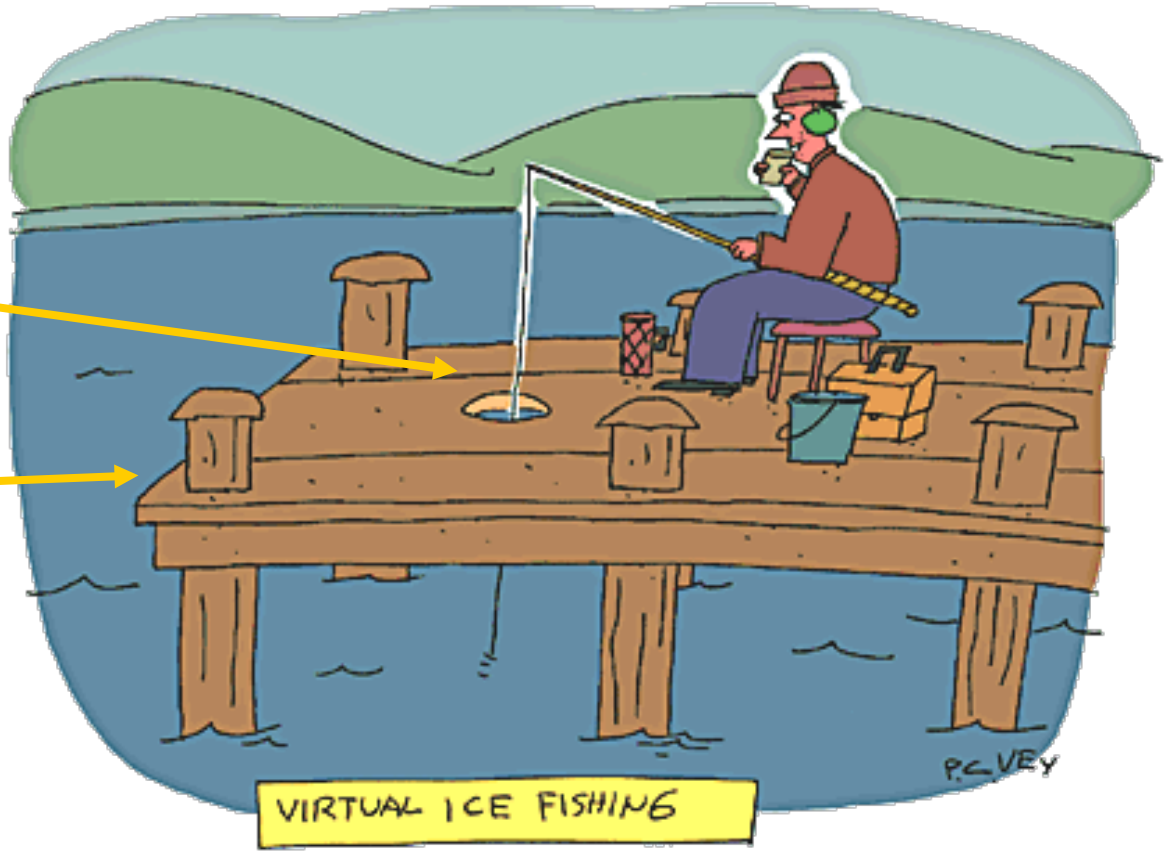


- ☒**Separates presentation of resources to users from actual resources**
- ☒**Aggregates pools of resources for allocation to users as virtual resources**

Virtualization Gives Users Idealized Resources

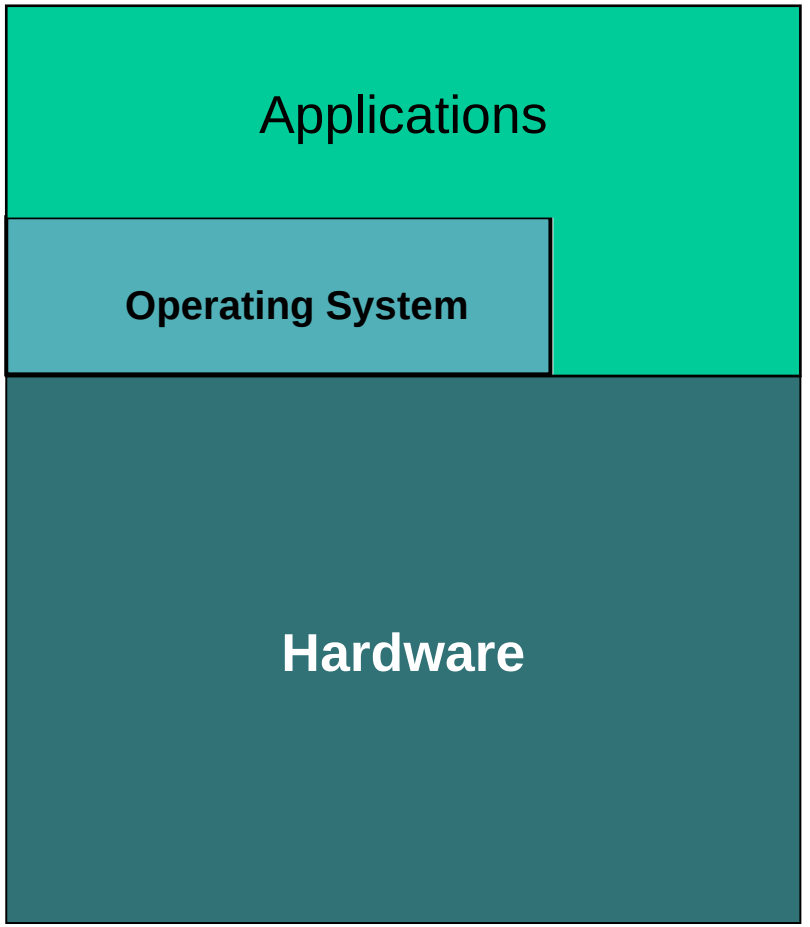
Architected Fishing Interface

Virtual Ice
Has better RAS than real ice



Virtualization…for what?

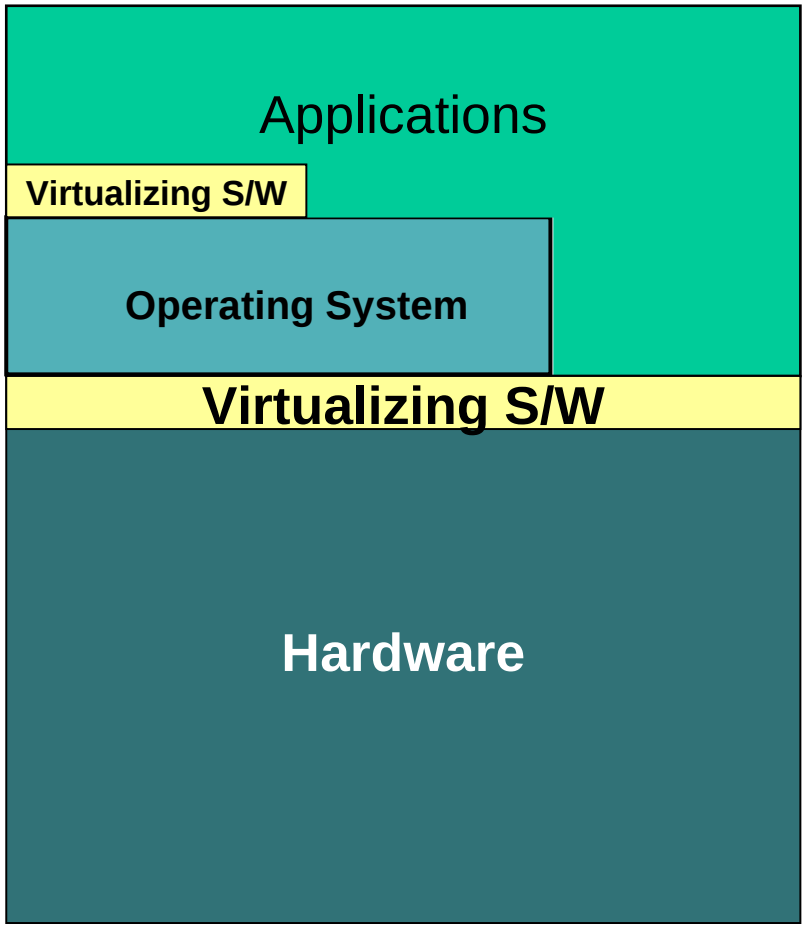
- Used for hiding the physical characteristics of computing resources from the way in which other systems, applications, or end users interact with those resources.
- For example, the operating system creates the illusion of multiple processes, each executing on its own processor with its own (virtual) memory.



Virtualization…How?

- Virtualization… How?
  - By adding a layer between execution stack layers.

