Virtualization Technologies

Basic Idea

- Observation
  - Hardware resources are typically under-utilized
  - Hardware resources directly relate to cost
- Goal
  - Improve hardware utilization
- How
  - Share hardware resources across multiple machines
  - May make sense for network attached storage, but what about processor, memory, etc.?
- Approach
  - Decouple machine from hardware
- Virtual Machine (VM)
  - A machine decoupled from the hardware, i.e. does not necessarily correspond to the hardware
  - Multiple "Virtual Machines" on the same physical host could share the underlying hardware
  - First VM: IBM System/360 Model 40 VM [1965]

Why Virtualize?

- Consolidate resources
  - Server consolidation
  - Client consolidation
- Improve system management
  - For both hardware and software
  - From the desktop to the data center
- Improve the software lifecycle
  - Develop, debug, deploy and maintain applications in virtual machines
- Increase application availability
  - Fast, automated recovery

Consolidate resources

- 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

Improve system management

- Data center management
  - VM portability and live migration a key enabler
  - automate resource scheduling across a pool of servers
  - optimize for performance and/or power consumption
  - allocate resources for new applications on the fly
  - add/remove servers without application downtime
- Desktop management
  - centralize management of desktop VM images
  - automate deployment and patching of desktop VMs
  - run desktop VMs on servers or on client machines
- Industry-cited 10x increase in sysadmin efficiency

Improve the software lifecycle

- Develop, debug, deploy and maintain applications in virtual machines
- Power tool for software developers
  - record/replay application execution deterministically
  - trace application behavior online and offline
  - model distributed hardware for multi-tier applications
- Application and OS flexibility
  - run any application or operating system
- Virtual appliances
  - a complete, portable application execution environment

Increase application availability

- Fast, automated recovery
  - automated failover/restart within a cluster
  - disaster recovery across sites
  - VM portability enables this to work reliably across potentially different hardware configurations
- Fault tolerance
  - hypervisor-based fault tolerance against hardware failures [Bressoud and Schneider, SOSP 1995]
  - run two identical VMs on two different machines, backup VM takes over if primary VM's hardware crashes
  - commercial prototypes beginning to emerge (2008)

ISTITUTO DI SCIENZA E TECNOLOGIE
DELL'INFORMAZIONE "A. FAEDO"

Background

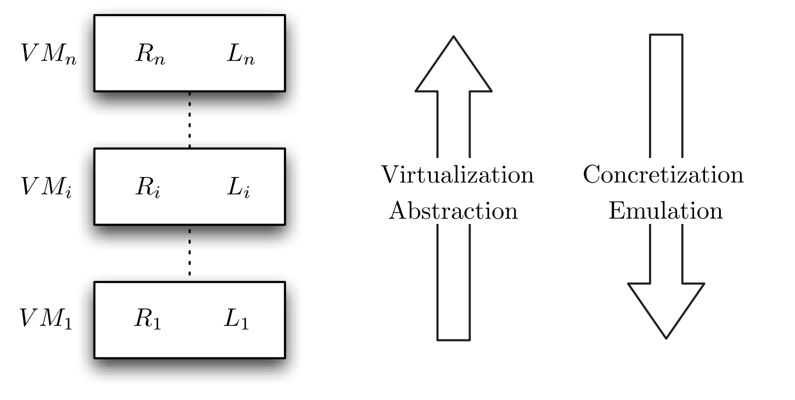
Modern Computer System

- Modern computer system is very complex
  - Hundreds of millions of transistors
  - Interconnected high-speed I/O devices
  - Networking infrastructures
  - Operating systems, libraries, applications
  - Graphics and networking software
- To manage this complexity: **Levels of Abstractions**
  - Allows implementation details at lower levels of design to be ***ignored*** or ***simplified***
  - Each level is separated by well-defined interfaces, so that the design of a higher level can be decoupled from the lower levels

ISTITUTO DI SCIENZA E TECNOLOGIE DELL'INFORMAZIONE "A. FAEDO"

Layers of Abstraction

- Abstraction
  - used to manage complexity
  - typically defined in layers (VM_i)
  - each layer has its own language (L_i) and data structures (R_i)
  - lowest layers implemented in hardware
  - higher layers implemented in software



