Issues

- Also, subject to quirks of the architecture
  - Example: x86
  - fails silently if some privileged instructions execute without privilege
  - doesn’t have clean separation between privileged and non privileged instructions
  - Backward compatibility means difficult to improve

Issues

- Consider Intel x86 popf instruction
  - Loads CPU flags register from contents of the stack
  - If CPU in privileged mode -> all flags replaced
  - If CPU in user mode -> on some flags replaced
  - No trap is generated

Binary translation

- Use binary translation to modify OS to rewrite silent failure instructions

- More aggressive translation can be used

  - Translate OS mode instructions to equivalent VMM instructions

  - Some operations still expensive
    Cache for future use
    Used by VMWare ESXi and Microsoft Virtual Server

- Performance on x86 typically ~80-95% of native

Paravirtualization

- Modify the OS to make it aware of the hypervisor
  - Can avoid the tricky features
    Aware of the fact it is virtualized
  - Can implement optimizations
- Comparison to binary translation?
- Amount of code change?
  – 1.36% of Linux, 0.04% for Windows

Hardware assistance

- All virtualization needs some HW support
    - More support -> more feature rich, stable, better performance of guests

- Intel added new VT-x instructions in 2005 and AMD the AMD-V instructions in 2006
    - CPUs with these instructions remove need for binary translation
    - Generally define more CPU modes – “guest” and “host”
    - VMM can enable host mode, define characteristics of each guest VM, switch to guest mode and guest(s) on CPU(s)
    - In guest mode, guest OS thinks it is running natively, sees devices (as defined by VMM for that guest)
    - Access to virtualized device, privileged instructions cause trap to VMM
    - CPU maintains VCPU, context switches it as needed
    - HW support for Nested Page Tables, DMA, interrupts as well over time

Next: Virtualization 2