Introduction to Virtual Machines

[1] E. Bugnion, J. Nieh, and D. Tsafrir, "Hardware and Software Support For Virtualization," *Synth. Lect. Comput. Archit.*, vol. 12, no. 1, pp. 1–206 , Feb. 2017.

Readings: Chapter 1 [1]

Abstraction and Layering

- **Abstraction**: only way of dealing with complex systems
  - Divide world into objects, each with an…
    - **Interface**: knobs, behaviors, knobs → behaviors
    - **Implementation**: "black box"
  - Specialists deal with implementation; others interface
  - Example: car drivers vs. mechanics

- **Layering**: abstraction discipline makes life even simpler
  - Removes need to even know interfaces of most objects
  - Divide objects in system into layers
  - Layer X objects
    - Implemented in terms of interfaces of layer X-1 objects
    - Don't even need to know interfaces of layer X-2 objects
  - Example: cab passenger vs. mechanics

Abstraction, Layering, and Computers

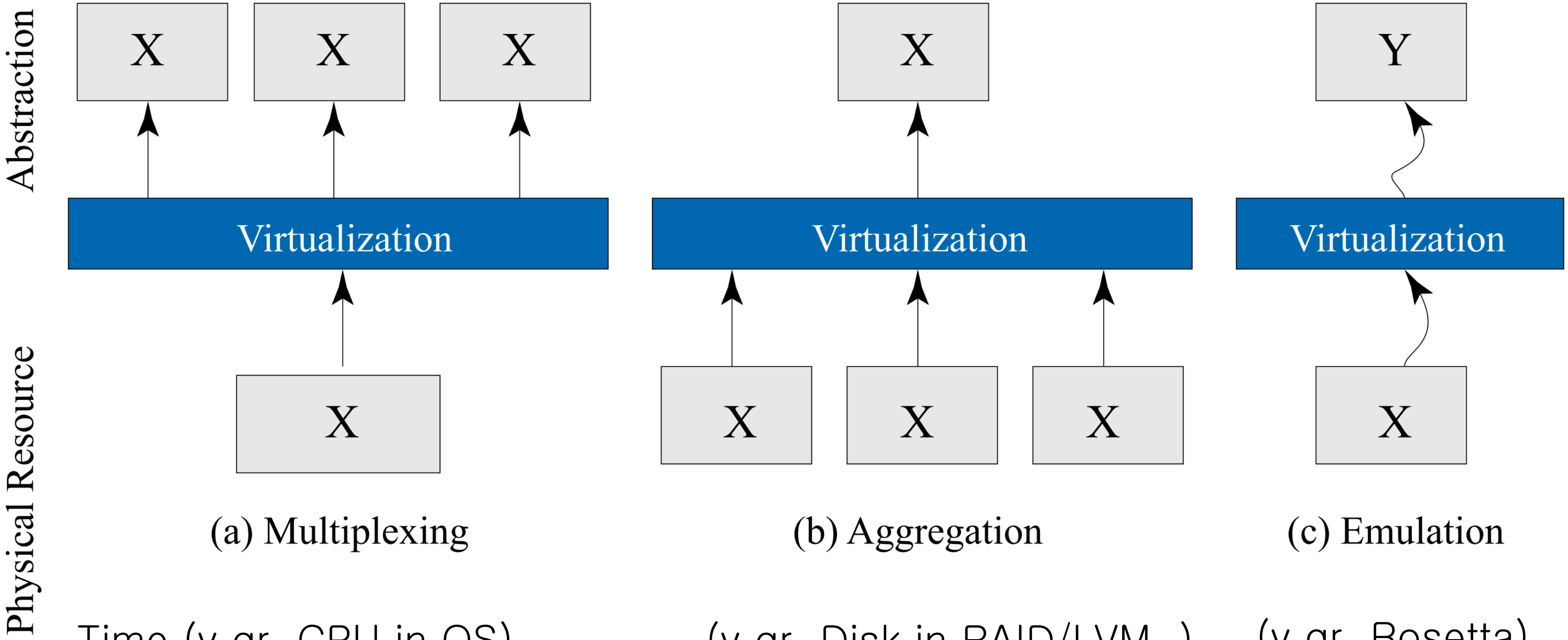
- Computers are complex systems, built in layers
  - Applications
  - O/S, compiler
  - Firmware, device drivers
  - Processor, memory, raw I/O devices
  - Digital circuits, digital/analog converters
  - Gates
  - Transistors
- 99% of users don't know hardware layers implementation
- 90% of users don't know implementation of any layer
- That's OK, world still works just fine
  - But unfortunately, the layers sometimes breakdown
  - Someone needs to understand what's "under the hood"