Types of Virtualization

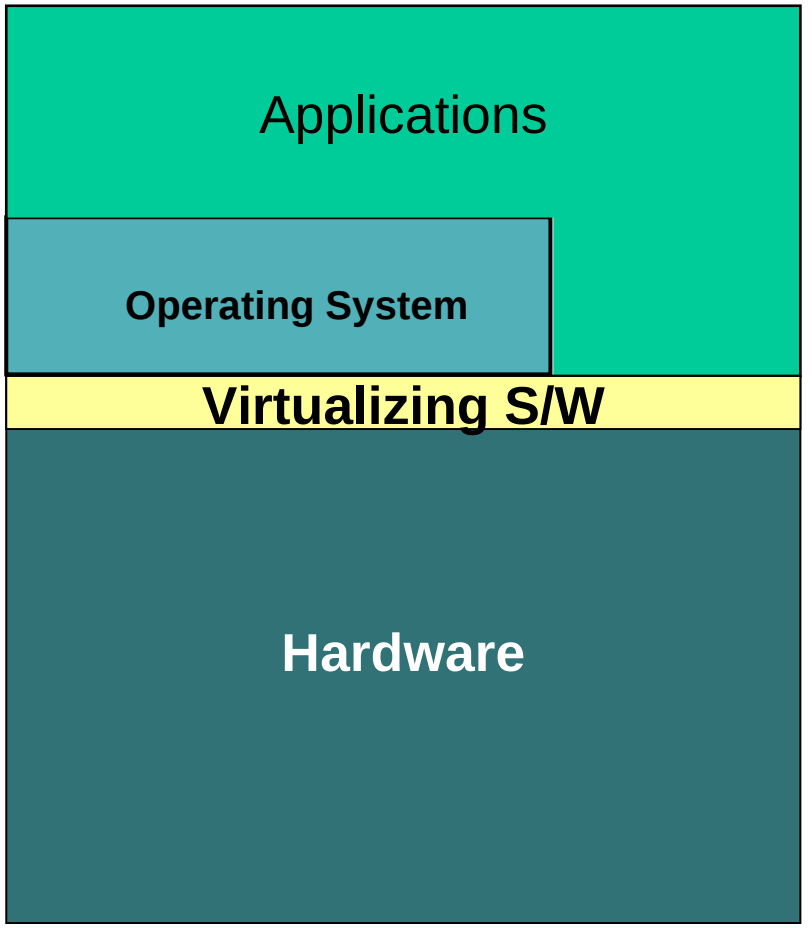


Virtualization…How?

Hardware-level virtualization

- VMM* is placed between the hardware and the OS.
- Could provide a different ISA.
- e.g. Emulators.
- Tasks:
  - Maps virtual resources to real ones.
  - Translate virtual instructions to real ones.

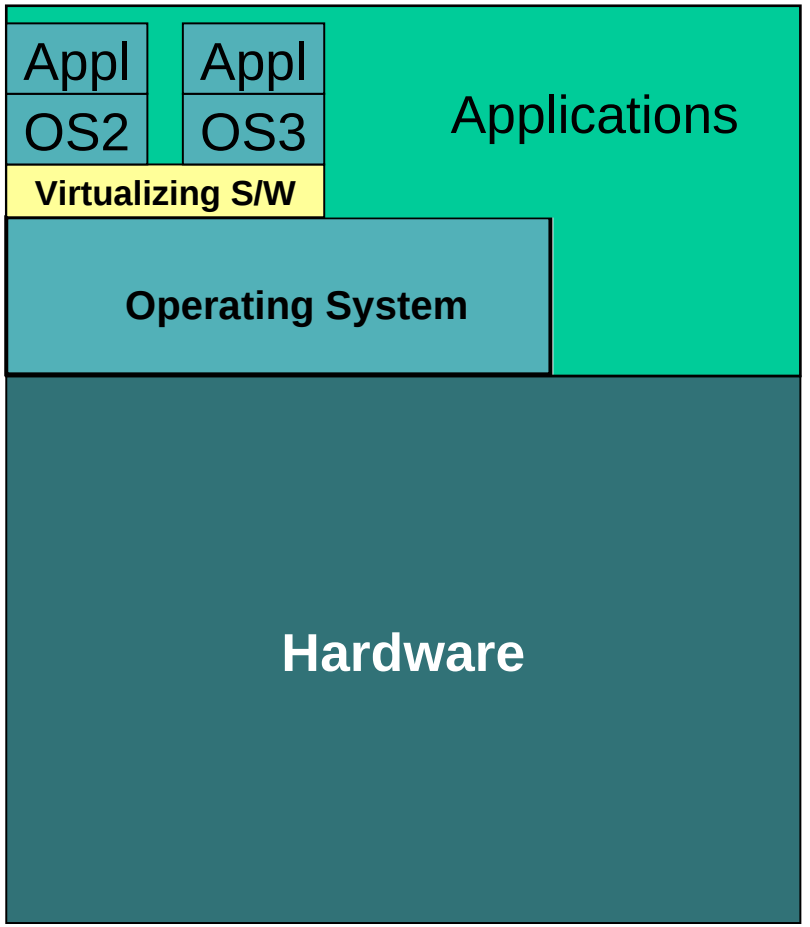
* VMM = Virtual Machine Monitor



Virtualization…How?

System-level virtualization

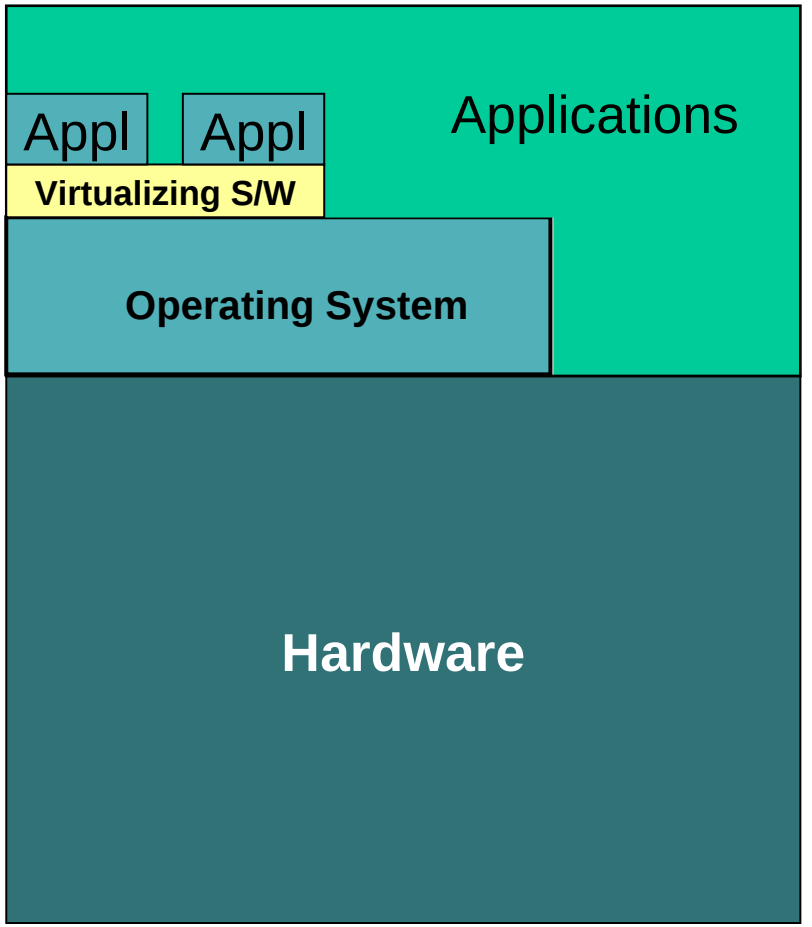
- placed between the OS and other OS
- e.g. VMware Wks and UML
- Enable several OS to run on single hardware.
- Terminology
  - Host OS - Guest OS
- Guest OS and VM run in Application privilege ring.



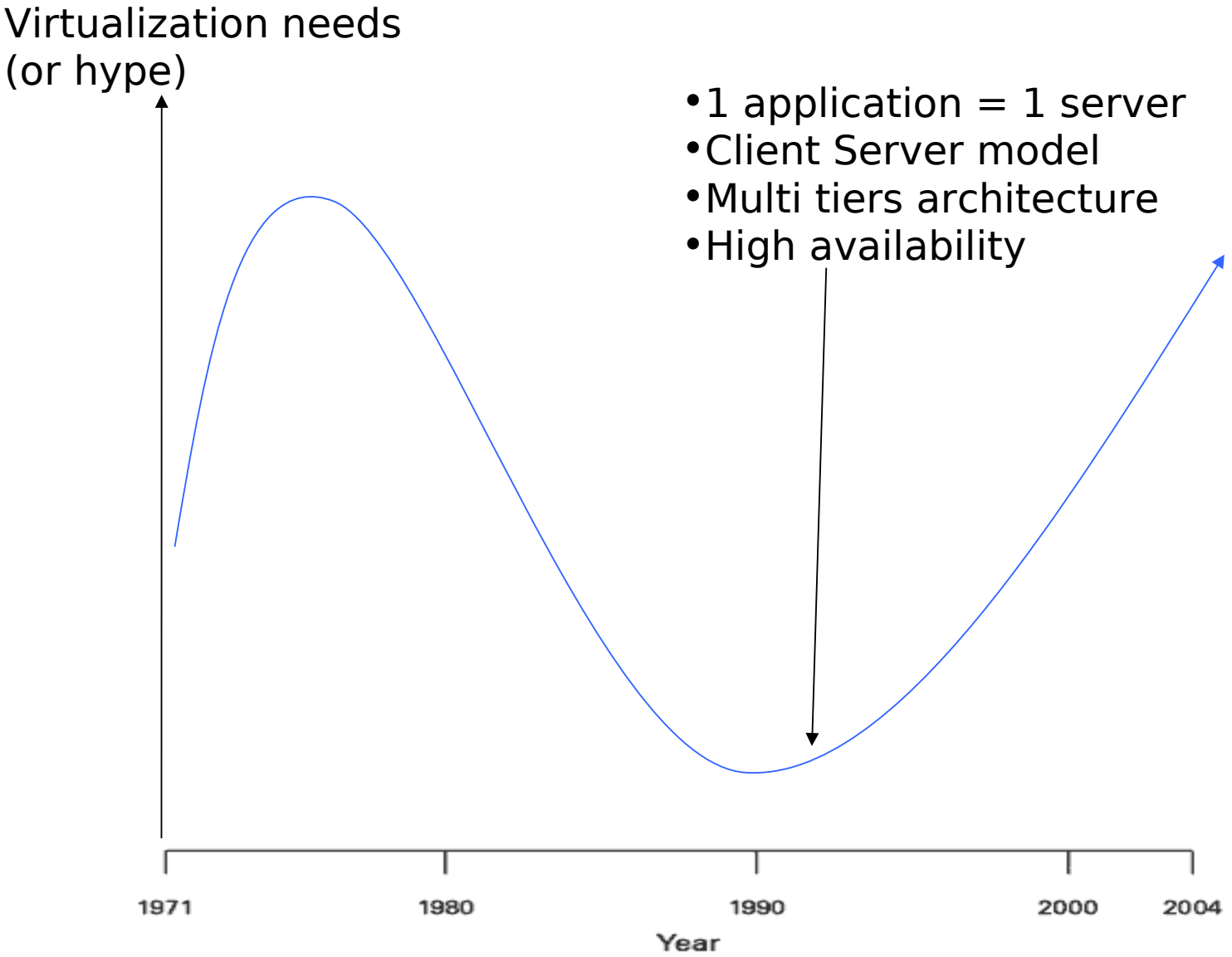
Virtualization…How?

