ECE 5984 Virtualization Technologies

Introduction to Virtualization

Pierre Olivier

Outline

1) Virtualization quick definition and use cases

2) Virtualization: In-depth definition

3) Virtual Machines

4) Hypervisors

5) Memory denomination, full/hardware/para-virtualization

Outline

1) Virtualization quick definition and use cases
2) Virtualization: In-depth definition
3) Virtual Machines
4) Hypervisors
5) Memory denomination, full/hardware/para-virtualization

Virtualization: definition

In the context of this course

- **Quick and easy definition in the context of this course:**

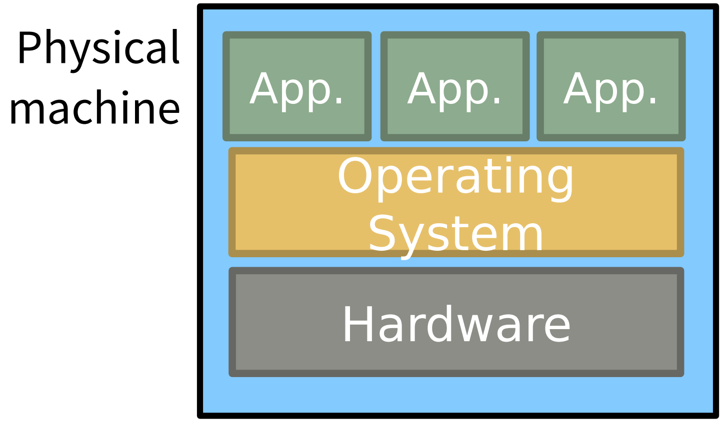
Virtualization technologies is the set of software and hardware components that allow running multiple operating systems at the same time on the same physical machine

Virtualization: definition

In the context of this course

- **Quick and easy definition in the context of this course:**

Virtualization technologies is the set of software and hardware components that allow running multiple operating systems at the same time on the same physical machine

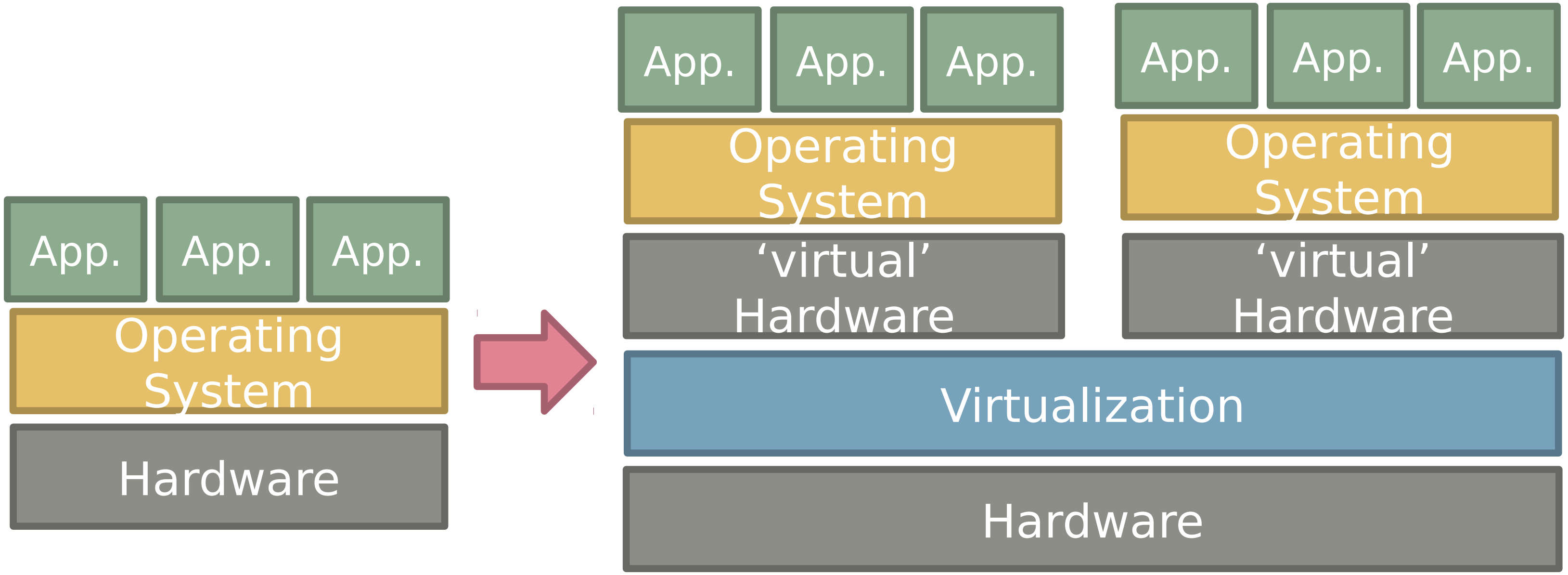


Virtualization: definition

In the context of this course

- **Quick and easy definition in the context of this course:**

Virtualization technologies is the set of software and hardware components that allow running multiple operating systems at the same time on the same physical machine

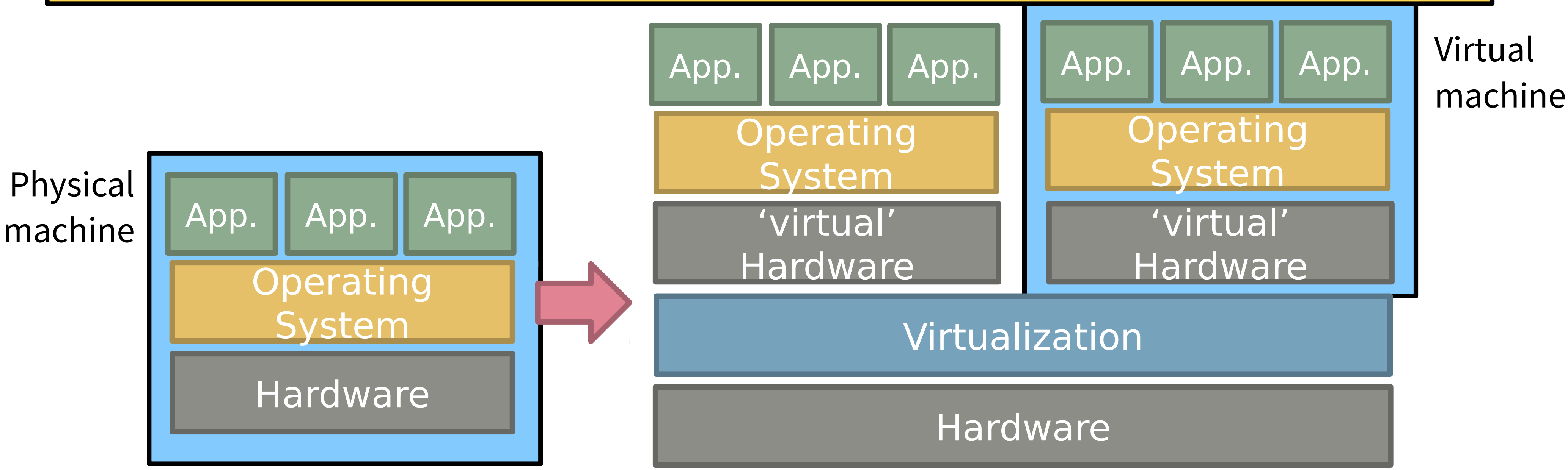


Virtualization: definition

In the context of this course

- **Quick and easy definition in the context of this course:**

Virtualization technologies is the set of software and hardware components that allow running multiple operating systems at the same time on the same physical machine



Virtualization: what for?
A (tiny) bit of history

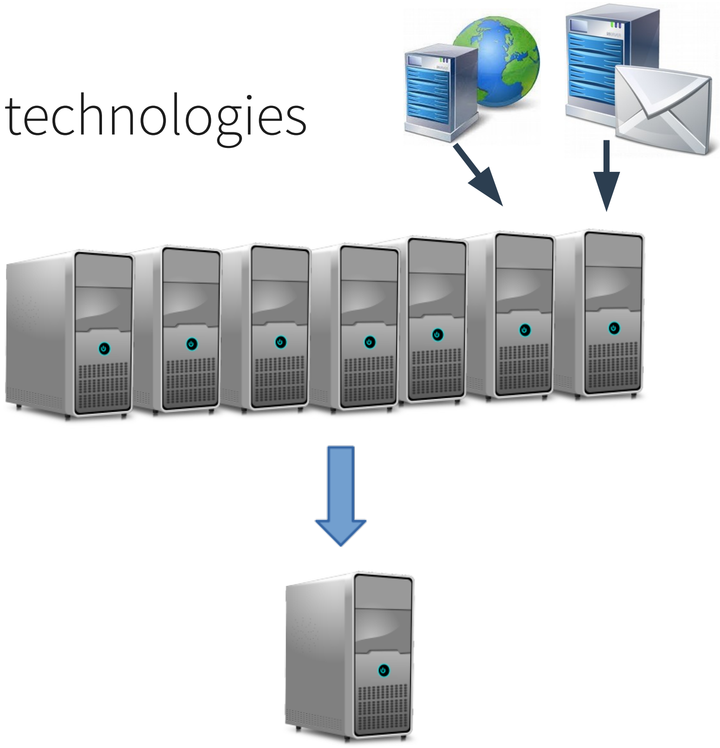
- **1960s: IBM's VM (1960s)**
  - Project System/360 (S/360) sold between 1965 and 1978
  - Family of computers of various sizes built using the same architecture
    - Client can buy a small model for testing then a big mainframe later
  - Clients then wanted to **move software running on multiple small models to a single large one: *consolidation***
  - **Model 67: virtualizable ISA**
    - Machine can appear as multiple, less powerful versions of itself
    - CP-67 (Control program ~ OS service): successor of the CP-40 research prototype
- **1974: Popek & Goldberg theorem**
  - Seminal paper: *Formal Requirements for Virtualizable Third Generation Architectures*
- **1990s: *Disco***
  - *Hypervisor from Stanford, researchers then found Vmware*
- ***2000s: Xen, KVM, VirtualBox, Hyper-V, etc.***

Source: wikipedia

Virtualization: what for?

Consolidation

- **Consolidation is the process of creating X virtual machines from X physical ones and running them on Y physical hosts**
  - With Y < X
  - Historical motivation for developing virtualization technologies
  - Gives the **benefits of multi-computer systems without the $/management costs:**
    - Software dependencies
    - Reliability
    - Security



Virtualization: what for?

Software development

- **Flexible OS diversity: different OS on the same machine**
  - Ex: VirtualBox with Linux for kernel development
- **Rapid provisioning**
  - Way faster than a physical machine
- **VMs are self contained**
  - Practical way to "pack" an application with all its software dependencies
    - Model and version of the OS, libraries, etc.
  - Useful for development and automated testing

Virtualization: what for?

