10/16 CS240 - x86 VMM

Announcements

For next class (Tuesday 10/21):

- In-class **Midterm Exam**
  - Open notes (i.e. printed material). No electronics.
  - Material through today's lecture (x86 vmm hw/sw)
  - Similar to sample exams on course website

Looking ahead:

- Lab 1 is due end of day November 2nd.

Paper background

- ASPLOS 2006 - International Conference on Architectural Support for Programming Languages and Operating Systems
  - Ole Agesen - Stanford PhD - Self Programming Language
  - Keith Adams - Top Brown CS undergraduate, later did Facebook's HipHop VM
- VMware software x86 virtualization launch in 1999
  - VMM had an unusual memory size constraint with software engineering implications
- x86 architecture vendor added virtualization support in 2005 (Intel) / 2006 (AMD)
  - Memory virtualization support added in 2008 (AMD) / (2009) Intel

What does it mean to be virtualizable?

- POPEK, G. J., AND GOLDBERG, R. P. Formal requirements for virtualizable third generation architectures. Commun. ACM 17, 7 (**1974**), 412–421.
    - By 1985 (x86 protected mode introduced), virtualizable wasn't considered a requirement
- Run environment in less privileged mode
- Trap and emulate
    -

x86 was considered not virtualizable

- Visibility of privileged state
- Lack of traps for some privileged instructions
  - Instruction semantics differ between privilege levels
- Some x86 modes virtualizable (can use direct execution)
  - Most OSes run user-mode code in ring 3
    - Discuss SGDT
  - v8086 mode
- Commonly used non-virtualizable mode:
  - Kernel mode - ring 0 execution
  - Several of the legacy modes (real mode, etc.)

Example of problematic x86 (kernel mode) ring 0 code

```
pushf               ; save current EFLAGS (including interrupt flag IF)
cli                 ; clear interrupt flag – disable interrupts

    ; --- critical section begins ---
    mov  ax, [shared_counter]
    inc  ax
    mov  [shared_counter], ax
    ; --- critical section ends ---

popf                ; restore saved flags (re-enabling interrupts if enabled)
ret
```

Basic virtualization approach

- Create in-memory shadow copy of privilege hardware state
    - Emulation used this in-memory state copy
    - VMM can use the real hardware state
- List of x86 privileged hardware state
    - Privileged registers: CR0 mode bits, CR2 fault address, CR3 PT address, CR4 features
    - Segment descriptor tables (LDT, GDT, IDT)
    - Parts of EFLAGS and %cs
    - Page tables

VMware's software VMM - Dynamic Binary Translation

- Dynamic binary translation history
  - Fast interpreters: Shade -> Embra/SIMOS
  - Just-in-time compilers: Xerox PARC Smalltalk runtime, Self -> Java HotSpot
  - In parallel as VMware: Transmeta CPU
- Let's do an inner loop of an x86 dynamic binary translator
  - IDENT, SIMULATOR translations
  - Chaining
    - Frequently the jmp is omitted on fall through. Why?
  - Adaptive translations?
- Challenges of x86 code:
  - Jump into the middle of an instruction sequence
  - Dynamically modified instructions

Protecting the translator

- Steal %gs segment register
- Truncate segments
- Non-IDENT translations
  - PC-relative addressing (call, jmp, jcc)
  - Direct control flow
  - Indirect control flow (ret, jmp *via register or memory*)
  - Privileged instructions

Memory tracing

- `memory_trace(address, handler_to_call)`
  - Types: page tables, GDT, LDT, IDT, code pages
- How is it implemented?
- Overheads?
  - How does the binary translator eliminate trap overheads?

Hardware Virtualization: Intel VT and AMD SVM

- Virtual Machine Control Block
    - Shadow privilege state
    - Access in special VMM mode with special instructions
- `vmrun` Virtual Machine Control Block until it needs to exit back to VMM
    - Can directly execute the guest OS and applications.
    - Need VMM to do device emulation and **memory virtualization**

Memory virtualization - Shadow page tables

- Recall from ESX lecture
- Consider the tradeoff:
  - Trace costs
  - Hidden page faults
  - Context switch cost

