Virtual machines and operating systems

Krzysztof Lichota
lichota@mimuw.edu.pl

Agenda

- Virtual machines and operating systems interactions
- x86 processor architecture problems
- Pacifica/VT
- XEN
- Security

Full virtualization based on VirtualBox code

Virtual machines and operating systems

- Guest system thinks it is working on real machine and has all the resources for himself
- Host system has to provide simulated environment for guest
- Host systems has to isolate virtual machines and share the resources among them

Virtualization basics

- Guest system is put in unprivileged (or less privileged) mode
- Privileged instructions cause exception and pass control to virtual machine manager (hypervisor)
- Hypervisor emulates proper behaviour by modifying machine state and passes control back to machine

Virtualization of hardware devices

- Harware device drivers work by programming device registers (using I/O ports and MMIO) and acting upon interrupts
- Virtual machine emulates real hardware device
- IN/OUT instructions are intercepted as they are privileged
- MMIO can be intercepted by protecting pages

Virtualization of hardware devices (2)

- Virtual machine simulated device driver performs operation as if device did it (reads some data from virtual disk, sends some data to virtual network, etc.)
- When operation is complete, virtual machine simulates interrupt on virtual machine
- Examples: simulating IRQ, attaching to VM simulated port, VM interface for virtual drivers

Virtualization of memory management

- Hypervisor cannot allow guest direct access to physical memory
- All modifications of memory management structures must be intercepted and somehow simulated (or ignored) by modifying real page tables
- Writes to page table register (CR3 on x86) can be easily intercepted as it is privileged operation
- But guest OS can also modify page tables

Page tables modification

- Hypervisor creates “shadow page tables”
- Page tables modified by guest OS are dummy, writes to CR3 are intercepted
- Hypervisor intercepts modifications to guest OS page tables (by write-protecting them) and transforms them into proper operations on real page tables
- Note: if guest was granted direct access to hardware device this might cause problems, as it does not know physical location and it might not be contiguous

Other OS actions

- Interrupts - obviously guest cannot modify interrupt handling – interrupt handler tables are shadowed, enabling/disabling interrupts is intercepted
- Writes to control registers (switching processor mode, etc.) are also intercepted
- TLB flushes, ...

Guest to host services

- Some services are useful and require cooperation between host and guest (clipboard sharing, drag-and-drop, etc.)
- Guest must somehow pass information to VMM and get results back
- Implemented by causing special exceptions which cause VMM to execute action and return result by modifying machine state
- Exception invoked using interrupt (Vmware) or illegal instruction (Virtual PC)

x86 processor architecture problems

- Dual-purpose instructions (example: POPF instruction can modify Interrupt Flag in kernel mode, but does not generate trap in user mode – it is ignored)
- Store/load instruction pairs where load instruction is privileged, but store does not generate trap (e.g. SIDT/LIDT)
- Instructions which compare privilege level or allow storing it from segment registers (LAR, LSL, PUSH CS, ...)
- Real mode (8086 mode) – must emulate direct access to hardware and real mode

Solving x86 architecture problems

- Use Pacifica/VT-x extensions which address these problems
- Dynamically scan the code and replace bad instructions with call to emulator using INT3 (callout-and-emulate):
  - read-protect pages and perform scanning upon execution
  - write-protect pages after scanning to prevent modification of code
  - use execute-only tricks to prevent self-examining code

Solving x86 architecture problems (2)

- Perform binary translation of code by rewriting it to run emulation code
- 8086 mode – dynamically recompile and emulate real-mode code

VT-x/Pacifica

- Instruction set extensions which implement virtualization, designed separately by Intel and AMD, incompatible with each other but using similar techniques
- New privilege level is added for hypervisor, sometimes called ring (-1)
- Hypervisor works in ring -1 and can start virtual machines in ring 0 under supervision

VT-x/Pacifica (2)

- Hypervisor can control which events in ring 0 cause exceptions
- Troublesome instructions in ring 0 (like POPF or POP CS) also cause exceptions, so inconsistencies can be avoided
- Exceptions can be turned off to boost performance, some exceptions are selective (e.g. change in PE bit in CR0) ->
- It is possible to intercept VMRUN instructions so nesting virtual machines is possible

VT-x/Pacifica (3)

- Tagging of TLB (VMM and separately each VM address space) is added to avoid penalty when switching between hypervisor and guest
- It is possible to route IRQ to specific guest, intercept guest IRQs and invoke simulated interrupts in guests
- It is possible to intecept I/O ports in guests selectively (and allow some to be accesses directly)

VT-x/Pacifica (4)

- Virtual machine state is kept in VMCB (Virtual Machine Control Block) which has 4 KB size and contains information which instructions and events to intercept, state of VM (registers, cause of exit to VMM, “interrupt shadow” after STI, ...)
- At the moment of exit to VMM hypervisor can check in detail machine state based on VMCB

VT-x/Pacifica (5)

- Paged real mode has been added to allow efficient simulation of real mode
- The possibility to limit hardware devices access to certain regions of physical memory has been added to provide isolation of virtual machines which have direct access to hardware devices (using IOMMU)

VT-x/Pacifica (6)

- Nested paging has been added
- Addresses generated in VM are in guest linear space, which is further translated using paging into host linear space addresses
- This mechanism eliminates need for shadow page tables – OS in virtual machine can maintain page tables on its own, no need of intercepting

Performance

- Usermode code (ring 3) runs with native speed
- Ring 0 code causes performance penalty as privileged instructions must be emulated
- Simplest emulation (trap-and-emulate) has poor performance, VT-x/Pacifica in most VMMs is used in limited way!
- Callout-and-emulate has better performance, but also has significant overhead (interrupt)

Performance (2)

- Binary translation performs best in most cases.
- Example: RDTSC
  - trap-and-emulate: 2030 cycles
  - callout-and-emulate: 1254 cycles
  - binary translation: 216 cycles

Performance (3)

- In standard x86 virtualization exception handling has big overhead:
  - context switch from guest ring 3 to VMM
  - detecting that trap should be passed to VM
  - context switch to VM guest kernel
- VT-x/Pacifica provide better performance for traps – no need to route traps through VMM, they go directly to guest kernel

Xen

- Everything is easier and faster if we modify guest and it is aware it is running as guest (so called paravirtualization)
- Guest calls hypervisor (using hypercall) to perform some privileged action (modifying page tables, routing IRQs, modifying registers)
- Hypervisor checks if requested operation is OK and performs action instead of guest

Xen (2)

- Guests implemented as another architecture for Linux (also other OSs)
- Example: sti()
  - Original Linux implementation (link)
  - Xen architecture implementation (link)
- Host kernel also modified to work as domain 0 (hypervisor)

Security

- Virtual machines interact in complicated way with hypervisor
- Hypervisor is very complicated code and there are always bugs in complicated code
- Result: it is possible to force bugs in VMM and thus break out of VM or cause some nasty behavior
- Ring -1 can be used to create super-rootkit (BluePill)

Bibliography

- http://www.virtualbox.org/browser/trunk/sr
- http://www.virtualbox.org/wiki/VirtualBox_
- http://www.cs.wisc.edu/areas/os/Seminar/s
- http://www.cs.nps.navy.mil/people/faculty/
- http://www.cs.utexas.edu/users/hunt/class
- http://softwarecommunity.intel.com/Wiki/V
- http://www.intel.com/technology/itj/2006/v
- http://www.blackhat.com/presentations/bh

- http://www.symantec.com/avcenter/refere
- http://www.dell.com/downloads/global/pow