Application-level virtualization

- VM is placed between the OS and the applications.
- e.g. Java Virtual Machine (JVM)
- Provide same interface to all Application, irrespective of OS.
- Provide Application Sand-boxing.
- Tasks:
  - Translate Application byte code to OS-specific executable.



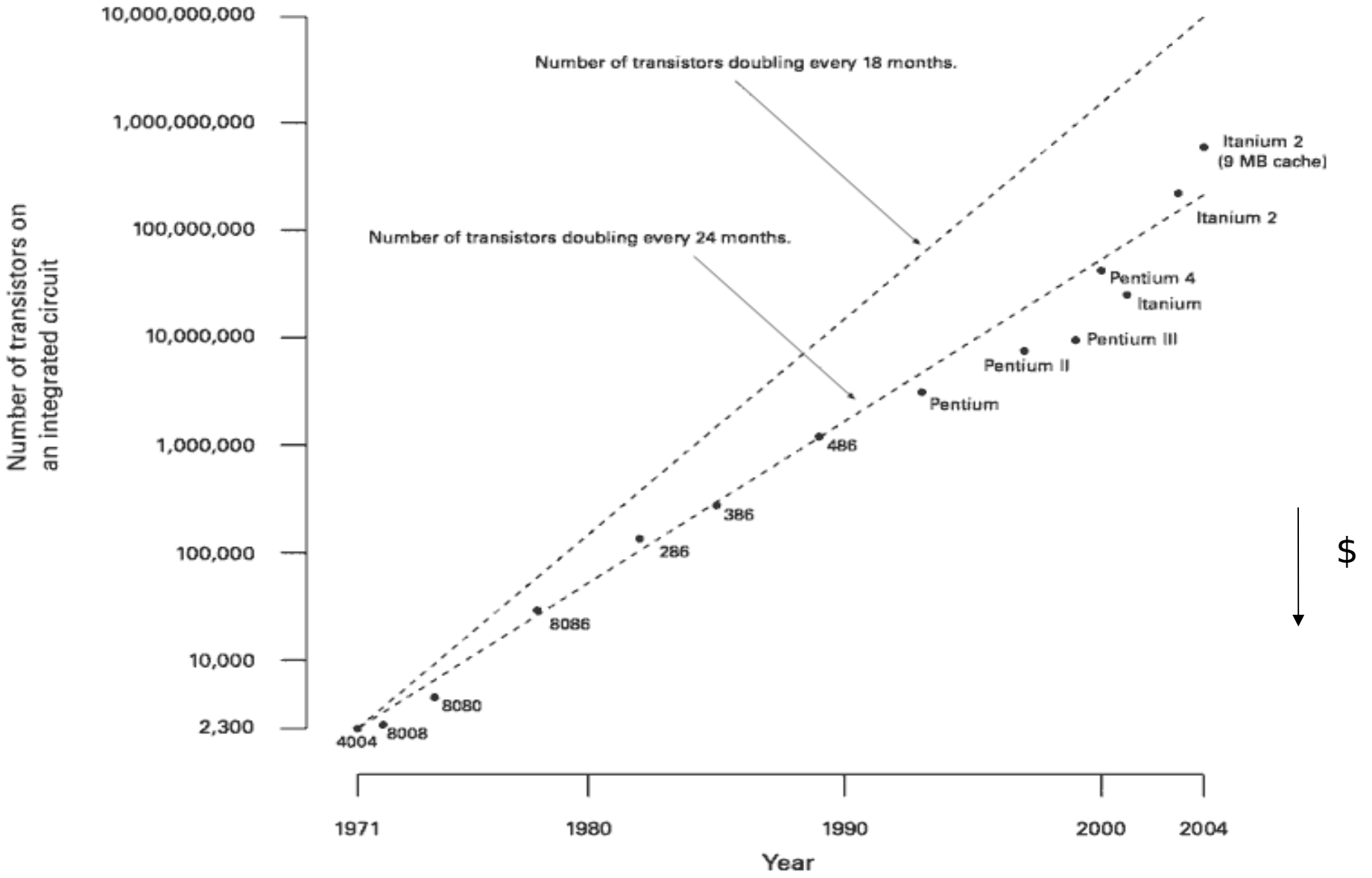
The Rise and Fall of Virtual Machines



The Rise and Fall of Virtual Machines

- 1970s: Appeared a good idea
  - IBM VM/370 –A VMM for IBM mainframes
  - Multiplex multiple OS environments on expensive hardware
  - Desirable when few machine around
- 1980s: Looks like a dumb idea
  - Hardware got cheap but wimpy
  - VMM neither desirable nor possible

Virtualization...Why?



Key Concepts: Native vs. Hosted

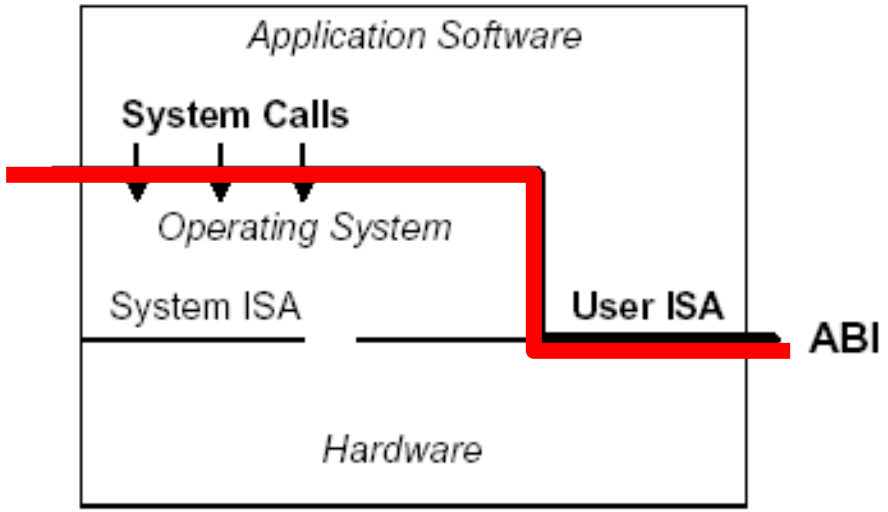
- Native VMs
  - VMM is installed on the bare machine, no host OS
  - All other VMs are then created through the VMM
  - Pros: Clean Architecture, Efficient
  - Cons: Complicated VMM due to device drivers
  - Example: VMware ESX Server

- Hosted VMs
  - VMM is installed on top of a host OS
  - User-mode: VMM runs in non-privileged mode
  - Dual-mode: VMM runs partly in privileged mode (as a driver on the host OS) and partly in unprivileged mode (like an application)
  - Pros: VMM uses drivers in the host OS for I/O → Thin VMM
  - Cons: Inefficient for I/O intensive applications
  - Example: Microsoft Virtual Server, VMware WKS

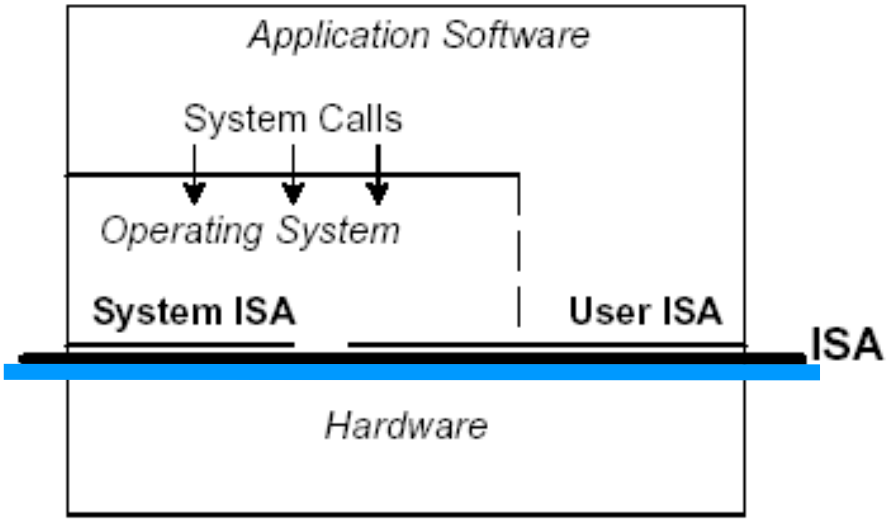
ISA and ABI

ABI - Application Binary Interface interfaces processes with environment

ISA - Instruction Set Architecture interfaces with hardware



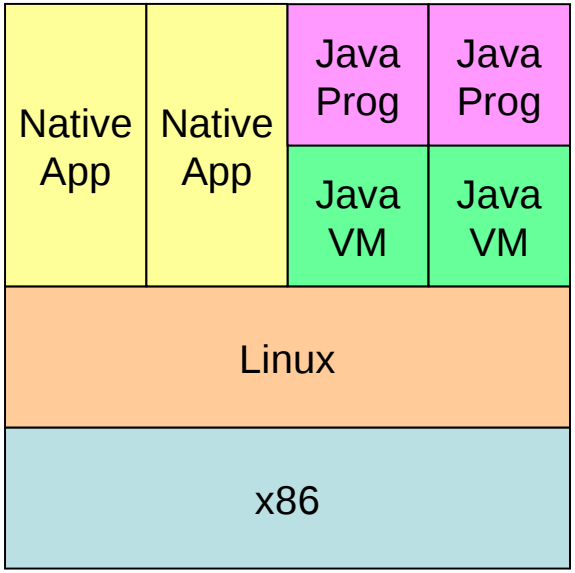
ABI



ISA

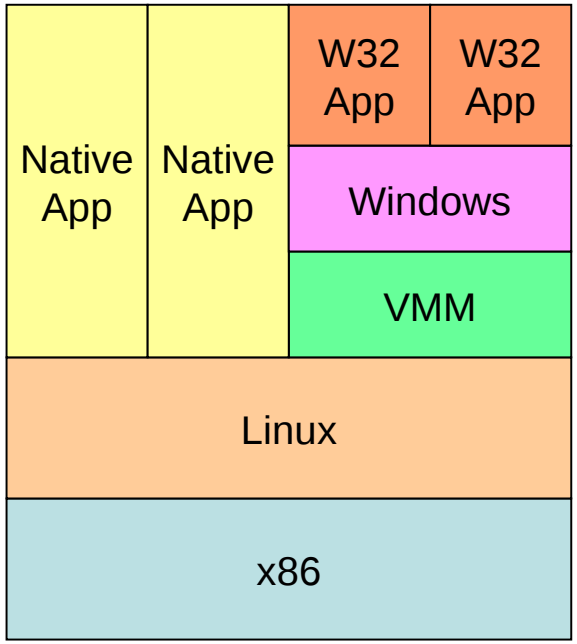
Key Concepts: Process vs. System

- There are two kinds of virtual machines: process and system.
  - Process virtual machine can support an individual process.
  - System virtual machine can run a complete OS plus environment.



Process VM

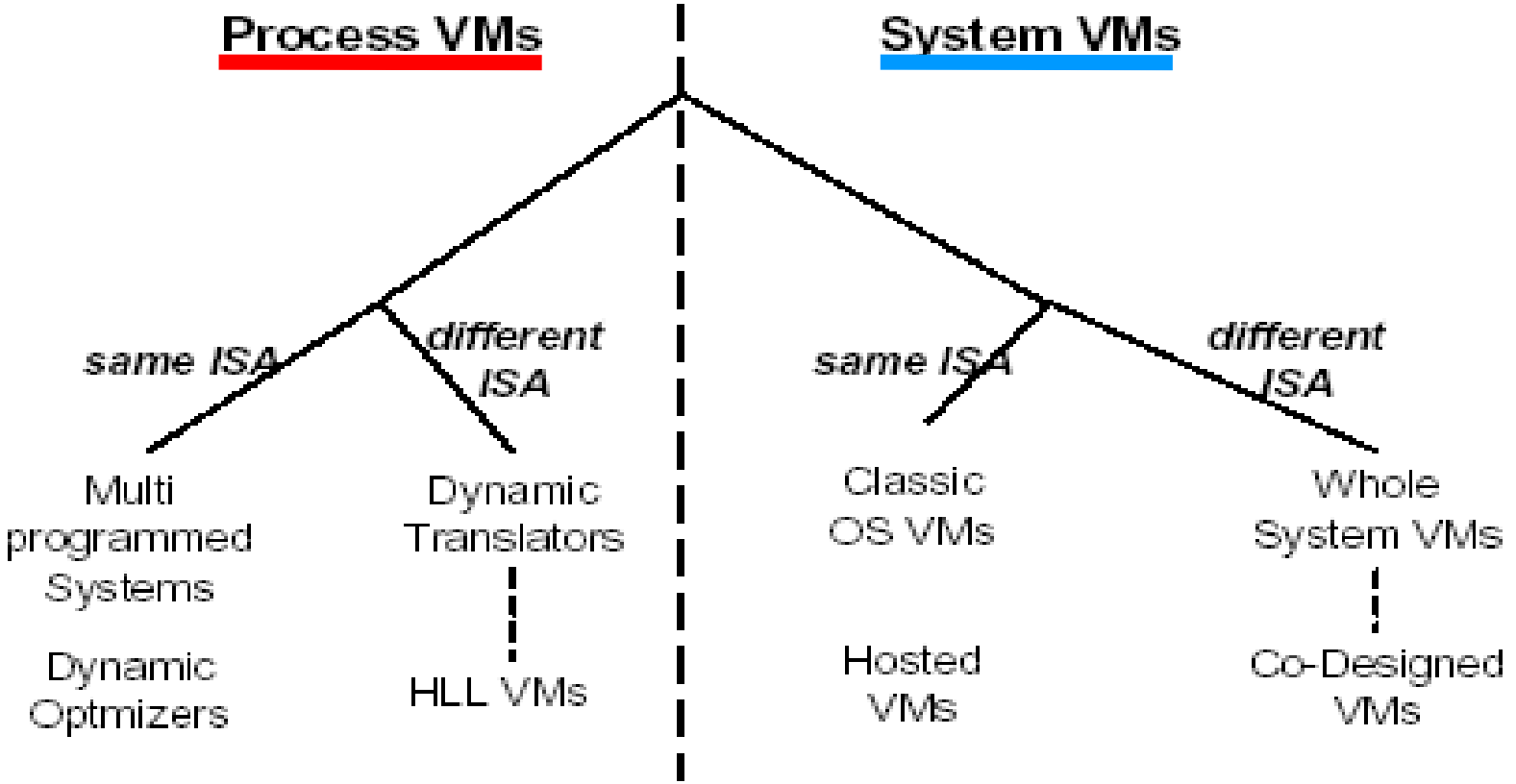
E.g. running an x86 application on a PowerPc



System VM

E.g. running an instance of Linux (and its applications) on Windows

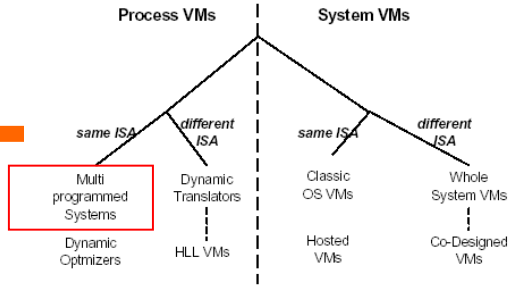
Taxonomy Overview



Exposed Interfaces:

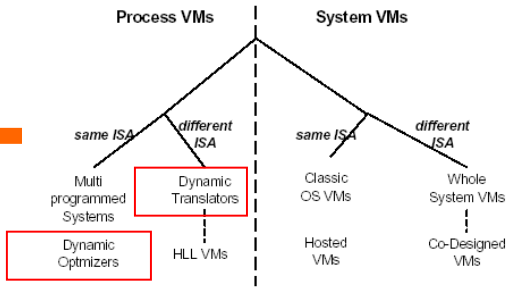
1. ABI
2. ISA

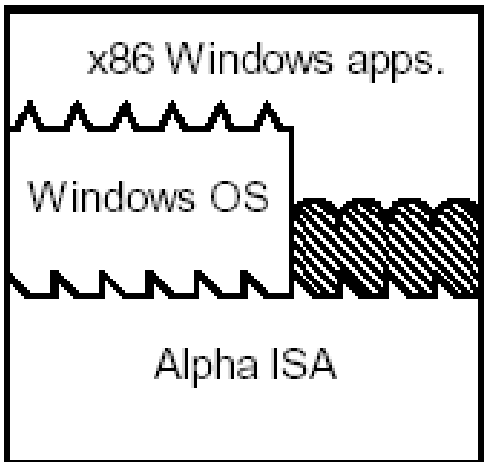
Process VMs (1)

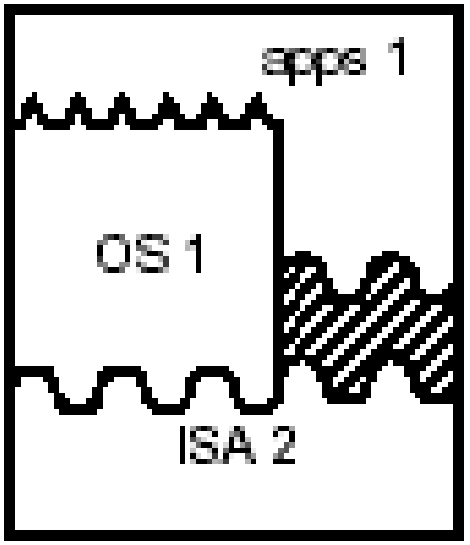


- Multiprogramming
  - same ISA, same OS
  - replicates ABI so that each process thinks it has its own machine
  - standard in “modern” OS
  - it can be argued whether to call this a VM

