12. Introduction to Virtual Machines

- Modern Applications
- Challenges of Virtual Machine Monitors
- Historical Perspective
- Classification

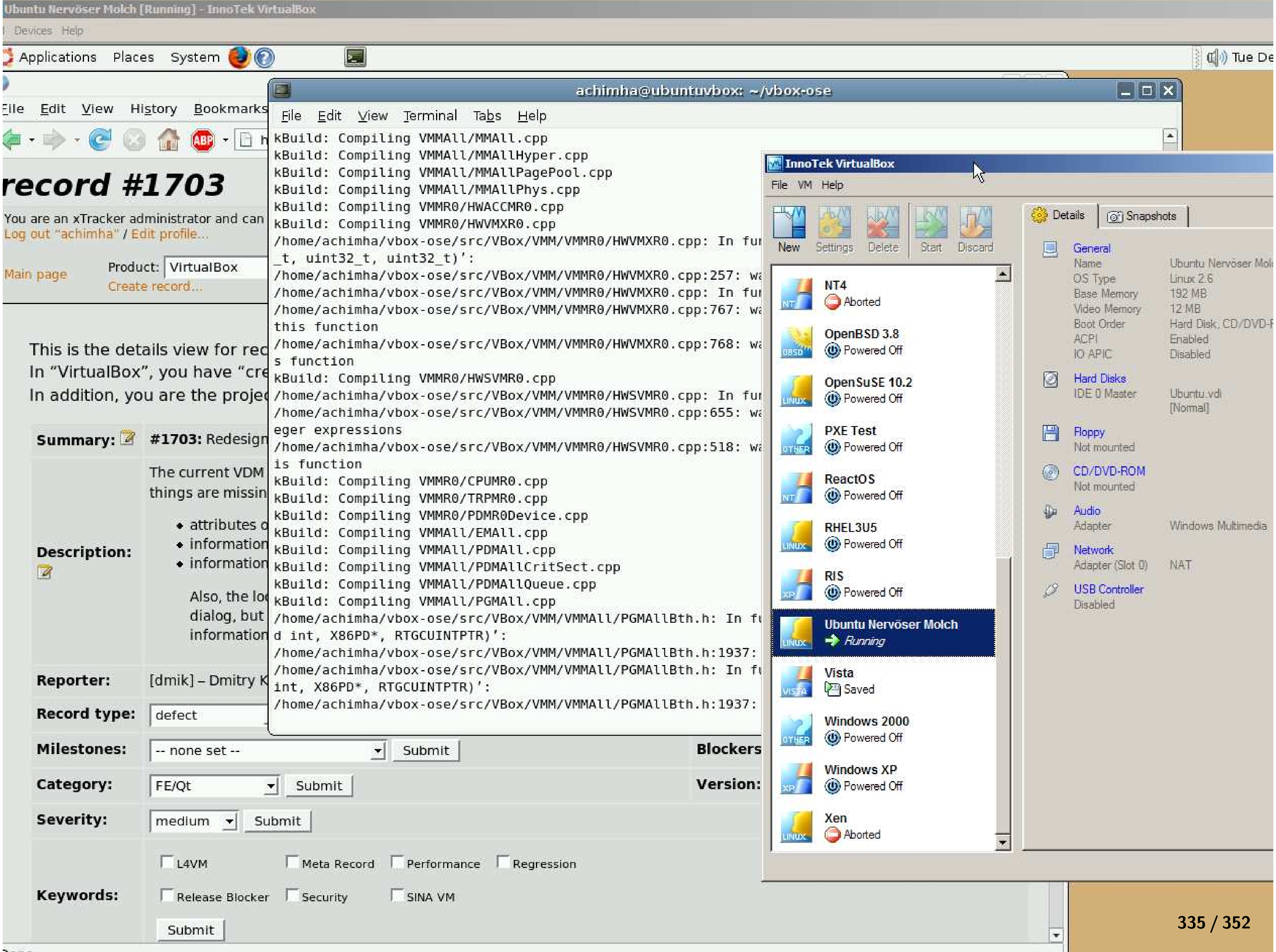
References

- First Attempt to Formalize and Classify: Popek and Goldberg, 1974
- The core element of a virtual machine: the *virtual machine monitor*
  - The virtual machine is analogous to an operating system, and the virtual machine monitor to its kernel

