Virtualization Concepts

Presented by: Mariano Diaz

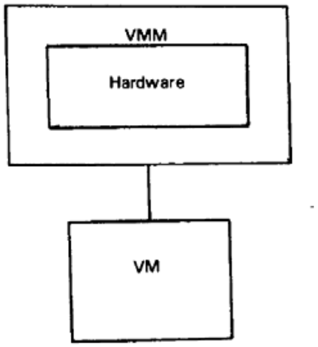
What is virtualization?

Virtualization Intro

- A layer that maps the interface of a system (virtual machine) or component (i.e., I/O device) onto the interface and resources of an underlying, possibly different, real system.
- Purposes:
  - Abstraction, Replication, Isolation, Cross compatibility/Encapsulation
- Doesn’t necessarily aim to simplify or hide details.
- Managed by a virtual machine monitor (VMM) [Host, Guest OS].

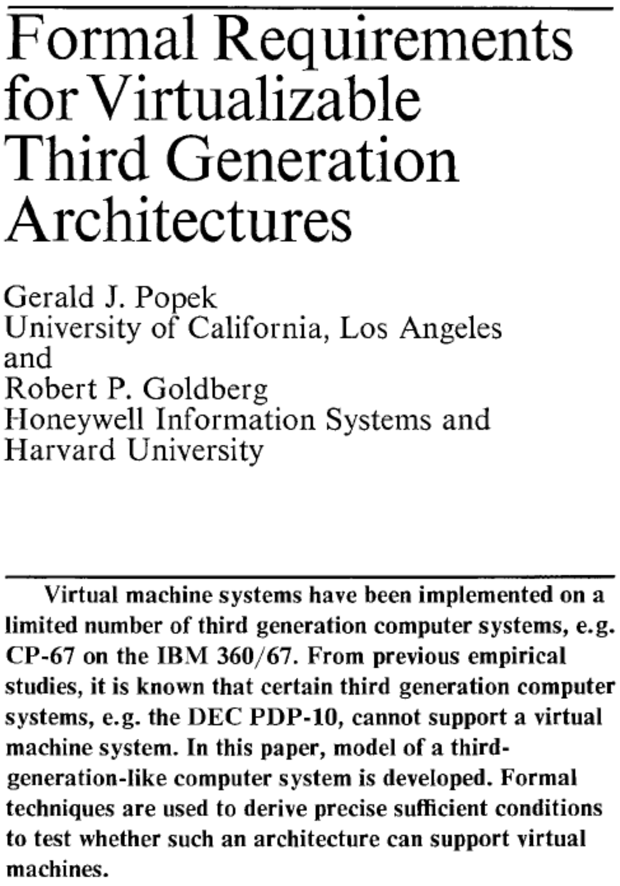
Origins – Principles (Kafura)

Fig. 1. The virtual machine monitor.



“an *efficient, isolated duplicate* of the real machine”

- Efficiency
  - □ Innocuous instructions should execute directly on the hardware
- Resource control
  - □ Executed programs may not affect the system resources
- Equivalence
  - □ The behavior of a program executing under the VMM should be the same as if the program were executed directly on the hardware (except possibly for timing and resource availability)



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

Virtual machine systems have been implemented on a limited number of third generation computer systems, e.g. CP-67 on the IBM 360/67. From previous empirical studies, it is known that certain third generation computer systems, e.g. the DEC PDP-10, cannot support a virtual machine system. In this paper, model of a third-generation-like computer system is developed. Formal techniques are used to derive precise sufficient conditions to test whether such an architecture can support virtual machines.

Communications of the ACM, vol 17, no 7, 1974, pp.412-421

Origins – Principles (Kafura)

Instruction types

- Privileged

  an instruction traps in unprivileged (user) mode but not in privileged (supervisor) mode.
- Sensitive
  - □Control sensitive –

    attempts to change the memory allocation or privilege mode
  - □Behavior sensitive
    - Location sensitive – execution behavior depends on location in memory
    - Mode sensitive – execution behavior depends on the privilege mode
- Innocuous – an instruction that is not sensitive

Theorem

For any conventional third generation computer, a virtual machine monitor may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.

