

IT ARCHITECTURE

VIRTUALIZATIONS

Willy Sudiarto Raharjo

Teknik Informatika
Universitas Kristen Duta Wacana

History

- Since 1960
- Purpose:
 - ◆ Enhance resource sharing
 - ◆ Improve computer performance
 - Resource utilization
 - Application flexibility
- Idea:
 - ◆ Separate hardware from software
 - ◆ Results: better system efficiency

Before Virtualization



After Virtualization



- 1:1 ratio of OSs and applications per server

- 1:Many ratio of OSs and applications per server
- Additional layer to manage and secure

Application level

JVM / .NET CLR / Panot

Library (user-level API) level

WINE/ WABI/ LxRun / Visual MainWin / vCUDA

Operating system level

Jail / Virtual Environment / Ensim's VPS / FVM

Hardware abstraction layer (HAL) level

VMware / Virtual PC / Denali / Xen / L4 /
Plex 86 / User mode Linux / Cooperative Linux

Instruction set architecture (ISA) level

Bochs / Crusoe / QEMU / BIRD / Dynamo

Instruction Set Architecture Level

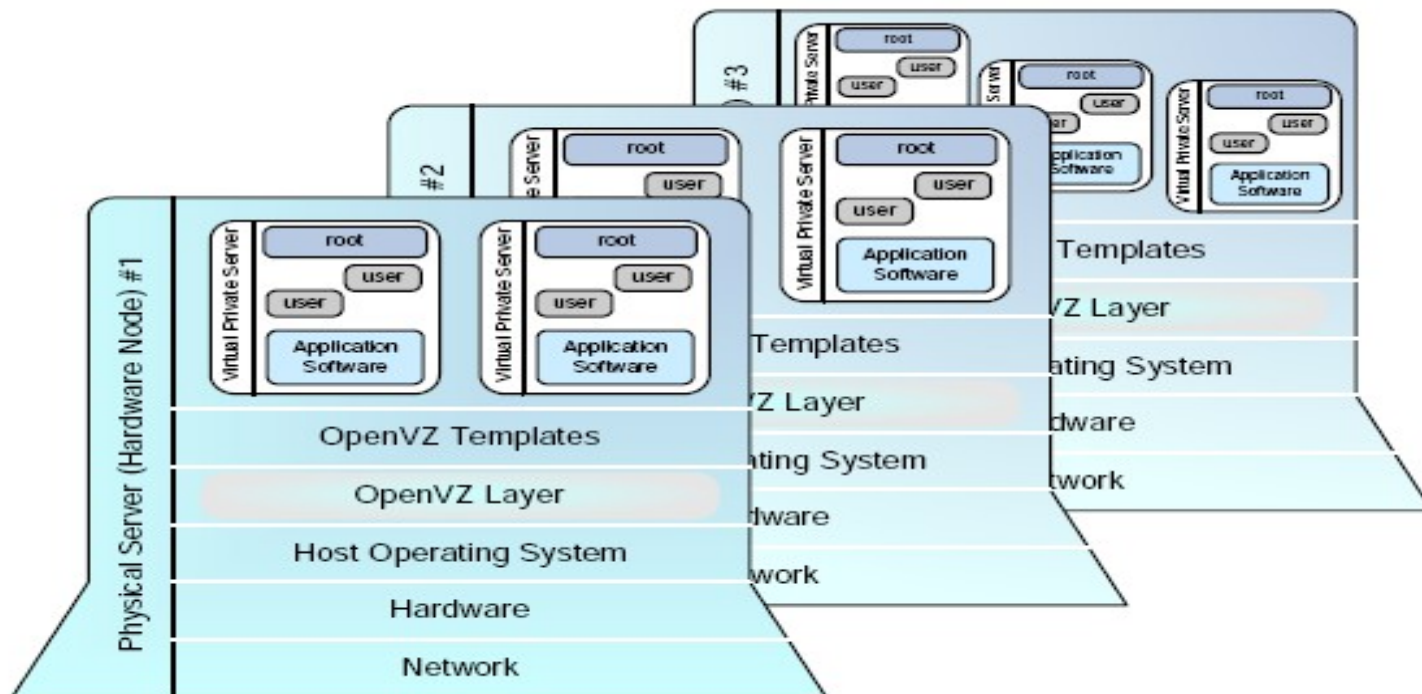
- Emulating ISA by the ISA of the host machine
- Allows legacy binary code to be runnable on any new hardware host machine
- Basic emulation method: code interpretation
 - ◆ Relatively slow
 - ◆ Solution: dynamic binary translations
 - Adding processor specific software translation layer to compiler

