System Virtual Machines (2)

Contents

- Resource virtualization – I/O
- Performance Enhancement of System Virtual Machines

Virtualizing I/O Device

- Constructing a virtual version of a device and virtualizing its I/O activity
  - E.g., Providing a virtual NIC for each VM with a single, physical NIC
  - When a guest VM requests to use a virtual device, VMM intercepts the request and convert it to equivalent requests to the physical device
- I/O virtualization depends on the type of the devices

Virtualizing I/O Device

- Dedicated devices
  - E.g., displays, keyboards, mouse, …
  - Does not necessarily have to be virtualized
  - Requests to and from the device could bypass the VMM, in theory
  - However, they are handled by the VMM first since OS is running in user mode
- Partitioned device
  - E.g., a very large disk
  - Can be partitioned into several smaller virtual disks as dedicated to VMs
  - VMM translates track/sector numbers to those of the physical disk

Virtualizing I/O Device

- Shared devices
  - E.g., a network card
  - Can be shared among guest VMs
  - Each guest VM has its own virtual state, e.g., a virtual network address
    - Maintained by the VMM for each guest VM
  - VMM translate requests from a VM to one with the physical device

- Nonexistent physical device
  - Virtual devices “attached” to a virtual machine for which there is no corresponding physical device
  - For example, a network card used for communicating with other virtual machines on the same platform without a physical network card
  - VMM intercepts requests from a VM, buffer it and interrupt the other VM

Virtualizing I/O Device

- Spooled Device
  - Use a two-level spool tables

Virtual Machine 1 Spool Table

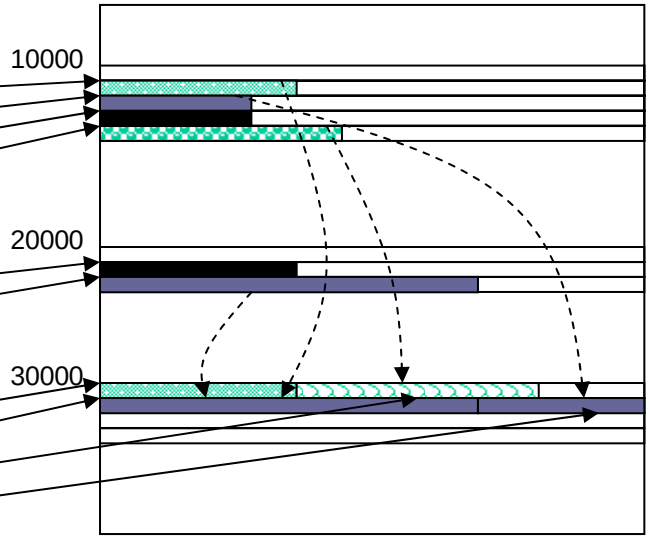
Program	Status	Location	Real loc	Size
A	Printed	1000	11000	400
B	Completed	2000	12000	200
C	Running	3000	13000	200
D	Completed	4000	14000	500

Virtual Machine 2 Spool Table

Program	Status	Location	Real loc	Size
P	Running	1000	21000	400
Q	Completed	2000	22000	800

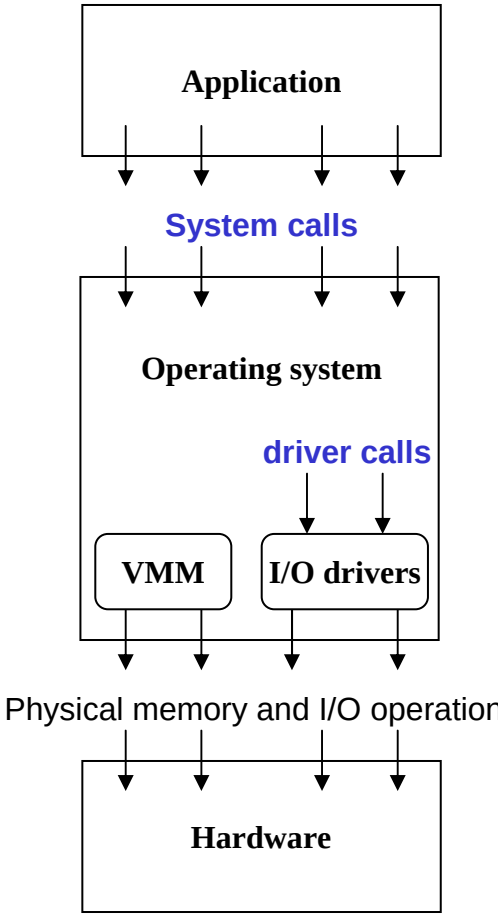
VMM Spool Table

VM	Status	Status	Real loc	Size
1	A	Printed	30000	400
2	Q	Printing	31000	800
1	B	Waiting	31800	200
1	D	Waiting	30400	500



Virtualizing I/O Activities

- I/O activities are abstracted by system call interface and device driver interface
  - An application program makes a device-independent I/O request (e.g., an open() system call)
  - OS converts the request into a call to a device driver routine (e.g., an open() device driver call)
  - Device driver performs device-specific I/O operations
    - Based on I/O instructions or memory-mapped I/O
- VMM intercepts a guest's I/O action and convert it from a virtual device action to a real device action at
  - System call interface
  - Device driver interface
  - Operational-level interface