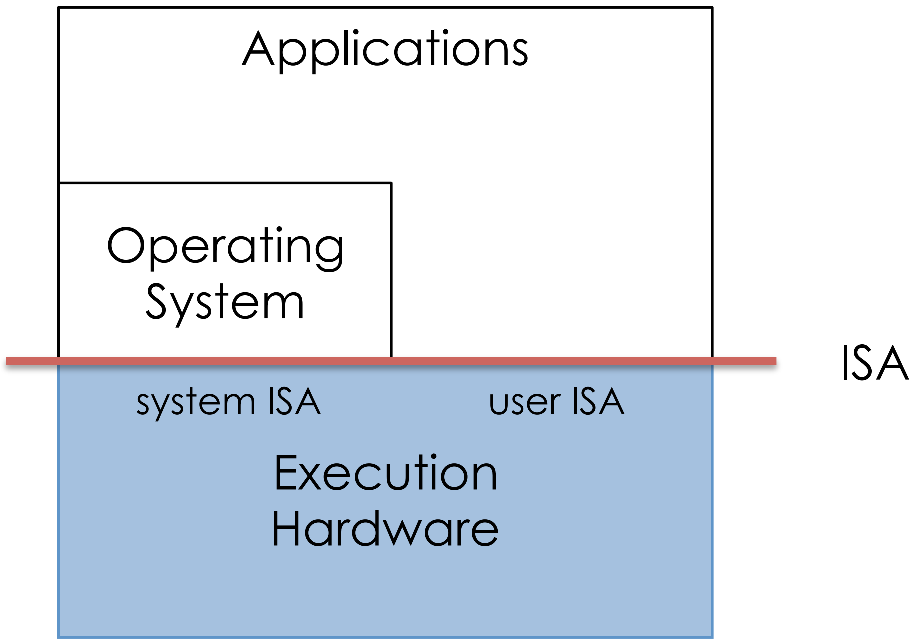
- Machine: denotes the system on which software is executed.
  - to an operating system this is generally the physical system
  - to an application program a machine is defined by the combination of hardware and OS-implemented abstractions

- Typical Layers
  - VM_4: Applications
  - VM_3: Operating System
  - VM_2: Assembler Machine
  - VM_1: Firmware Machine
  - VM_0: Hardware Machine

ISTITUTO DI SCIENZA E TECNOLOGIE DELL'INFORMAZIONE "A. FAEDO"

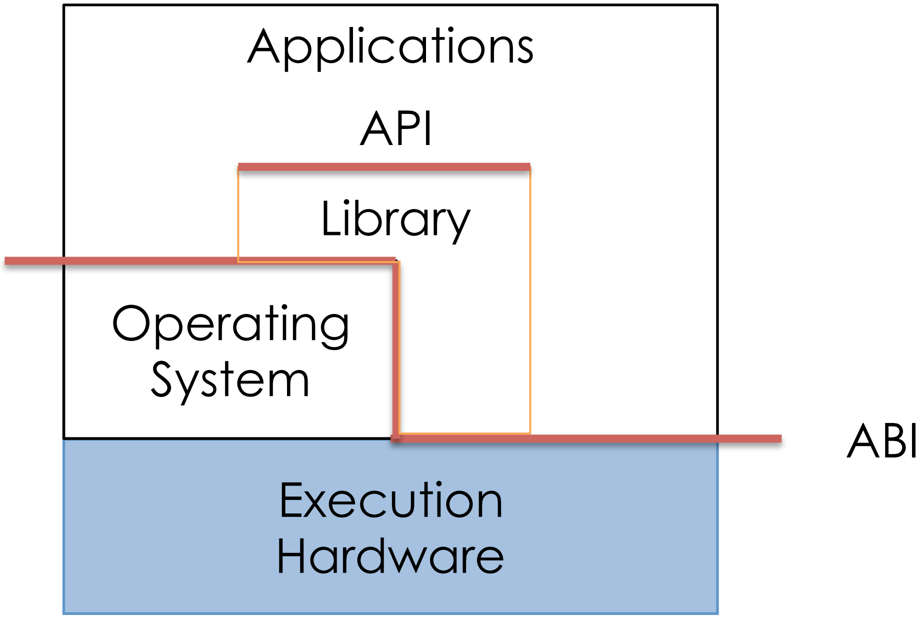
Interfaces (I)

