Virtual Machines

- Overview
- History
- Benefits / Features
- Building Blocks
- Implementations
- Virtualization and Operating System Components
- Examples
Key Ideas

- User only knows and uses what the interface allows
- System may support different users, with different interfaces

Overview

- Fundamental idea
  
  "abstract hardware for different execution environments"

- Several components
  - Host – underlying hardware system
  - Virtual machine manager (VMM) or hypervisor – creates and runs virtual machines by providing interface that is identical to the host
    - Except for paravirtualization
  - Guest – process provided with virtual copy of the host
    - Usually, an operating system
  - Single physical machine can run multiple operating systems concurrently, each in its own virtual machine
System Model

VMM Implementation

- Vary greatly, with options including:
  - Type 0 hypervisors
    - all hardware via firmware
    - could provide dedicated CPUS, memory, I/O for each
    - IBM LPARs and Oracle LDOMs are examples
  - Type 1 hypervisors
    - runs on bare metal
    - OS-like software built to provide virtualization
    - VMware ESX, Joyent SmartOS, and Citrix XenServer
    - includes general-purpose operating systems that provide standard functions as well as VMM functions
      - Microsoft Windows Server / HyperV, RedHat Linux with KVM
  - Type 2 hypervisors
    - Applications that run on standard operating systems but provide VMM features to guest operating systems
    - VMware Workstation and Fusion, Parallels Desktop, Oracle VirtualBox, QEMU
Other variations include:

- **Paravirtualization** - Guest operating system is modified to work *in cooperation* with the VMM to optimize performance
- **Programming-environment virtualization** - VMMs do not virtualize real hardware but instead create an optimized virtual system
  - Used by Oracle Java and Microsoft.Net
- **Emulators** – Allow applications written for one hardware environment to run on a very different hardware environment, such as a different type of CPU
- **Application containment** - Not virtualization at all but rather provides virtualization-like features by segregating applications from the operating system, making them more secure, manageable
  - Including Oracle Solaris Zones, BSD Jails, and IBM AIX WPARs
- **Much variation due to breadth, depth and importance of virtualization in modern computing**

## Benefits

- **Host protected from VMs, VMs protected from each other**
  - viruses less likely to spread
  - sharing is provided via shared file system volume, network
- **Freeze, suspend, running VM**
  - Move, copy somewhere else and resume
  - Snapshot and restore back to that state
  - Clone by creating copy and running both original and copy
- **Great for OS research**
  - better system development efficiency
- **Run multiple, different OSes on a single machine**
  - Consolidation, app dev, …
Benefits cont…

- **Templating**
  - create an OS + application VM, provide it to customers, use it to create multiple instances of that combination

- **Live migration** –
  - move a running VM from one host to another!
  - no interruption of user access

- **All those features taken together ➞ cloud computing**
  - Using APIs, programs tell cloud infrastructure (servers, networking, storage) to create new guests, VMs, virtual desktops

---

Building Block - Trap and Emulate

- **Dual mode CPU means guest executes in user mode**
  - Kernel runs in kernel mode
  - Not safe to let guest kernel run in kernel mode
  - VM needs two modes:
    - virtual user mode
    - virtual kernel mode
    - both run in real user mode

- **How does switch from virtual user mode to virtual kernel mode occur?**
  - Attempting a privileged instruction in user mode causes an error -&gt; trap
  - VMM gains control, analyzes error, *executes operation as attempted by guest*
  - Returns control to guest in user mode
  - Known as *trap-and-emulate*

- **Guest user mode code in runs at same speed**
  - kernel mode privileged code runs slower
  - especially a problem with multiple guests

- **CPUs adding hardware support mode**
  - more modes improves virtualization performance
Building Block need for binary Translation

- Some CPUs blur line between privileged and non-privileged
  - Earlier Intel x86 CPUs are among them
  - Earliest Intel CPU designed for a calculator
- Backward compatibility means difficult to improve
- Consider Intel x86 `popf` instruction
  - Loads CPU flags register from contents of the stack
  - CPU in privileged mode ⇒ all flags replaced
  - CPU in user mode ⇒ only some flags replaced
    - No trap generated
- Other special instructions as well

Binary translation…
Binary translation to the rescue

- **Binary translation**
  - Basics are simple, but implementation complex
  - If guest VCPU is in user mode, guest runs instructions natively
  - If guest VCPU in kernel mode
    - VMM examines every instruction guest is about to execute by reading a few instructions ahead of program counter
    - Non-special-instructions run natively
    - Special instructions translated into new set of instructions that perform equivalent task (for example changing the flags in the VCPU)

- **Implemented by translation of code within VMM**
  - Code reads native instructions dynamically from guest, on demand, generates native binary code that executes in place of original code

- **Performance of this method would be poor without optimizations**
  - Products like VMware use caching:
    - Translate once, cache, and reuse when guest executes code again
    - Testing showed booting Windows XP as guest caused 950,000 translations, at 3 microseconds each, or 3 second (5%) slowdown over native (obviously an old data point)
Nested Page Tables \(NPTs\)

- How can VMM keep page-table state for guests?
  - Each guest maintains page tables to translate virtual to physical addresses
  - VMM maintains per-guest NPTs
  - When guest OS tries to change page tables:
    - VMM makes equivalent change to NPTs and its own page tables
  - Can cause many more TLB misses
    - potentially large performance impact

Hardware Assistance

- All virtualization needs some HW support
  - More support \(\Rightarrow\) more features, stability, performance
- Intel added VT-x instructions in 2005 and AMD the AMD-V instructions in 2006
  - CPUs with these instructions remove need for binary translation
  - define more CPU modes – “guest” and “host”
  - VMM can enable host mode, define characteristics of each guest VM, switch to guest mode and guest(s) on CPU(s)
  - In guest mode, guest OS thinks it is running natively, sees devices (as defined by VMM for that guest)
    - Access to virtualized device, priv instructions cause trap to VMM
    - CPU maintains VCPU, context switches it as needed
- HW continues to improve
  - support for Nested Page Tables
  - DMA
  - interrupts
Similar acronym: page modification logging (PML) is a hardware feature that tracks modified memory pages of VMs

Paravirtualization *guest OS knows the matrix is real*

- Not really virtualization
  - VMM provides an *abstraction* of hardware
  - *Guest is modified* to account for the virtualization layers
    - increased performance
    - less need for hardware support
- Xen techniques:
  - Clean and simple device abstractions, leads to:
    - Efficient I/O
    - Good communication between guest and VMM for I/O
    - Each device has circular buffer shared by guests and VMM via shared memory
Paravirtualization *not exact duplicate of hardware*

Xen I/O via shared circular buffer

Xen cont.

- More Xen techniques:
  - Memory management does not include nested page tables
    - Each guest has own read-only tables
    - Guest uses *hypercall* (call to hypervisor) when page-table changes are needed

- Paravirtualization allowed virtualization of older x86 CPUs (and others) without binary translation
  - Guest had to be modified to run on paravirtualized VMM

- On modern CPUs Xen doesn't require guest modification
  - not paravirtualization any more
Types of VMs application containment

- Some goals of virtualization are:
  - segregation of apps
  - performance and resource management
  - easy start, stop, move
  - management
- Can do those things without full-fledged virtualization
  - If applications compiled for the host operating system, don’t need full virtualization to meet these goals
- Docker containers create virtual layer between OS and apps
  - only one kernel running – host OS
  - container only has application layer of OS
  - OS and devices are virtualized
  - Container have their own:
    - applications
    - networking stack, addresses, and ports
    - runtime systems
    - user accounts, etc.

Much research ongoing both industry and academia

- basic tech used by cloud providers