Process VMs (2)

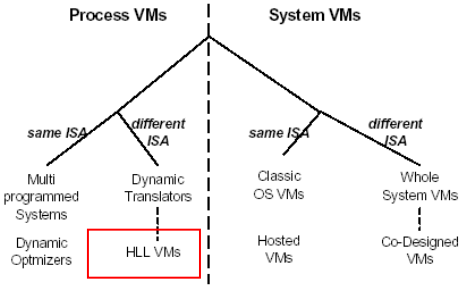






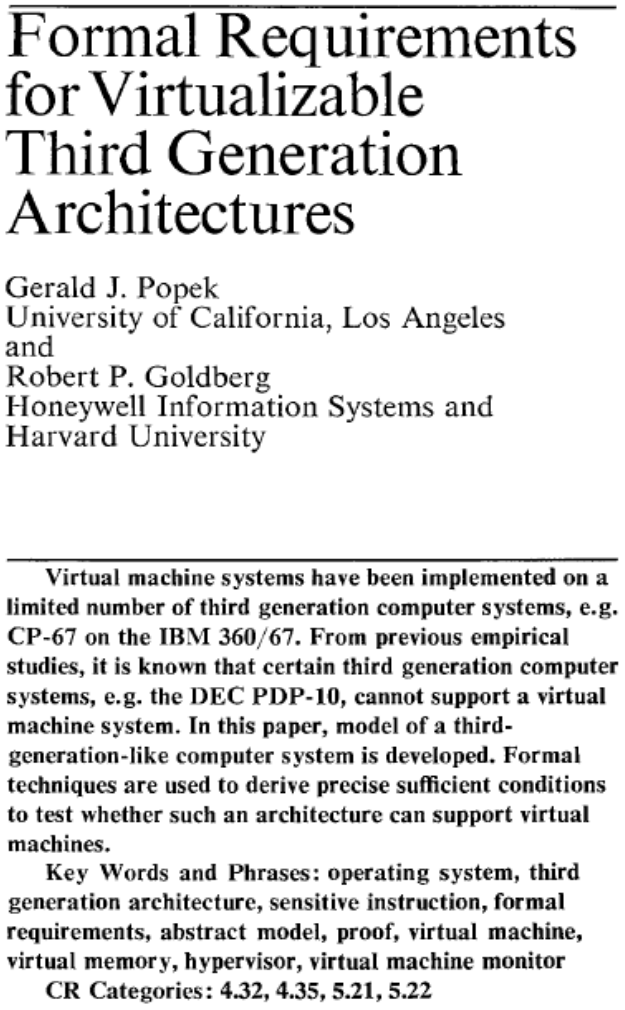
- "Emulation" and "Dynamic Binary Translation"
  - OS is same, ISA is different
  - better known as "interpretation" and "compilation"
- Dynamic Optimizers
  - same OS and ISA
  - Targeted to achieve better performances

Process VMs (3)



- High-level VMs
  - E.g. Java
  - Maximal platform independence
    - Except for OS calls, etc.
  - Performance penalty

References - I



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

Virtual machine systems have been implemented on a limited number of third generation computer systems, e.g. CP-67 on the IBM 360/67. From previous empirical studies, it is known that certain third generation computer systems, e.g. the DEC PDP-10, cannot support a virtual machine system. In this paper, model of a third-generation-like computer system is developed. Formal techniques are used to derive precise sufficient conditions to test whether such an architecture can support virtual machines.

Key Words and Phrases: operating system, third generation architecture, sensitive instruction, formal requirements, abstract model, proof, virtual machine, virtual memory, hypervisor, virtual machine monitor

CR Categories: 4.32, 4.35, 5.21, 5.22

Formal Requirements for Virtualizable Third Generation Architectures

By G.J. Popek, R. P. Goldberg

Communications of the ACM
July 1974, Vol 17, pages 412-421

Downloadable from: http://labs.vmware.com/download/75/

P&G Defined a Simplified Computer Model ...

The processor is a conventional one with two modes of operation, supervisor and user. In supervisor mode, the complete instruction repertoire is available to the precessor. In user mode, it is not. Memory addressing is done relative to the contents of a relocation register.

[...]

The machine can exist in any one of a finite number of states where each state has four components: executable storage E, processor mode M, program counter P, and relocation-bounds register R.

[...]

One key restriction in the model is the exclusion of I/O devices and instructions.

… its Virtual Machine Monitor (VMM) …

- Any program running under the VMM should exhibit an effect identical with that demonstrated if the program had been run on the original machine directly, except for:
  - Availability of resources (can be constrained)
  - Timing dependencies (issue with device drivers)
- A statistically dominant subset of the virtual processor's instructions are executed directly by the real processor.
  - Popek and Goldberg say that a virtual machine is different from an emulator. An emulator intervenes and analyzes every instruction performed by the virtual processor, whereas a virtual machine occasionally relinquishes the real processor to the virtual processor.
- The VMM is in complete control of system resources.

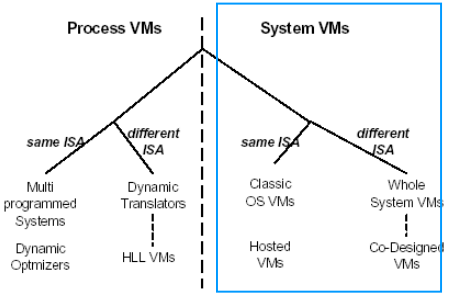
... the Concepts of Privileged Instructions ...

- Trap:
  - *An instruction is said to trap if its execution automatically saves the current state of the machine and passes control of a prespecified routine by changing the processor mode, the relocation bounds register R* [memory protection] *and the program counter to the values specified*
  - *A memory trap is a trap caused as a result of an attempt by an instruction to develop an address which is greater than the bounds in R*
- Privileged Instructions:
  - *An instruction is privileged if and only if for any pair of states S1 and S2 which differ only by the protection level and in which it does not memory trap, traps when executed in user mode only.*

... and Sensitive Instructions.

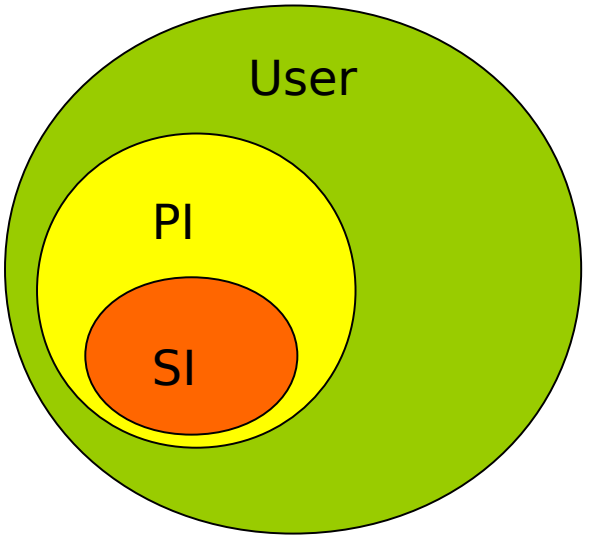
- Control Sensitive Instructions:
  - *An instruction is control sensitive if it attempts to change the amount of (memory) resources available, or affects the processor mode without going through the memory trap sequence*
- Behavior Sensitive Instructions:
  - *An instruction is behavior sensitive if the effect of its execution depends on the value of the relocation-bounds register, i.e. upon its location in real memory, or on the mode*
- Sensitive and innocuous instructions:
  - *An instruction is sensitive if it is either control sensitive or behavior sensitive. If it is not sensitive, then it is innocuous*

ISA Virtualizability

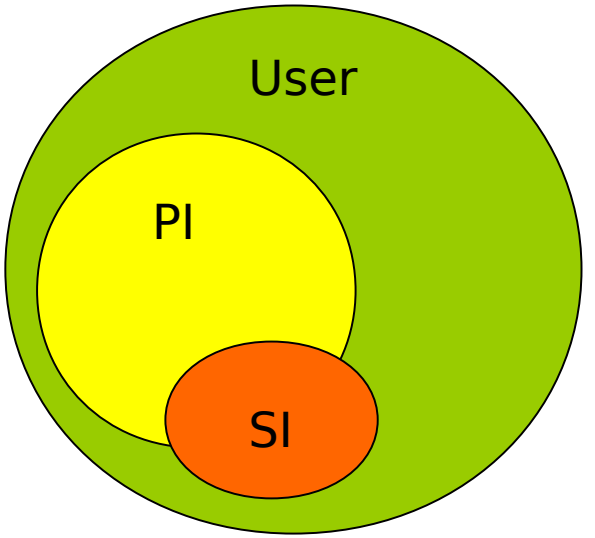


Popek/Goldberg Theorem:

A virtual machine monitor may be constructed if the set of SI for a given processor is a subset of PI