Hardware Abstraction Layer

- Performed on top of the bare hardware
- Generates virtual hardware environment
- Purpose: increase hardware utilization rate
 - ◆ Processors
 - ◆ Memory
 - ◆ I/O devices

OS Level

- Abstraction layer between traditional OS and user apps
- Creating isolated containers on a single physical server
- Must have the same kind of guest OS



Library Support Level

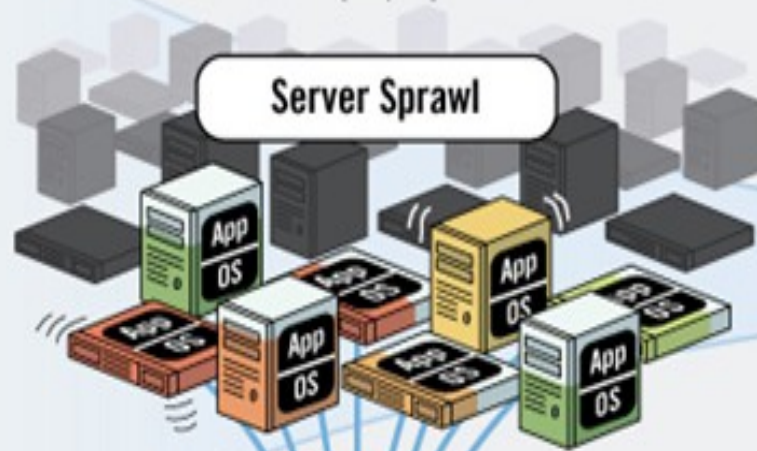
- Most application use API rather than system calls
- APIs are well documented, so we can virtualize it
- API Call are intercepted and remapped
 - ◆ Creating execution environment rather than the entire OS
 - ◆ Results: applications on one platform can be executed in other platform
 - ◆ Example: WINE project
- Also known as Application Binary Interface (API Emulation)

User-Application Level

- Virtualize application as VM
- Traditional OS: application runs as a process
 - ◆ → process-level virtualization
- Virtualization layer sits as an application on top of the OS
 - ◆ Exports abstraction of a VM
 - ◆ Run program written and compiled to particular abstract machine definition
 - ◆ Enables multiplatform application, such as Java-based apps

