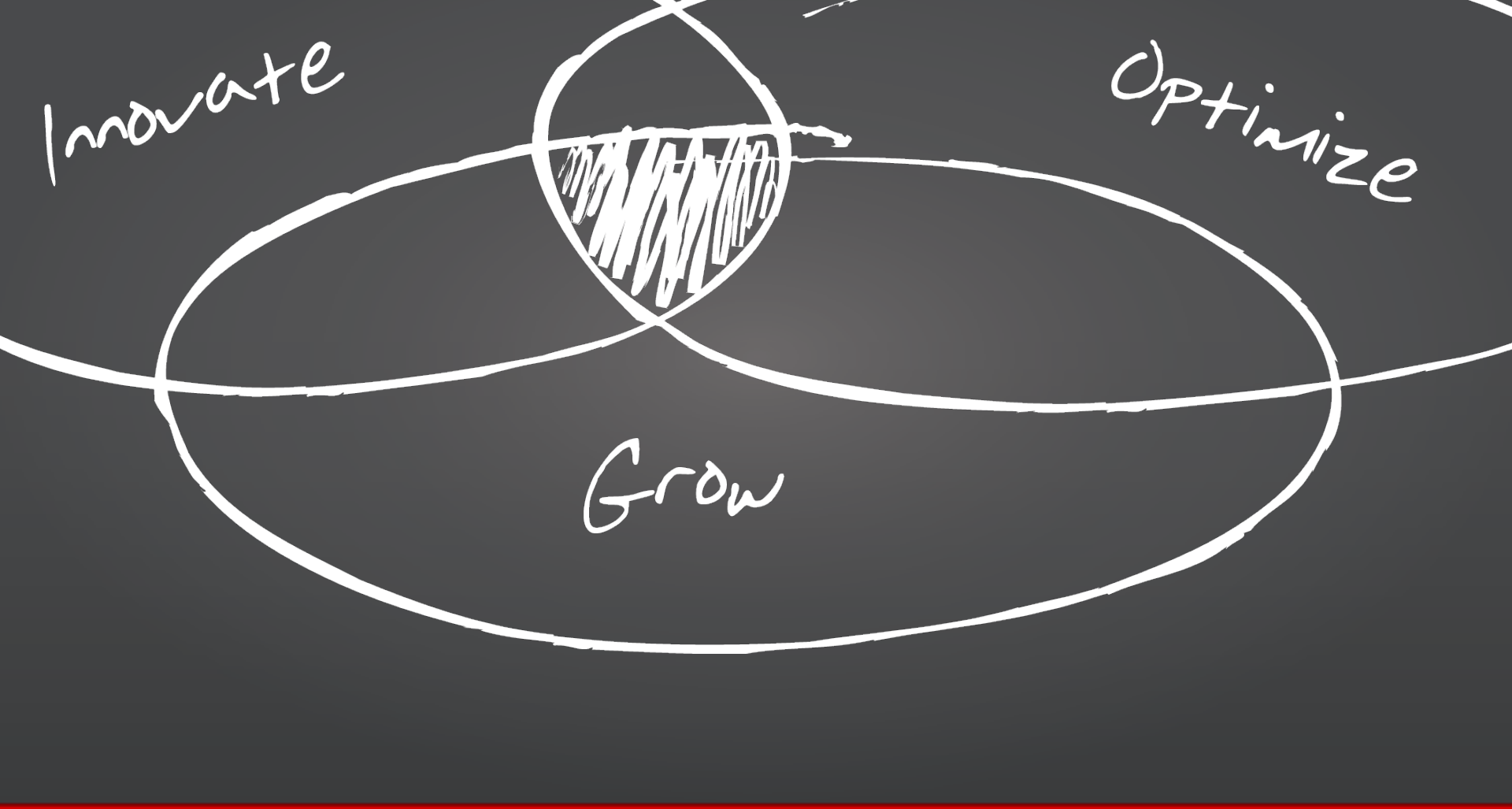