- James E. Smith and Ravi Nair: *Virtual Machines: Versatile Platforms for Systems and Processes*, 2005

12. Introduction to Virtual Machines

- Modern Applications
- Challenges of Virtual Machine Monitors
- Historical Perspective
- Classification


Ubuntu Nervöser Molch [Running] - InnoTek VirtualBox
Devices Help
Applications Places System
Tue De
achimha@ubuntuvbox: ~/vbox-ose
File Edit View Terminal Tabs Help
kBuild: Compiling VMMAll/MMAll.cpp
kBuild: Compiling VMMAll/MMAllHyper.cpp
kBuild: Compiling VMMAll/MMAllPagePool.cpp
kBuild: Compiling VMMAll/MMAllPhys.cpp
kBuild: Compiling VMMR0/HWACCMR0.cpp
kBuild: Compiling VMMR0/HWVMXR0.cpp
kBuild: Compiling VMMR0/HWSVMR0.cpp
kBuild: Compiling VMMR0/CPUMR0.cpp
kBuild: Compiling VMMR0/TRPMR0.cpp
kBuild: Compiling VMMR0/PDMR0Device.cpp
kBuild: Compiling VMMAll/EMAll.cpp
kBuild: Compiling VMMAll/PDMAll.cpp
kBuild: Compiling VMMAll/PDMAllCritSect.cpp
kBuild: Compiling VMMAll/PDMAllQueue.cpp
kBuild: Compiling VMMAll/PGMAll.cpp
record #1703
You are an xTracker administrator and can
Log out "achimha" / Edit profile...
Main page
Product: VirtualBox
Create record...
Summary: #1703: Redesign
Description:
Reporter: [dmik] – Dmitry K
Record type: defect
Milestones: -- none set --
Submit
Blockers
Category: FE/Qt
Version:
Severity: medium
Keywords:
L4VM
Meta Record
Performance
Regression
Release Blocker
Security
SINA VM
InnoTek VirtualBox
File VM Help
New
Settings
Delete
Start
Discard
Details
Snapshots
NT4
Aborted
OpenBSD 3.8
Powered Off
OpenSuSE 10.2
PXE Test
ReactOS
RHEL3U5
RIS
Ubuntu Nervöser Molch
Running
Vista
Saved
Windows 2000
Windows XP
Xen
General
Name
OS Type
Linux 2.6
Base Memory
192 MB
Video Memory
12 MB
Boot Order
ACPI
Enabled
IO APIC
Disabled
Hard Disks
IDE 0 Master
Ubuntu.vdi
[Normal]
Floppy
Not mounted
CD/DVD-ROM
Not mounted
Audio
Adapter
Windows Multimedia
Network
Adapter (Slot 0)
NAT
USB Controller
Disabled

2007: Virtualization Everywhere

Machine-Level Virtualization

- *VMware*, *Parallels Desktop*, *Virtual Box* (Linux, MacOS X, Windows)
  - Driven by the convergence of server, desktop and embedded computing
  - Break some of the artificial constraints imposed by proprietary software
  - VMs replace processes in a secure environments: all communications use high-level distributed system interfaces on top of INET sockets
  - Build feature-rich kernels over small, device-specific kernels

2007: Virtualization Everywhere

Processor-Level Virtualization

- *VMware*, *Virtual Box*, *QEMU*, *Rosetta*
  - Translate machine instructions of the guest processor to run on the host system
  - Fast translation schemes: binary translation, code caches, adaptive translation, binary-level optimization, link-time optimization