- Abstraction layers have well defined interfaces
  - A processors instruction set defines such an interface: IA-32, IBM PowerPC, ARM
- Instruction Set Architecture (ISA)
  - defines hardware/software boundary
  - *user ISA*: portion of architecture visible to an application program
  - *system ISA*: portion of architecture visible to the supervisor software (e.g., OS)



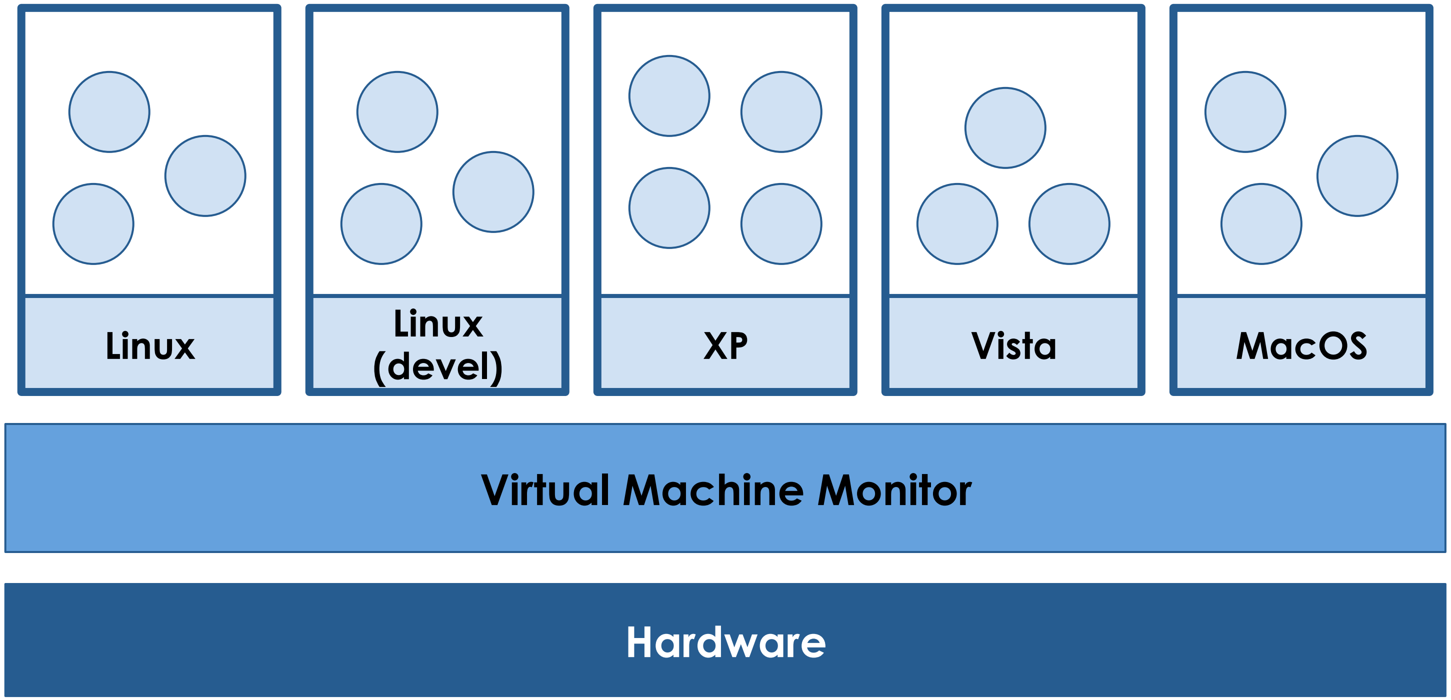
Interfaces (II)

- Application Binary Interface (ABI)
  - defines program interface to the hardware resources and services
  - user ISA
    - system instructions are not included in the ABI
    - user instructions allow program direct access to hardware
  - system calls
    - indirect interface for accessing shared system resources and services
    - implemented by the supervisor software
- Application Programming Interface (API)
  - defined in terms of a high-level language (HHL)
  - typically implemented as a system library and defined at the source level (for example libc which is linked into program's address space)
  - specifies operations available by system which are implemented by the operating system or other system software



Virtualization

What is Virtualization



General concept

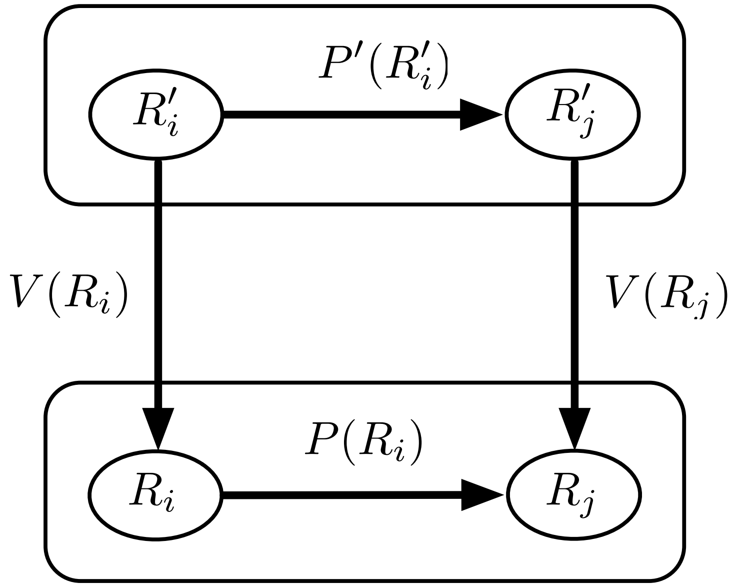
Construction of an isomorphism that maps a **guest** VM to an existing **host** VM such that:

– maps the guest state R_i (collection of guest virtualization objects) onto the host state R_i' through some function V() such that

$$V(R_i) = R_i'$$

– for every policy P() transforming the state R_i in state R_j in the guest, there is a corresponding policy P'() in the host that performs an equivalent modification of the host state

$$P' \circ V(R_i) = V \circ P(R_i)$$



UNIVERSITÀ DI PISA
ISTITUTO DI SCIENZA E TECNOLOGIE DELL'INFORMAZIONE "A. FAEDO"

Virtualization Properties

- Isolation
- Encapsulation
- Interposition

Isolation

- Fault Isolation
  – Fundamental property of virtualization
- Software Isolation
  – Software versioning
  – DLL Hell
- Performance Isolation
  – Accomplished through scheduling and resource allocation

Encapsulation

- All VM state can be captured into a file
  - Operate on VM by operating on file
  - mv, cp, rm
- Complexity
  - Proportional to virtual HW model
  - Independent of guest software configuration