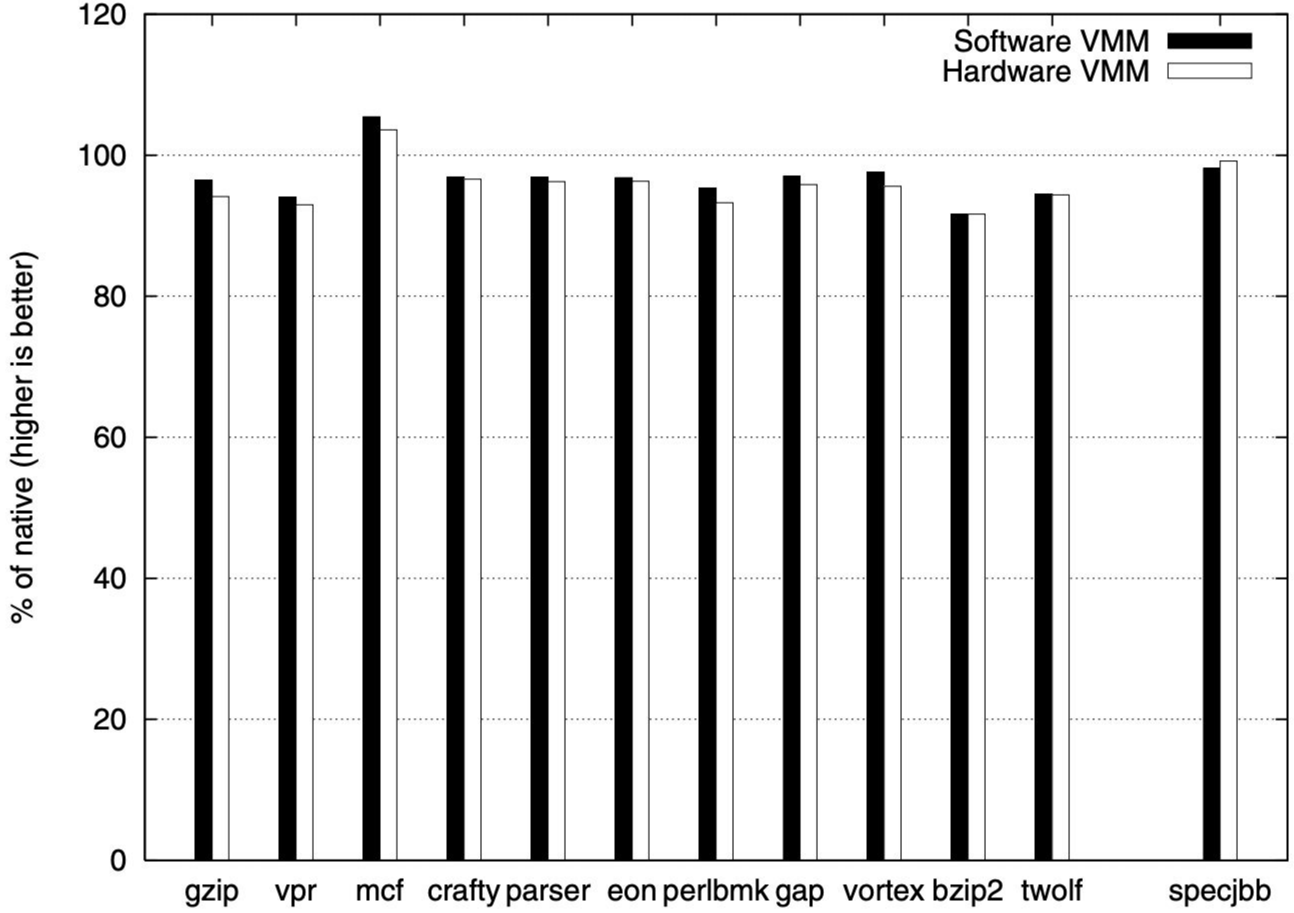

Figure 2. SPECint 2000 and SPECjbb 2005.