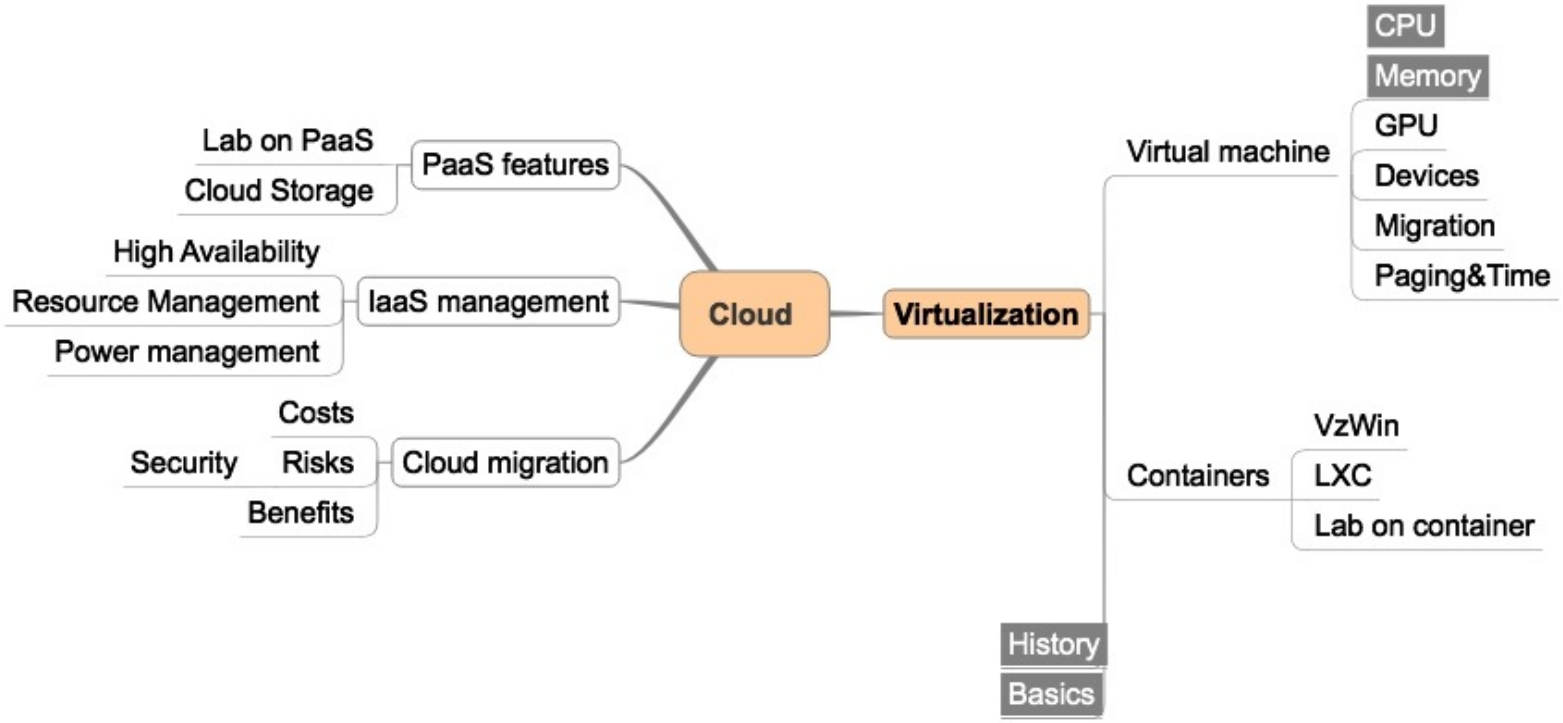