Virtualization

- Definition:
  - *Virtualization is the application of the layering principle through* ***enforced modularity****, whereby the exposed virtual resource is identical to the underlying physical resource being virtualized.*
- Virtualization Example in Computer Architecture
  - Virtual memory modularity enforced through MMU
- Virtualization within the Operating System
  - Expose real resources (CPU, Memory, I/O) to processes in a controlled way
- Virtualization in I/O subsystems
  - RAID controllers and Disks

Implementation Techniques in Virtualization

- Combination of these three techniques in the hypervisor



(a) Multiplexing

(b) Aggregation

(c) Emulation

Time (v.gr. CPU in OS)
Space (v.gr. Mem in OS)

(v.gr. Disk in RAID/LVM,)
DIMMs)

(v.gr. Rosetta)

Virtual Machines: Definitions

- **Virtualization** is the application of the layering principle through enforced modularity, whereby the exposed virtual resource is identical to the underlying physical resource being virtualized

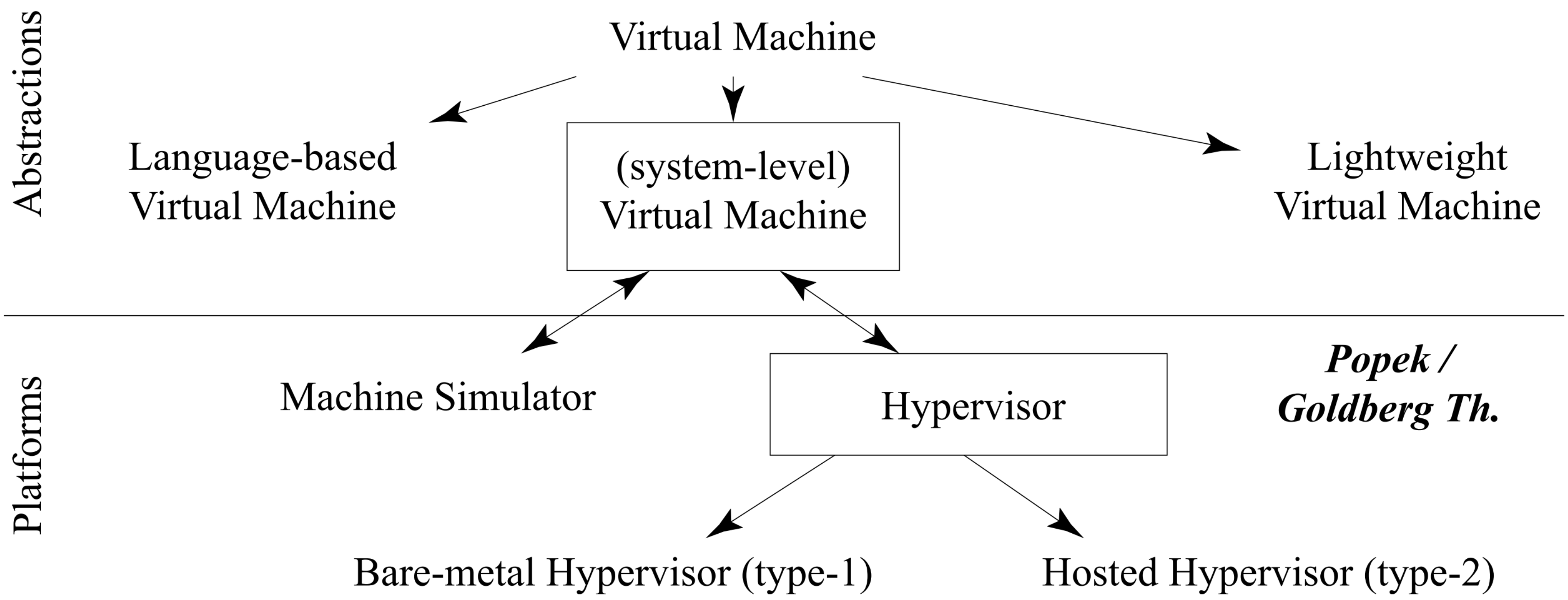
- **A virtual machine** is an abstraction of a complete compute environment through the combined virtualization of the processor, memory, and I/O components of a computer.

