

Virtualisation Technologies

RECAP:

- Can never virtualise time
- APIs:
 - Abstraction (no details of hardware needed)
 - No interference/safety
- PROCESS vs. SYSTEM Virtualisation
 - JVM, Dynamic binary translators (Rosetta), Valgrind
 - dynamic binary
bootstrappers
- System Virtualisation
 - make UNIVERSAL
 - write once, COMPILE anywhere
 - language ↗
 - ↗ Most portable
 - ↗ More portable than Java
 - 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 OS Virtualisation ↗ FULL
 - 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

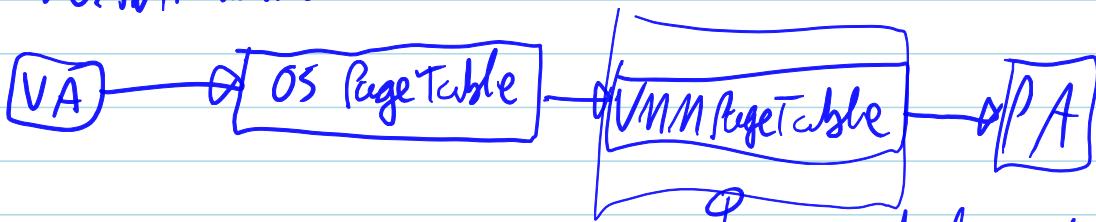
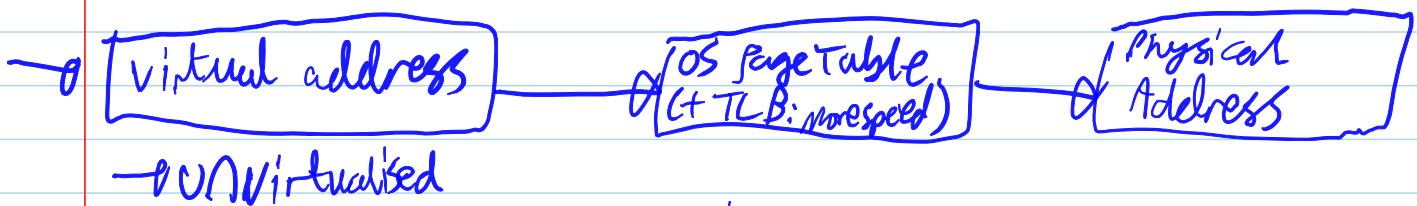
→ Device Control Reg.

→ 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?



new component to make it
VIR TUALISED

→ Interfacing Guest OS's and VMM

→ Static software

→ Dynamic software

→ Don't change or determine unusual instructions
ahead of time

→ Hardware (dynamic)

- ParaVirtualisation
 - call VM for privileged operations, I/O
 - Cooperate with VM over shared page table
 - (1)
 - have to modify OS

- Detect & Fix interfaces in VM

↑
writeprotect
guestOS
page
tables

(!)Complete(!)

- last bit in lecture slides:

(1) = fastest, here to stay

(2) = must change hardware and shift