The total virtualization

Virualizing time; virtualizing paging

Course overview



Content

- ✓ Virtualizing time:
  - ✓ Time in x86
  - ✓ Splendours and miseries of timekeeping
- ✓ Virtualizing paging:
  - ✓ Paging in depth
  - ✓ Shadow paging
  - ✓ EPT/RVI

Paging is a popular data structure you'd never learn from books. Time is the master of the universe

Time

- What do you know about time keeping in OS?

- What type of timers do you know?

Time

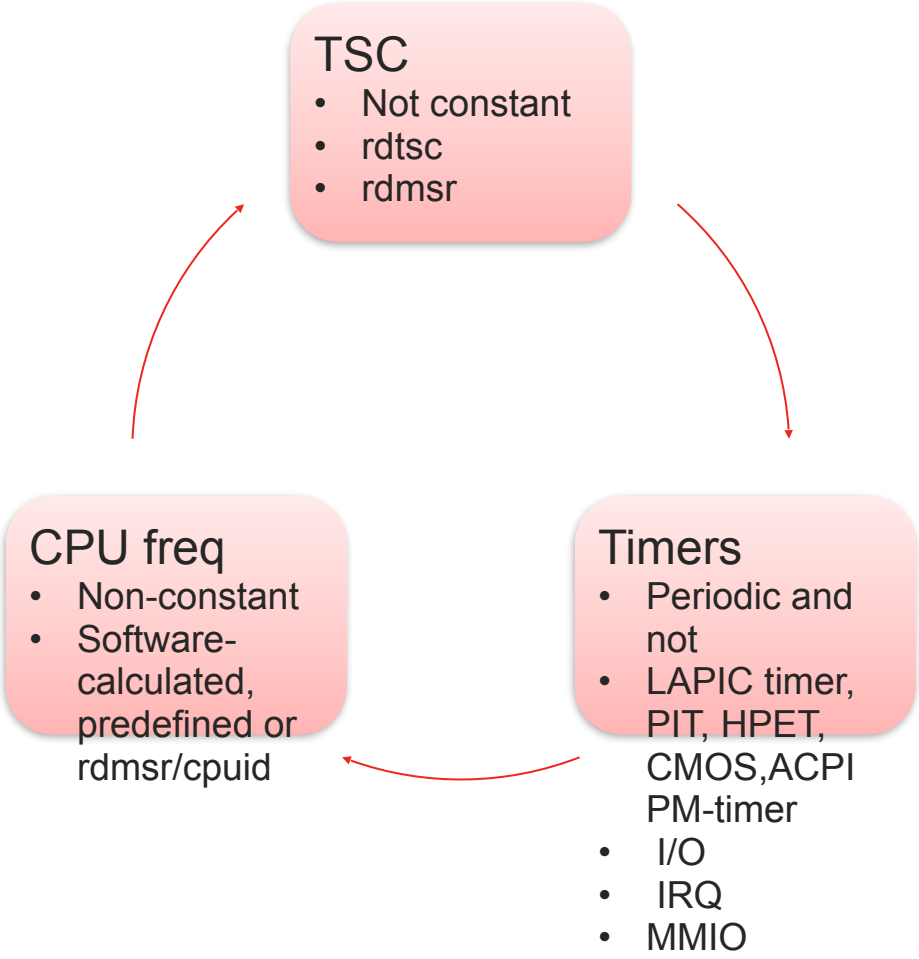
- ✓ Time stamp counter, MHz

- ✓ Timers
  - ✓ CMOS
  - ✓ PIT
  - ✓ APIC timer
  - ✓ ACPI PM timer
  - ✓ HPET

Time in OS

- ✓Tick counting
- ✓Tickless scheme
- ✓Wall-clock

Time: components



Time: what's wrong with TSC?

- ✓ Different TSC on different CPU (invariant TSC)
- ✓ Dependency of CPU frequency from ACPI P-states (constant TSC)
- ✓ C-state (non-stop TSC)

Timekeeping problems

- ✓ vCPU could be rescheduled:
  - ✓ Lags
    - ✓ More frequent interrupts and speed-up of timers counters
    - ✓ Lost tick
  - ✓ No timer guarantee
- ✓ Guest could go natively without VMM exits for too long
  - ✓ No guest timer guarantee
- ✓ Performance

Timing:

Interrupt delivery

- ✓ Guarantees for in-time delivery (guest and host problems)