- **The hypervisor** is a specialized piece of system software that manages and runs virtual machines.

- **The virtual machine monitor** (VMM) refers to the portion of the hypervisor that focuses on the CPU and memory virtualization (beware)

Virtual Machines

- A **virtual machine** is a complete compute environment with its own isolated processing capabilities, memory, and communication channels.


Abstractions
Platforms
Virtual Machine
Language-based Virtual Machine
(system-level) Virtual Machine
Lightweight Virtual Machine
Machine Simulator
Hypervisor
Popek / Goldberg Th.
Bare-metal Hypervisor (type-1)
Hosted Hypervisor (type-2)

Classes of Virtual Machines

- Language-based virtual machines
  - For portability reasons, synthetic ISA to compile.
  - ISA is translated on runtime into hardware ISA
  - Not our focus
  - Ej: JVM

- Lightweight virtual machines,
  - Rely on a combination of hardware and software isolation mechanisms to ensure that applications **running directly on the processor** (e.g., as native x86 code) are securely isolated.
  - Ej: containers

- **System Level Virtual Machines**

System Level Virtual Machines

- Compute environment that **resembles** the hardware of a computer with enough detail to run a standard, **commodity operating** system and its applications
- **Full isolation** from the other virtual machines and the rest of the environment.
- Applies the virtualization principle to **an entire computer system**.
- Each virtual machine has its own copy of the underlying hardware, or at least, its own copy of **some**
- Each virtual machine runs its own independent operating system instance, called the **guest operating system**

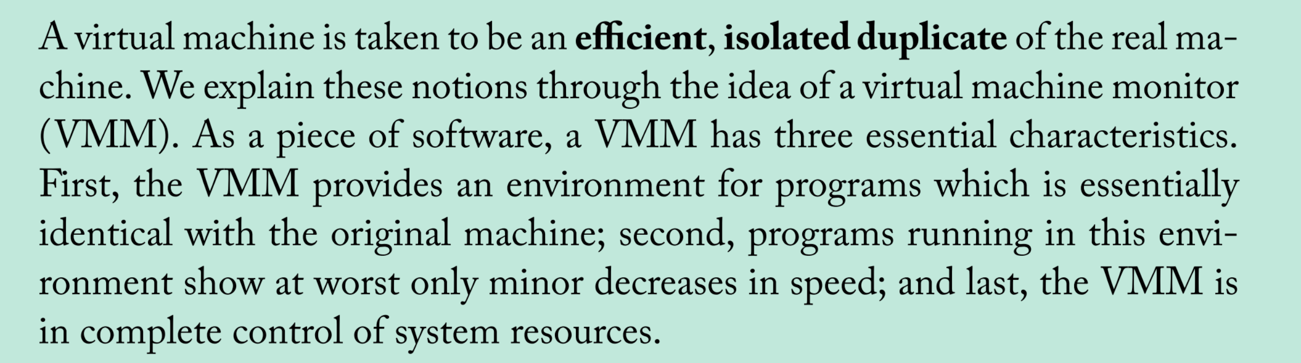
Platforms

- Machine Simulators (full system simulators)
  - Implemented as a user-level application
  - Models functionally (and timing) hardware details of the platform to study (processor, memory, I/O, etc). Platform can be single-system or muti-system
  - Very slow
  - Useful to hardware/software codesign, computer architecture research, etc..
  - Example: **gem5**