2007: Virtualization Everywhere

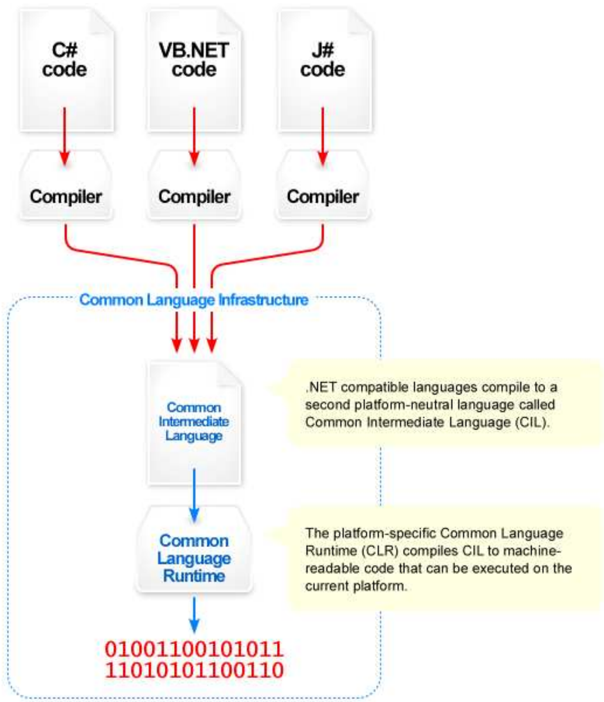
System-Level Virtualization

- Para-Virtualization with *Xen* (Linux only)
  - Ease OS development
  - Customize access control of embedded VMs in a secure environment
  - Virtual hosting, remote administration consoles

2007: Virtualization Everywhere

Language-Level Virtualization

- Abstract machine integrated into the semantics of a programming language
  - *JVM* (Sun Java)
  - *ECMA CLI* (MS .NET)
- Features
  - Portability, code size improvements, original dynamic optimizations
  - High-productivity features (garbage collection, distributed components)
  - Sandbox (robustness, security management, fault-tolerance)



12. Introduction to Virtual Machines

- Modern Applications
- Challenges of Virtual Machine Monitors
- Historical Perspective
- Classification

Virtual Machine Monitor

Classical Challenges

- Influence of the *guest-host* relationship
  - Homogeneous: intercept any *guest*-specific action to redirect it to the *host*'s interface
  - Heterogeneous: instruction-set *emulation* and *binary translation*
- Excessive memory usage
- Project drivers of the guest operating system to host devices

Virtual Machine Monitor

Virtualization of Priviledged Code

- Common issues
  - Memory-mapped I/O
  - Exceptions and interrupts
  - Esoteric machine language instructions
- Good design: IBM System/360 instruction set and I/O architecture
- Bad design: Intel x86 and IBM PC I/O
  - Port I/O, accesses to system buses, memory-mapped I/O, control registers and exceptions/interrupts *could not* be reconfigured to (selectively) trigger *host* exceptions
  - Require a conservative emulation layer to execute priviledged code
  - Fixed in the Core Duo 2 processor: *native virtualization*