Signficance

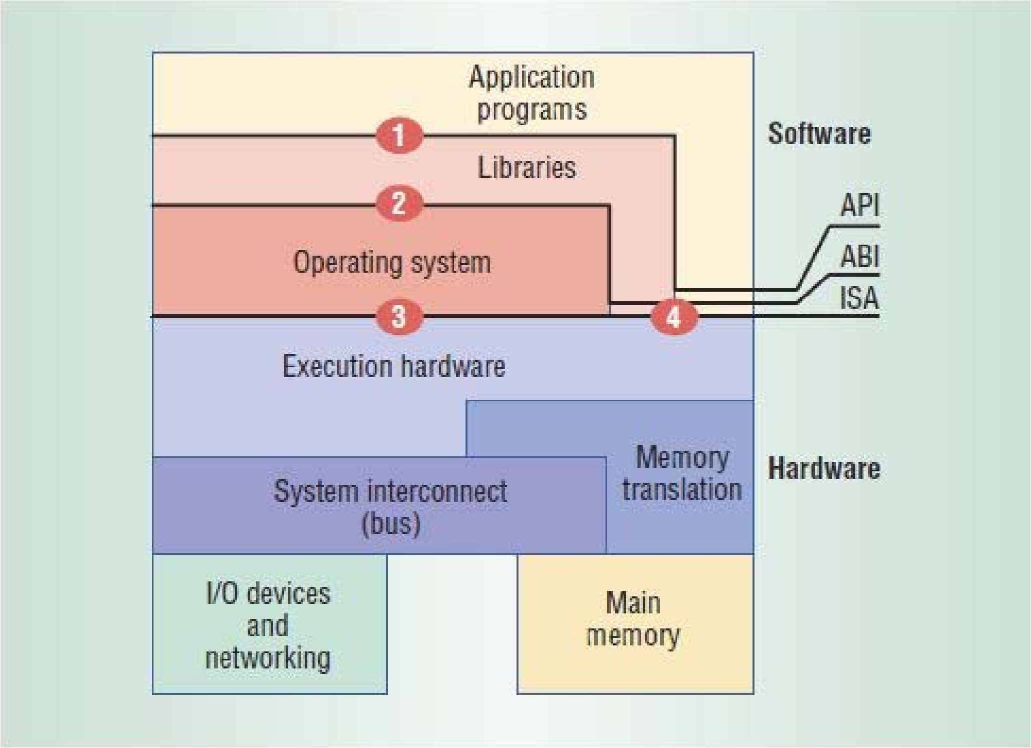
The IA-32/x86 architecture is not virtualizable.

Origins – History

- General computing consisted of mainframes, a scarce resource, during the 70s.
- Virtualization declined beginning in the 80s, shift in processing power.
- Revival in 2000s due to server sprawl and as a means to improve data-center management.
- New advances include:
  - □ Whole system migration
  - □ Complementary security mechanism
  - □ Centralization of services

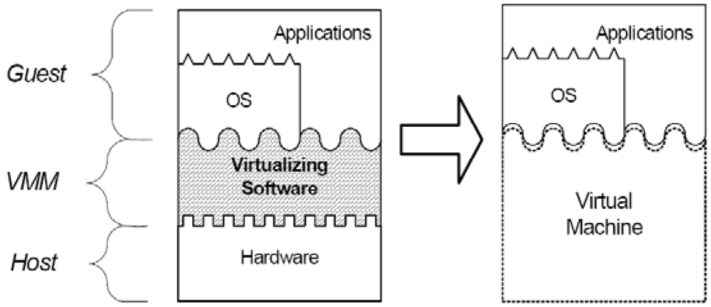
Computer Architecture

- Architecture: formal specification of an interface in the system.
- Interfaces: ISA, ABI, API



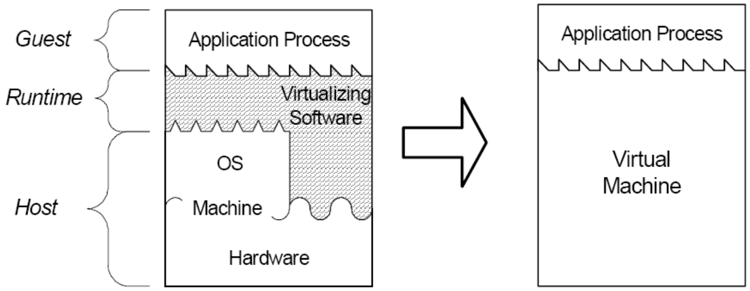
VMM Types (Kafura)

- System



- □ Provides ABI interface
- □ Efficient execution
- □ Can add OS-independent services (e.g., migration, intrustion detection)
- □ AKA Classic
- □ Persistent

