Virtual Machine Overview
What's Virtual Machine?

- **Virtual machine (VM)**
  A software implementation of a machine (computer) that executes programs like a physical machine.

- **Types of virtual machine categories:**
  - System virtual machines - Hardware virtual machine
    - VMWare, Xen, KVM, VirtualBOX
  - Application virtual machine
    - Citrix XenApp, Microsoft App-V, VMWare ThinApp

(From Wikipedia)
Virtual Architecture

Traditional Architecture

Virtual Architecture

(Quote from VMware)
How Can Virtualization Help us?

- Server consolidation
- Server Migration
- Ease of management
- Multiple OS & Application Supports
- Test before we go
- Increase hardware utilization:
  - Rapid provisioning, dynamic fault tolerance..
  - Fault/performance Isolation
- Better utilization of multicore
Virtualization Tradeoffs

- **There is a performance trade-off**
  - Applications that used to own the whole processor must now share it
  - Hypervisor adds some runtime overhead too
  - Full virtualization without hardware support means software emulation

- **Increase in management complexity**
  - Old scenario: two software stacks + two hardware systems
  - New scenario: two software stacks + one hardware system + one host kernel

- **More abstraction, more software layers, more complexity...**
  - More bugs

- **Increases size of Trusted Computing Base**
- **Increases impact of (unpredicted) hardware failure**
Why Virtualization?

- **Mainly for servers**
  - Too many servers for too little work
- **Server consolidation**
  - Saves space and power consumption
Why Virtualization?

- **Mainly for servers**
  - Too many servers for too little work

- **Server consolidation**
  - Saves space and power consumption
  - Dynamic load balancing
Why Virtualization?

- **Mainly for servers**
  - Too many servers for too little work

- **Server consolidation**
  - Saves space and power consumption
  - Dynamic load balancing

- **Isolation, and thus security**
  - Honey pot for security
Why Virtualization?

- Mainly for servers
  - Too many servers for too little work

- Server consolidation
  - Saves space and power consumption
  - Dynamic load balancing

- Isolation, and thus security
  - Honey pot for attackers
  - VM’s are isolated

VMM

VM1

Hardware

Safe!
## Other Reasons

- VM(OS) monitoring
- High availability through replication of server
- Flexible hardware emulation
- Testing/quality assurance
- Software migration - live update
- Instant provisioning
- ....

![Main VM and Cloned VM Diagram](image)
Embedded Systems

- Mobile phones, DTV, ..

Motivations
- Run Linux or MS Windows while running RTOS securely

Problems
- Real time hypervisor is needed
- Microcontrollers do not have memory protection and user/kernel mode

Less need for full-virtualization
- Para-virtualization is applicable and sufficient
Embedded Virtualization

- Workload consolidation
- Flexible resource provisioning
- License barrier
- Legacy software support
  - Especially important with dozens or hundreds of embedded operating systems, commercial and home-brew

- Improve reliability
- Improve security
Soon, I will be upgraded
I am not fully tested
I am 10M lines of code
Don’t believe me
I don’t like this OS
I need EDF

If I die in the field ...
my device driver has bugs

Embedded System Dilemmas

hardware

operating system

main service
conver -gence
3rd party application
Types of VMM

- Depending on what sits right on HW

**Type-1: VMM on HW**
- Xen, VMware ESX server, Hyper-V
- Mostly for server, but not limited
- VMM by default
- OS-independent VMM

**Type-2: Host OS on HW**
- KVM, VMware Workstation, VirtualBox
- Mostly for client devices, but not limited
- VMM on demand
- OS-dependent VMM
VM Implementations

- **Full-virtualization**
  - No guest OS modification
  - Without HW supports, it is a little tricky
  - More CPU’s are adding HW supports

- **Para-virtualization**
  - Modify OS for supporting virtualization
  - What if you don’t have the OS source code?
  - Better performance
  - Better functionality
Hardware Supports

- Efficient virtualization requires hardware support
  - Goal: minimize performance overhead and modifications to guests

- Architecture support
  - High-end x86 (Intel VT, AMD SVM)
  - High-end PowerPC (PowerPC 970)
  - Embedded PowerPC virtualization architecture announced
  - ARM virtualization extension
Intel® Virtualization Technology Evolution

Vector 3: IO Focus
VT capabilities build upon one another to address full-system virtualization issues in sync with VMM software evolution

Vector 2: Chipset Focus
Assists for IO-device sharing (e.g., multi-context IO devices, etc.)