Device emulation

- ✓ I/O ports (CMOS, PIT, ACPI)
- ✓ MemI/O (LAPIC, HPET)
- ✓ MSR (TSC, x2Lapic)
- ✓ RDTSC

Timing: hardware virtualization

- ✓Native RDTSC
- ✓VMX-preemption timer
- ✓Native access to some timer fields (RDMSR x2APIC)

Timing: paravirtualization

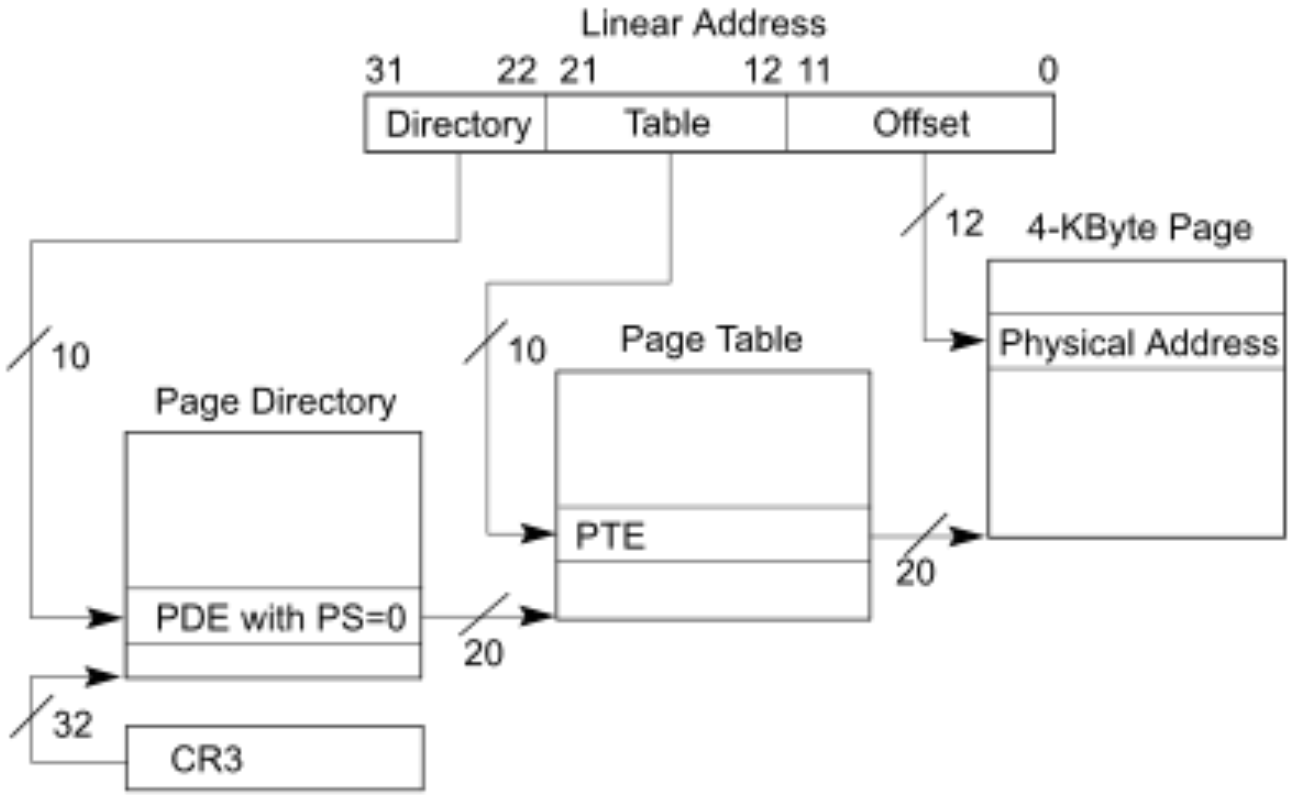
- ✓Hyper-V (Partition Reference Time Enlightenment)
- ✓Kvm_clock
- ✓Provider-specific paravirt (Windows ACPI-PM)
- ✓Guest tools (for time sync)