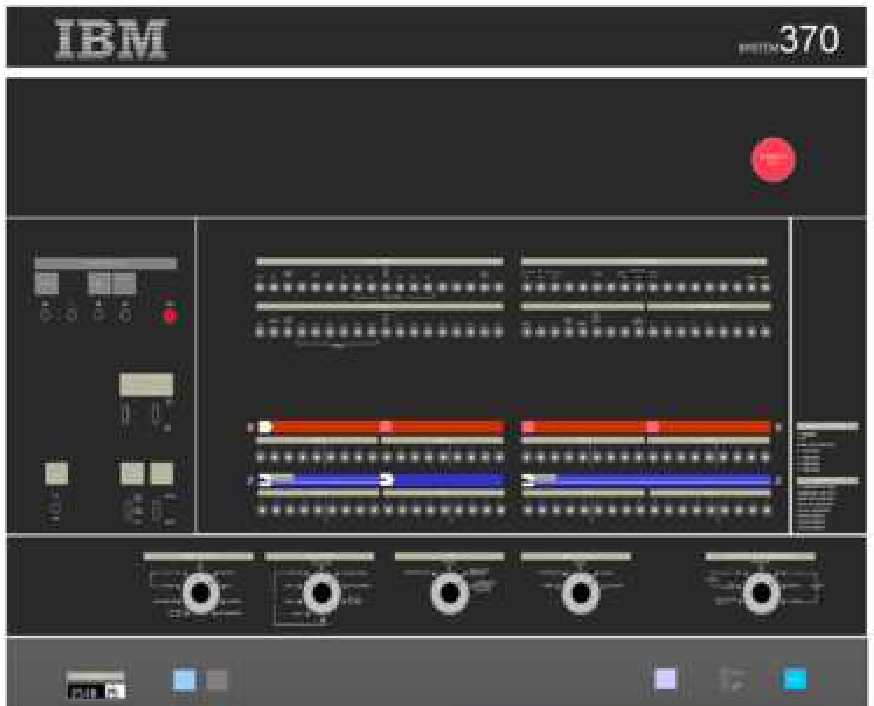
12. Introduction to Virtual Machines

- Modern Applications
- Challenges of Virtual Machine Monitors
- Historical Perspective
- Classification

Historical Perspective

IBM VM System/370: 1967

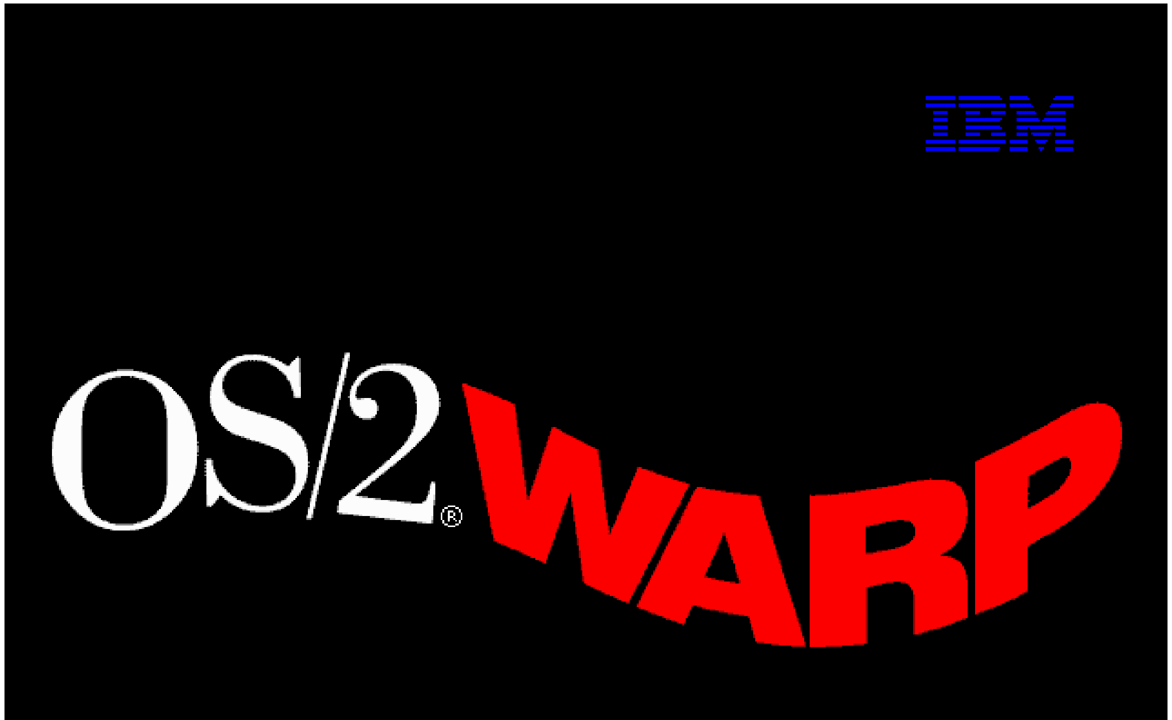
- First virtual machine
  - Offer long-term portability of System/360 applications over a wide range of machines and peripherals, although the processor's machine instructions where quite different
- Implementation: binary translation and emulation of foreign code
  - Unpriviledged code
  - Compatible guest and host system API