Before

Server Sprawl



Network Speed
1Gb Ethernet



Storage Sprawl

After

Virtualized Servers

App	App	App	App	App	App
OS	OS	OS	OS	OS	OS



Network Speed
10Gb / 20Gb+
Ethernet / InfiniBand

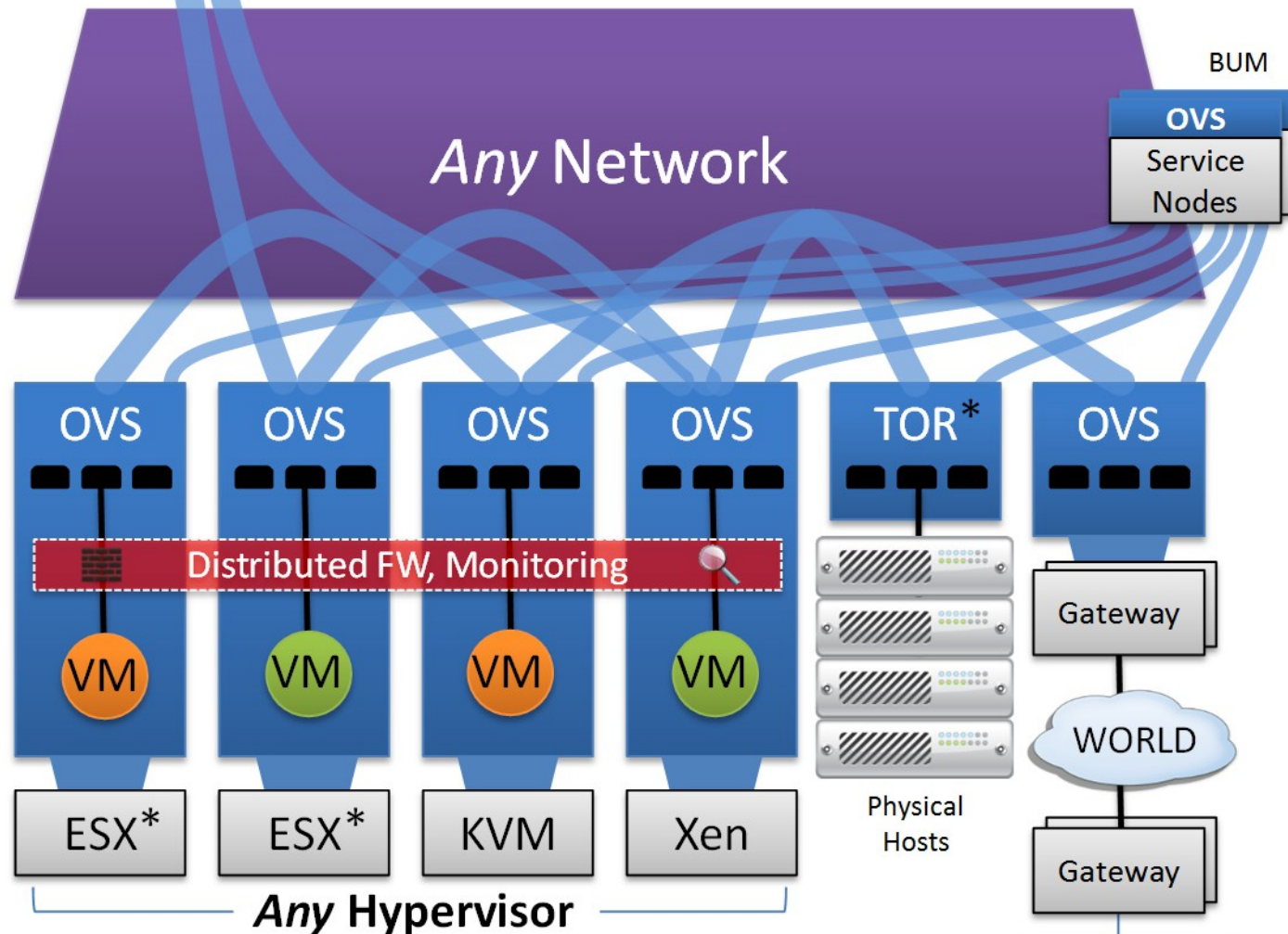
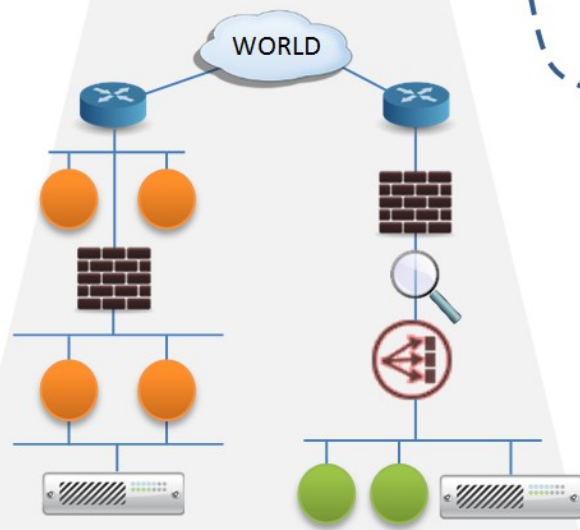
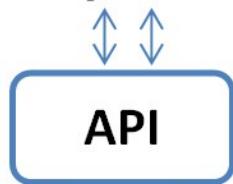
Virtualized/Clustered Storage



Isilon: Cleaning up your environment

Network Virtualization Platform

vmware® nicira



BRAD HEDLUND .com

*roadmap item

Remote Site

Benefits of Virtualization



- Reduction in operating costs

(For 300 servers)	Physical	Virtual
Floor space (Assuming \$2/month/sq.ft.)	150 sq. ft. (25 racks) \$3,600	12 sq. ft. (2 racks) \$288
Cooling requirements (HVAC tonnage) (Assuming \$0.08/kWh)	8,760 hours/year 42 \$29,434	8,760 hours/year 27 \$18,922
Power requirements (Assuming \$0.08/kWh)	125kW \$105,120	95.4kW \$66,856
Hardware costs	\$3.84MM	\$2.7MM

- Total savings by virtualization: \$1.2 Million

Virtualization Structures/Tools

- Hypervisor Architecture / Virtual Machine Monitor
- Para-virtualization
- Host-based virtualization