Stop the tea party! Make time happen

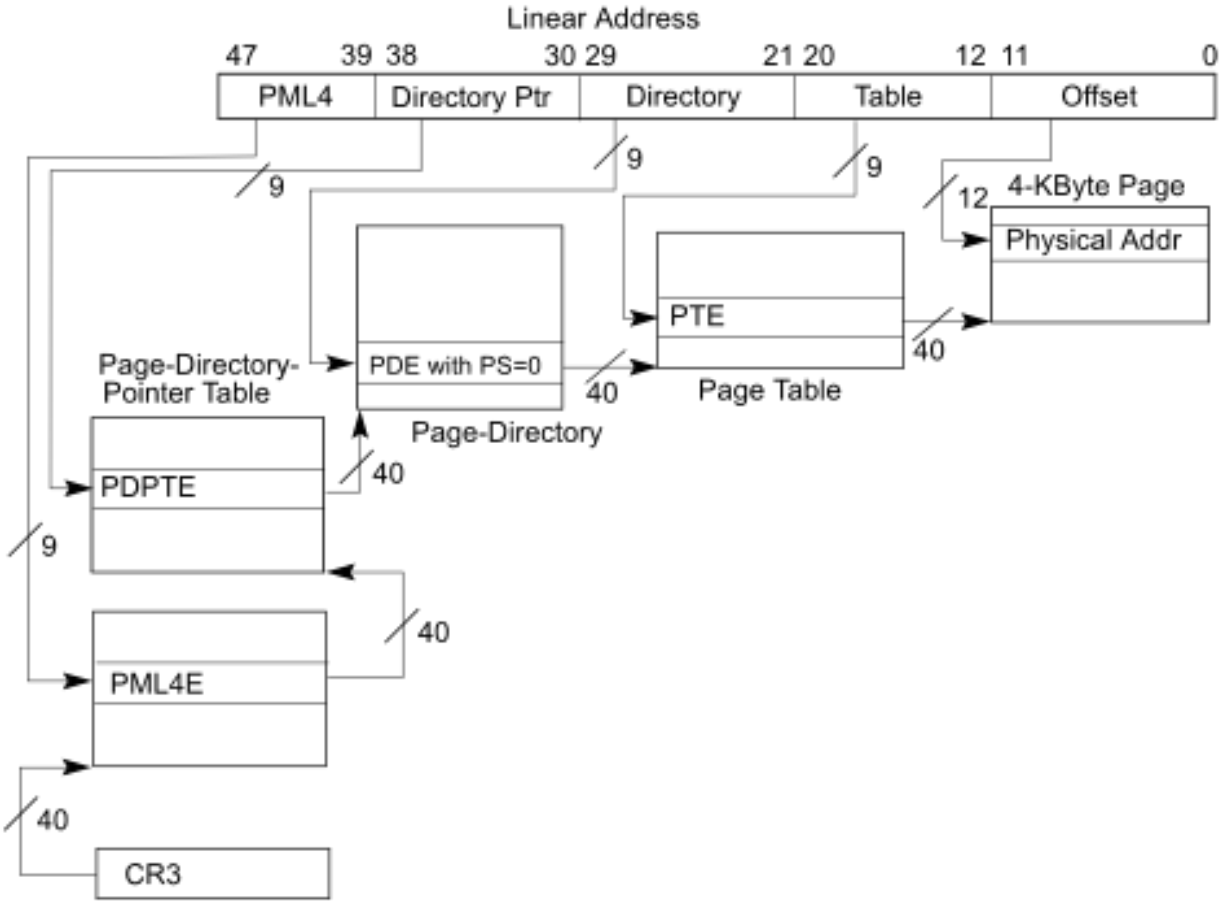
Questions?

Paging

Paging (x32)


Linear Address
31
22
21
12
11
0
Directory
Table
Offset
12
4-KByte Page
Physical Address
10
Page Table
10
Page Directory
PTE
20
PDE with PS=0
20
32
CR3

Paging (x64)


Linear Address
47
39 38
30 29
21 20
12 11
0
PML4
Directory Ptr
Directory
Table
Offset
9
9
9
12
4-KByte Page
Physical Addr
PTE
40
Page-Directory-
Pointer Table
PDE with PS=0
40
Page Table
Page-Directory
40
PDPTE
9
40
PML4E
40
CR3

Paging

- ✓ Why do we need paging?
- ✓ What if no paging?
- ✓ Paging level tradeoff
- ✓ Page size tradeoff

Paging & CPU

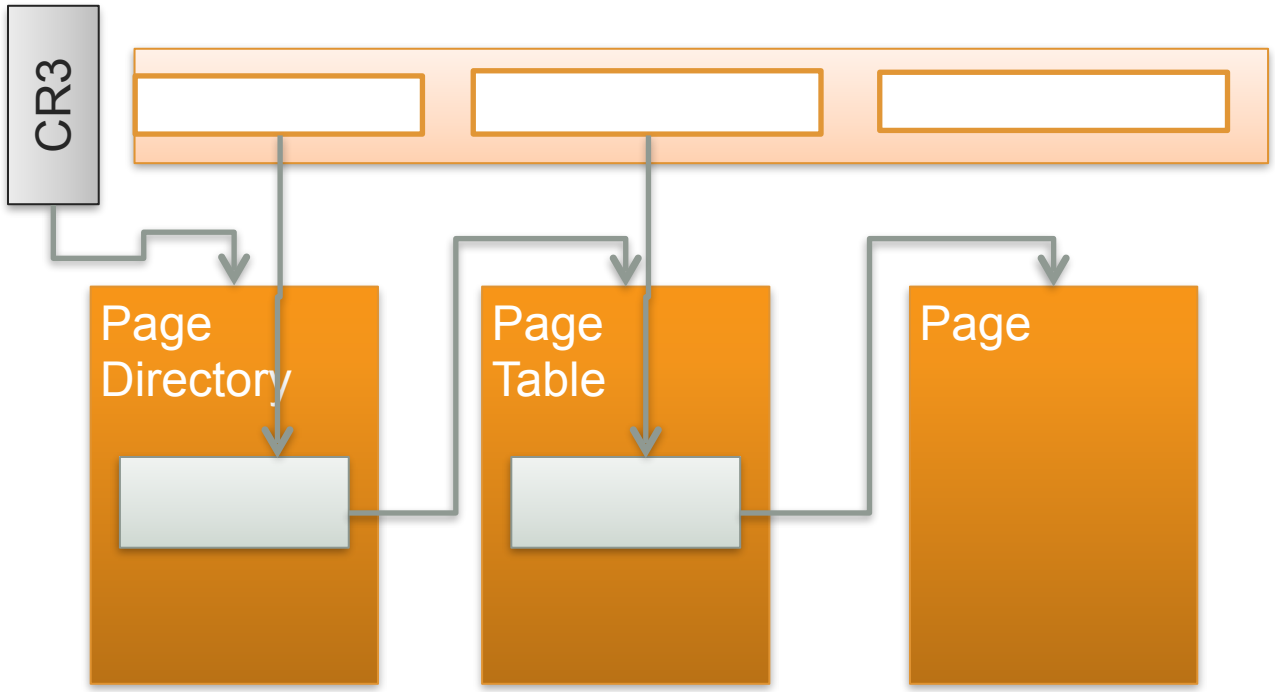
- ✓Paging: single or multiple (per CPU)
- ✓Cache: single or multiple (per CPU)

- ✓How to sync pagings and caches?
  - ✓CR3
  - ✓IPI
  - ✓Invlpg
  - ✓Global pages

Paging is …

- ✓ Page walk (LA -> PA)
- ✓ Protection (RO, NXE, SU/US)
- ✓ Paging modes (CR0, CR4)
- ✓ TLB
- ✓ Paging root (CR3)

