Virtualisation Technologies

RECAP:

- Can never virtualise time
- AIMS:
  - Abstraction (no details of hardware needed)
  - No interference/safety
- PROCESS vs. SYSTEM Virtualisation
  - JVM, Dynamic binary translators (Rosetta), Valgrind (dynamic binary optimisers)
- System Virtualisation
  - make UNIVERSAL
  - write once, COMPILE anywhere
    - language :-) #MostPortable #MorePortableThanJava
    - need source code, no guarantee of recompilation :((problem of write once, COMPILE anywhere)
  - Hosted Virtualisation: normal 'Hosted' OS with hypervisor/virtual monitor machine (VMM) ontop of it
- XEN Guest 0 Virtualisation (!FUN!)

- OS handles physical resources
- VMM controls all attempts at guest OS' accesses to physical resource (isolated from resources)

- Guarded Physical Resources
  - Timer
  - CPU Reg.
    - Interrupt Enable
    - Page Table Base
  - Device Control Reg.
    - Programmed I/O — not used often
    - Interrupt I/O — end of operation, input available
    - Direct Memory Access I/O —

- How do you divide Physical Memory amongst guest OS's?

- virtual address → OS Page Table (+ TLB: more speed) → Physical Address
  - UN-Virtualised

VA → OS Page Table → VMM Page Table → PA

new component to make it VIRTUALISED

- Interfacing Guest OS's and VMM
  - Static software
  - Dynamic software
    - Don't change or determine unusual instructions ahead of time
  - Hardware (dynamic)

- ParaVirtualization
  - call VMM for privileged operations, I/O
  - Cooperate with VMM over shared page table
  - (:|) have to modify OS

- Detect & Fix interfaces in VMM
  - write-protect guest OS page tables
  - (!) Complete (!)

- Less bit in lecture slides:
  - (:) = faster, here to stay
  - (:|) = must change hardware and shit