Interposition

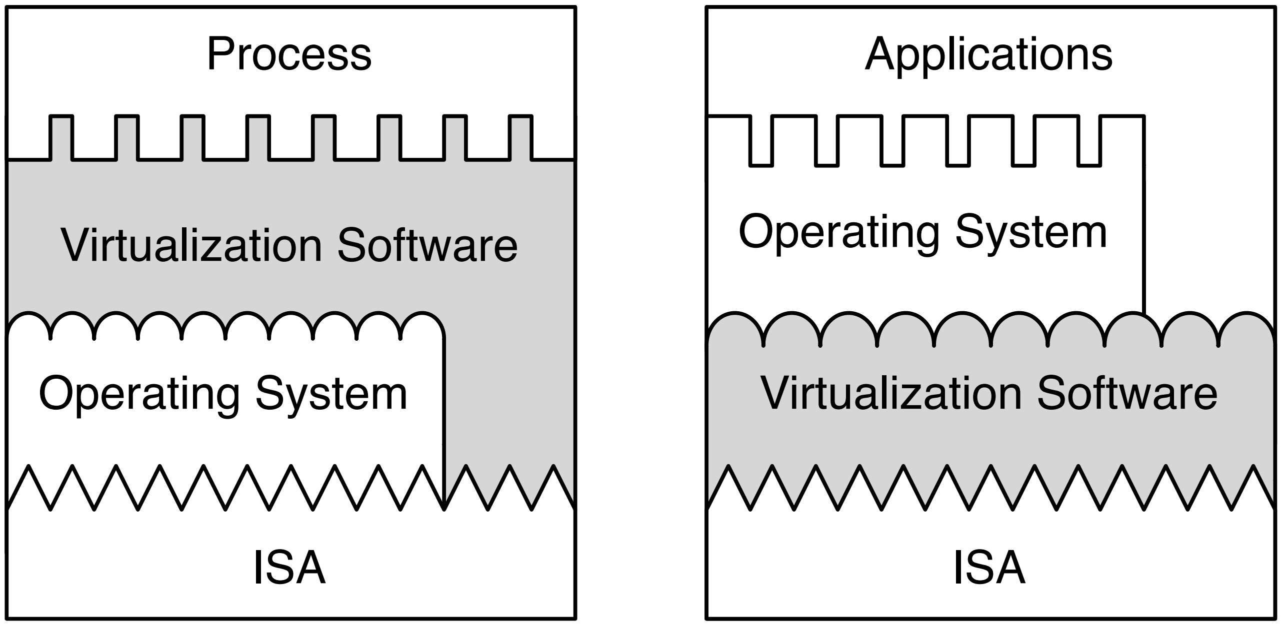
- All guest actions go through monitor
- Monitor can inspect, modify, deny operations
- Examples:
  - Compression
  - Encryption
  - Profiling
  - Translation

Concrete Perspectives

- *Process perspective*: The system ABI defines the interface between the process and machine
  - user-level hardware access: logical memory space, user-level registers and instructions
  - OS mediated: Machine I/O or any shared resource or operations requiring system privilege.
- *Operating system perspective*: ISA defines the interface between OS and machine
  - system is defined by the underlying machine
  - direct access to all resources
  - manage sharing
- *Virtual machine* executes software (process or operating system) in the same manner as target machine
  - Implemented with both hardware and software
  - VM resources may differ from that of the physical machine
  - Generally not necessary for VM to have equivalent performace

Where is the VM?

- *Process virtual machine*: supports an individual process
  - Emulates user-level instructions and operating system calls
  - Virtualizing software placed at the ABI layer
- *System Virtual Machines*: emulates the target hardware ISA
  - guest and host environment may use the same ISA
- Virtual Machines are implemented as combination of
  - Real hardware
  - Virtualizing software