Virtualizing paging

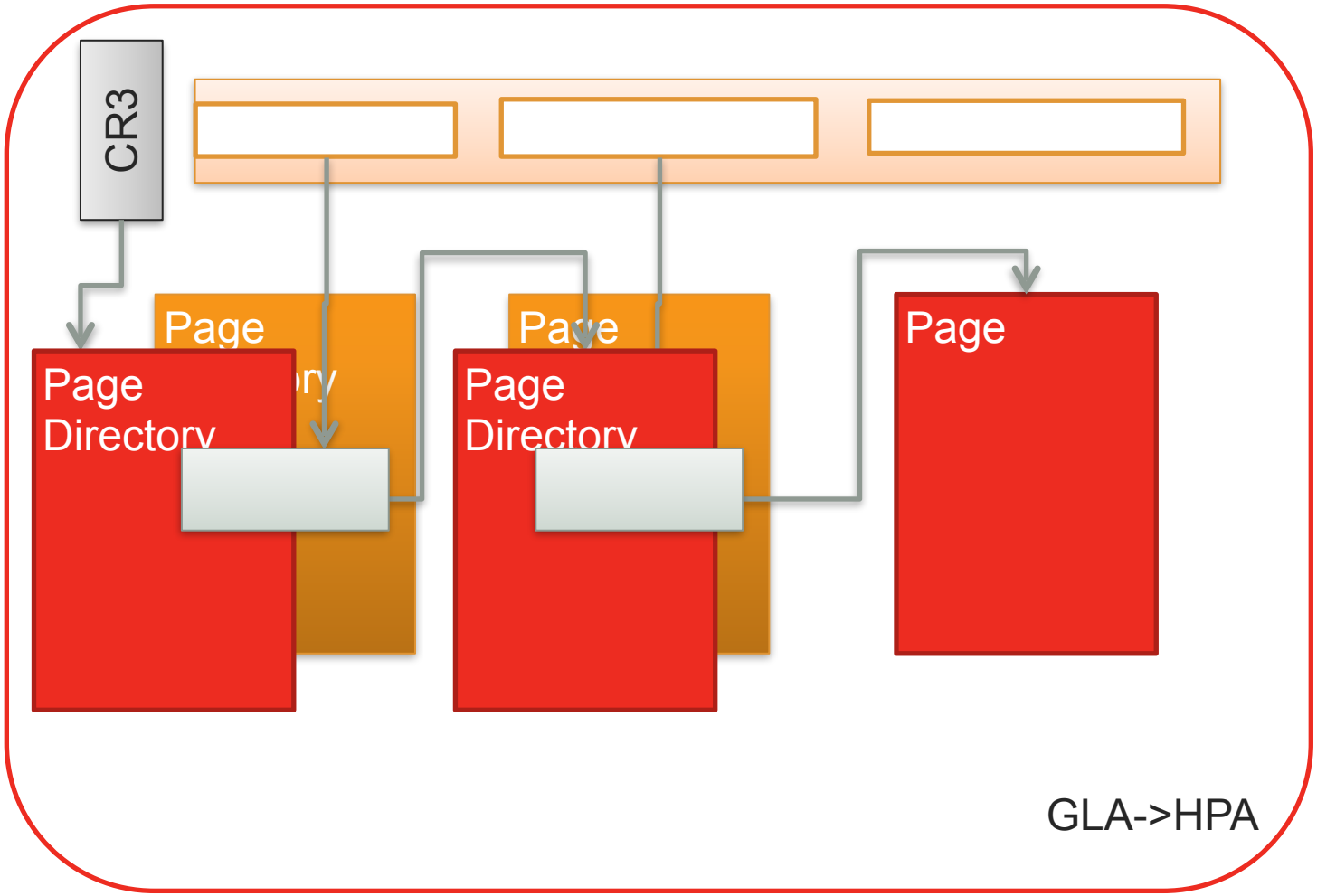


GLA->GPA

Virtualizing paging

- ✓ #PF
  - ✓ Shadow paging maintenance
  - ✓ Memory Mapped I/O
  - ✓ GPA -> HPA
- ✓ Change paging mode (CR0, CR4)
- ✓ Change paging (CR3)
- ✓ Invlpg
- ✓ Hide-and-seek: vmm vs guest

