Virtualization
COMP 252 - Lecture 6

Antoniu Pop

antoniu.pop@manchester.ac.uk

16 February 2018

Previous Lecture: Virtualization Technologies

- **Aims of virtualization**
  - Multiplex resources

Give the illusion that you own the resources.

  - Isolation/abstraction

Software does not need to know the details of the hardware on which it runs.

  - (avoid interference, safety, etc.)

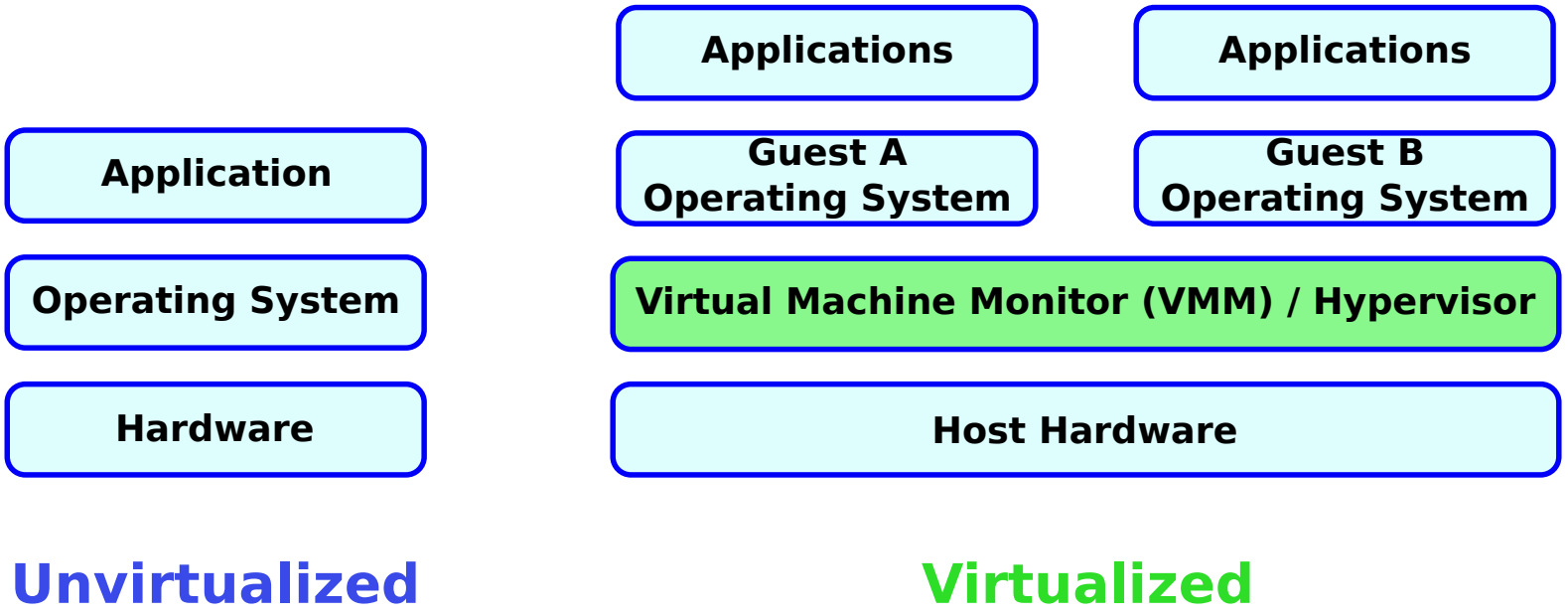
- **Process vs. System Virtualization**

- **Process virtualization**
  - JVM (“write once, run everywhere” model)
  - Dynamic Binary Translators (ISA: Rosetta, Mambo; OS&library calls: Wine)
  - Dynamic Binary Optimizers – program shepherding (Pin, Valgrind)