Migration, checkpoint/restart

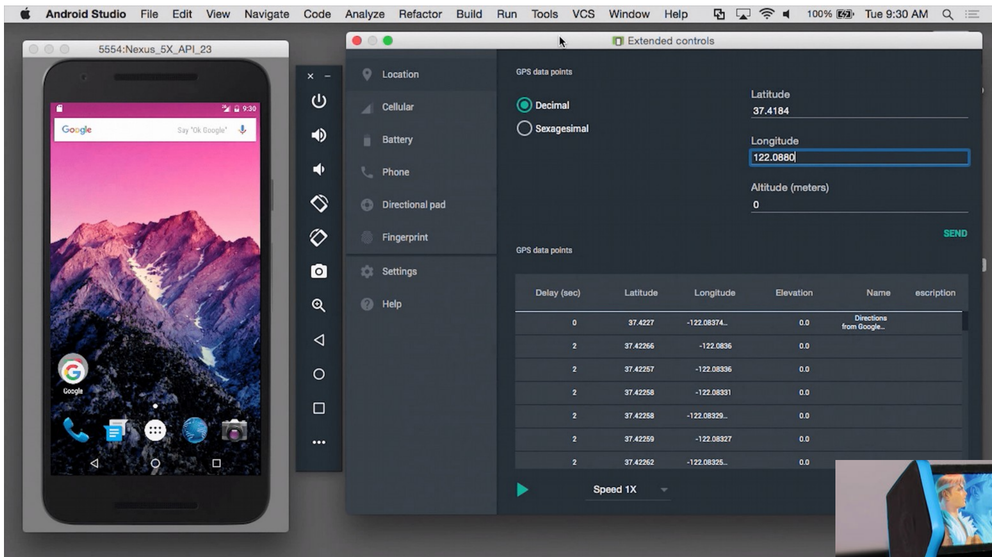
- **VMs are self-contained and can be migrated between hosts**
  - Live migration transparent from the VM user point of view
    - ➔ Ex: Quake 3 server under Xen migrated with 60 ms downtime[1]
  - Freeing resources
    - ➔ For maintenance
    - ➔ When a fault is expected
  - Increased performance
  - Distributed resources scheduling: for example load balancing, consolidation for power savings, etc.
- **Checkpoint/restart for long-running jobs**
  - Dump the VM state to disk in order to resume it later
    - Or *to be able to* resume it later (ex: after a crash)
- **Both techniques straightforward for VMs**
  - As opposed to process migration

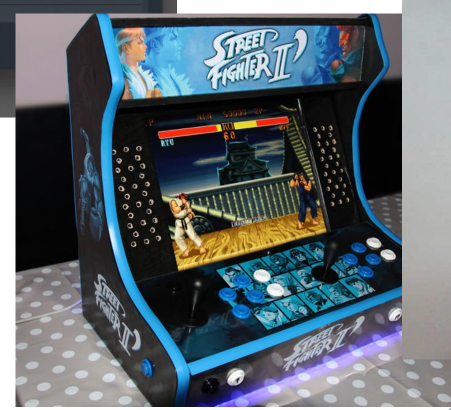
[1]Clark, Christopher, et al. "Live migration of virtual machines." Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2. USENIX Association, 2005.

Virtualization: what for?

Hardware emulation, legacy/backward compatbility

- Virtualization can be used to emulate old/different hardware





Virtualization: what for?
Cloud computing

- **Virtualization enabled cloud computing (Security, isolation, flexibility)**
  - Offloading local tasks to remote computing resources
    - Rent a VM to put a webserver (IaaS)
    - Fully develop and run a web application using Google app engine (PaaS)
    - Offload mail server to gmail (ex VT) (SaaS)
  - To save on management, infrastructure, development, maintenance costs
    - Pricing: pre-purchase (rent) or on-demand

AWS EC2 Instance

Virtualization: what for?

Security

- **Virtualization provides very strong isolation between guests**
  - Sandboxing
    - Virus/malware analysis
    - Honeypots
    - Process/task level isolation through virtualization
      - ➔ Ex: QubesOS, Bromium
- **VM introspection**
  - Analysis of the guest behavior from a privileged level higher than the OS's
    - Guest OS cannot be trusted
    - Ex: LibVMI

Outline

1) Virtualization quick definition and use cases

2) **Virtualization: In-depth definition**

3) Virtual Machines

4) Hypervisors

5) Memory denomination, full/hardware/para-virtualization

Virtualization: definition

High-level definition

- **For the textbook:**

> *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: definition

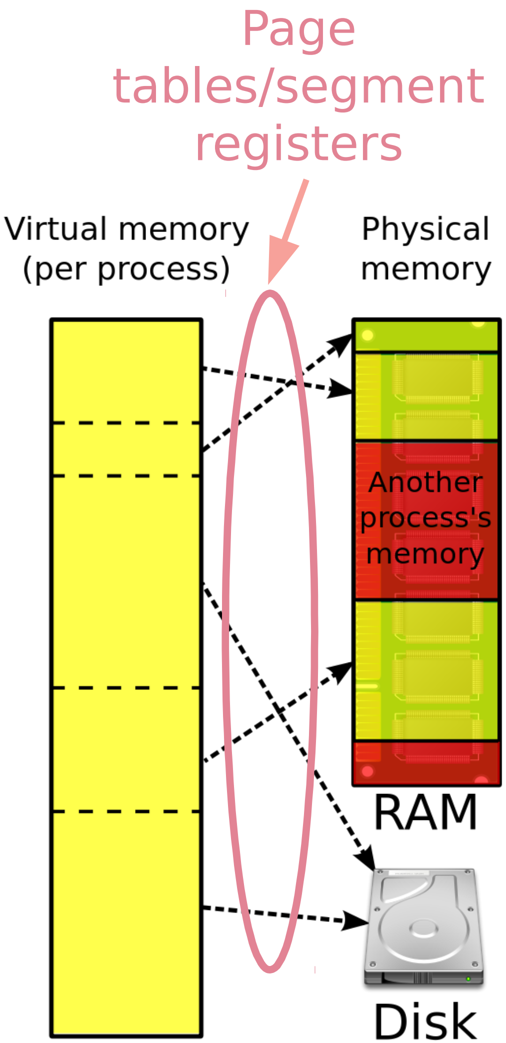
High-level definition (2)

> *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*

- **Layering principle**
  - Abstraction of one or several components using an indirection layer
  - Uses a well-defined interface to expose the abstraction
- **Enforced modularity**
  - Abstraction layer cannot be bypassed by its clients
- **Exposed virtual resource is identical to the virtualized physical one:**
  - Conceptual equivalence between the real and abstracted component
    - Clients of the virtual component should be as similar as possible to the clients of the physical component
      - ➔ If possible they should run unmodified
- **In a general sense, virtualization does not *only* refer to the abstraction of an entire computer (a virtual machine)**

Virtualization: definition
High-level definition: example 1

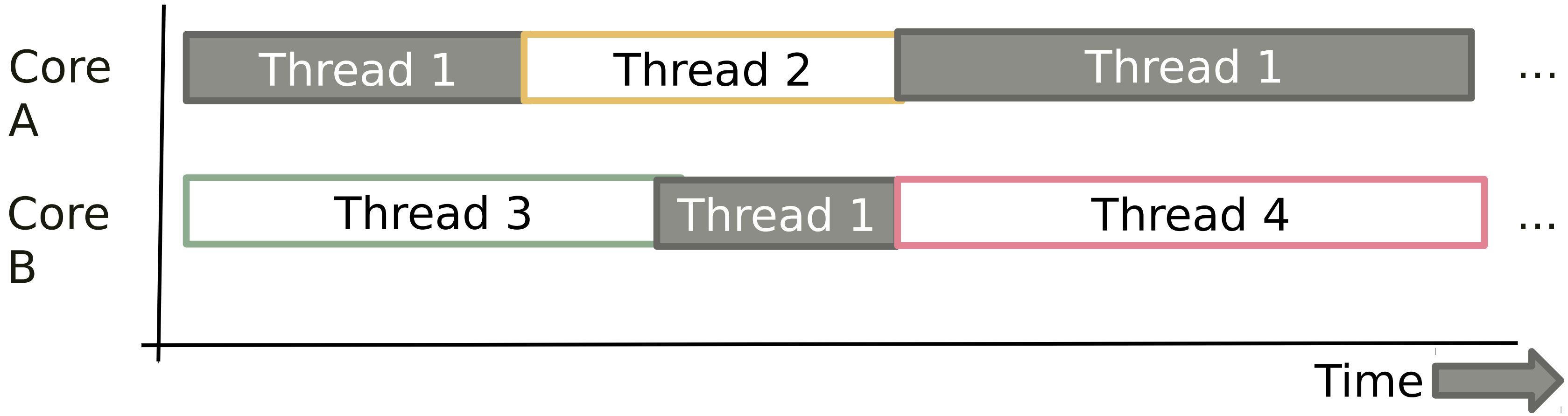
- **Virtual memory**
  - Memory Management Unit (MMU) abstracts physical RAM
    - Segmentation & Paging (**indirection layer**)
    - Gives the process the illusion it has access to all the RAM
      - ➔ Address space
    - Allows swapping memory pages to disk
    - Multiple other benefits with the page-fault mechanism
  - **Modularity**: only the kernel can modify the virtual to physical mapping
    - Process cannot access kernel space or other process address spaces directly
  - **Equivalence**: memory still accessed through load/stores



(source: wikipedia)

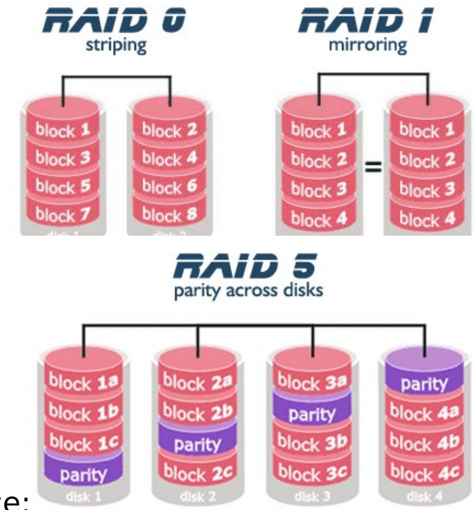
Virtualization: definition
High-level definition: example 2

- **Operating Systems**
  - OS is a virtualization layer abstracting physical resources and exposing these abstractions to the processes
  - Virtual memory in cooperation with the MMU
  - Through scheduling, OS virtualizes the physical CPU cores and multiplexes them among processes/threads



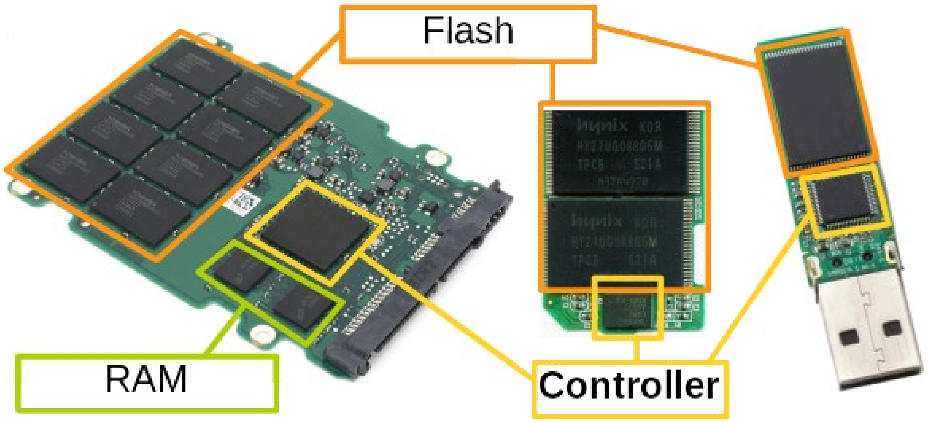
Virtualization: definition
High-level definition: example 3