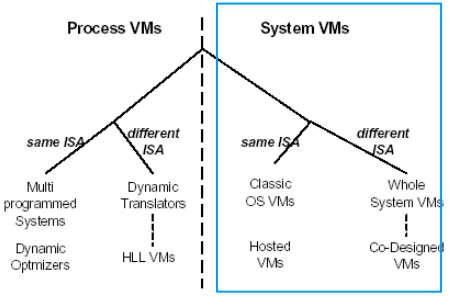


ISA is Virtualizable



ISA is “NOT” Virtualizable

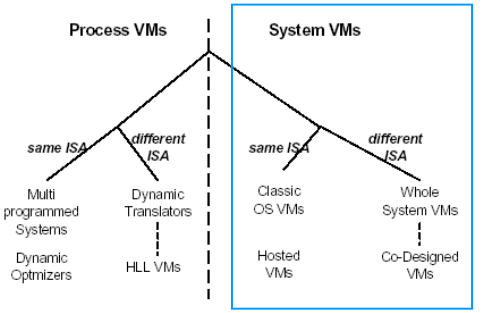
When ISA is not Virtualizable



- All is not lost if an ISA violates Popek/Goldberg requirement
  - However, it brings in additional complications and inefficiencies in implementation of VMM

- Critical instructions:
  - Instructions that are *sensitive* but not *privileged*
  - x86 has 17 critical instructions
  - All critical instructions must be trapped and emulated in the VMM

System VMs

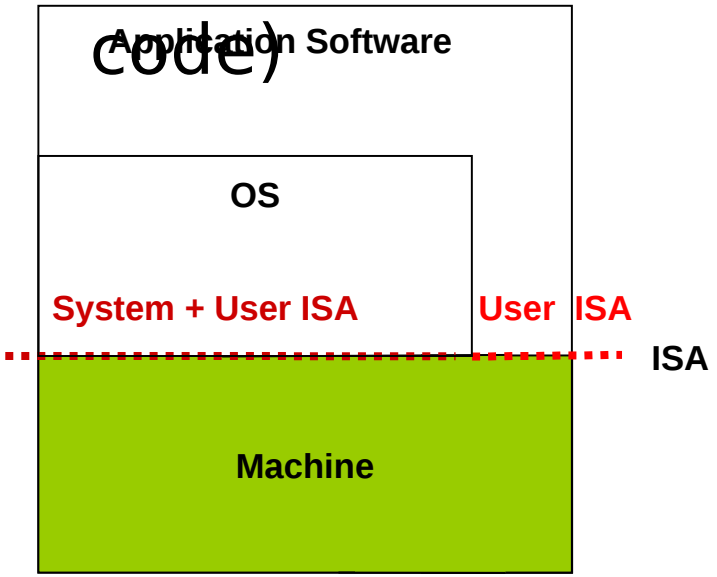


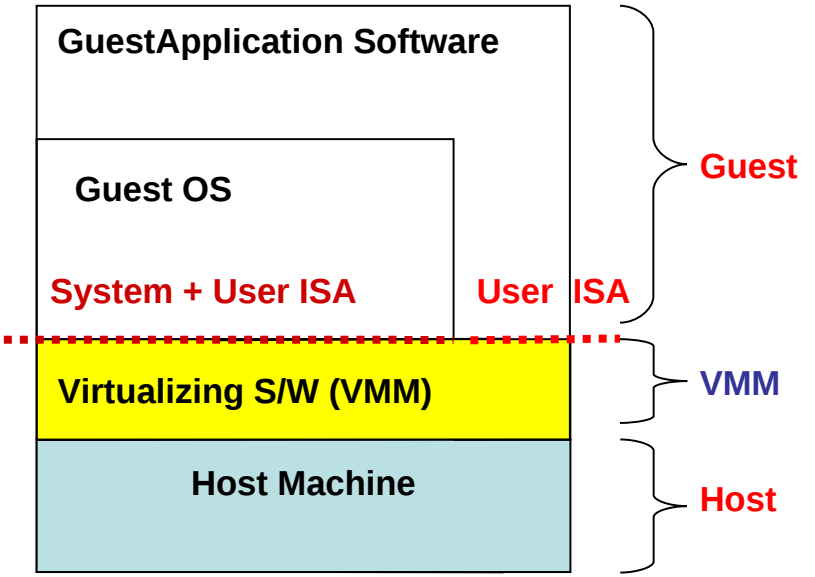
- Same ISA
  - "Classic"
    - VMM built directly on top of hardware.
    - Most efficient.
    - Requires device drivers in the VMM.
  - Hosted
    - VMM built on top of existing OS.
    - Easier to deploy
    - Devices drivers supplied by host OS, VMM uses facilities provided by host OS.

- Different ISA
  - Whole System VMs: Emulation
    - ISA not the same, must emulate everything.
  - Co-Designed VMs: Optimization
    - Hardware designed to support VMs.
    - Provides a clean design for virtualization.
    - Can be significantly more efficient.

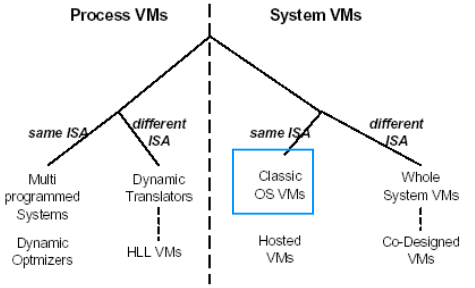
System VM

- Provides a complete guest system environment
- Run whole OS & executables
  - Virtualizes the ISA layer
  - VM, called a VMM, emulates both user-level & system-level ISA (i.e., both application code and OS code)

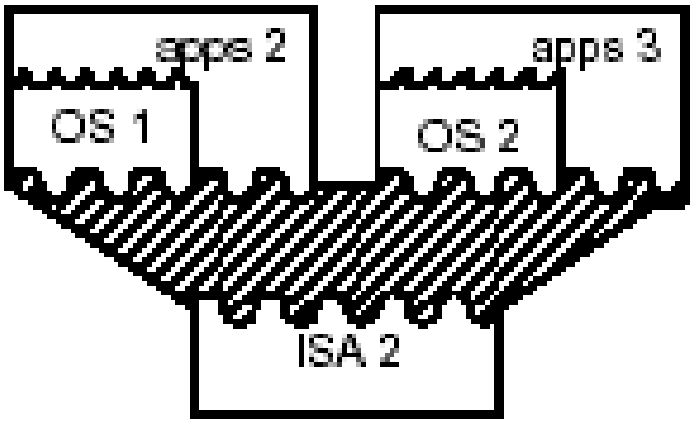




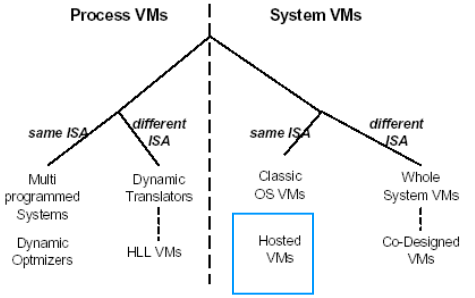
System VMs (1)



- Classic System VMs
  - VMM provides replication and resource management
  - possible benefits: flexibility, utilization, isolation
  - similar to what an OS does for processes
  - sits on hardware
  - (super)privileged mode



System VMs (2)

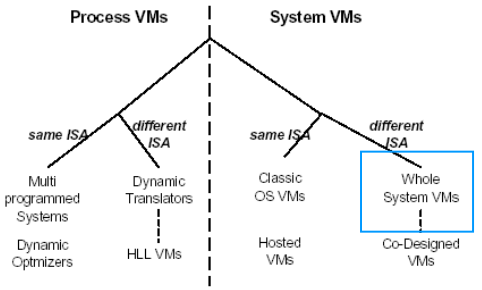


- Hosted System VMs
  - Like classic system VM, but operates within process space
  - Can play tricks to overcome limitations

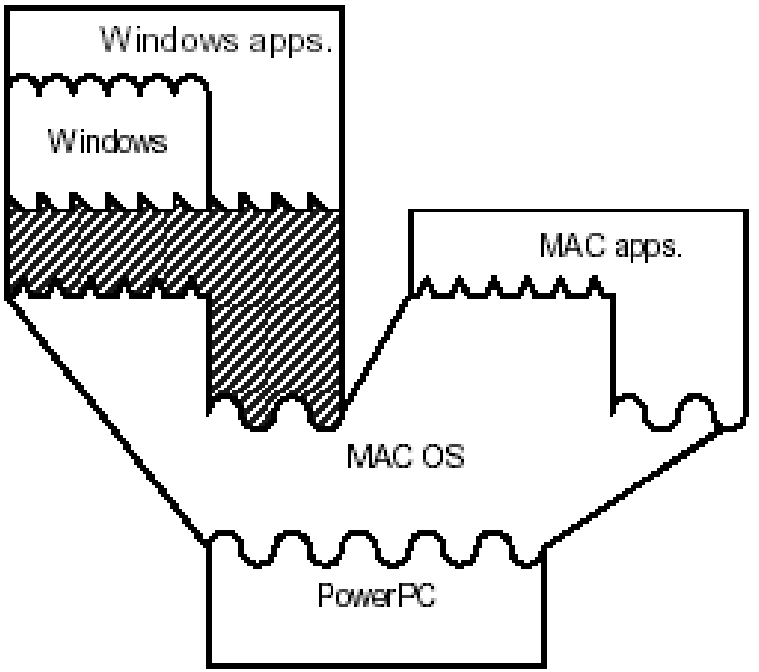
Common to Classic/Hosted System VM:

- try to do as much as you can natively

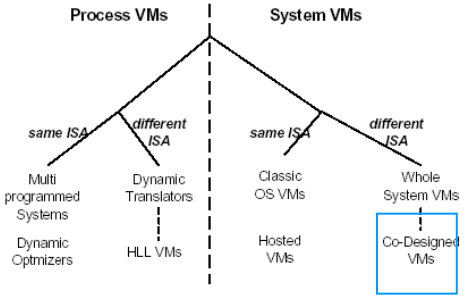
System VMs (3)



- Whole System VMs
  - ISA is different
  - no ‘native’ execution possib Complete emulation/translation required.
  - Usually done as a hosted VM

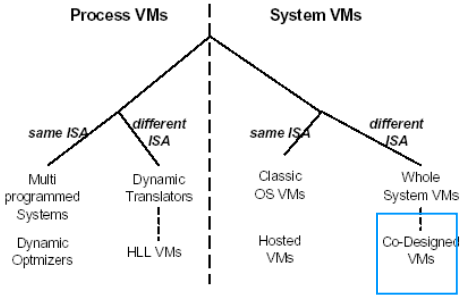


System VMs (4)

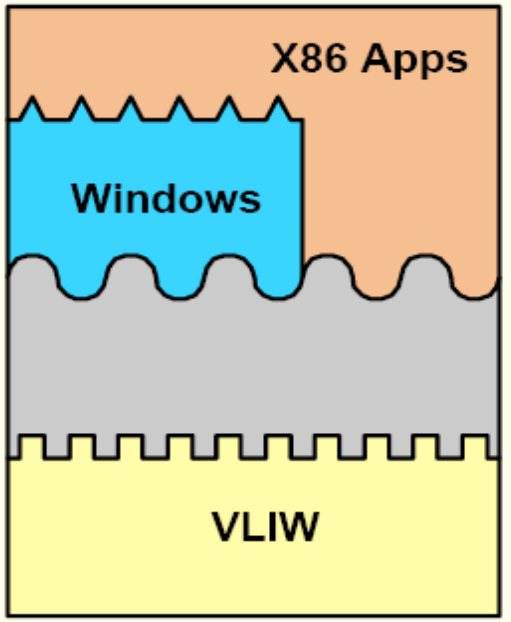


- Co-Designed VMs
  - use synthetic custom ISA at bottom
  - goal: reconcile diverging requirements between ISA and microarchitecture
  - no ‘native’ execution possible
  - Emulation/translation
    - joint effort by hardware and software
    - can be made completely transparent

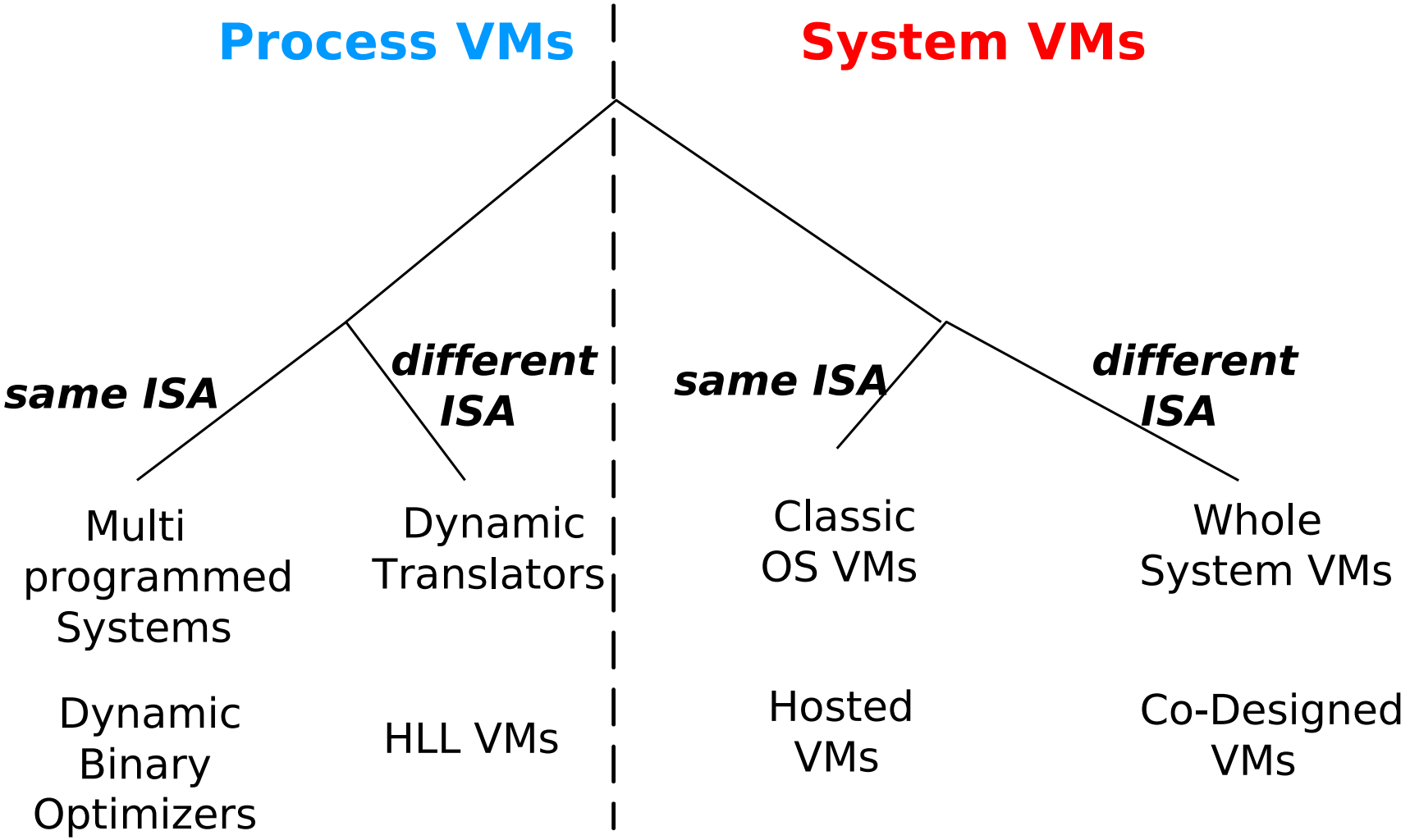
Co-designed Virtual Machines



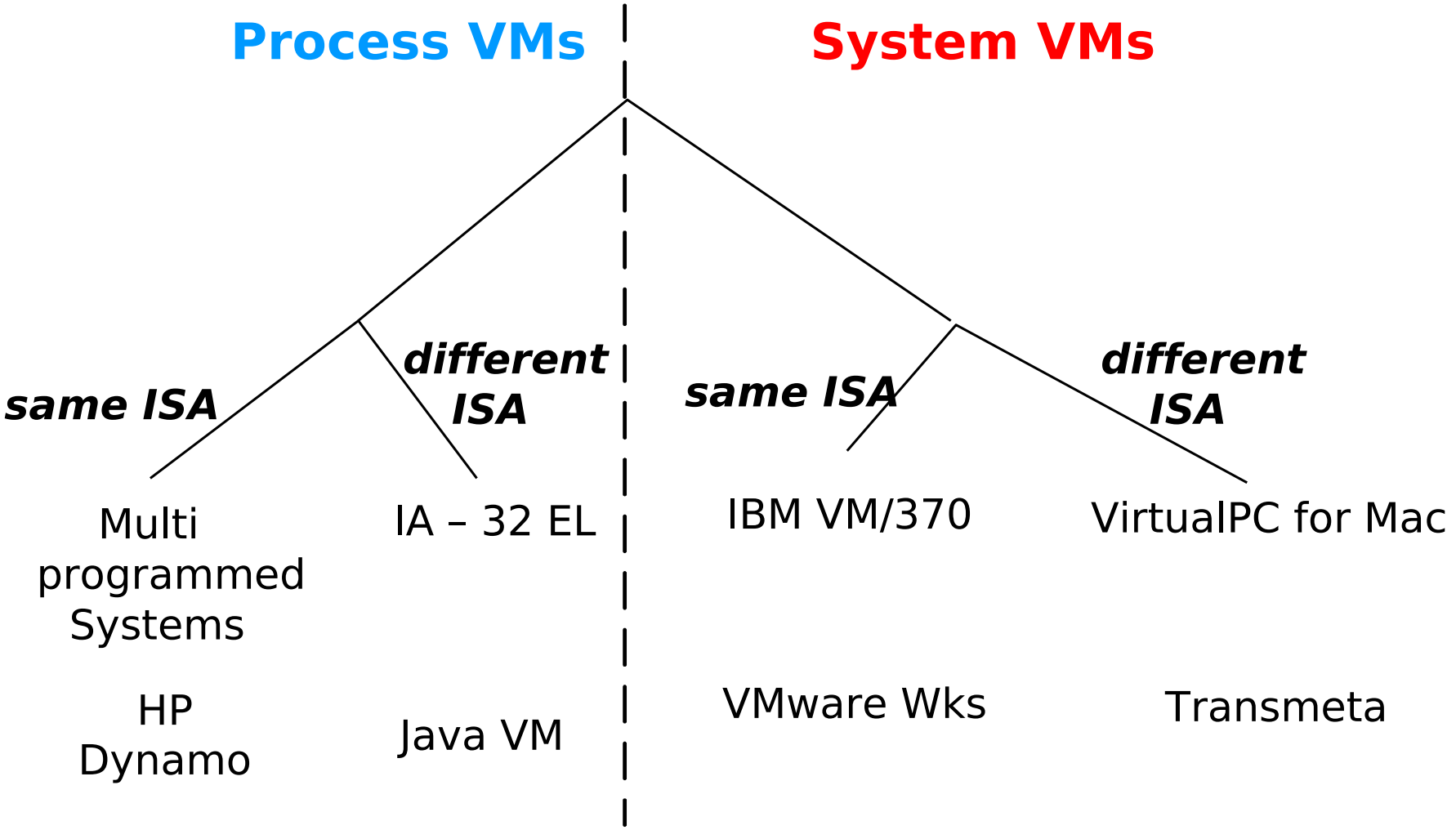
- Perform both translation and optimization
- Primary goal is performance or power efficiency
- VM provides interface between emulated and native ISA
- VM is a part of hardware design
- Example: Transme S/400, ...