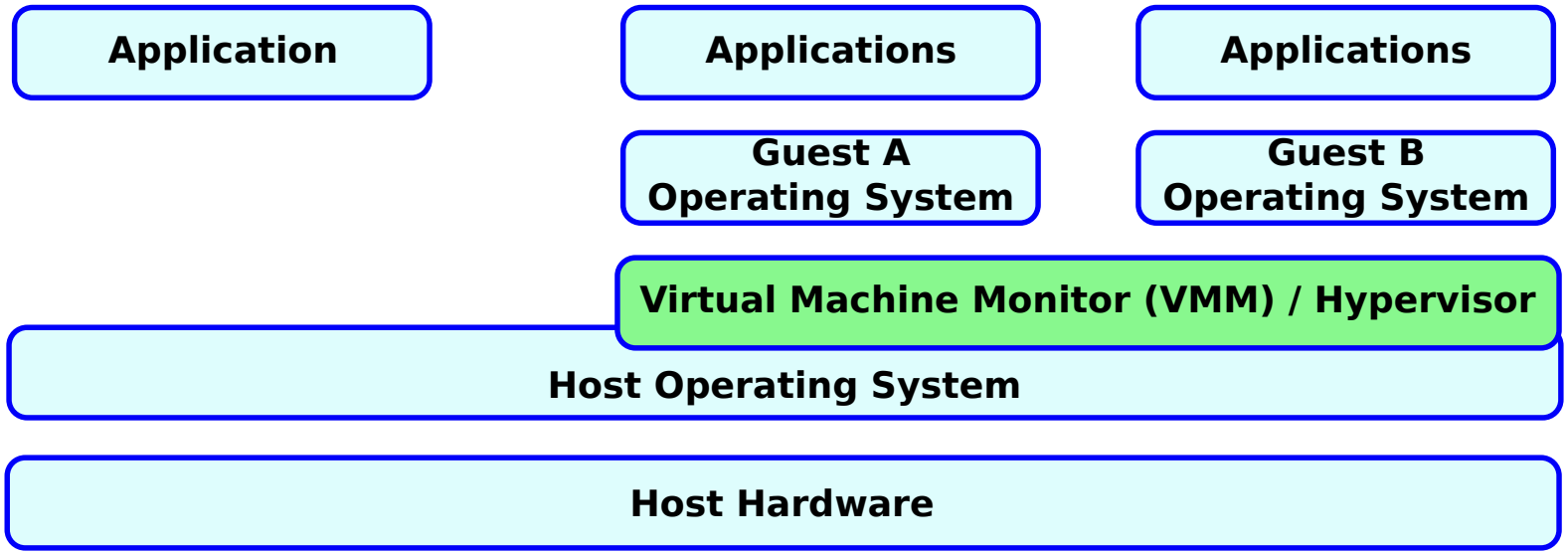
Today's Lecture – Learning Objectives

- To understand the implementation choices and details of System Virtualization
  - how virtualization works in modern architectures
  - what are the choices and characteristics of such implementations