- **I/O subsystems: the RAID**
  - Redundant Array of Independent/ Inexpensive Disks
  - Abstract multiple disks in a single logical volume
    - Reliability & performance advantages



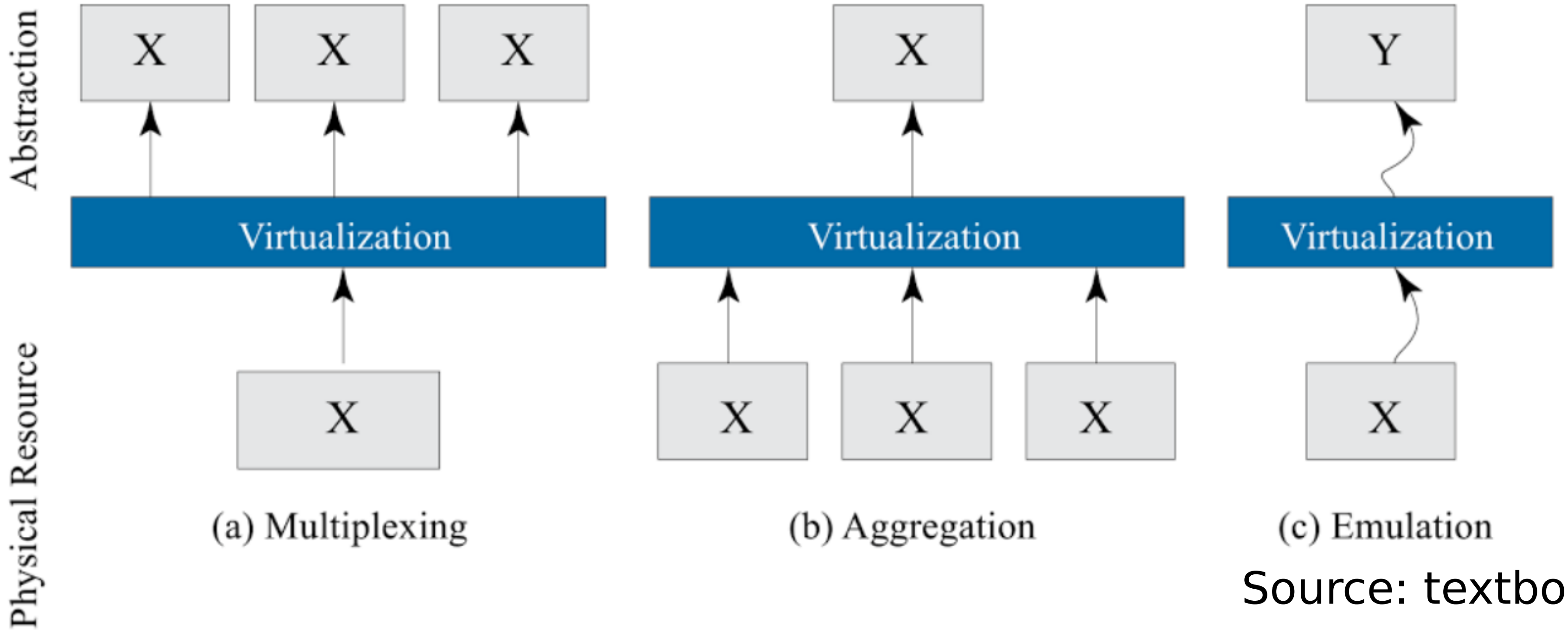
Source: http://www.sertdatarecovery.com

- **I/O subsystems: the FTL**
  - Flash Translation Layer
    - Abstracts flash memory specificity and make the flash device look like a hard disk



Virtualization: definition
Multiplexing, aggregation and emulation

- Virtualization achieved by using/combining three main principles: multiplexing, aggregation and emulation



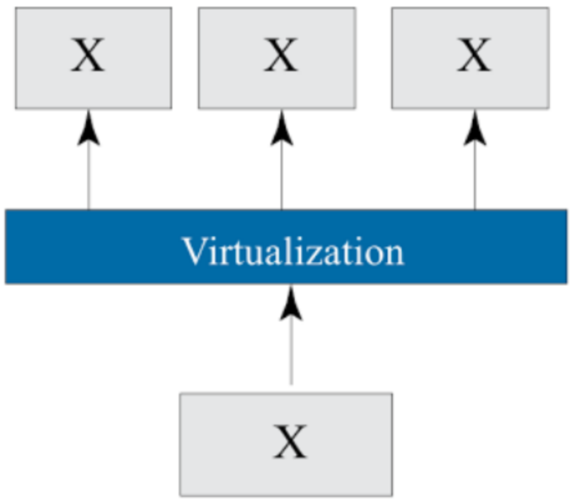
(a) Multiplexing (b) Aggregation (c) Emulation

Source: textbook

Virtualization: definition

Multiplexing

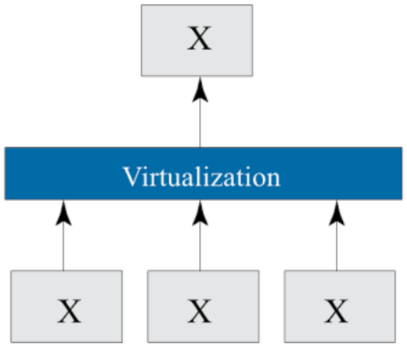
- **Multiplexing: exposes an abstraction of a single component as multiple entities**
  - In space: partitioning
    - Ex; virtual memory, virtual disks
    - Etc.
  - In time: scheduling
    - Ex: thread scheduling on CPUs
    - Etc.



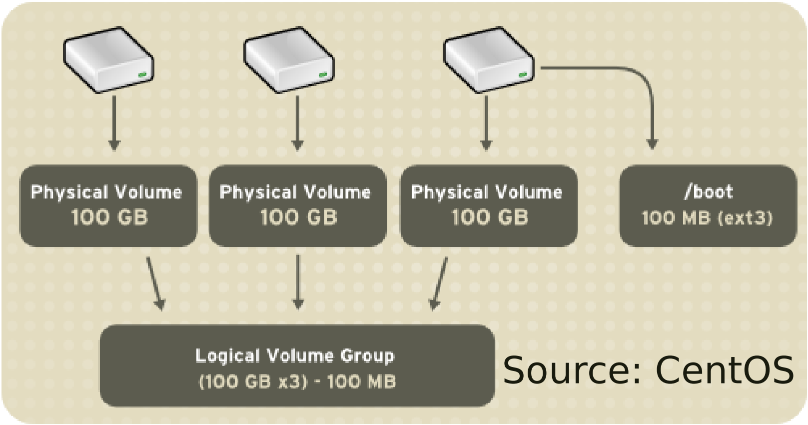
(a) Multiplexing

Virtualization: definition
Aggregation

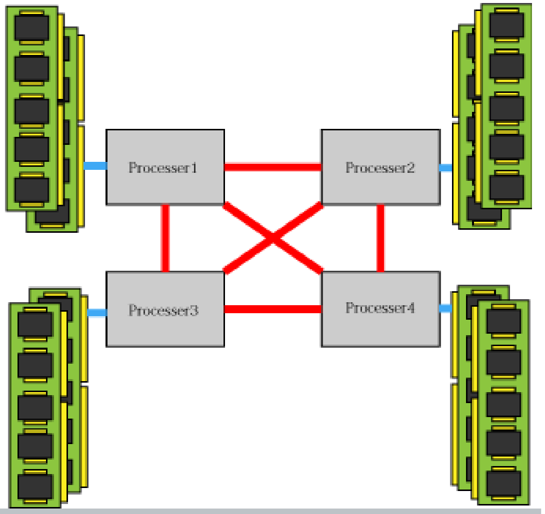
- **Aggregation: merges multiple resources of the same type into a single abstraction**
  - RAID
  - Logical Volume Manager
  - NUMA systems
  - Etc.



(b) Aggregation



Source: CentOS wiki

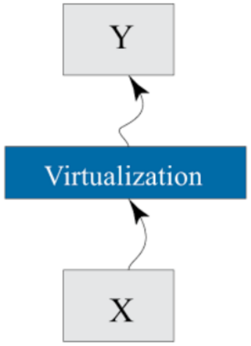


Source: https://github.com/ryran/xsos/issues/101

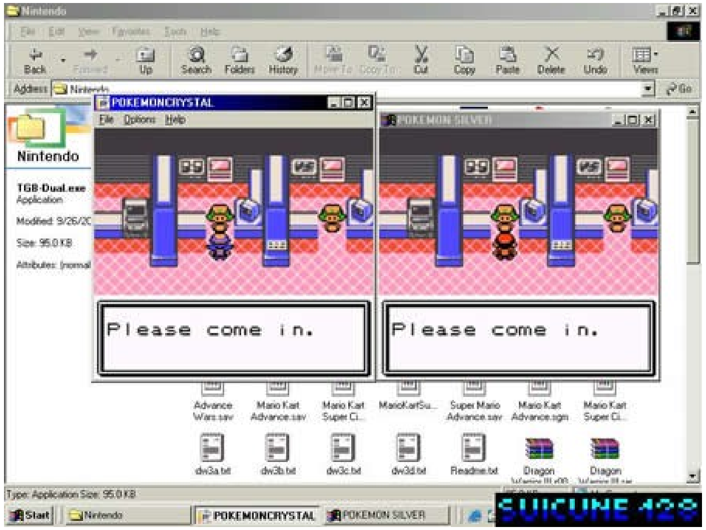
Virtualization: definition

Emulation

- **Emulation: presents on a computer a software model of a physical resource even if it is not physically present**
  - Example: use disk/RAM to emulate RAM/disk
    - Swap/Ramdisk
  - Example: cross-architectural simulators
    - Apple Rosetta running PowerPC software on x86 for retro-compatibility
    - Qemu running ARM software on x86, for example for Android development
    - Etc.