Virtualizing at I/O operation Level

- Two ISA approaches to I/O: memory-mapped I/O and I/O instructions
  - ISA only provides mechanism for addressing I/O device and transfering data
  - Memory-mapped I/O
    - Loads/stores to specific region of real memory are interpreted as command to devices
    - OS only can access them
    - Used in RISC processors, sometimes in addition to I/O instructions
  - I/O instructions
    - Special input/output instructions with special addresses
    - Used in IA-32 (in and out)
  - I/O instructions are privileged and the memory-mapped I/O region is protected

- Due to its privileged nature, these I/O operations will be trapped to the VMM
- But it is hard for VMM to get the whole picture with each trapped instruction

Virtualizing at the Device Driver Level

- A system call is translated by OS to a call to device driver routine
- If VMM can intercept the call to a virtual device driver, it can convert
  - Virtual device information to the corresponding physical device
  - Virtual device driver call to a physical device driver call
- This requires VMM developer to know well OS and device driver interfaces
  - If Windows or Linux are considered only, virtual device drivers can easily be made for each type of device and distributed to the users together with the VMM

Virtualizing at the System call Level

- Virtualization would be more efficient if I/O system calls can be intercepted
  - Then, the entire I/O action can be done by the VMM
- Requires shadowed I/O system call routines which mimics the same
  - Much more substantial than writing virtual device drivers

Performance Enhancement of System VM

- Performance in early system VM was low (21% of native system, 1979)
- Reasons for performance degradation
  - Setup overhead
    - Setting resources when a VM is activated
  - Emulation
    - Sensitive instructions and possible others must be emulated
  - Interrupt handling
    - Interrupts must be handled by VMM first even though they are handled by OS
  - State saving
    - Saving state of VM when control is transferred to VMM
  - Bookkeeping
    - E.g., Accounting of time charged to user is different
  - Time elongation
    - E.g., Accessing shadow page table as well as local page tables for memory access

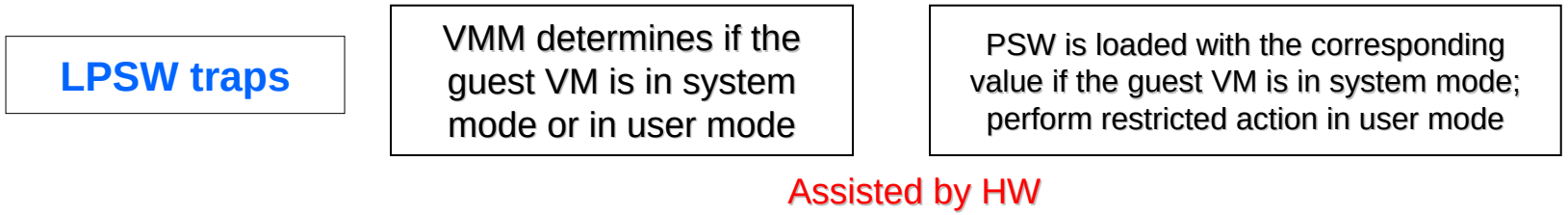
Solution – VM Assist

- H/W extension to improve performance
- e.g., IBM VM/370 assist collection

Assist Collection		Number of functions
Virtual Machine Assists (VMA)		13
Extended control program support	Control program assist	22
	Expanded virtual machine assist	12
	Virtual interval timer assist	1
Shadow table bypass assist		8
Preferred machine assist		22
Dual address space assist		20
Extended storage key assist		3
Total		101

Instruction Emulation Assist

- Instruction emulation assists
  - The HW (via microcode) performs the emulation of special instructions
  - E.g., In System/370, 13 instructions are assisted by HW
    - **LOAD PSW (LPSW)**, INSERT PSW KEY (IPK), INSERT STORAGE KEY (ISK), LOAD REAL ADDRESS (LRA), RESET REFERENCE BIT (RRB), SUPERVISOR CALL (SVC), SET STORAGE KEY (SSK), SET SYSTEM MASK (SSM), STORE CONTROL (STCTL), STORE AND AND SYSTEM MASK (STNSM), STORE THEN OR SYSTEM MASK (STOSM), SET PSW KEY FROM ADDRESS (SPKA)



VMM Assist

- Virtual machine monitor assists (1)
  - Context switch between VM and VMM
    - HW save/restore registers
  - Decoding of privileged instructions
    - Privileged instructions are executed frequently in OS which would cause a significant overhead of trapping
    - HW decoding of privileged instructions reduces some of this overhead

VMM Assist

- Virtual machine monitor assists (2)
  - Virtual interval timer
    - While the guest VM is running, virtual timer in a certain memory location is decremented automatically by the real timer
    - H/W assist gives a timer interrupt to the VM when the virtual time becomes zero
  - Additional instruction set
    - Obtain free space from free storage area
    - Return space to free storage
    - Page lock/unlock
    - Translate virtual address and test for shared page
    - Invalidate segment/page table

Improving Performance of Guest VM

- If guest OS knows if it is running on a VM and if it can provide some information to the VMM, we can improve performance by removing redundant actions in OS and VMM
- If VMM can access to the internals of the OS, we can also improve the performance
- We can also add H/W features for the fast execution of the VMM

- Actually, these approaches are emerging again with para-virtualization or with a H/W support these days

Schedule for the Remaining Classes

- We will have an exam on Nov 7th (Wed)
- The presentation schedule
  - Nov 12:
  - Nov 14: Reserved for an outside speaker
  - Nov 19:
  - Nov 21:
  - Nov 26:
  - Nov 28: Reserved for an outside speaker
  - Dec 3:
  - Dec 5:
  - Dec 10: