CS 695 Lecture #3

A1
- 25th Jan 11.59 pm (Saturday)

* Design of VMMs
~ Popek & Goldberg (virtualization principles)
efficiency, resource control, equivalence.

CPU virtualization for VMs

Design 0: trap-and-emulate (OS-inspired)

x86

Baremetal OS	VMM	Ring / privilege level
Apps	Apps	3 ~ lowest PL
	OS	1
OS	VMM	0 ~ highest PL

example.

(i) guest OS → lcr3 %eax ← load CR3 w/ contents of eax regs.
lcr3 %mem

traps guest OS execution as PL does not match

falls to trap handler in PL 0
switches

validations, modifications to the update.

CR3 → source — guestOS, nonguestOS
memory
bound check on ear/address
translate the address to another address.

Virtualization: interception for checks & translation

(ii) set interval timer

(iii) write to IDTR register

(iv) timer interrupt

controls the time quantum of process on CPU

on trap vmm may decide to write a partial value.

(*) yohoo! cpu virtualization done!

NO!

└ set of instructions that do not generate a trap when min PL on CPU does not match.

example x86

(i) popf ~ pops ~~current~~ top of stack and writes to FLAGS register

9th bit of the FLAGS regs.

is IF ~ interrupt flag enable/disable interrupts.

when guest OS is a Ring 1

popf ⇒ nop and no trap.

(ii) sidt sgdt sldt

└ works w/ Ring 1

~ 17 instructions of the x86 ISA that were not virtualizable.

Design of VMMs. (Categorization)

① Bare-metal hypervisors. (Type 1) vmware, xen vs Hosted VMMs (Type 2) virtualbox, kvm

② Full virtualization (interface to OS is same as bare-metal) vs Para-virtualization (OS knows that it is on a VM) & makes VMM-specific changes to itself.

Design 1: Scan-and-patch / binary translation (vmware) — full virtualization

- at start/end of each basic code block
- scan for critical instructions (sensitive but do not generate trap)
- replace instruction w/ a trap
- make critical instruction privileged
- handle critical inst^r in trap handler