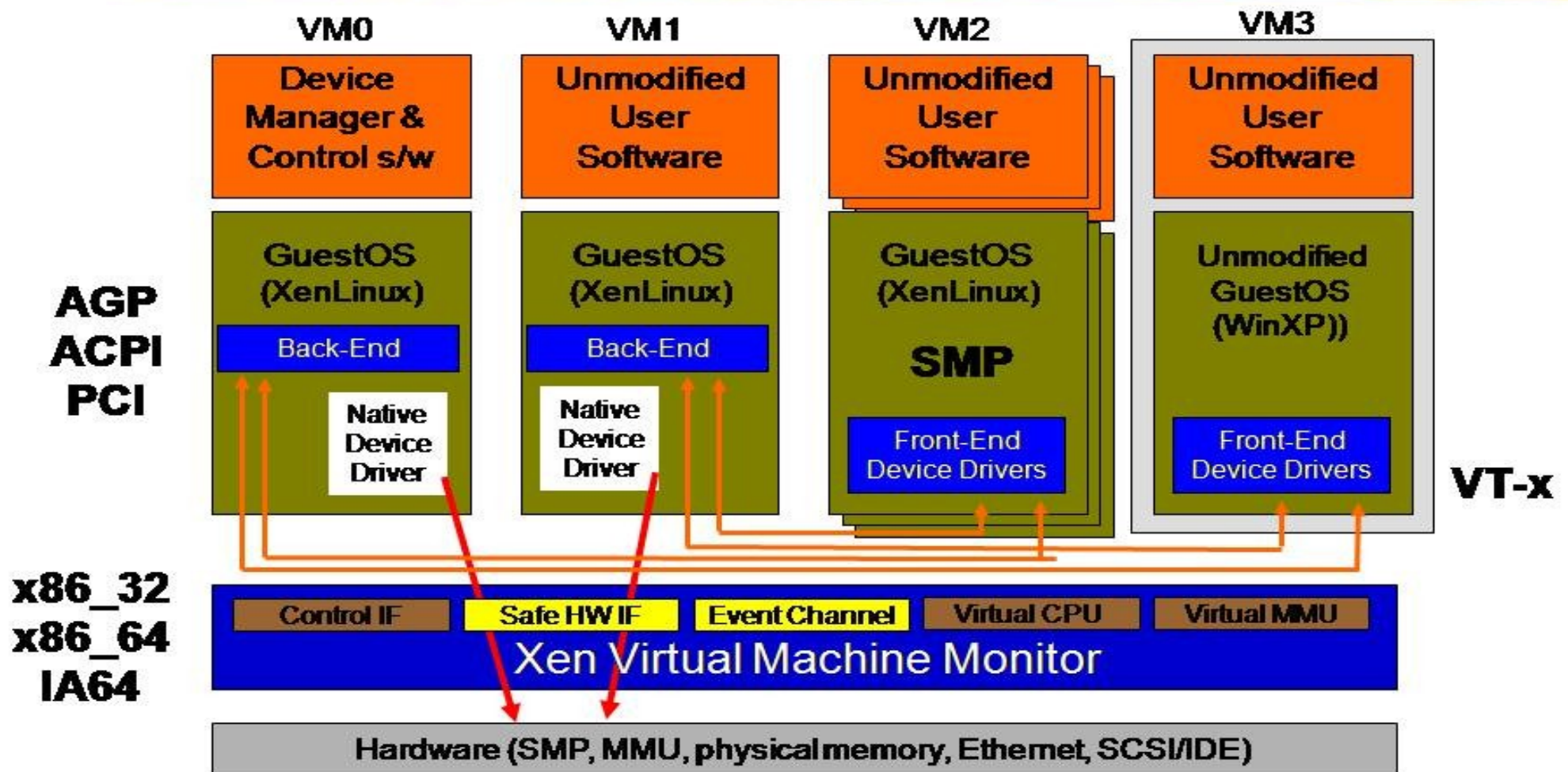
Hypervisor

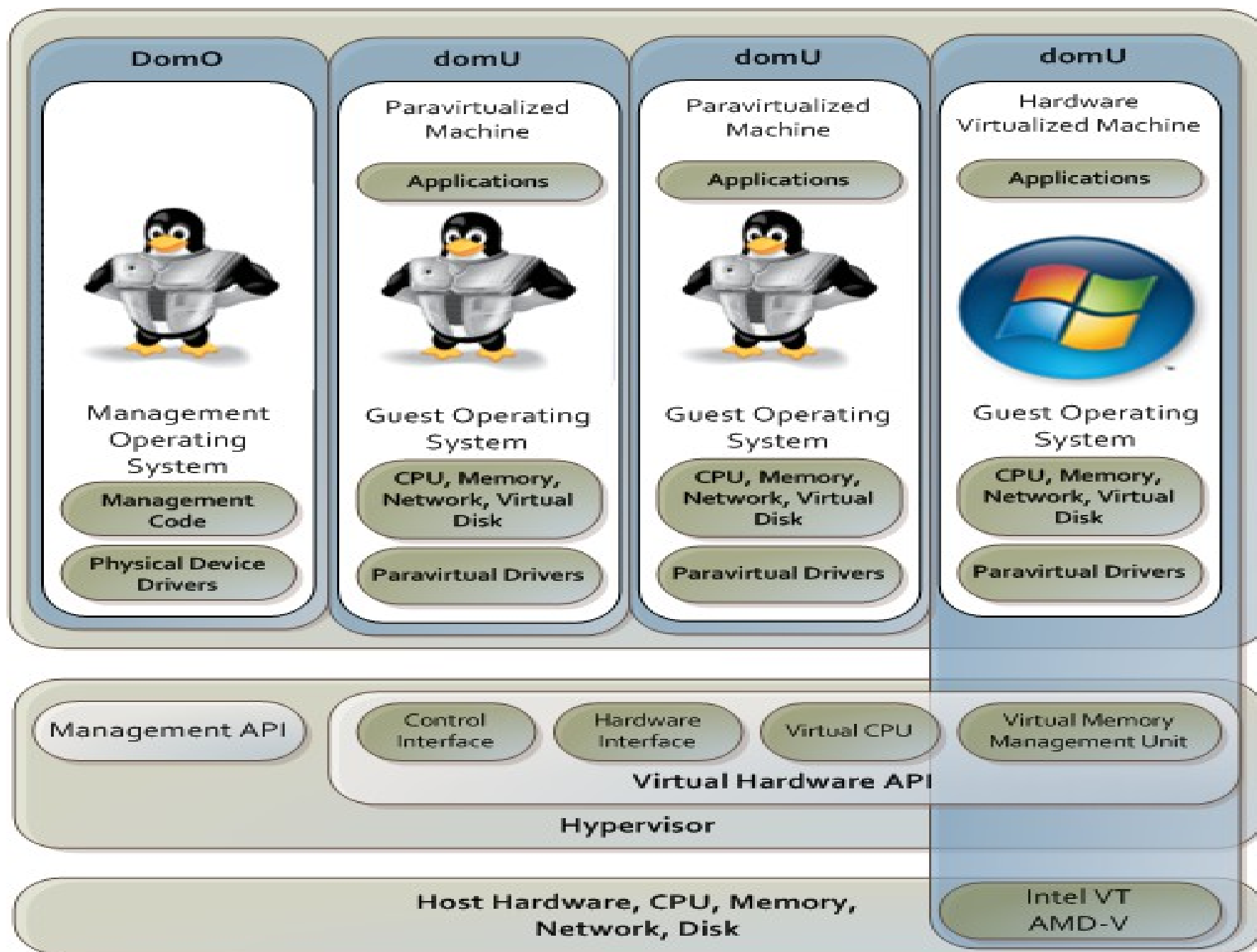
- Supports hardware-level virtualization
- Located between physical hardware and OS
- Micro-kernel hypervisor
 - ◆ Basic and unchanging functions
 - memory management and processor scheduling
 - ◆ Other changeable components are outside hypervisor
 - ◆ Example: Microsoft Hyper-V
- Monolithic hypervisor
 - ◆ Implements all functions
 - ◆ Bigger in size

Xen Architecture

- Open source hypervisor program by Cambridge University
- Micro-kernel hypervisor
- Do not include any device driver natively
- Enables guest OS to have direct access to physical device
- Guest OS that has control ability → Domain 0
 - ◆ Loaded when Xen boots
 - ◆ Allocate and map hardware resources for guest domains
- The rest are called Domain U (guest domains)

Xen 3.0 Architecture





Implementation Technologies

- Hardware virtualization
 - ◆ Full virtualization
 - No need to modify the host OS
 - Relies on binary translations to trap and virtualize execution of instructions
 - Non critical instructions runs on hardware directly
 - Critical instructions replaced with traps into VMM and emulated by software
 - ◆ Host-based virtualization
 - Install virtualization layer on top of the host OS
 - Host OS responsible for managing hardware
 - Flexible, but performance is slow

Para-virtualization

- Modification to guest OS is needed by providing special API
- Virtualization layer can be inserted at different position
- Problems:
 - ◆ Compatibility and portability
 - ◆ Maintaining cost is high
 - Require deep OS kernel modifications
 - ◆ Performance varies due to workload variations
- Example: Linux KVM

Hardware-Assisted Virtualization

- Support virtualization by adding special mode and instructions
 - ◆ VMM and guest OS run in different mode
 - ◆ Sensitive instruction is trapped in VMM
- Applies to several components
 - ◆ CPU
 - ◆ Memory
 - ◆ I/O devices

Intel® Virtualization Technology Evolution

Vector 3:
IO Device Focus

Vector 2:
Chipset Focus

Vector 1:
Processor Focus

VMM
Software
Evolution