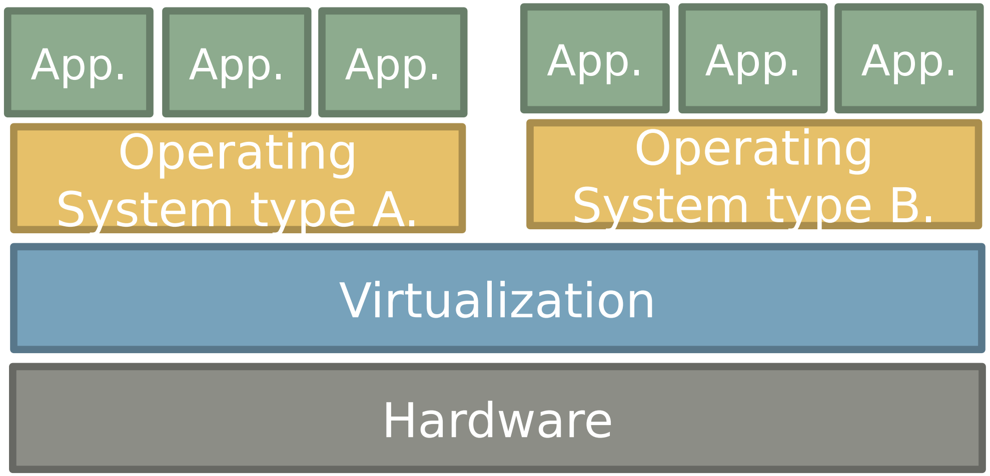
(c) Emulation



Virtualization: definition

Back to the course context

- In this course we are interested in virtualization used to run multiple OS (potentially different) on a single host



Outline

1) Virtualization quick definition and use cases
2) Virtualization: In-depth definition
3) Virtual Machines
4) Hypervisors
5) Memory denomination, full/hardware/para-virtualization