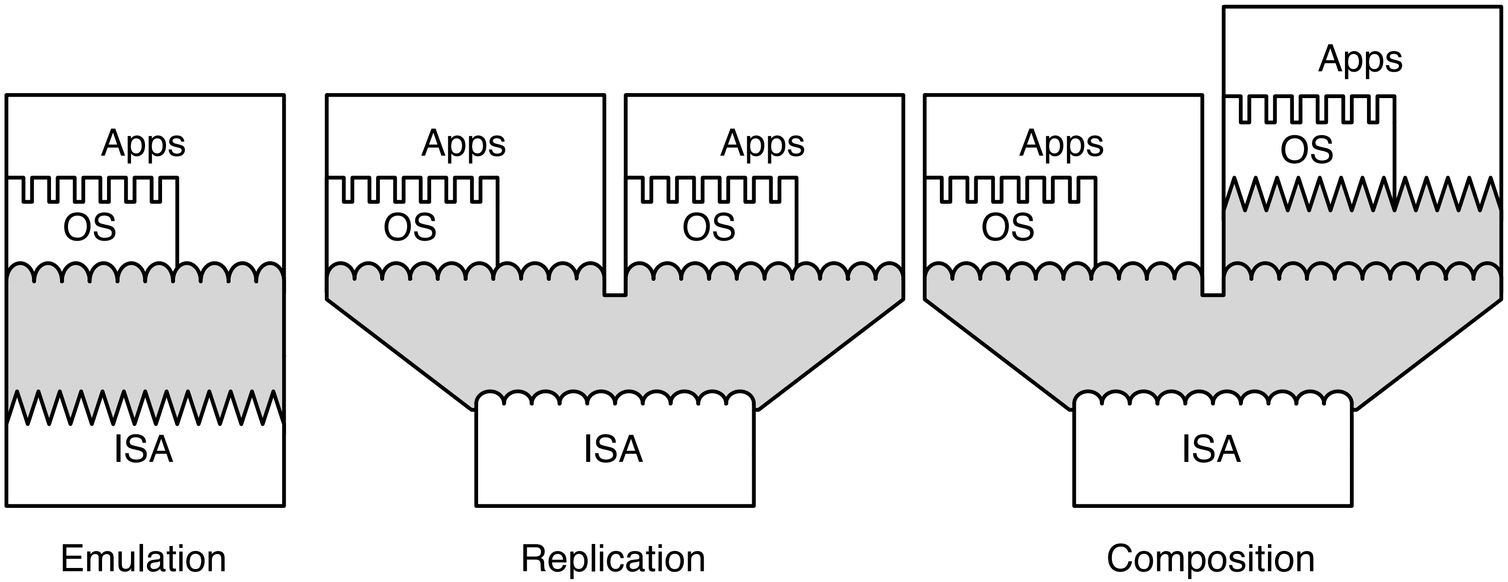
Terminology

- **host** environment: layers under the VM
- **guest** environment: layers above the VM
- **runtime**: virtualizing software in process VMs.
- **virtual machine monitor (VMM)**: virtualizing software in system VMs

VM Capabilities

Virtual machines can provide *emulation*, *optimization* and *replication*

- **emulation**: cross platform compatibility
- **optimization**: by considering implementation specific information
- **replication**: making a single resource or platform appear as many

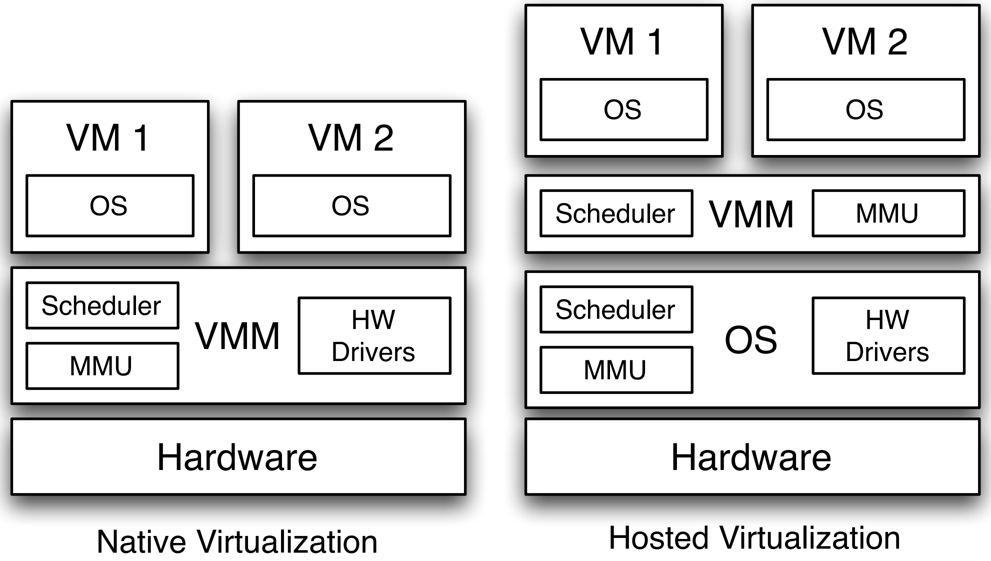


Process VM Examples

- Same ISA
  - Multiprogrammed Systems
  - Binary Optimizers
- Different ISA
  - Emulators & Dynamic Binary Translators
  - Higher Level Language (HLL) VMs