- Hypervisor
  - Relies on direct execution on the CPU
  - Should emulate non-user level instructions somehow
  - Models functionally other less performance sensitive components (Disk, net)
  - … if supported, can be used directly
  - Example: **Xen**, **KVM**

Hypervisor

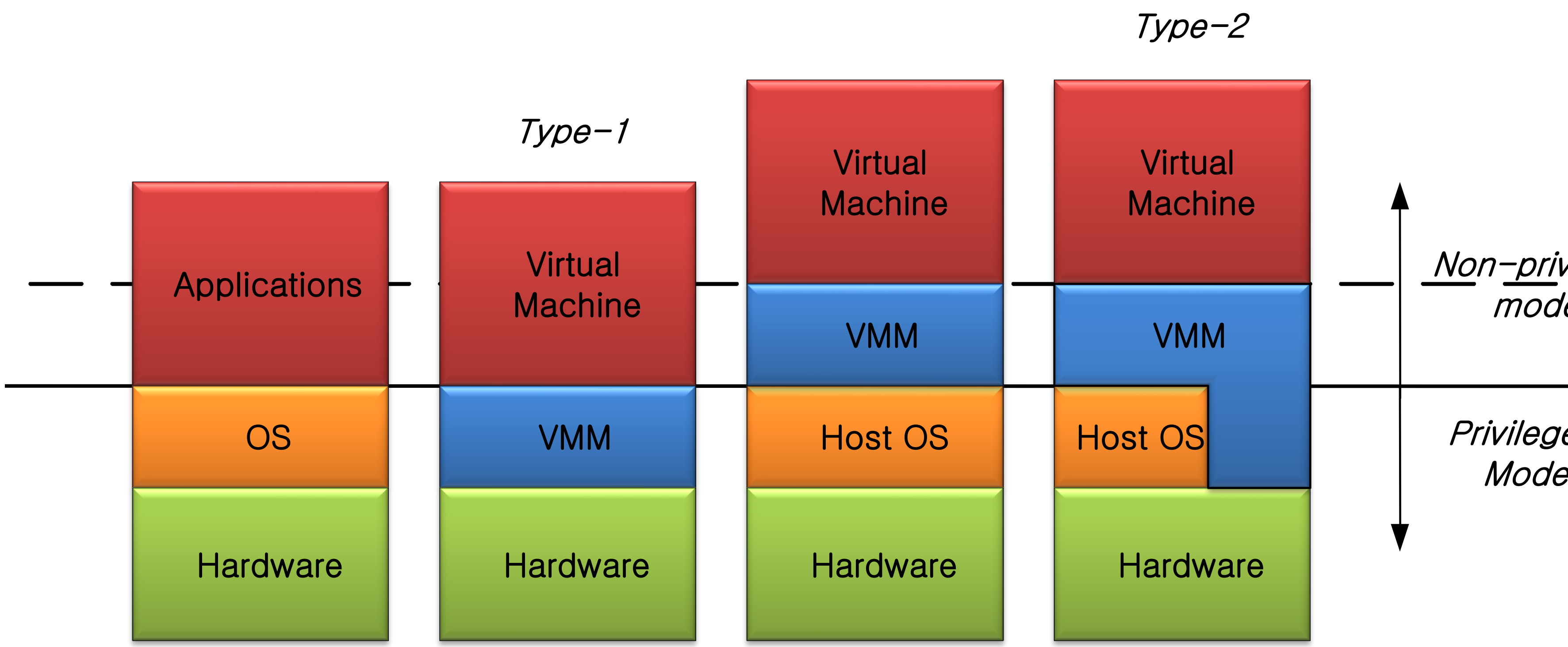
- Popek and Goldberg Definition



> A virtual machine is taken to be an **efficient**, **isolated duplicate** of the real machine. We explain these notions through the idea of a virtual machine monitor (VMM). As a piece of software, a VMM has three essential characteristics. First, the VMM provides an environment for programs which is essentially identical with the original machine; second, programs running in this environment show at worst only minor decreases in speed; and last, the VMM is in complete control of system resources.