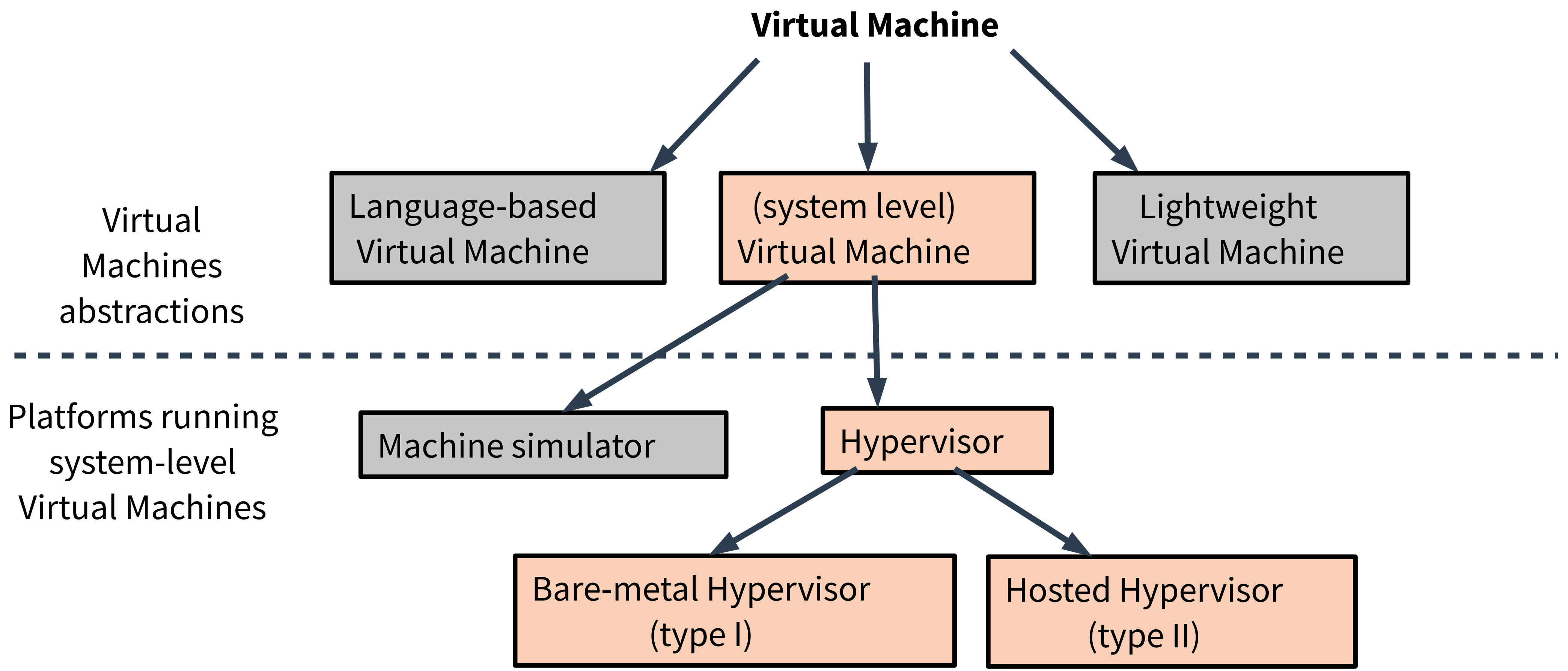
Virtual Machine

- **Textbook definition:**

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

Virtual Machine

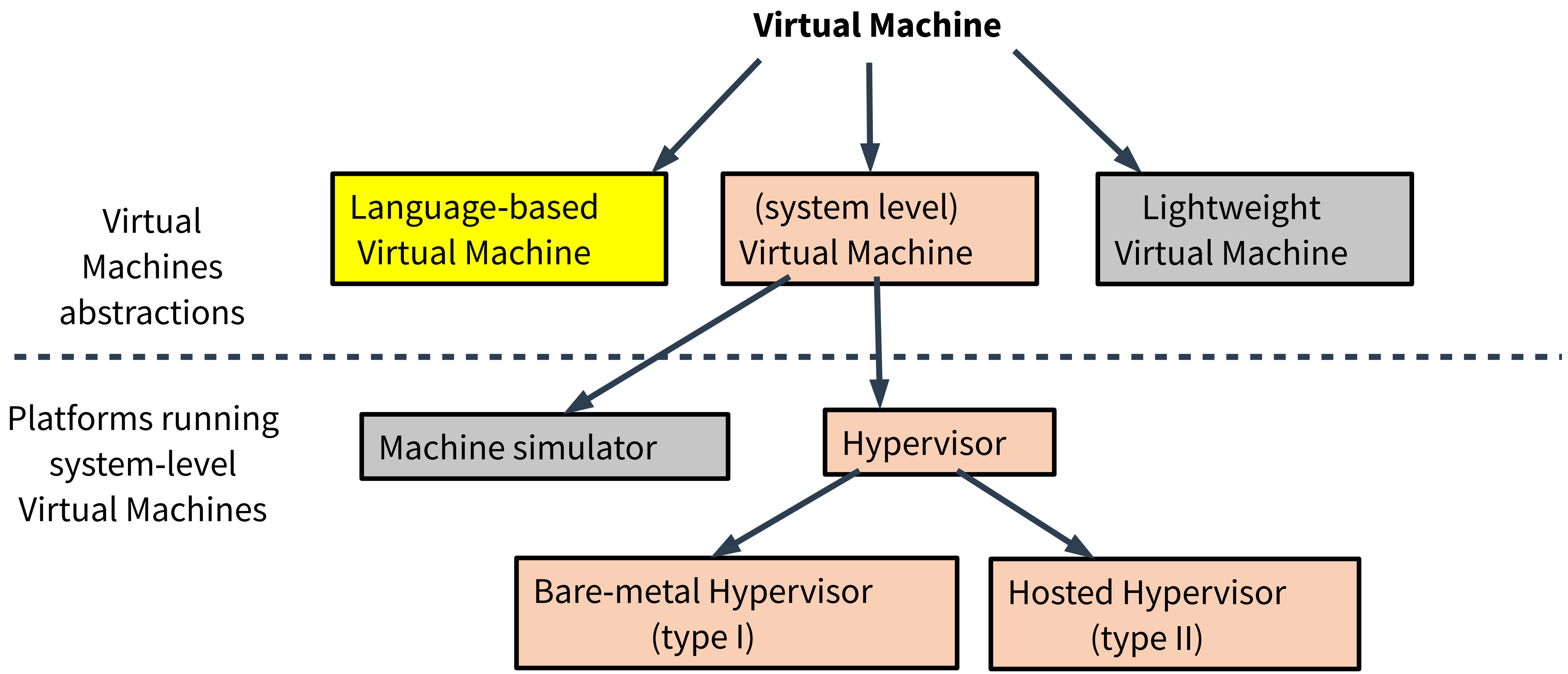
VM & running platforms classification



Adapted from the textbook

Virtual Machine

VM & running platforms classification

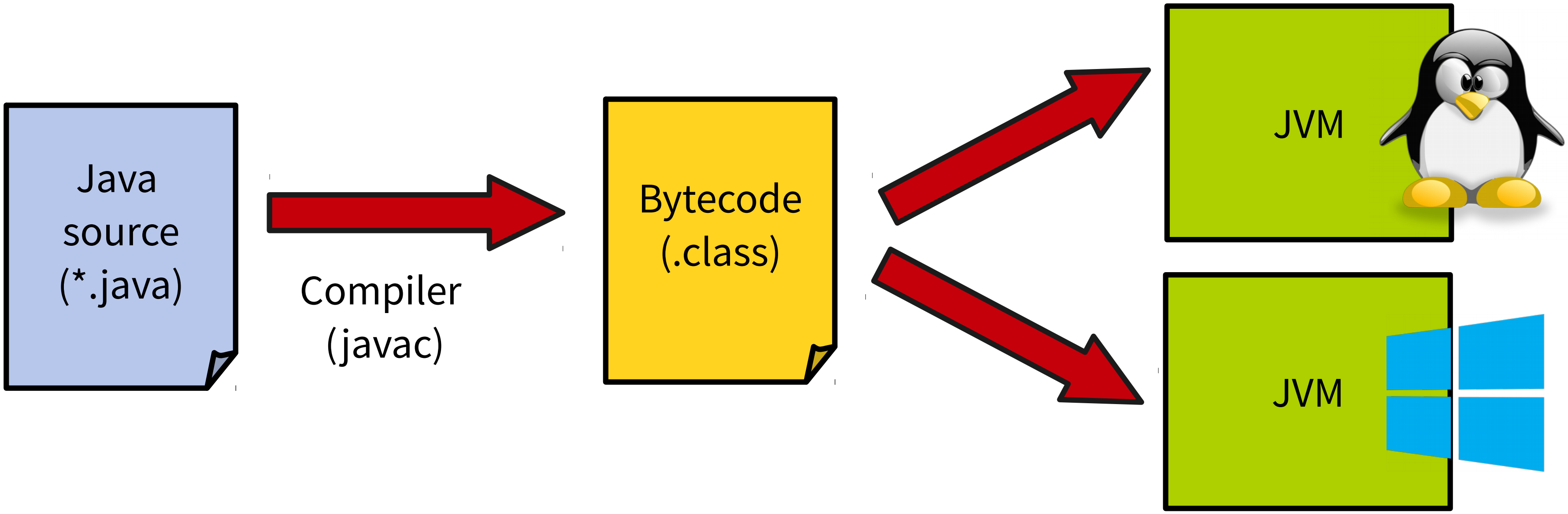


Adapted from the textbook

Virtual Machine

VM abstractions: language-based

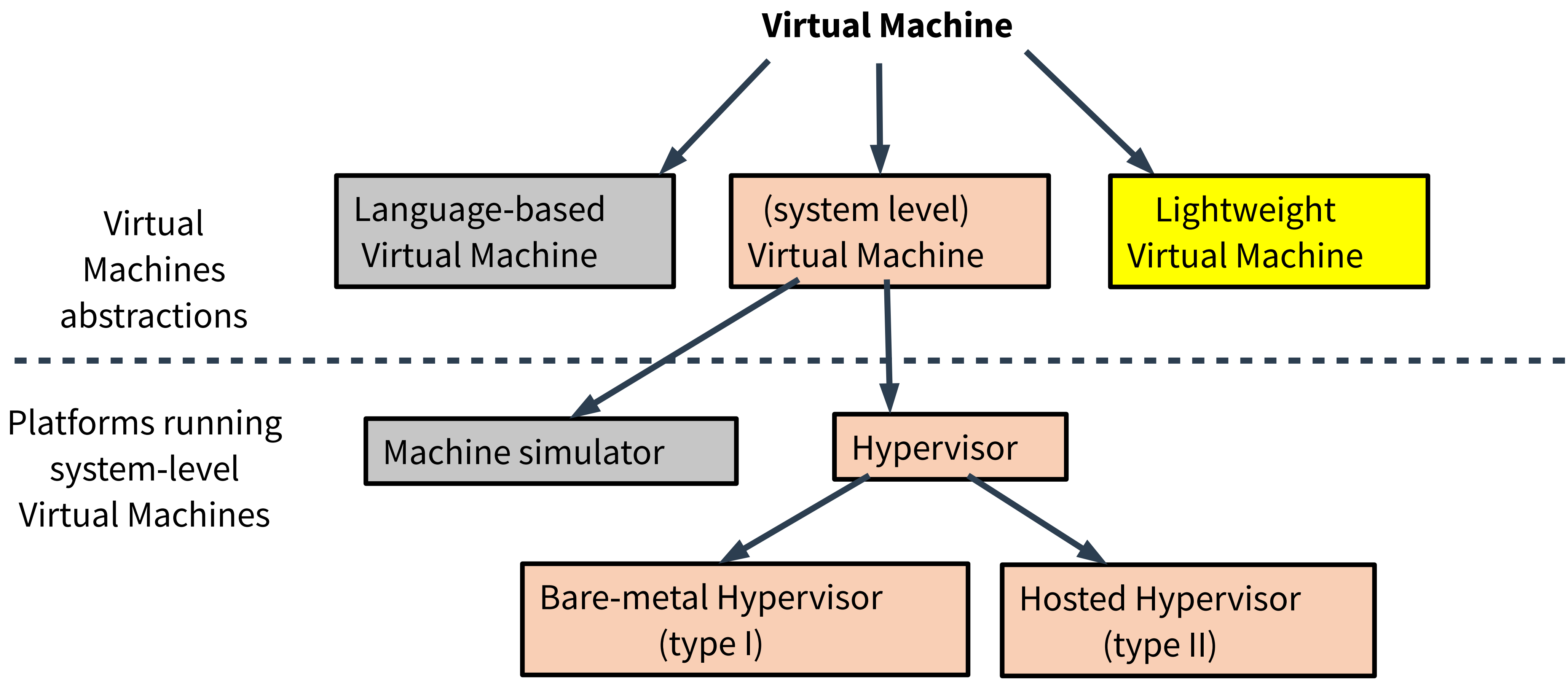
- **Java Virtual Machine, JavaScript engines, Microsoft Common Language Runtime**
  - Designed to run single applications → not the target of this course



'Interpret' bytecode,
convert it to machine code

Virtual Machine

VM & running platforms classification



Adapted from the textbook

Virtual Machine

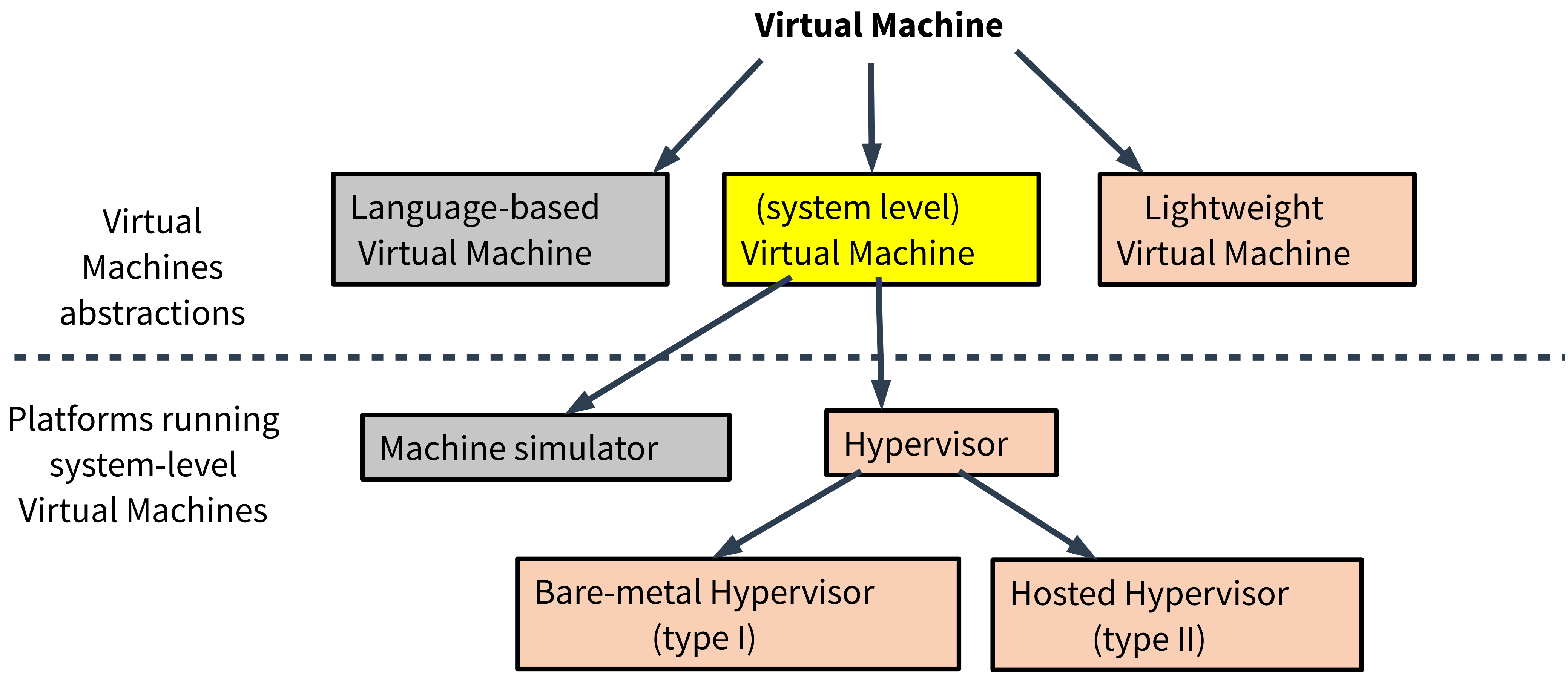
VM abstractions: lightweight virtual machines

- **Isolation of native code running directly on the CPU though hardware/software mechanisms**
  - Sandboxing
  - Stronger than regular process isolation
    - Customized OS offering more isolation guarantees than regular process-based isolation
- **No attempt is made to virtualize the hardware**
  - Isolation enforced at the OS/framework level
  - In some cases this breaks backward compatibility
    - Cannot run unmodified OS
- **Examples: Denali[1], Google Native Client[2], Vx32[3]**
- **Examples: Linux containers**

[1]Whitaker, Andrew, Marianne Shaw, and Steven D. Gribble. "Scale and performance in the Denali isolation kernel." ACM SIGOPS Operating Systems Review 36.SI (2002): 195-209.
[2] Yee, Bennet, et al. "Native client: A sandbox for portable, untrusted x86 native code." Security and Privacy, 2009 30th IEEE Symposium on. IEEE, 2009.
[3]Ford, Bryan, and Russ Cox. "Vx32: Lightweight User-level Sandboxing on the x86." USENIX Annual Technical Conference. 2008.