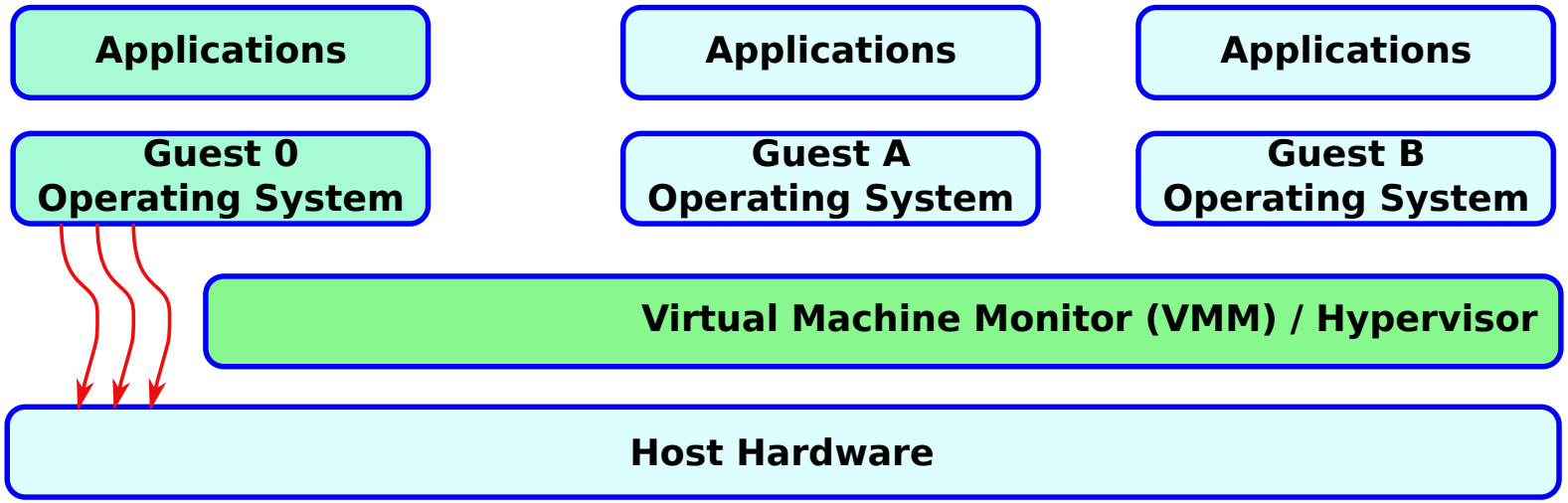
Aims and Definitions



Hosted Virtualization

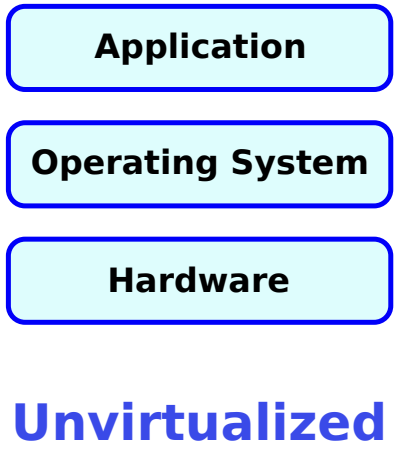


XEN Guest 0 Virtualization



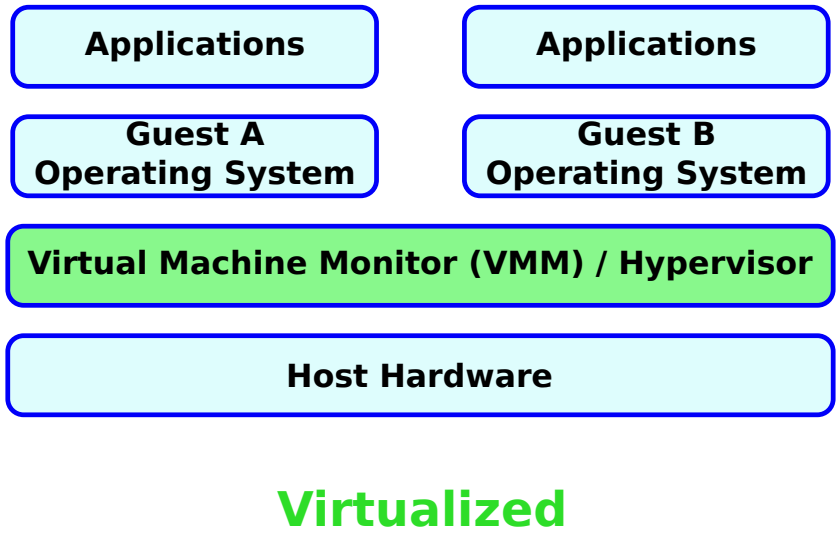
Revision: OS Protection/Privilege

- OS handles physical resources
  - Privileged
- Application isolated from resources
  - Non-privileged



Virtualization Protection/Privilege

- VMM handles physical resources
  - Privileged
- Guest OS isolated from resources
  - non- (or less-) privileged