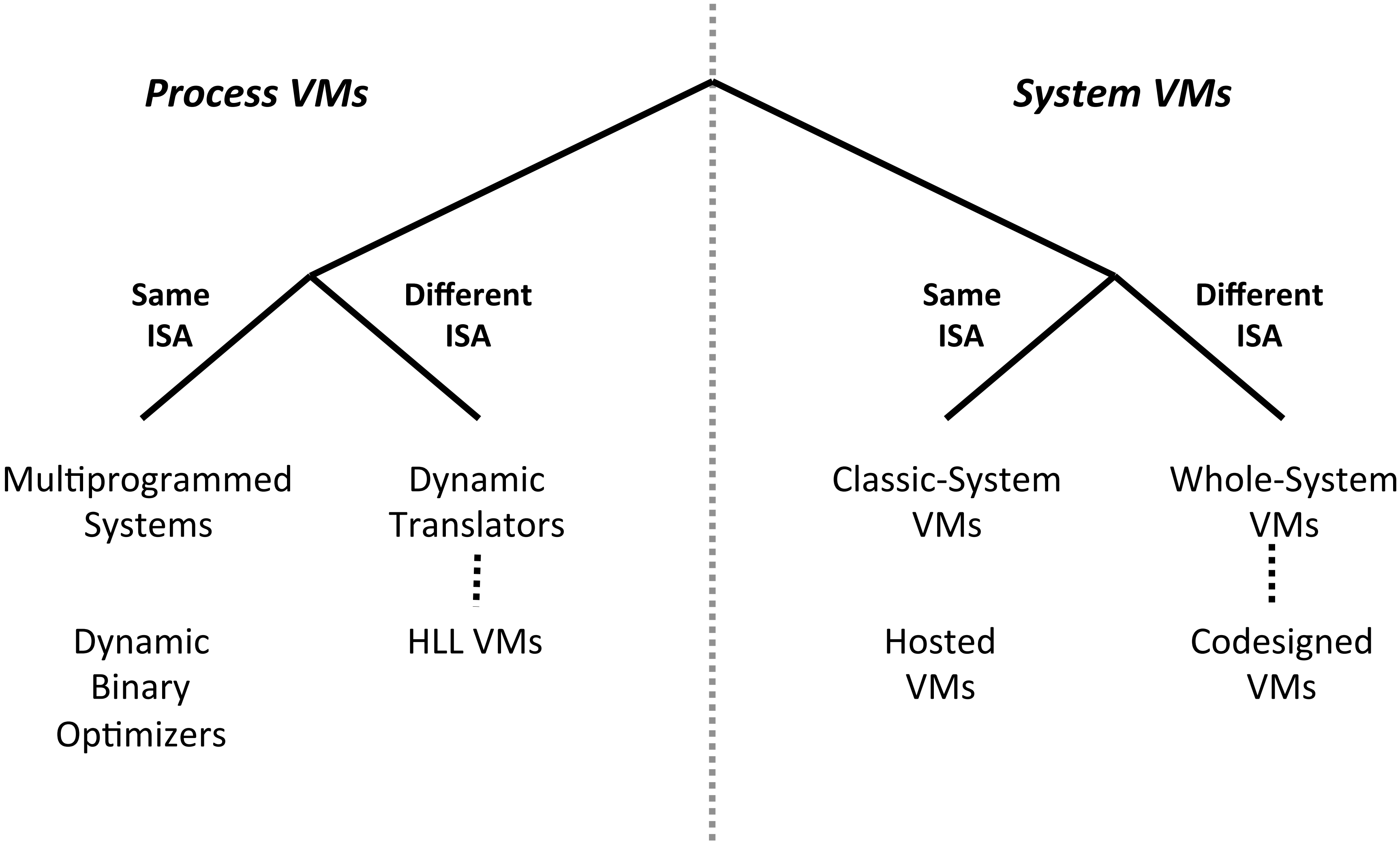
System VM Examples

- Whole System
  - Same/Different ISA
  - Bare Hardware/Native/Bare Metal/Type 1
  - Hosted/Type 2



- Codesigned
  - innovative ISAs and/or hardware implementations for improved performance, power efficiency, or both.

Taxonomy


Process VMs
System VMs
Same ISA
Different ISA
Same ISA
Different ISA
Multiprogrammed Systems
Dynamic Translators
Dynamic Binary Optimizers
HLL VMs
Classic-System VMs
Whole-System VMs
Hosted VMs
Codesigned VMs