Virtual Machines (section 5.6)

- Why do we have operating systems
  - Protect system and hardware resources
  - Isolate and protect processes
  - Share resources among processes (better utilization)
  - Emulate services not directly available in hardware (example: timers)
- Sharing resources in an OS:
  - Time sharing (multiplexing) the processor
  - Space sharing (multiplexing) the memory
  - On-demand sharing of I/O
- Overhead of having an OS:
  - No overhead when process is executing and does not perform system calls
  - System call overhead when process invokes the OS
  - Overhead of managing and scheduling processes

Virtual Machine Monitors (VMM)

- Multiple operating systems executing simultaneously on the hardware
- VMM emulates guest operating system and machine resources
  - Each OS gets the illusion that it owns the hardware
  - Improved isolation of multiple guest OSs
  - Avoids security and reliability problems
- Virtualization has some performance impact
  - Feasible with modern high-performance computers
- Examples
  - IBM VM/370 (1970s technology!)
  - VMWare
  - Microsoft Virtual PC
**Virtual Machine Monitor**

- Maps virtual resources to physical resources
  - Memory, I/O devices, CPUs
- Guest code runs on native machine in user mode
  - Traps to VMM on instructions that access protected resources
- VMM handles real I/O devices (emulates virtual I/O devices)

If no virtual machine, access hardware by trapping to OS

To access hardware, application traps to OS, which traps to VMM.

---

**Example: Virtualization of interrupts**

- On an interrupt in the native machine, the OS suspends the current process, handles the interrupt, then either resumes the process or selects another process to run.
- With a Virtual Machine Monitor, an interrupt suspends the current VM, handles the interrupt on behalf of the appropriate VM, then either resumes the suspended VM or selects another VM to run.

**Example: Virtualizing a hardware timer:**

- One hardware timer can be virtualized to interrupt multiple OSs
  - If OS$_1$ requests an interrupt at time $t_1$
  - and OS$_2$ requests an interrupt at time $t_2$
  - the times $t_1$ and $t_2$ are kept in a software list and the hardware timer is set to the minimum of $t_1$ and $t_2$. 
Question: calculate the CPI for the system listed above assuming that there are no accesses to I/O. What is the CPI if the performance impact double or cut in half?. What is the longest penalty to trap to VMM for a maximum of at most 10% performance degradation?

With virtualization, the CPI = 1.5 + (120/10000) * (15+175) = 3.78
If VMM performance impact doubles -> CPI = 1.5 + (120/10000) * (15+350) = 5.88
If VMM performance impact halves -> CPI = 1.5 + (120/10000) * (15+87.5) = 2.73

Without virtualization, the CPI = 1.5 + (120/10000) * (15) = 1.68
For a 10% performance degradation, the longest possible penalty for a trap to VMM, T, should be given by:

1.5 + (120/10000) * (15+T) < 1.1 * 1.68
That is, T < 14

For a non-virtualized system,

CPI = 1.5 + (120/10000) * (15) + (30/10000) * 1100 = 4.98

For a virtualized system,

CPI = 1.5 + (120/10000) * (15+175) + (30/10000) * (1100+175) = 7.6

I/O traps usually require long periods of execution spent in the guest O/S, with only a small portion of that time spent in the VMM. As such, the impact of virtualization is less for I/O bound applications. This is confirmed by the result indicating that the CPI increases from 1.68 to 3.78 (a factor of 2.25) when there is no I/O but increases from 3.33 to 5.69 (a factor of 1.7) when the I/O rate is low and increases from 4.98 to 7.6 (a factor of 1.53) when the I/O rate is high.