- Equivalence
  - Duplicating real resources
- Safety
  - Isolation between VM and hypervisor
- Performance
  - Separates Hypervisors from Simulators

Type-1 and Type-2 hypervisors



Type-1

Type-2

Applications

OS

Hardware

Virtual Machine

VMM

Hardware

Virtual Machine

VMM

Host OS

Hardware

Virtual Machine

VMM

Host OS

Hardware

Non-priv mode

Privilege Mode

Traditional system

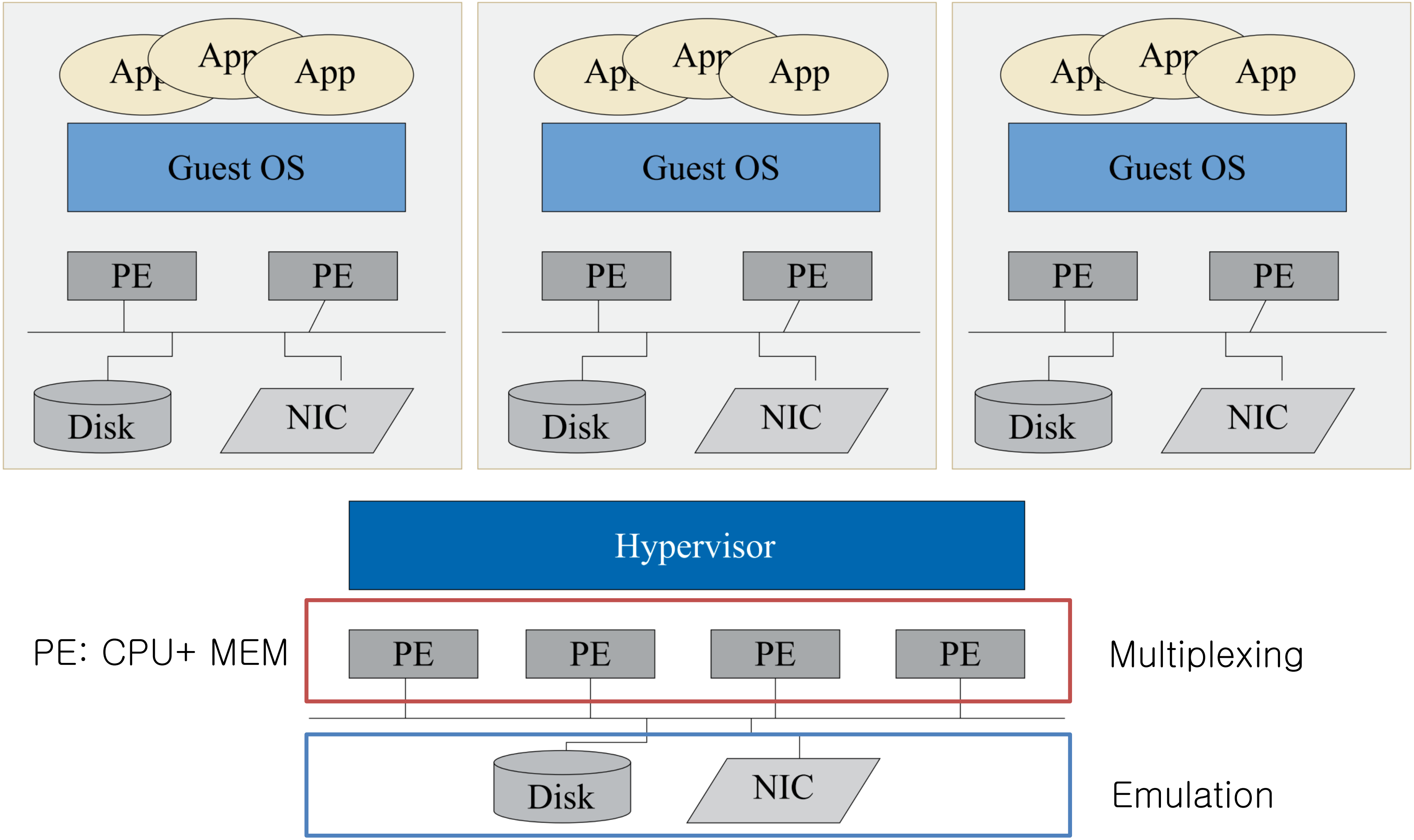
Native VM system

(e.g. z/VM, PS3, Xen, VMW-ESX, HYPER-V…)