- Process

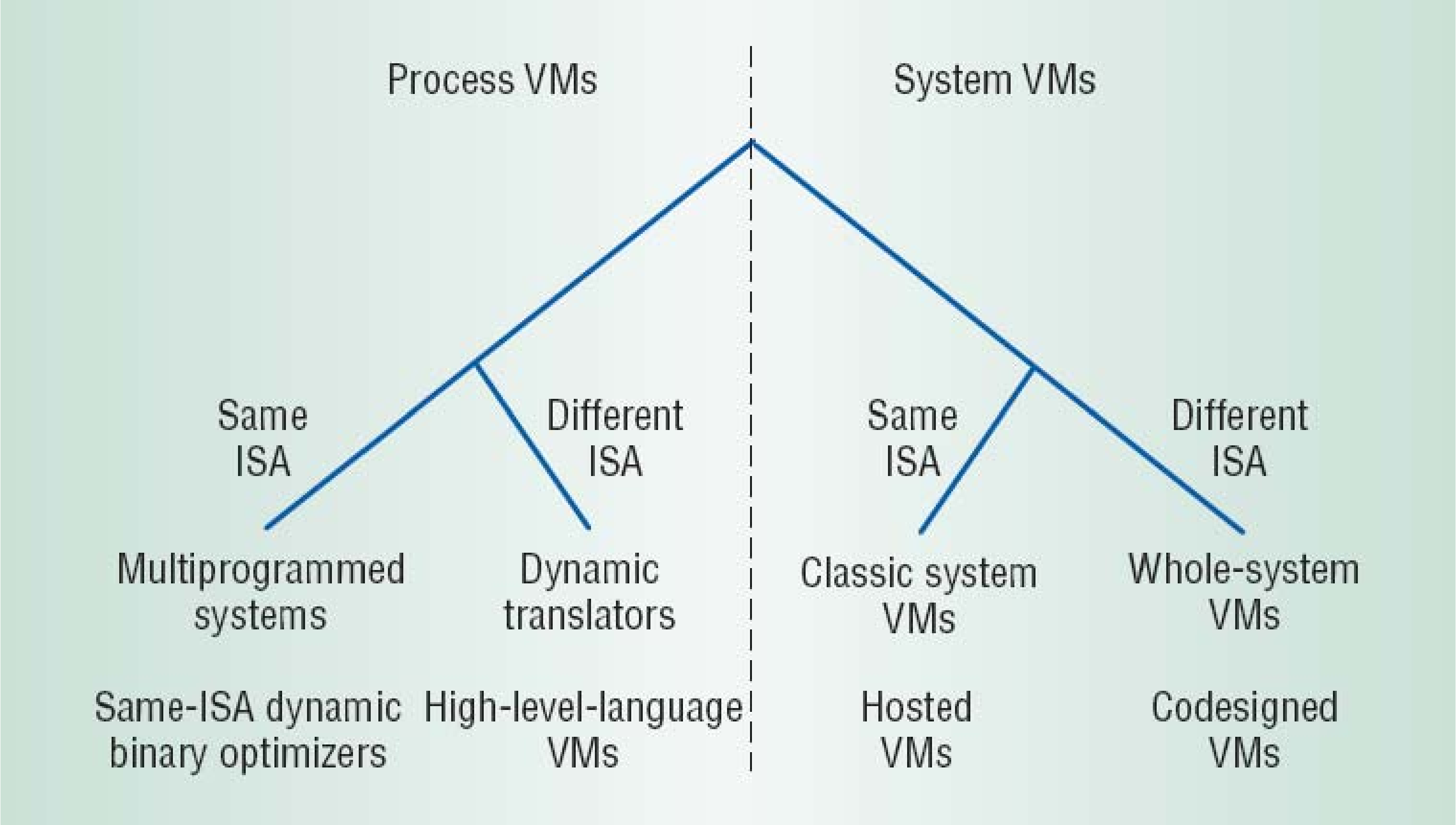


- □ Provides API interface
- □ Easier installation
- □ Leverage OS services (e.g., device drivers)
- □ Execution overhead (possibly mitigated by just-in-time compilation)

Techniques to implement

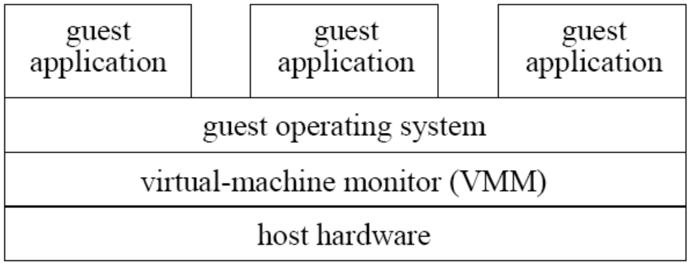
- Full virtualization
  - □ Direct execution (can be combined w/ binary translation)
  - □ Guest OS doesn’t need modification
  - □ Architecture provides trap semantics AKA “virtualizable”
  - □ Example: VMWare
- Paravirtualization
  - □ Addresses nonvirtualizable portions of instruction set (x86) (ie., POPF)
  - □ Guest OS kernel must be modified
  - □ Not cross-compatible
  - □ Example: Xen

Taxonomy of VMs



Systems VMM

- Two different kinds:
  - □ Type 1: runs directly on host hardware, high performanse (VMWare ESX Server, OS/370, Xen)



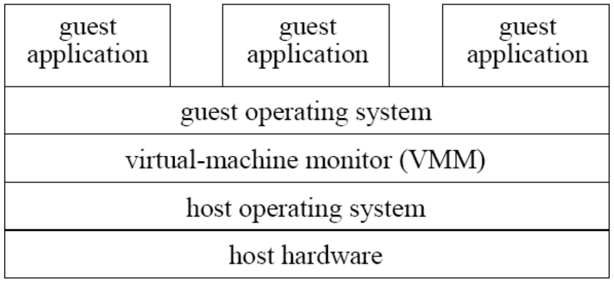
guest application | guest application | guest application

guest operating system

virtual-machine monitor (VMM)

host hardware

  - □ Type 2: runs on host OS, cross compatible, easy to install (User-mode linux)



guest application | guest application | guest application

guest operating system

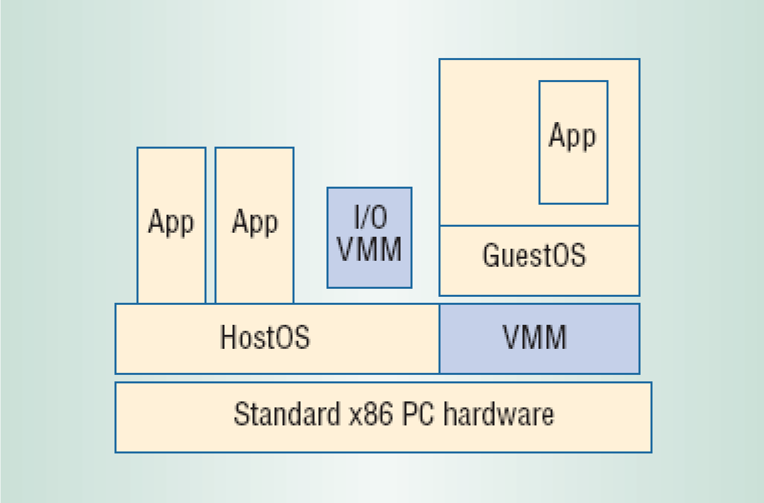
virtual-machine monitor (VMM)

host operating system

host hardware

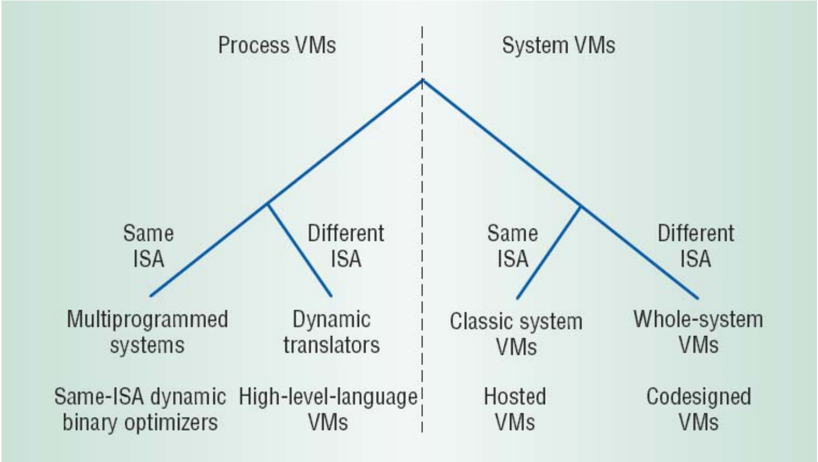
Hosted VMM

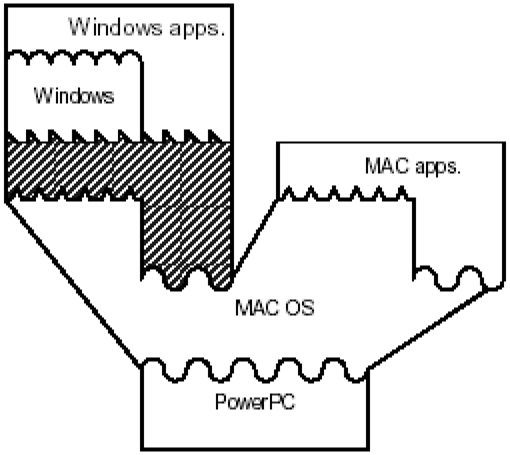
- Additional hosted VMM, hybrid of types 1 and 2
  - □ Improved performance over Type 2 (hosted)
  - □ Leverage device drivers for popular OS's
  - □ Requires same ISA as hardware
  - □ Example: VMWare Workstation



Whole-system VM

- Host and guest don’t have common ISA.
- Complete binary translation necessary.
- High overhead.
- Example: Virtual PC





Binary translation

- Trap and emulate.
- Binary translation:
  - □ Runs privileged instructions that are nonvirtualizable (x86).
  - □ Can be combined with direct execution (Example: VMWare Workstation).
  - □ Can optimize direct execution by lowering virtualization overhead.
  - □ Can use a trace cache.
- Memory and I/O virtualization discussed in subsequent presentations.

Questions?