Historical Perspective

IBM OS/2: 1987

- Provide modern OS features to legacy MSDOS applications
  - Multitasking (non-preemptive at first)
  - Virtual memory (protection and management of more than 640kB of RAM)
- Implementation: embed a 1MB MSDOS memory image in the virtual memory frame of a single process (called task)
- Initially supported by Microsoft... then came Windows 3.1 and then NT



Historical Perspective

Apple (and Transitive) Rosetta: 1994 and 2006

- Execute Motorola 680x0 code on PowerPC
  - Emulation and some binary translation
  - User code only
- Execute PowerPC code on x86
  - Priviledged code, support for a full-fledged UNIX OS
- Compatible guest and host system API, low-overhead implementation
- Original project: *DAISY* in 1992 (IBM Research, PowerPC → VLIW instruction set)
- Enhancements
  - Full system virtualization (heterogeneous): *VMware*
  - Performance (emphasis on dynamic optimization), multi-OS (Linux, HPUX, Windows): *IA36EL* (Intel)

Historical Perspective

Transmeta Crusoe: 2000

- Translate x86 code to a VLIW ISA
  - Binary translation code embedded on the chip itself: *Code Morphing*
  - Pros: low overhead (avoids instruction cache pollution, dedicated hardware), energy and bandwidth savings (on-chip memory accesses)
  - Cons: peep-hole optimizations only, hard to maintain precise exception semantics
- Discrete advantage: fix hardware bugs, shorter testing (most expensive)
- Untold advantage: hide original processor specifications, including energy management and proprietary VLIW instruction set

12. Introduction to Virtual Machines

- Modern Applications
- Challenges of Virtual Machine Monitors
- Historical Perspective
- Classification

Taxonomy of Virtual Machine Monitors

Software Implementations

- *Hypervisor*: most general case, when native virtualization is not possible or to implement a *sandbox*
- Emulation and full-system virtualization: e.g., *QEMU*, *VMware*
- Computer architecture simulation (cycle-accurate or transaction-level): e.g., *Simics* (Chalmers), *UNISIM* (INRIA, Princeton, UPC)
- Binary translation, code cache and dynamic optimization: e.g., *DAISY* (IBM), *Dynamo* and *DELI* (HPLabs) *Rosetta* (Apple), *IA32EL* (Intel)
- Para-virtualization: resource sharing, security and sandboxing, e.g., *Xen* or *User Mode Linux*

Taxonomy of Virtual Machine Monitors

Hardware Implementations

- Homogeneous instruction sets
  - Native virtualization with a *lightweight hypervisor*: “trap” on specific instructions or address ranges, e.g., *Parallels Desktop* or *kvm* (with QEMU) on Intel Core 2 and later x86 processors
  - Reduce exception overhead and cost of switching to kernel mode
- Heterogeneous instruction sets
  - Support to accelerate instruction decoding (PowerPC)
  - Additional instruction pipeline stages (x86 $\longrightarrow$ internal RISC microcode)
  - Hybrid binary translation to reduce overhead: *Code Morphing* (Transmeta Crusoe),

Virtualization Stack

Intricate Example

Java application
↓ Just-in-time compilation and Java virtual machine
MacOS X PowerPC
↓ Binary Translation (Rosetta)
MacOS X x86
↓ Full system virtualization (Parallels Desktop)
Linux x86
↓ Para-virtualization (Xen)
Linux x86
↓ Binary translation (Transmeta Code Morphing)
Transmeta Crusoe (VLIW)