Introduction to Virtualization

Florent Gluck - Florent.Gluck@hesge.ch

February 21, 2023

*Thanks to Prof. Ada Gavrilovska for letting me use some of the contents of her course "Introduction to Operating Systems" tought at Georgia Institute of Technology

What is virtualization?

- Process of creating a **virtual representation** of something through abstractions, such as virtual computer hardware platforms, storage devices, or computer network resources

- **Not new**: began in the 1960s, as a method of logically dividing the system resources provided by mainframe computers between different operating systems and applications

Types of virtualization

Common (none-exhaustive) types of virtualization:

- Platform (hardware) virtualization

Types of virtualization

Common (none-exhaustive) types of virtualization:

- Platform (hardware) virtualization
- Operating system (OS) virtualization

Types of virtualization

Common (none-exhaustive) types of virtualization:

- Platform (hardware) virtualization
- Operating system (OS) virtualization
- Storage virtualization

Types of virtualization

Common (none-exhaustive) types of virtualization:

- Platform (hardware) virtualization
- Operating system (OS) virtualization
- Storage virtualization
- Network virtualization

Types of virtualization

Common (none-exhaustive) types of virtualization:

- Platform (hardware) virtualization
- Operating system (OS) virtualization
- Storage virtualization
- Network virtualization
- Language-based virtualization

Platform (hardware) virtualization

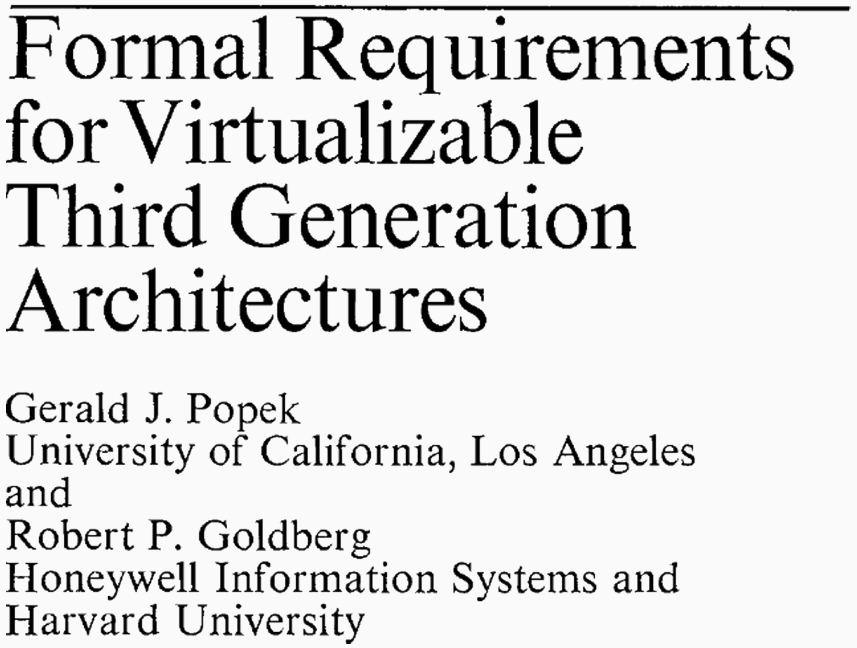
- Platform virtualization originated in the 1960s at IBM when norm of computing was few large mainframe computers shared by many users and many business services
- **Idea**: allow different users to access a computer as if they had each total control of the physical hardware
- A **virtualization layer** created, for each OS running on top of it, all the facilities of the underlying hardware in **virtual form**

- First virtual machine OS
- Supported address relocation h/w and 4 OSes

Virtual Machine

- **"Virtual Machine" (VM)** originally defined by Popek and Goldberg in 1974

- **"An efficient, isolated duplicate of a real computer machine"**

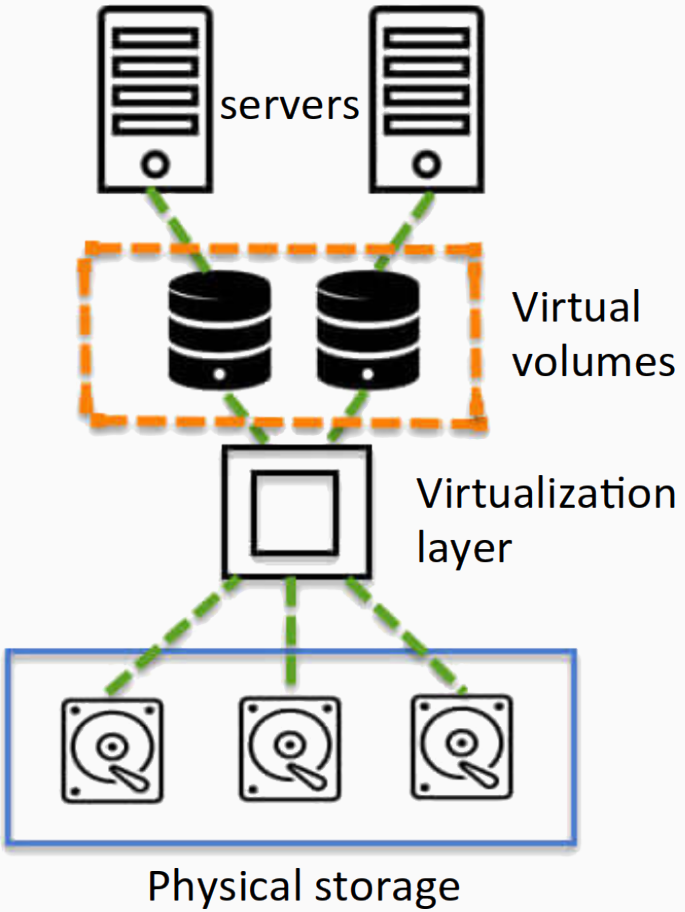


Formal Requirements for Virtualizable Third Generation Architectures

Gerald J. Popek
University of California, Los Angeles
and
Robert P. Goldberg
Honeywell Information Systems and
Harvard University

Storage virtualization

- Storage virtualization **abstracts** storage-management software from underlying hardware infrastructure
- **Benefits**:
  - flexibility
  - scalability



Storage virtualization examples

- Examples?

Storage virtualization examples

- Examples?
  - LVM (Logical Volume Manager)

Storage virtualization examples

- Examples?
  - LVM (Logical Volume Manager)
  - BTRFS filesystem

Storage virtualization examples

- Examples?
  - LVM (Logical Volume Manager)
  - BTRFS filesystem
  - Hardware RAID (Redundant Array of Independent Disks)

Storage virtualization examples

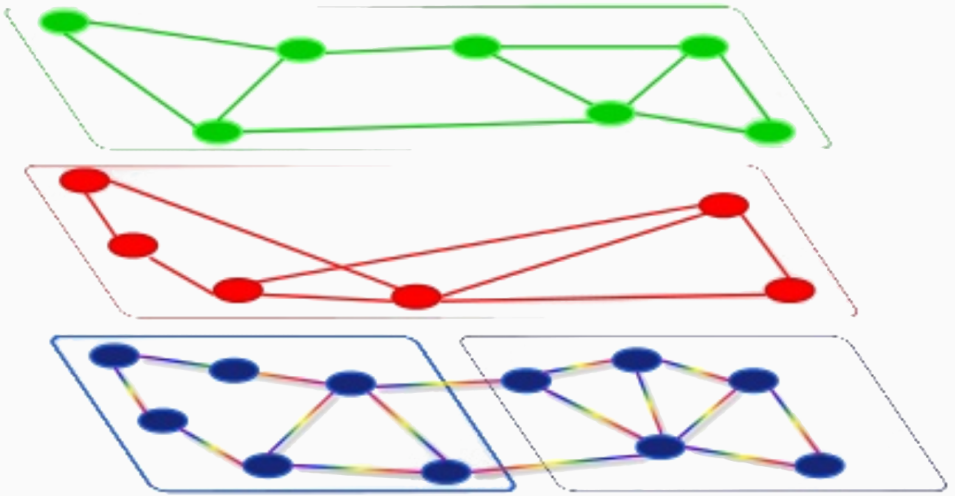
- Examples?
  - LVM (Logical Volume Manager)
  - BTRFS filesystem
  - Hardware RAID (Redundant Array of Independent Disks)
  - Software RAID (mdraid in Linux and dmraid over LVM)

Storage virtualization examples

- Examples?
  - LVM (Logical Volume Manager)
  - BTRFS filesystem
  - Hardware RAID (Redundant Array of Independent Disks)
  - Software RAID (mdraid in Linux and dmraid over LVM)
  - Hardware SAN (Storage Area Network)

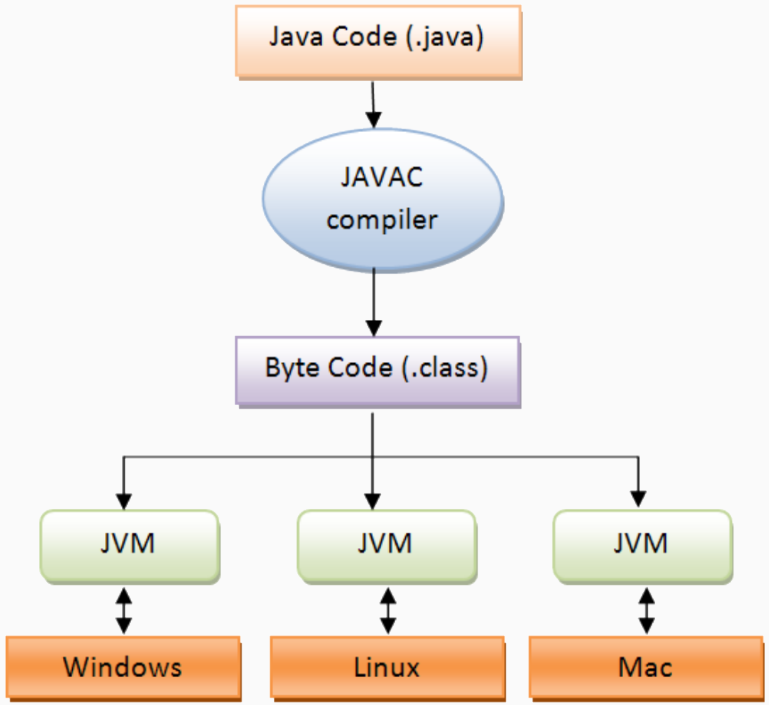
Network virtualization

- Network virtualization is the **abstraction** of the physical network:
  - **abstracts** the networking resources into a logical model so that the same set of physical resources can be shared by multiple tenants in a secure and isolated manner
- Example: software-defined network (SDN)



Language-based virtualization

- Langage compiled into machine instructions targeting an **architecture that does not physically exists** $\rightarrow$ virtual architecture
- **Benefit**: architecture independence which provides application (binary) portability between platforms
- **However**: requires a VM for each platform to support



Language-based virtualization examples

- Examples?

Language-based virtualization examples

- Examples?
  - JVM

Language-based virtualization examples

- Examples?
  - JVM
  - Dalvik

Language-based virtualization examples

- Examples?
  - JVM
  - Dalvik
  - Python VM

Language-based virtualization examples

- Examples?
  - JVM
  - Dalvik
  - Python VM
  - V8 js VM

Language-based virtualization examples

- Examples?
  - JVM
  - Dalvik
  - Python VM
  - V8 js VM
  - .NET CLR

System simulation

- Software **simulates** hardware components that make up a target machine
- Interpreter executes each instruction & updates the software representation of the hardware state
- Approach is **very accurate but very slow**
- Used to develop software for particular types of embedded hardware

System simulation examples

- Examples?

System simulation examples

- Examples?
  - Logisim

System simulation examples

- Examples?
  - Logisim
  - Wind River Simics

System simulation examples

- Examples?
  - Logisim
  - Wind River Simics
  - Ngspice

System emulation

- **Emulate** just enough of system hardware (hw) components to create an accurate "user experience"
- Typically, CPU & memory subsystems are emulated, but buses are not
- Many **shortcuts** taken to achieve better performance
  - Reduces overall system accuracy
  - Code designed to run correctly on real hw executes “pretty well”
  - Code not designed to run correctly on real hw exhibits wildly divergent behavior

System emulation examples

- Examples?

System emulation examples

- Examples?
  - QEMU

System emulation examples

- Examples?
  - QEMU
  - Bochs

System emulation examples

- Examples?
  - QEMU
  - Bochs
  - DOSBox

System emulation examples

- Examples?
  - QEMU
  - Bochs
  - DOSBox
  - MAME (Arcade)

System emulation examples

- Examples?
  - QEMU
  - Bochs
  - DOSBox
  - MAME (Arcade)
  - PCSX2 (PS2)

System emulation examples

- Examples?
  - QEMU
  - Bochs
  - DOSBox
  - MAME (Arcade)
  - PCSX2 (PS2)
  - UAE (Amiga)

System emulation examples

- Examples?
  - QEMU
  - Bochs
  - DOSBox
  - MAME (Arcade)
  - PCSX2 (PS2)
  - UAE (Amiga)
  - ZSNes (Super Nintendo)

System emulation examples

- Examples?
  - QEMU
  - Bochs
  - DOSBox
  - MAME (Arcade)
  - PCSX2 (PS2)
  - UAE (Amiga)
  - ZSNes (Super Nintendo)
  - DGen (Sega Genesis)

System emulation examples

- Examples?
  - QEMU
  - Bochs
  - DOSBox
  - MAME (Arcade)
  - PCSX2 (PS2)
  - UAE (Amiga)
  - ZSNes (Super Nintendo)
  - DGen (Sega Genesis)
  - Vice (C64)

Application emulation

- System emulation emulates the **whole system**, including hardware
- Application emulation **only emulates** the programming interfaces used by an application compiled for a given OS, so it can run on another OS with different programming interfaces

Application emulation examples

- Examples?

Application emulation examples

- Examples?
  - Wine (https://www.winehq.org/)

Application emulation examples

- Examples?
  - Wine (https://www.winehq.org/)
    - compatibility layer capable of running Windows applications on several POSIX-compliant OSes (Linux, OSX, etc.)

Application emulation examples

- Examples?
  - Wine (https://www.winehq.org/)
    - compatibility layer capable of running Windows applications on several POSIX-compliant OSes (Linux, OSX, etc.)
    - translates Windows API calls into POSIX calls on-the-fly

Application emulation examples

- Examples?
  - Wine (https://www.winehq.org/)
    - compatibility layer capable of running Windows applications on several POSIX-compliant OSes (Linux, OSX, etc.)
    - translates Windows API calls into POSIX calls on-the-fly
  - WSL1 (Windows Subsystem for Linux, version 1)

Application emulation examples

- Examples?
  - Wine (https://www.winehq.org/)
    - compatibility layer capable of running Windows applications on several POSIX-compliant OSes (Linux, OSX, etc.)
    - translates Windows API calls into POSIX calls on-the-fly
  - WSL1 (Windows Subsystem for Linux, version 1)
    - translates Linux system calls to Windows kernel compatible calls/behavior

Virtualization vs emulation vs simulation

- **Virtual Machine**
  - Model a machine exactly and efficiently
  - Minimal slowdown
  - Must run on the physical machine it virtualizes (more or less)
- **System Emulator**
  - Provides a behavioural model of hardware (and possibly software)
  - Not fully accurate
  - Reasonably fast
- **System Simulator**
  - Provides a functionally accurate software model of a machine
  - May run on any hardware
  - Typically slow

Why virtualize? (1/2)

Unfortunate **coupling** between hardware resources and the OS:

- Hard to run **multiple OSes** on the same machine
- Difficult to **transfer software setups** to another machine, unless identical or nearly identical hardware
- Messy to **adjust hardware** resources to system needs $\rightarrow$ requires sticking your hands in the box
- Requires **static**, up-front **provisioning** of machine resources

Why virtualize? (2/2)

Lack of true **isolation** between multiple applications:

- Operating systems "**leak**" information between processes through file system and other channels
- Multiple applications may require **specific and conflicting software** packages to run
- Certain applications may have very specific OS configuration and tuning requirements
- Software vendors *may not provide support* if their application runs alongside **anything** else

Benefits of virtualization (1/2)

- **Security**: bugs and faults **isolation**
- **Availability & reliability**
  - OS + apps **decoupled** from physical hardware → **live migration** of VMs from one physical machine to another
  - increase services availability, greater reliability
- **Consolidation**: ability to run **multiple VMs** on a single platform → decrease cost, improve manageability

Benefits of virtualization (2/2)

- **Functionality**: ability to run a native app for a **different OS**
- **Flexibility**:
  - can easily replicate an entire machine image in order to duplicate it or move it
  - software packaging and distribution
- **Development**: kernel, debugging, systems research, new architectures
- **Support**: legacy operating systems & applications

Resources

- “Virtual Machines: Versatile Platforms for Systems and Processes”; J. Smith, R. Nair; Morgan Kaufmann, 2005
- “Introduction to Operating Systems” by Prof. Ada Gavrilovska, Georgia Institute of Technology
- “Operating Systems: Three Easy Pieces”; Remzi H. et Andrea C. Arpaci-Dusseau; Arpaci-Dusseau Books, 2020