Vector 1: Processor Focus
VT-x Close basic processor "virtualization holes" in IA-32 and IPF processors
VT-i On-going evolution of processor virtualization assists, some micro-architectural, others architectural (e.g., extended page tables, EPT)

VMM Software Evolution
Software-only VMMs
- Binary translation
- Paravirtualization
Simpler and more secure VMM through use of hardware support
Better IO/CPU perf and functionality via hardware-mediated access to memory
Richer IO-device functionality and IO resource sharing

Before Intel VT:
No HW Support
VMM software evolution over time with hardware support
Full Virtualization – x86

• **Rings in x86 CPU**
  - A range of protection levels
  - Ring 0 has the highest level privilege
    - Operating system kernel normally runs.
    - Called system space, kernel mode or supervisor mode.
  - All other code such as applications operate in less privileged rings, typically Ring 3.
Rings in Virtualization

- **Traditional systems**
  - Operating system runs in privileged mode in Ring 0 and owns the hardware
  - Applications run in Ring 3 with less privileges

- **Virtualized systems**
  - VMM runs in privileged mode in Ring 0
  - Guest OS inside VMs are fooled into thinking they are running in Ring 0
    - Privileged instructions are trapped and emulated by the VMM
  - Newer CPUs (AMD-V/Intel-VT) use a new privilege level called Ring -1 for the VMM to reside allowing for better performance as the VMM no longer needs to fool the Guest OS that it is running in Ring 0.
Para-virtualization - CPU

- **Arch Xen/x86**
  - Critical instructions are replaced with Xen hypercalls
    - Avoid scanning/patching and fault trapping

- **Xen runs in ring 0**
  - Ring 1/2 for guest OS, 3 for user apps.
    - Prevent guest OS from directly executing privileged instructions
  - Jump to Xen in ring 0 by hypercalls

- **Guest OS may install ‘fast trap’ handler**
  - Direct ring user-space to guest OS system calls
**MMU**

- MMU virtualization: shadow vs. direct
- Using shadow pages (in VMware)
Direct-mode (Xen)

- Guest OSes allocate and manage own page-tables
  - Hypercall to change page-table base
- Xen must validate page table update before use
  - Update may be queued and batch processed
- Validation rules
  - Guest may only map pages it owns
  - Page-table pages may only be mapped read-only
MMU

- Direct-mode (Xen)
• MMU Performance

Imbench results on Linux (L), Xen (X), VMWare Workstation (V), and UML (U)
Nested/Extended Page Tables

- Hardware-supported MMU virtualization
Asynchronous I/O ring

- **Request Consumer**
  - Private pointer in Xen

- **Request Producer**
  - Shared pointer updated by guest OS

- **Response Producer**
  - Shared pointer updated by Xen

- **Response Consumer**
  - Private pointer in guest OS

- **Request queue** - Descriptors queued by the VM but not yet accepted by Xen
- **Outstanding descriptors** - Descriptor slots awaiting a response from Xen
- **Response queue** - Descriptors returned by Xen in response to serviced requests
- **Unused descriptors**
I/O

- Device channel

**Guest Requests DMA:**
1. Grant Reference for Page P2 placed on device channel
2. IDD removes GR
3. Sends pin request to Xen

**Diagram:**
- **Grant Table**
  - P1
  - P2

- **Device Channel**
  - GR

- **Active Grant Table**
  - P1
  - P2

- **Isolated Device Driver (IDD)**
  - GR

**Steps:**
4. Xen looks up GR in active grant table
5. GR validated against Guest (if necessary)
6. Pinning is acknowledged to IDD
7. IDD sends DMA request to device
I/O

- TCP results

![Graph showing TCP bandwidth results for Linux (L), Xen (X), VMWare Workstation (V), and UML (U).](image)
Xen Architecture

FE: front-end, BE: back-end
Xen – Strengths/Features

- **Rich server features**
  - Migration, CPU/memory hotplug, I/O virtualization
  - Secure isolation, QoS control
  - Multiple architecture supports

- **Viable Linux solution**
  - RedHat, Novel, XenSource, VirtualIron,...

- **Thin hypervisor**

- **Better client support is under development**
VMware Product Family

- VMware VI3
  - ESX
  - ESXi

- Enterprise Desktop
  - View (formerly VDI)
  - ThinApp
  - ACE

- Hosted products
  - Workstation
  - Server
  - Fusion
  - Player
Bare-metal products

- ESX & ESXi run on the bare metal (type –I)
  - Higher performance: 83-98%
  - Runs on a narrower range of hardware.

- Used for server consolidation for Data Centers

- High performance and scalability

- Many advanced features for resource management, high availability and security

- Supports more VMs per physical CPU than hosted products do.
Hosted Products (type-II)

- **Workstation/Server/Fusion/Player**
  - Requires a host operating system (Windows/Linux/Mac), installs like an application
  - Virtual machines can use all the hardware resources that the host can see
  - Maximum hardware compatibility as the operating system supplies all the hardware device drivers
Hosted Products

- Overhead of a full general-purpose operating system between the virtual machines and the physical hardware results in performance 70-90-% of native
VMware, Xen, KVM

KVM:
- Untrusted Guest
- KVM
- Linux kernel
- x86 hardware

XEN:
- Dom0 Linux kernel
- Untrusted Guest
- Xen
- x86 hardware

VMware:
- Workstation/ GSX Server
- Windows/Linux
- x86 hardware

ESX Server VMkernel
- Service Console
- x86 hardware
KVM Architecture
KVM – Strengths/Features

- Inherits Linux features and HW support
  - Power management
  - SMP scalability and schedulers
  - Developer base

- Limited Complexity due to VMX dependency
  - 10K lines in drivers/kvm
  - Small enough to be understood by me

- A guest is a Linux process
  - Native Linux admin tools work

- Requires a modified QEMU