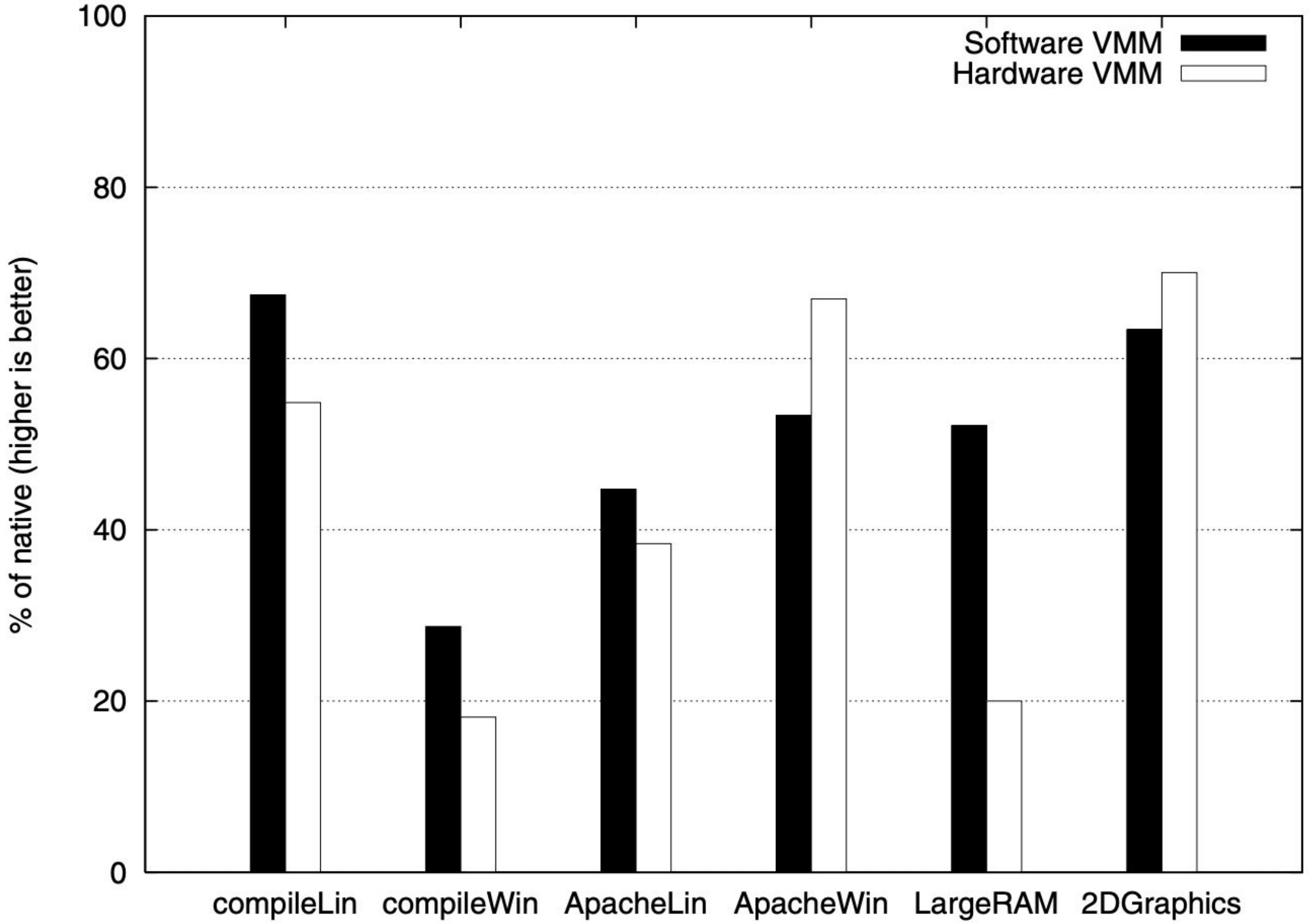

Figure 3. Macrobenchmarks.

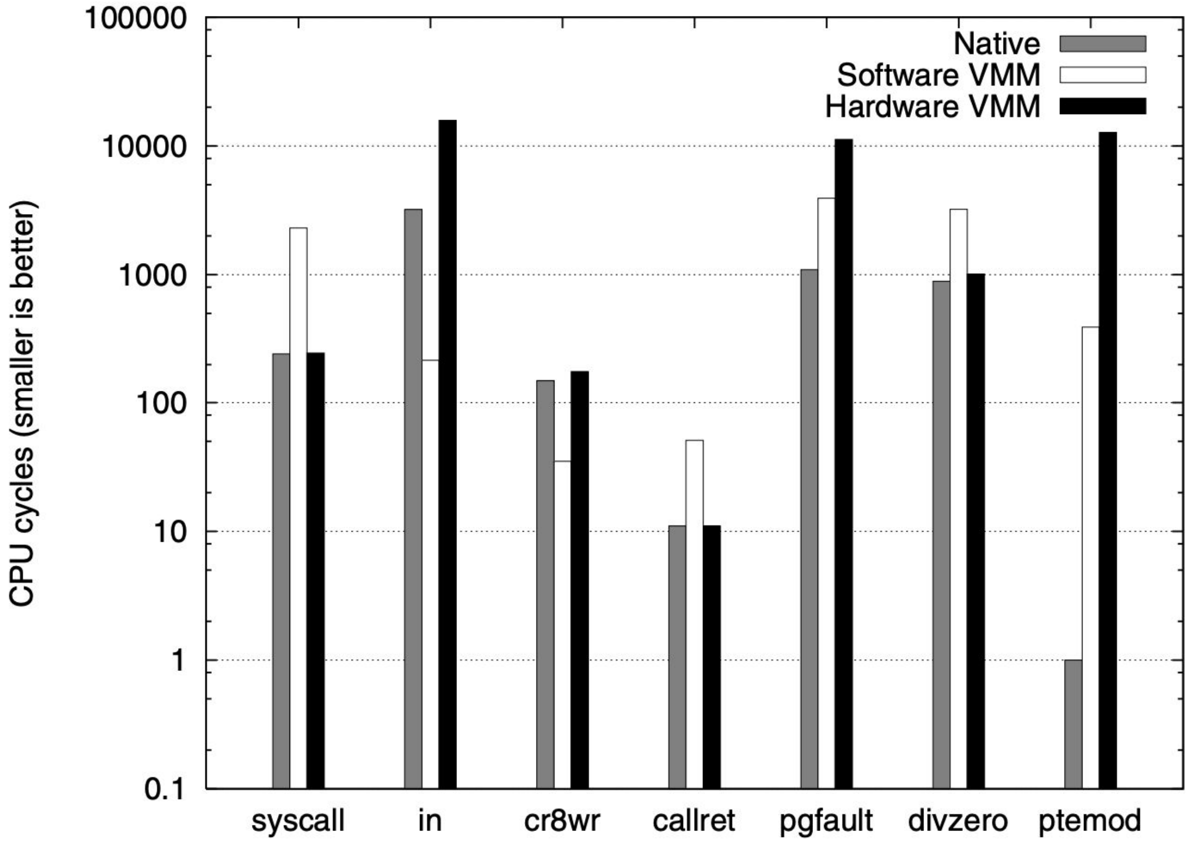

Figure 4. Virtualization nanobenchmarks.