Virtual Machine

VM & running platforms classification

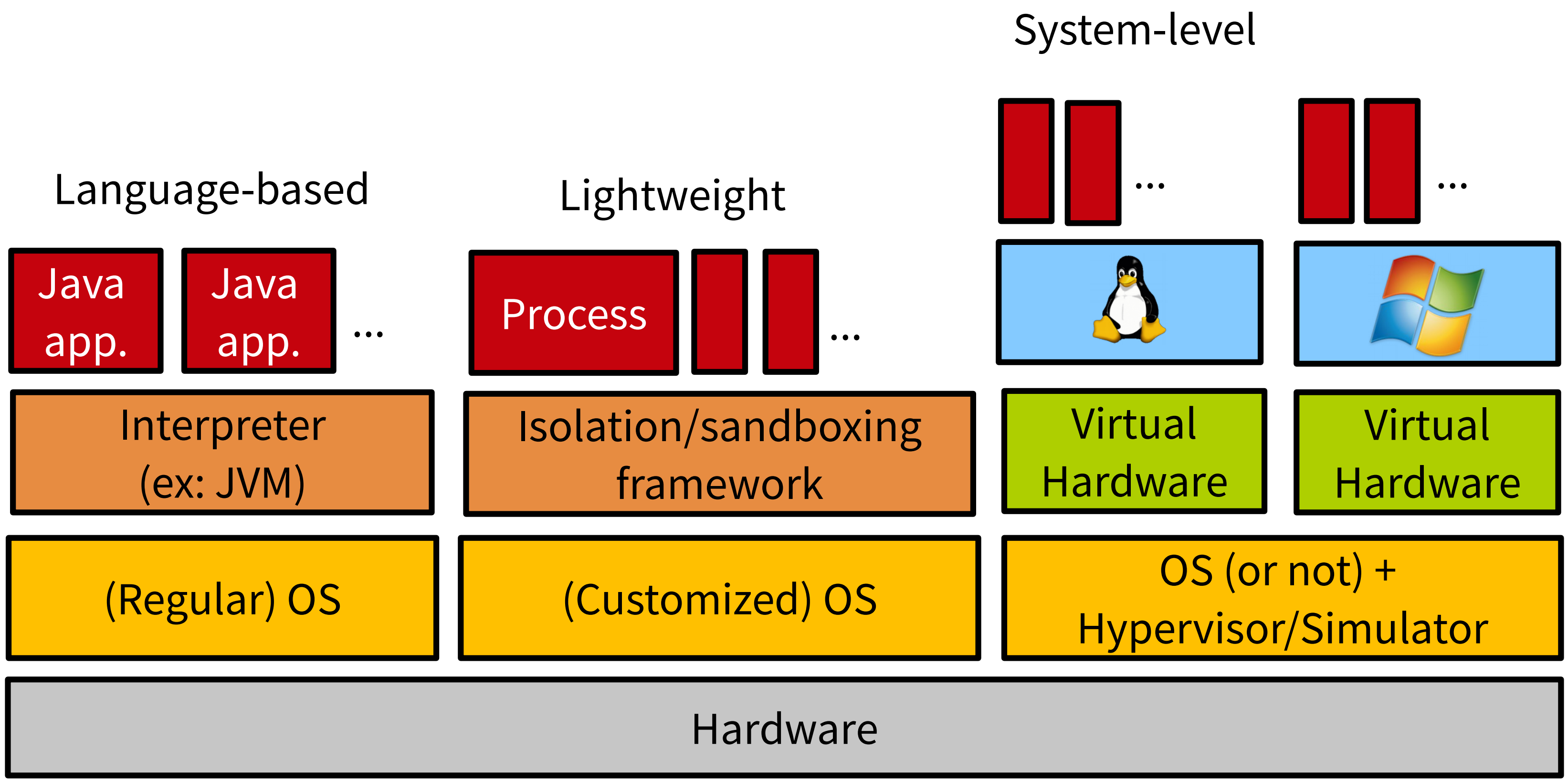


Adapted from the textbook

Virtual Machine

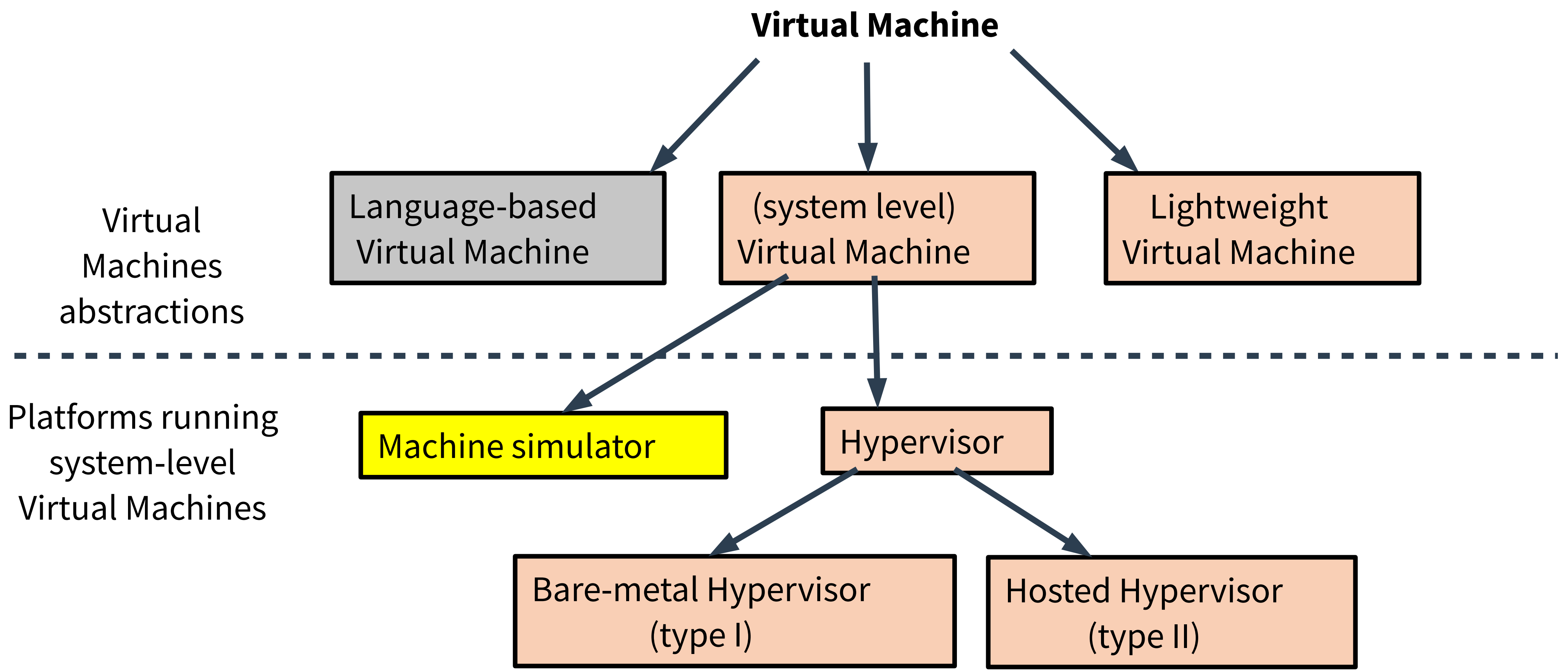
VM abstractions: system-level virtual machines

- **Creates a model of the *hardware* for a (mostly) unmodified operating system to run on top of it**
  - Each VM running on the computer has its own copy of the virtualized hardware



Virtual Machine

VM & running platforms classification



Adapted from the textbook

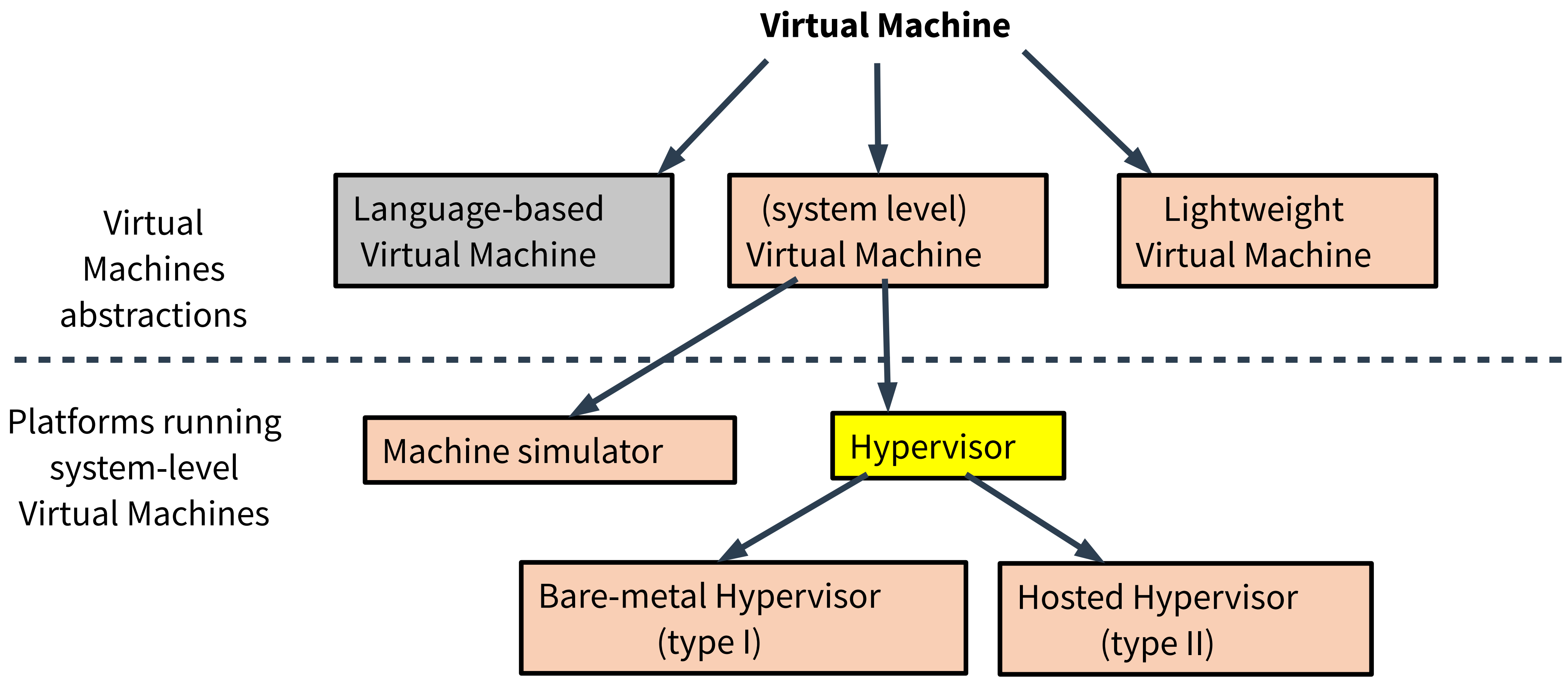
Virtual Machine

System-level platform: machine simulator

- **Normal user-level application**
  - Accurate emulation/simulation of the virtualized architecture
  - Cross-architectural emulators:
    - *Emulation: for usage as substitute*
    - Ex: Qemu in its full emulation (non-KVM) mode
      - ➔ Software prototyping, for example android (arm) simulator, embedded development on x86 development machine
  - Architecture simulators:
    - *Simulation: for analysis and study*
    - Ex: Gem5
      - ➔ Computer architecture prototyping, performance/power consumption analysis, research, etc.
- **Slow: 5x to 1000x slowdown compared to native execution**
  - Interpret each guest instruction in software

Virtual Machine

VM & running platforms classification



Adapted from the textbook

Virtual Machine

System-level platform: hypervisor

- **A hypervisor or Virtual Machine Monitor**
  - Relies on *direct execution* for performance reasons (close to native)
    - VM code executes directly on the physical CPU, at a *lower privilege level* than the hypervisor
    - Privileged instructions **trap** to the hypervisor
      - → They cannot execute in the VM context as the VM needs to be isolated
      - → Example: setting up page tables (`mov` to `cr3` in x86)
      - → Switch to the hypervisor which determines what do to with that instruction: **trap-and-emulate** model
      - → Then back to VM execution
  - Examples: Xen, Linux KVM, VMware ESXi, MS Hyper-V, Oracle VirtualBox, etc.

Outline

1) Virtualization quick definition and use cases

2) Virtualization: In-depth definition

3) Virtual Machines

4) Hypervisors

5) Memory denomination, full/hardware/para-virtualization

Hypervisors

- **The hypervisor or VMM**
  - Runs virtual machines while **minimizing virtualization overheads**
    - Tries to get as close as possible to native (non-virtualized) performance
  - **Multiplexes physical resources** between VMs
  - Ensures **isolation** between VMs and between VMs and the hypervisor
    - Isolates physical resources: for example memory/address spaces
    - Isolate performance
- **Popek & Goldberg, 1974, states that the hypervisor is:**
  - Efficient, isolated duplicate of the real machine
    - Provides an identical environment for programs
    - Runs programs with minor decrease in speed
    - Is in complete control of physical resources

Popek, Gerald J., and Robert P. Goldberg. "Formal requirements for virtualizable third generation architectures." Communications of the ACM 17.7 (1974): 412-421.