Virtual Machine Architectures



Virtual Machine Architectures

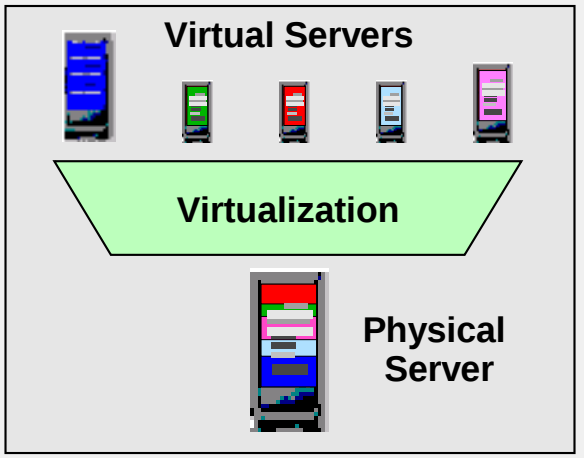


Virtualization: Why?

- Server consolidation
- Application Consolidation
- Sandboxing
- Multiple execution environments
- Virtual hardware
- Debugging
- Software migration (Mobility)
- Appliance (software)
- Testing/Quality Assurance

Server Consolidation

Roles:
- Consolidations
- Dynamic provisioning / hosting
- Workload management
- Workload isolation
- Software release migration
- Mixed production and test
- Mixed OS types/releases
- Reconfigurable clusters
- Low-cost backup servers



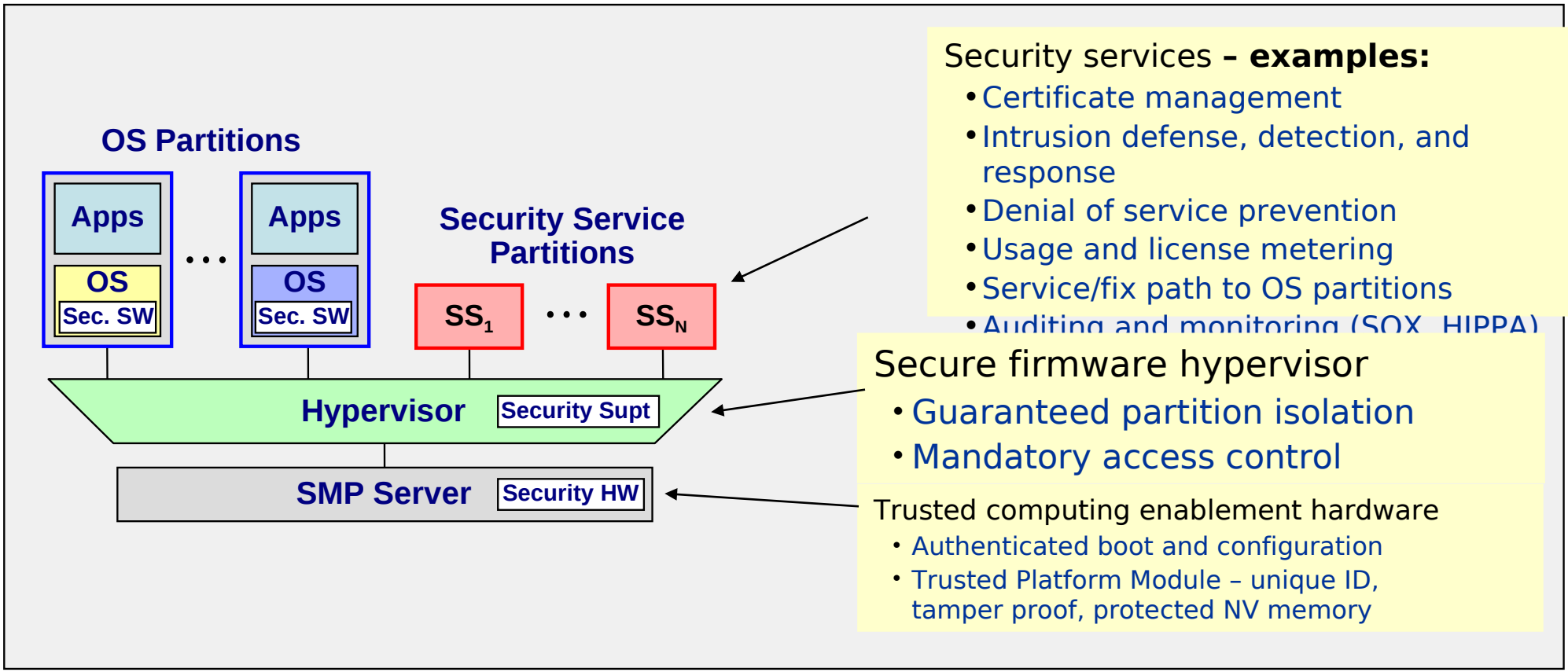
Benefits:
- Higher resource utilization
- Greater usage flexibility
- Improved workload QoS
- Higher availability / security
- Lower cost of availability
- Lower management costs
- Improved interoperability
- Legacy compatibility
- Investment protection

In the final analysis, the virtualization benefits take three forms:

- **Reduced hardware costs**
  - Higher physical resource utilization with smaller footprints.
- **Improved flexibility and responsiveness**
  - Virtual resources can be adjusted dynamically to meet new or changing needs.
  - Virtualization is a key enabler of on demand operating environments.
- **Reduced management costs**
  - Fewer physical servers to manage.

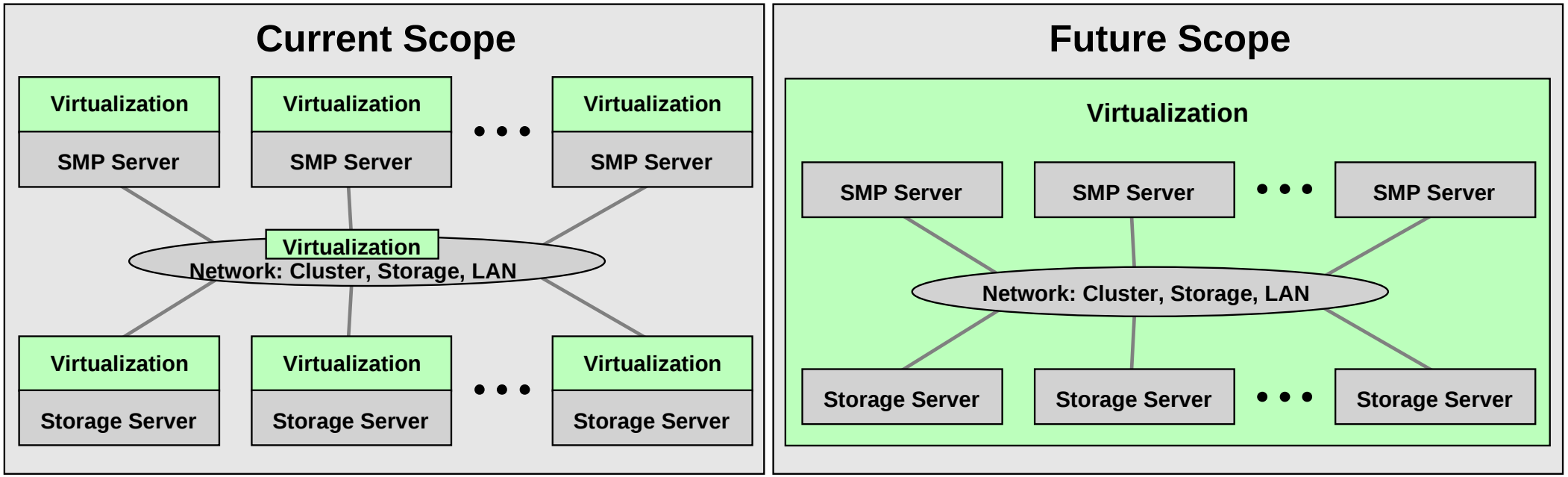
Emerging I/T Virtualization

Security Role of Hypervisors



- General purpose operating systems are a weak foundation for secure computing
  - Large and complex with many latent bugs
  - Constantly changing – security certifications are quickly made irrelevant
  - TCP/IP stacks make them vulnerable to attack by viruses, worms, and hackers
- Hypervisors will be used to establish a solid foundation for secure computing
  - Small enough to be fully inspected and certified

The Scope Of Virtualization Will Increase



- Server virtualization will be extended in scope from single servers to aggregations of servers, storage, and network components.
- Traditional view of virtualization:
  - Make a large system look like many – partitioning technology
- Evolving to the next level:
  - Make many small systems look like one from a management perspective

Security Is Limited By The Weakest Link