VMM gets control on every guest OS access to physical resource

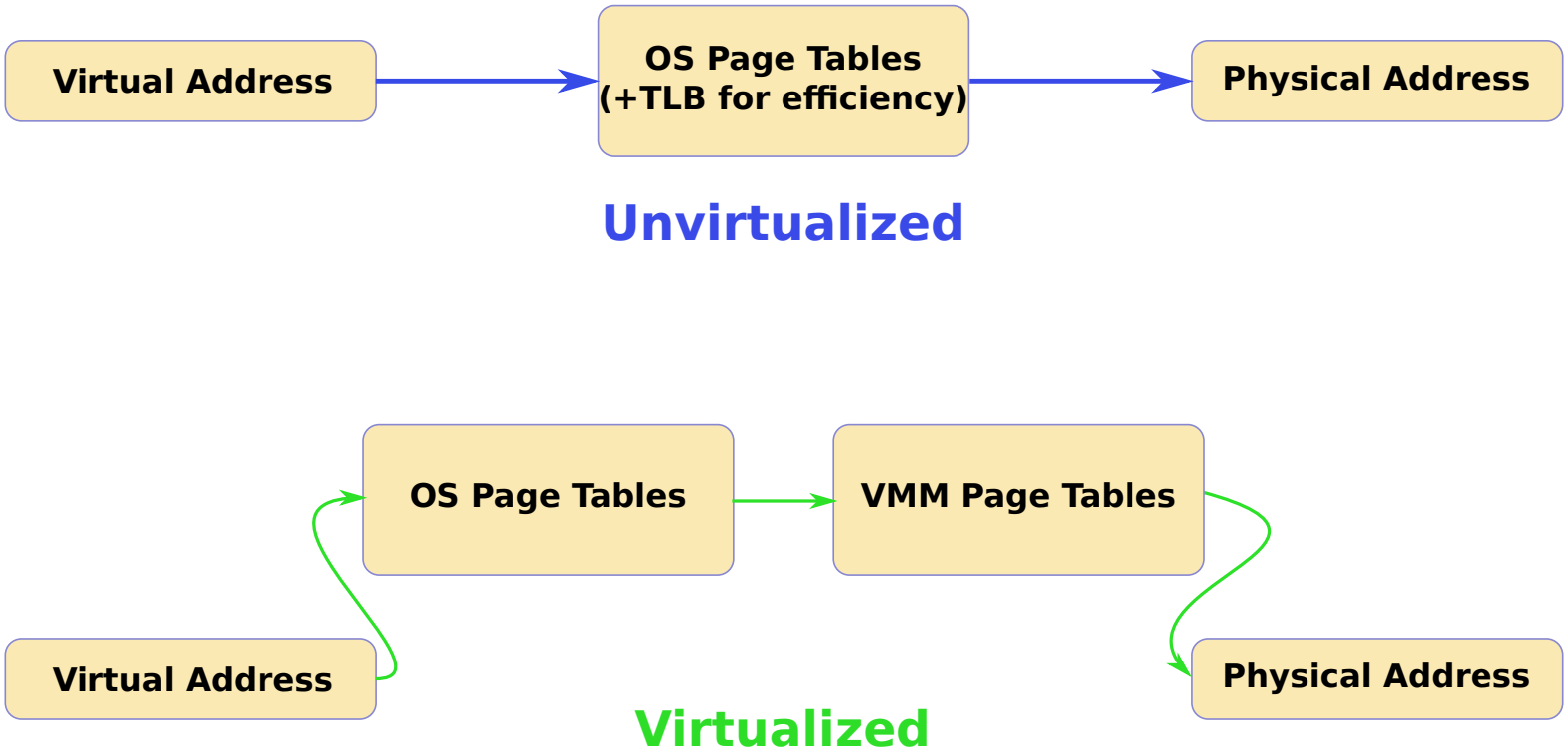
Guarded Physical Resources

- Timers
- CPU registers
  - Interrupt Enable
  - Page Table Base
- Device Control Registers
  - Programmed I/O?
  - Interrupt I/O?
  - DMA I/O?
- Interrupts (may be for different Guest?)
- Memory Mapping (page tables)

VMM Entry from Guest

- VMM designers are (a bit) lucky
  - Many Guest accesses to physical resources cause trap in non-privileged mode
  - So, running the OS in non-privileged mode suffices
- BUT some instructions behave differently (without trapping) in privileged and non-privileged mode (e.g. Intel “Store into Flags”)

Accessing Memory under Virtualization



What about TLBs?

Interfacing Guest OS and VMM

Three solutions today:

- Software (static)
- Software (dynamic)
- Hardware (dynamic)

ParaVirtualization

Modify Guest OS to be Virtualization-aware:

- call VMM for all privileged operations
- cooperate with VMM over shared page tables
- call VMM for input-output

Advantages? Disadvantages?

Detect and Fix Interfaces in VMM

- Detecting
  - Write-protect Guest OS page tables
  - Code-scan (Dynamic Binary Translation?) Guest OS for unsafe instructions – plant traps
- Fixing
  - Use write-error trap to detect guest page-table writes
  - Provide “shadow page tables” for hardware TLBs
  - Use “illegal instruction” and “trap” traps

Detect and Fix Interfaces in Hardware

- Requirement
  - VMM runs more-privileged than Guest OS
- Hardware provides Application/OS and VMM modes
- When Virtualization is active, all OS accesses to physical resources trap to VMM

Advantages? Disadvantages?