User-mode Hosted VM system

Dual-mode Hosted VM system (e.g. Virtual Box, VM-Fusion, ,KVM, VMW-GSX)

Multiplexing and Emulation

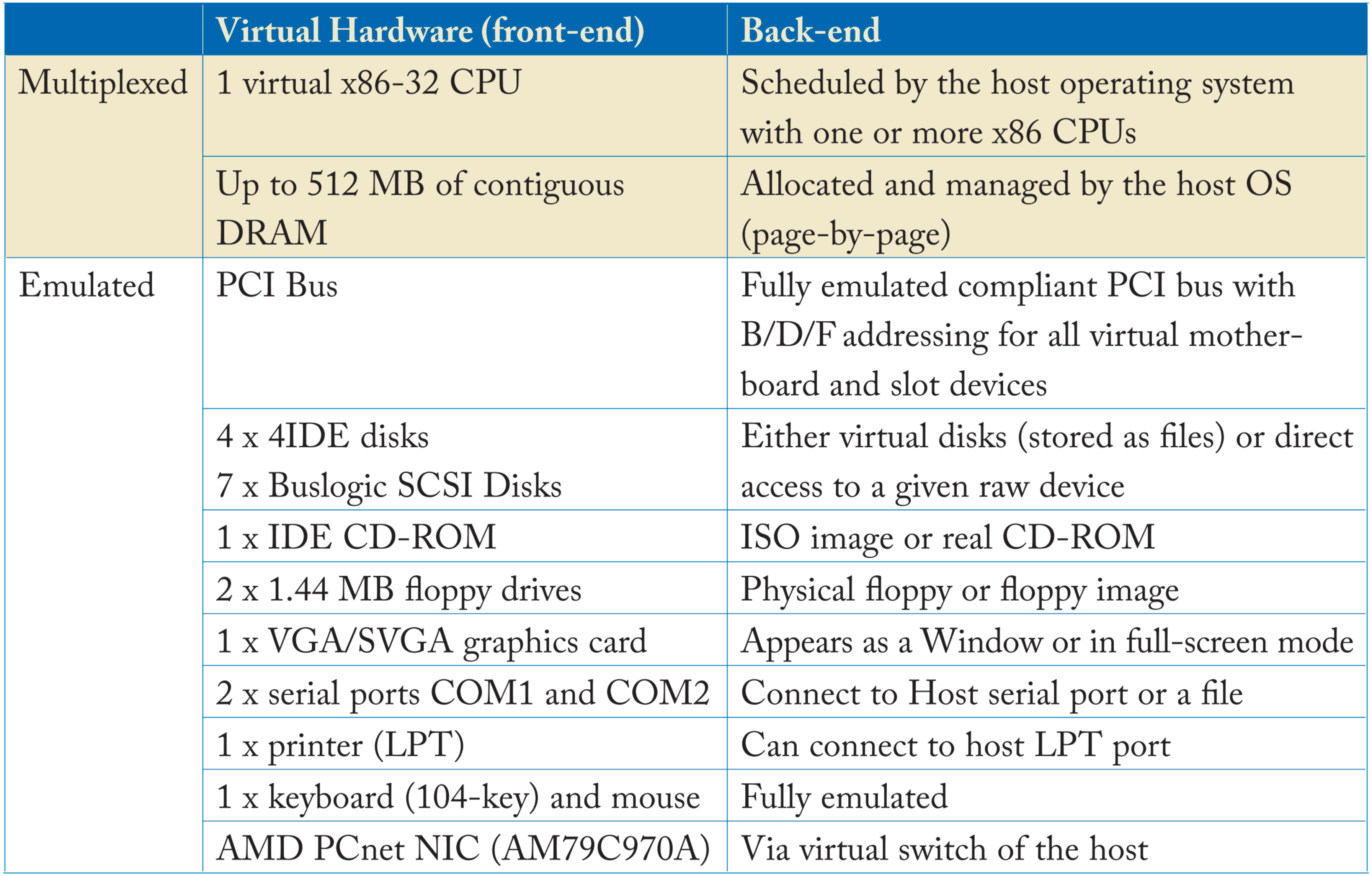


PE: CPU+ MEM

Multiplexing

Emulation

Case of VMWare (Early versions, No HW support)



	Virtual Hardware (front-end)	Back-end
Multiplexed	1 virtual x86-32 CPU	Scheduled by the host operating system with one or more x86 CPUs
	Up to 512 MB of contiguous DRAM	Allocated and managed by the host OS (page-by-page)
Emulated	PCI Bus	Fully emulated compliant PCI bus with B/D/F addressing for all virtual motherboard and slot devices
	4 x 4IDE disks 7 x Buslogic SCSI Disks	Either virtual disks (stored as files) or direct access to a given raw device
	1 x IDE CD-ROM	ISO image or real CD-ROM
	2 x 1.44 MB floppy drives	Physical floppy or floppy image
	1 x VGA/SVGA graphics card	Appears as a Window or in full-screen mode
	2 x serial ports COM1 and COM2	Connect to Host serial port or a file
	1 x printer (LPT)	Can connect to host LPT port
	1 x keyboard (104-key) and mouse	Fully emulated
	AMD PCnet NIC (AM79C970A)	Via virtual switch of the host

Names for Memory

- Eskimos and Snow, Computer Architects and Memory
  - **Virtual memory** concept is the most significant enhancement over the original Von-Neumman Model
- Virtual memory
  - Byte addressable **namespace** used by instruction sequences executed by the processor
- Physical Memory
  - Byte addressable resource accessed by the memory hierarchy (typically DRAM)
  - **Guest-physical** memory or **Host-physical** Memory in a VM

Approaches to Virtualization

- **Full** (software) **virtualization**
  - Hypervisors designed to maximize hardware compatibility
