A Comparison of Software and Hardware Techniques for x86 Virtualization

Presented by Mike Marty

The Renaissance of Virtualization

- 1970s: virtual machines first used
- 1990s:
  - x86 becomes prominent server platform
  - No vertical integration in x86
  - Lack of enterprise features in commodity OSs
- 1999: VMWare first product to virtualize x86
- 2006: AMD and Intel offer hardware support

Outline

- Classic Virtualization
- Software Virtualization
- Intel/AMD Hardware Virtualization
- Comparison and Results
- Discussion

Classic Virtualization

- Popek and Goldberg’s Criteria:
  1. Fidelity – run any software
  2. Performance – run it fairly fast
  3. Safety – VMM manages all hardware
- Trap-and-Emulate only real solution until recently
1. De-Privilege OS

Traps are expensive (~3000 cycles)

Many traps unavoidable
- E.g., page faults

Important enhancements
- "Paravirtualization" to reduce traps (e.g., Xen)
- Hardware VM modes (e.g., IBM s370)
**Can x86 Trap and Emulate?**

- **No**
  - Even with 4 execution modes!
  - Key problem: dual-purpose instructions don’t trap

- **Classic Example:** `popf` instruction
  - Same instruction behaves differently depending on execution mode
  - User Mode: changes ALU flags
  - Kernel Mode: changes ALU and system flags
  - Does not generate a trap in user mode

---

**Outline**

- Classic Virtualization
- Software Virtualization
- Intel/AMD Hardware Virtualization
- Comparison and Results
- Discussion

---

**Software Virtualization with VMWare**

- **Binary translation!** (mostly safe, user-mode)

![ VMware Binary Translation Diagram ]

- X86 → VMWare → X86

---

**VMWare’s Binary Translation**

- **On-the-fly**
- Only need to translate OS code
  - Makes SPEC run fast by default
- Most instruction sequences don’t change
- Instructions that **do change:**
  - Indirect control flow: call/ret, jmp
  - PC-relative addressing
  - Privileged instructions
- Adaptive Translation
  - “Innocent until proven guilty”
Performance Advantages of BT

- Translation sequences can be faster than native:
  - cli vs. vpu.flags.IF := 0

- Avoid privilege instruction traps
  - Example: rdtsc
    - Trap-and-emulate: 2030 cycles
    - Callout-and-emulate: 1254 cycles
    - BT emulation: 216 cycles (but TSC value is stale)

Outline

- Classic Virtualization
- Software Virtualization
- Intel/AMD Hardware Virtualization
- Comparison and Results
- Discussion

AMD SVM and Intel VT

- Extensions to x86-32 and x86-64
  - Allows classic trap-and-emulate!
  - Hardware VM modes to reduce traps
  - Details:
    - VMCB – virtual machine control block
    - VMX mode for running guest OSs
    - vmrun instruction to enter VMX mode
    - Many instructions and events cause VMX exits
    - Control fields in VMCB can change VMX exit behavior

Hardware VM Example: syscall

1. VMM fills in VMCB exception table for Guest OS
   - Sets bit in VMCB not exit on syscall exception
2. VMM executes vmrun
3. Application invokes syscall
4. CPU → CPL #0, does not trap, vectors to VMCB exception table
Software BT vs. Hardware VM

- Binary Translation VMM:
  - Converts traps to callouts
  - Callouts faster than trapping
  - Faster emulation routine
  - VMM does not need to reconstruct state
  - Avoids callouts entirely

- Hardware VMM:
  - Preserves code density
  - No precise exception overhead
  - Faster system calls

Bottomline: little difference for SPEC
Nanobenchmarks

- syscall
  - Native/Hardware VMM: same
  - Software VMM: +2000 cycles

- in
  - Native: 3209 cycles
  - Hardware VMM: 15826 cycles
  - Software VMM: 15x faster?

- call/ret
  - Native/Hardware VMM: 11 cycles
  - Software VMM: 51 cycles

Opportunities

- Faster Microarchitecture implementations
  - Intel Core Duo already much faster than P4

- Hardware VMM algorithms

- Software/Hardware Hybrid VMM

- Hardware MMU
  - Virtualize DMA

Catalysts for Discussion

- Is BT really faster for things that matter?
  - Process-based Apache on Linux?
  - Who configures a system to constantly page?

- VMWare is done, why bother with Hardware VM support?
  - Simplicity of VMM w/ Hardware support
  - New applications

- Will next-gen hardware make binary translation unnecessary?