Hypervisors

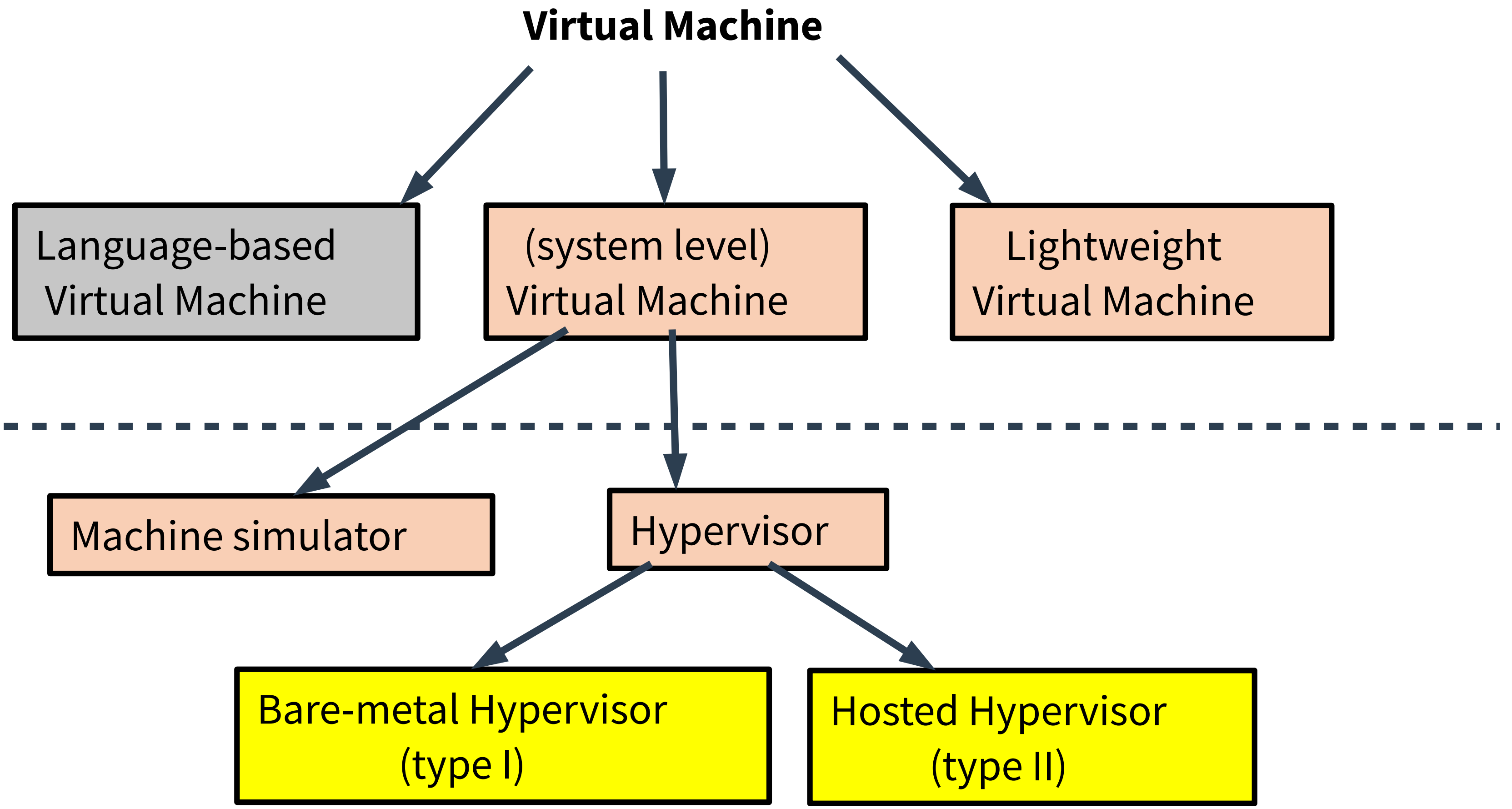
Popek, Gerald J., and Robert P. Goldberg. "Formal requirements for virtualizable third generation architectures." Communications of the ACM 17.7 (1974): 412-421.

- **The hypervisor provides virtualization applying the layering principle according to three specific criteria:**
  - Equivalence: VM is equivalent to the real machine
    - i.e. native os/programs should run (mostly) unmodified in the VM
  - Safety: VM are *isolated* from each other and from the hypervisor
    - No assumption about programs and OS running in the VM
      - ➔ They might be malicious!
  - Performance:
    - At most minor decrease in speed
      - ➔ Separates hypervisors from simulators/emulators

Hypervisors
Type I and II hypervisors

Virtual Machines abstractions

Platforms running system-level Virtual Machines

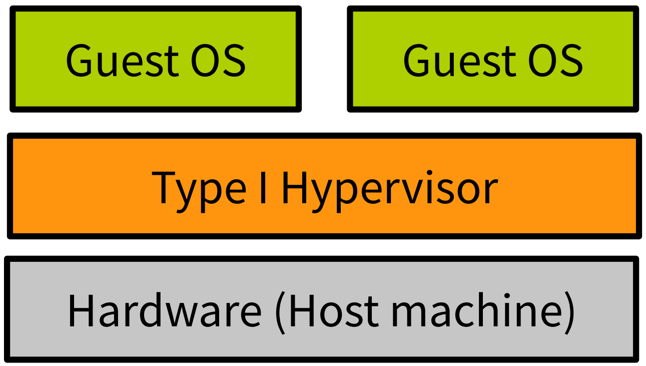


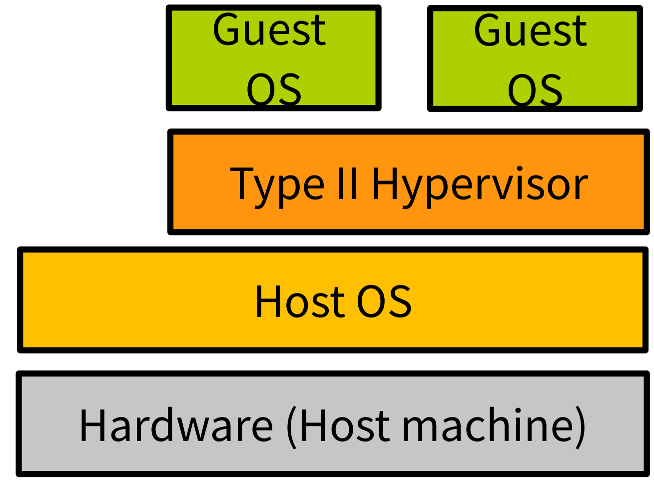
Adapted from the textbook

Hypervisors

Type I and II hypervisors

Type I: Bare-metal



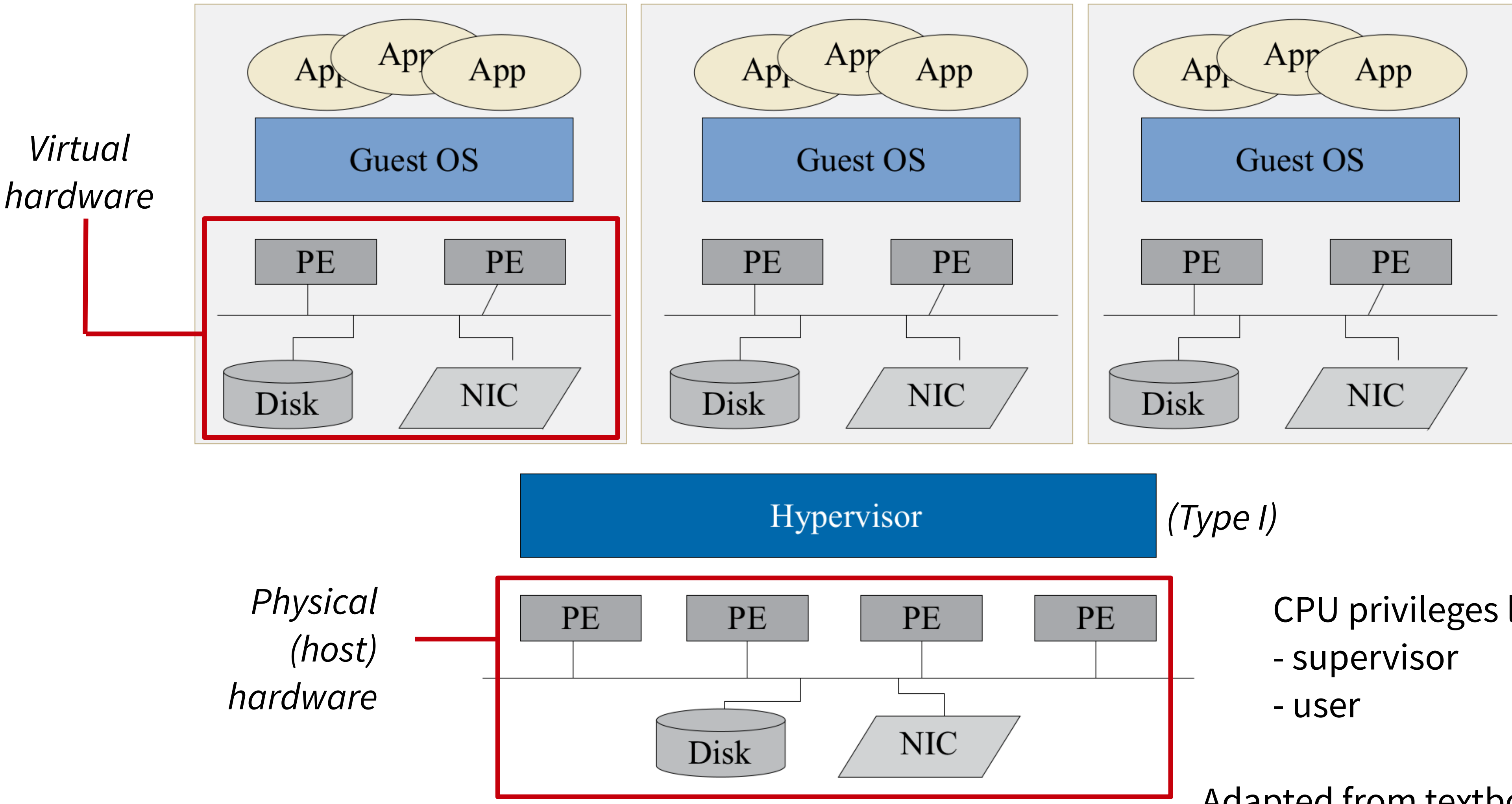


Type II: hosted

- **Resources allocation & scheduling**
  - Type I: done by the hypervisor
  - Type II: more involvement from the host OS
    - Host/guest denomination

Hypervisors

Simplified hypervisor sketch



CPU privileges levels:
- supervisor
- user

Adapted from textbook

Hypervisors
Simplified hypervisor sketch

- **Hypervisor:**
  - Multiplexes CPUs and memory between VMs
  - Emulates I/O bus and devices

Hypervisors

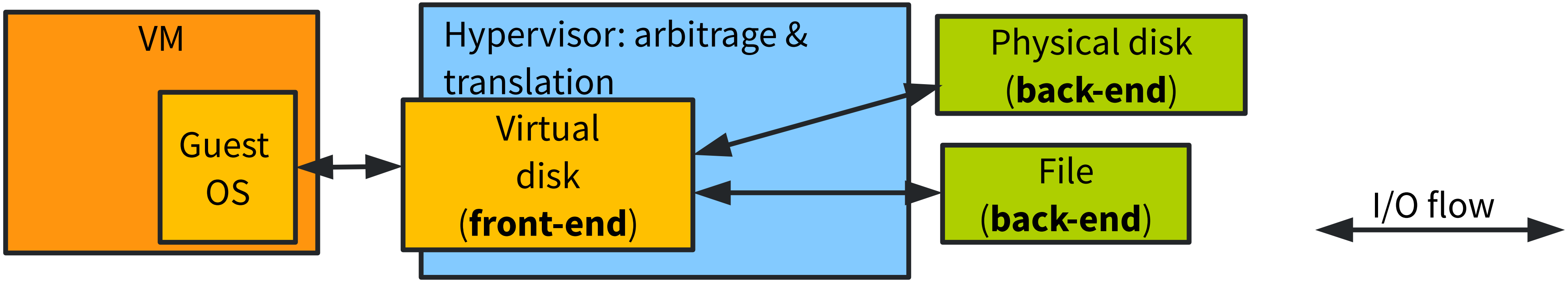
Simplified hypervisor sketch: CPU/memory multiplexing