  - Run unmodified operating systems on architectures without full support for it (usually by the means of dynamic binary translation)

- **Hardware Virtualization** (HVM)
  - Hypervisors built for hardware with architectural support for virtualization
  - (mostly) Rely on direct execution

- **Paravirtualization**
  - Initially, hypervisors for platform without architectural support for virtualization using modified operating system to avoid binary translation (via hyper-calls)
  - Today, a mix of paravirtualization and HVM is usual

Benefits of Virtual Machines

- Operating system diversity (on same hardware)
- Server consolidation
  - Best IT practices mandates single app per “server”
- Rapid provisioning
  - Simplify server deployment
- Security
  - Additional OS Isolation, OS introspection, external control (e.g. firewall interposing)
- High-availability
  - Near-zero operation impact of hardware disruption
- Distributed resource scheduling
  - Live migration techniques convert a group of hypervisors in a common resource pool
- Cloud computing
  - Mix customers (tenants) in a shared resource pool. Requires **network virtualization**

History of sVM

- Mid/late-60s IBM 360/67 -- CP-67
  - First 360 with VM.
  - CMS an essential part (Multiuser in a single user OS)

- Late 60s/early 70s
  - VMs blossomed as a research topic

- Early 70s several VM implementations
  - Honeywell
  - DEC
  - RCA
  - Several university projects
- ....

sVM on cheap HW (x86)

- Software Explosion
  - VMWare (v1.0 1999)
    - Targets Windows
    - From Stanford Disco Project (mainly SimOS)
  - Xen (v1.0 2003)
    - Targets Linux
    - AWS
    - From Cambridge Univ.

- Hardware
  - Intel: Vanderpool (VT-x): Pentium 4 (2005)
  - AMD: Pacifica (AMD-v) Athlon64 Processors (2006)