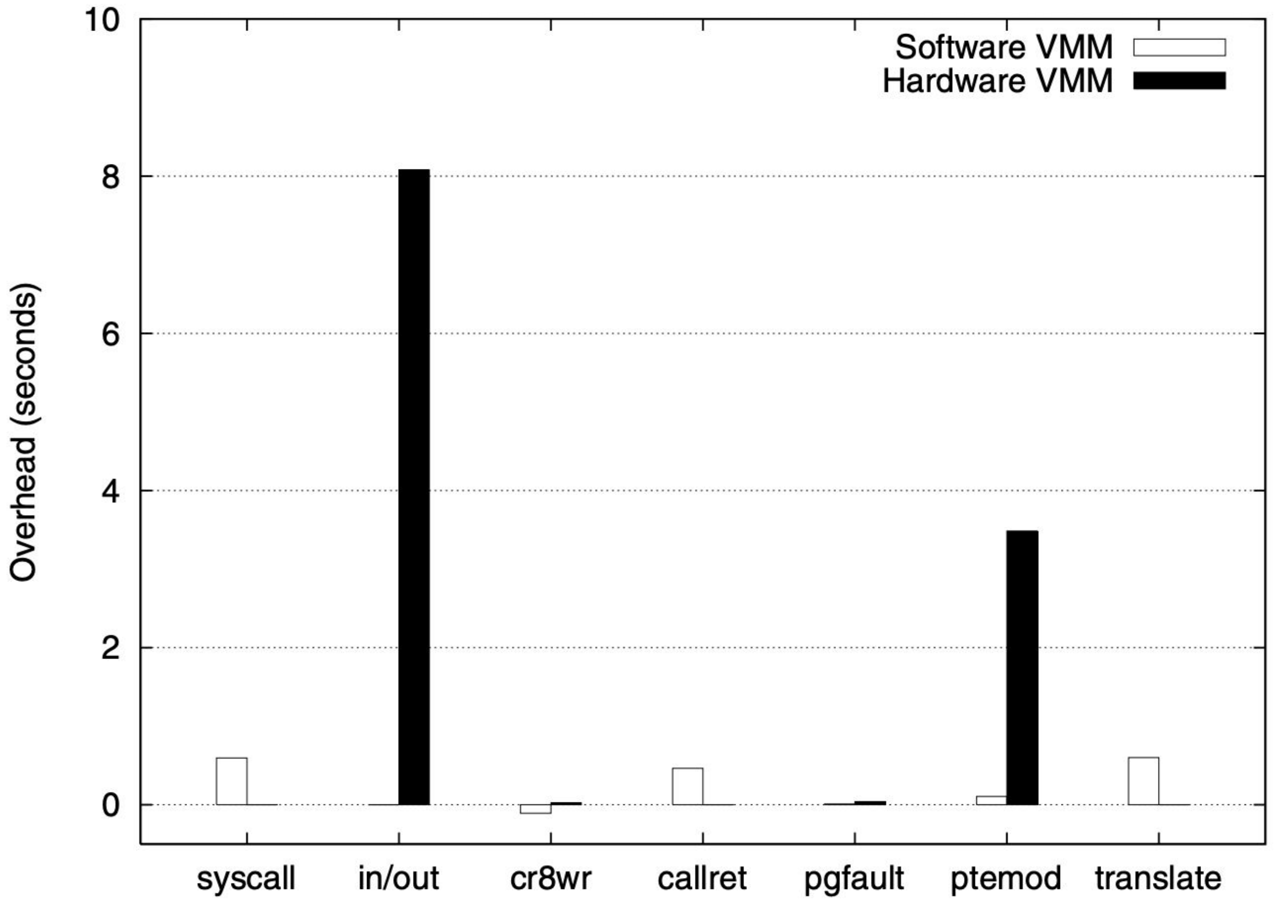

Figure 5. Sources of virtualization overhead in an XP boot/halt.

x86 memory virtualization

- Memory virtualization added to both Intel and AMD
  - Intel Extended Page Tables
  - AMD Nested Page Table

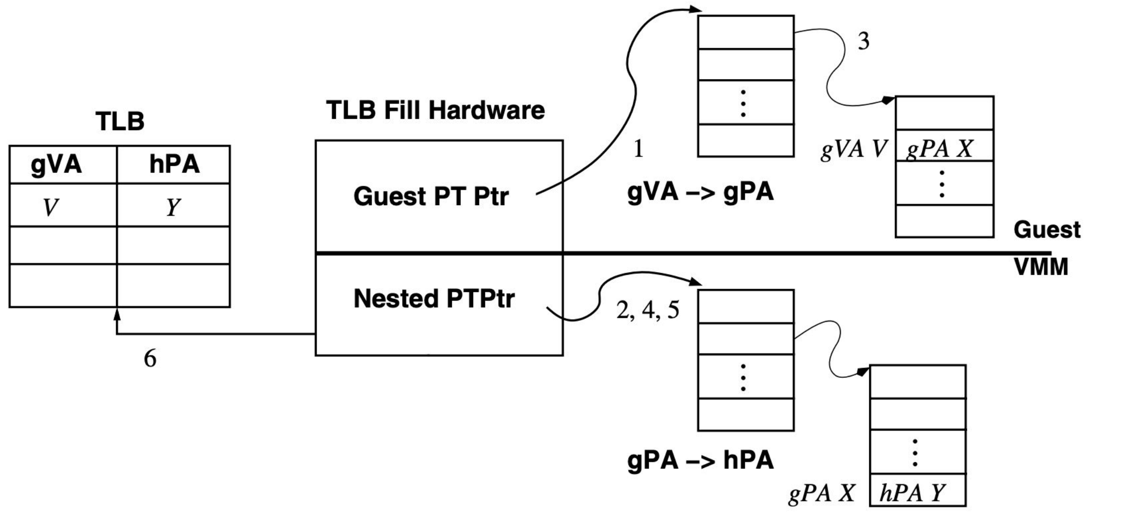

Figure 6. Nested paging hardware.