- **Hypervisor:**
  - Multiplexes CPUs and memory between VMs
  - Emulates I/O bus and devices
- **CPU multiplexing is done for performance reasons**
  - *Efficiency criteria*: direct execution as opposed to emulation
  - Safety criteria?
    - ➔ Virtual CPU runs with reduced privileges, it cannot execute privileged instructions
    - ➔ Traps to the hypervisor on such instructions (virtualization overhead)
  - Trap-and-emulate paradigm
- **Physical memory also multiplexed**
  - Challenge: virtualizing the MMU and offering to the VM kernel/user levels of execution while the VM actually executes in user mode

Hypervisors

Simplified hypervisor sketch: I/O emulation

- **Hypervisor:**
  - Multiplexes CPUs and memory between VMs
  - Emulates I/O bus and devices
    - For compatibility
      - A VM sees the same virtual I/O devices even when running on hosts with different devices
      - I/O devices have well defined interfaces, for example: send a set of network packets, read 128K from disk from sector X, etc.
      - The hypervisor emulate simple virtual devices (disk/nic) that can be accessed with commonly implemented drivers (ex IDE/SCSI): front-end
      - Hypervisor redirects I/O to actual devices or other abstraction, ex: disk or file: back-end



Outline

1) Virtualization quick definition and use cases

2) Virtualization: In-depth definition

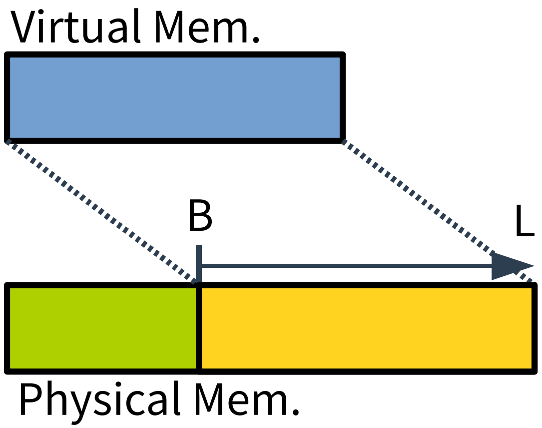
3) Virtual Machines

4) Hypervisors

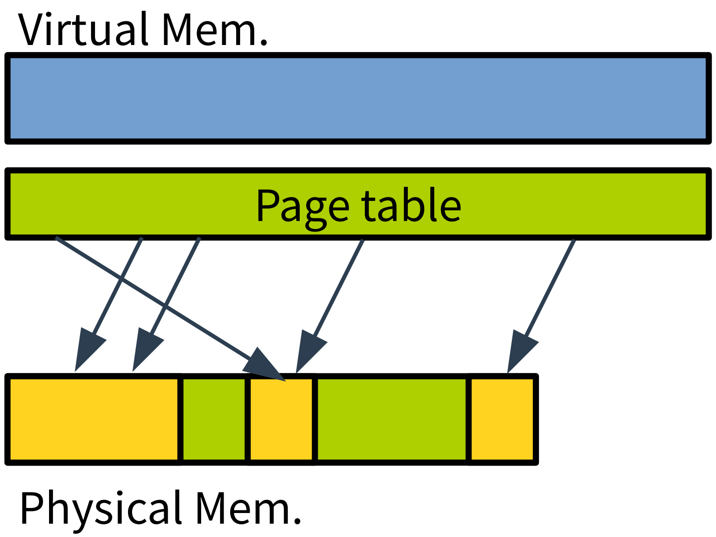
5) **Memory denomination, full/hardware/para-virtualization**

Memory denomination

- **Virtual Memory**
  - Addressable namespace accessed by the processor when virtual memory is enabled
    - Segmentation: base address + limit, set in segment registers
      - ➔ Translates to physical address by adding the base address and checking the limit for permission
    - Paging: defined by page tables
      - ➔ Translate using the page tables, permission are checked with metadata bits in the page tables entries
  - Size can be larger than the amount of physical RAM
    - Ex 256 TB for x86-64

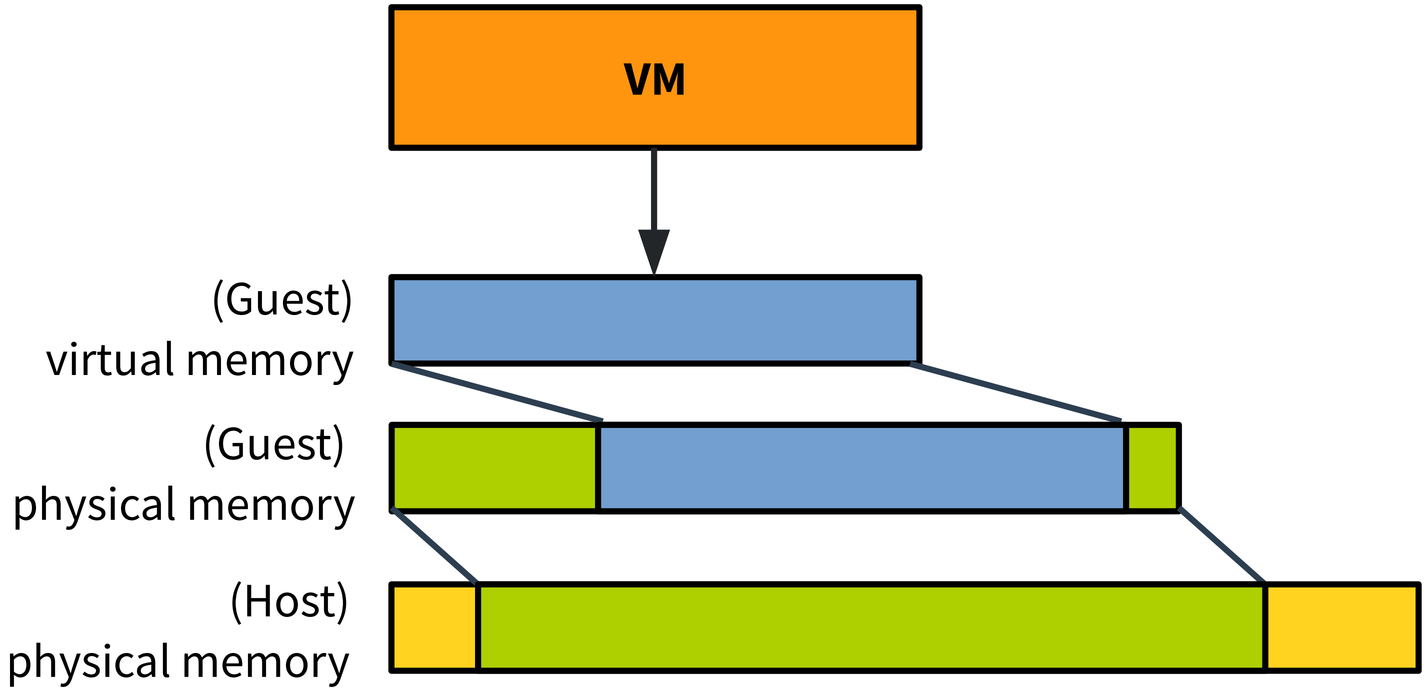


- **Physical memory**
  - Addressable physical resource, generally DRAM
  - Size equals the amount of RAM on the machine
  - In a virtualized system: difference between:
    - Guest physical memory: defined by the hypervisor and viewed by the VM
      - ➔ Also called pseudo-physical memory
    - Host physical memory: the host machine RAM
      - ➔ Also called machine memory



Memory denomination (2)

- **Guest physical memory: defined by the hypervisor and viewed by the VM**
  - Also called pseudo-physical memory
- **Host physical memory: the host machine RAM**
  - Also called machine memory



Approaches to virtualization

- Full (software) virtualization
- Hardware virtualization (HVM)
- Paravirtualization

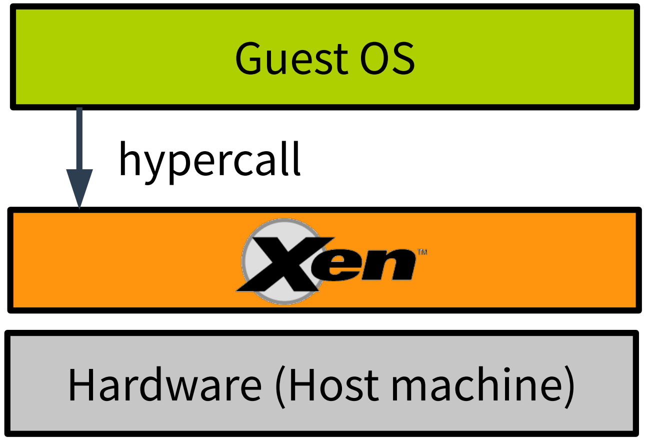
Approaches to virtualization

Full (software) virtualization and Hardware virtualization

- **Full (software virtualization):**
  - Hypervisor maximizing compatibility on non-virtualizable architecture
    - Running *completely unmodified* operating systems
    - Must interpret and translates some privileged guest instructions
    - Ex: early versions of VMware on x86-32
    - Also named software virtualization
- **Hardware virtualization**
  - Hypervisor running on architectures with hardware support for virtualization
    - Also runs completely unmodified guest OS
    - Relies (mostly) exclusively on *direct execution* to execute VM instructions
    - Also named HVM/HV
    - Ex: KVM

Approaches to virtualization
Paravirtualization

- **Relaxes compatibility constraints:**
  - Assumes the guest OS can be slightly modified
  - Guest OS privileged instructions are replaces by explicit calls to the hypervisor:
    - Hypercalls
- **Examples: Denali[1], Xen[2]**



[1]Whitaker, Andrew, Marianne Shaw, and Steven D. Gribble. "Scale and performance in the Denali isolation kernel." ACM SIGOPS Operating Systems Review 36.SI (2002): 195-209.
[2]Barham, Paul, et al. "Xen and the art of virtualization." ACM SIGOPS operating systems review. Vol. 37. No. 5. ACM, 2003.

Readings

- Textbook chapter 1
- Tanenbaum, Andrew S., and Herbert Bos. Modern operating systems. Prentice Hall Press, 2014.
  - Chapter 7: Virtualization and the cloud
- Chisnall, David. The definitive guide to the xen hypervisor. Pearson Education, 2008.
  - Chapter 1: The state of virtualization