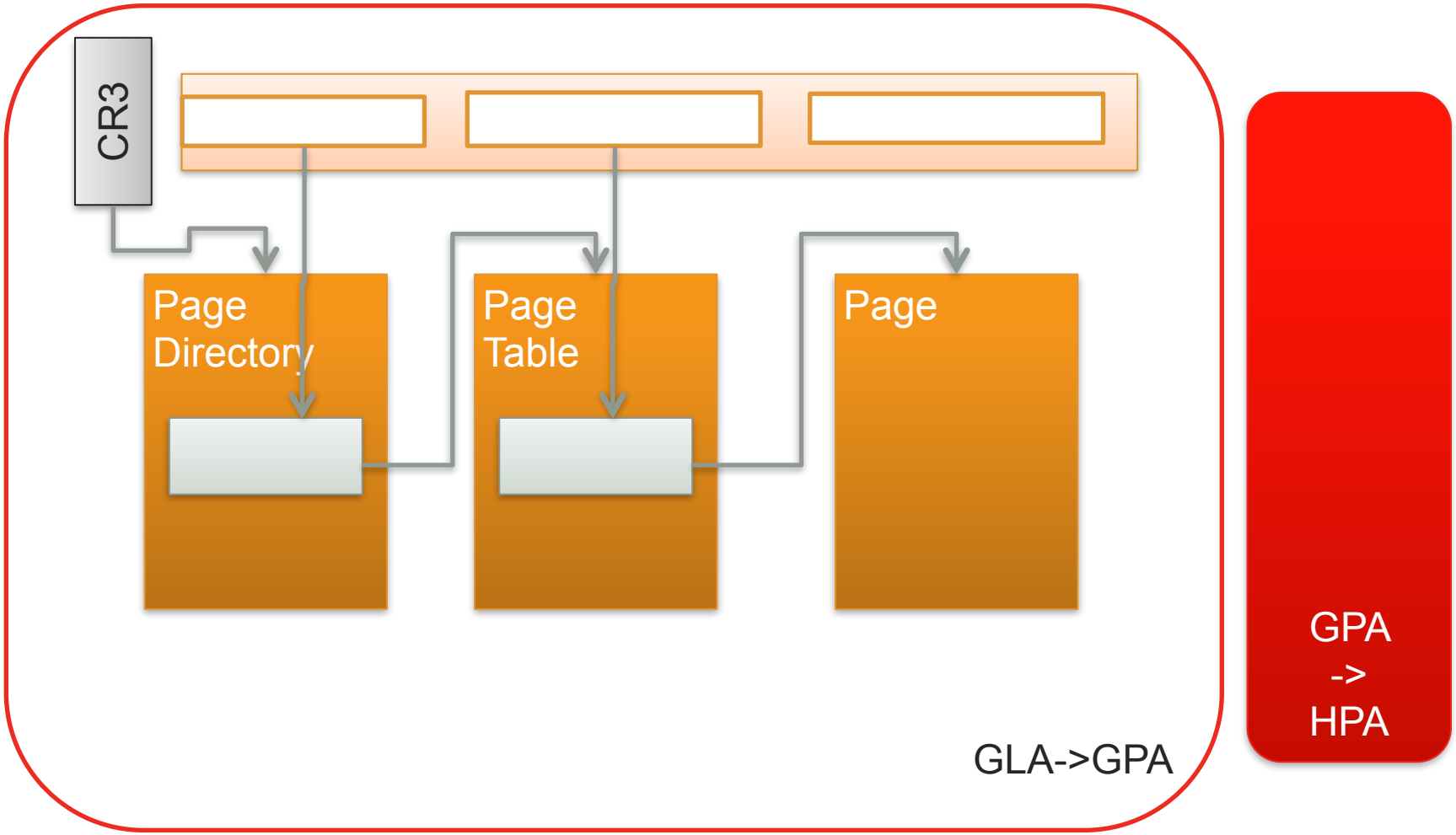
Virtualizing paging: shadow paging



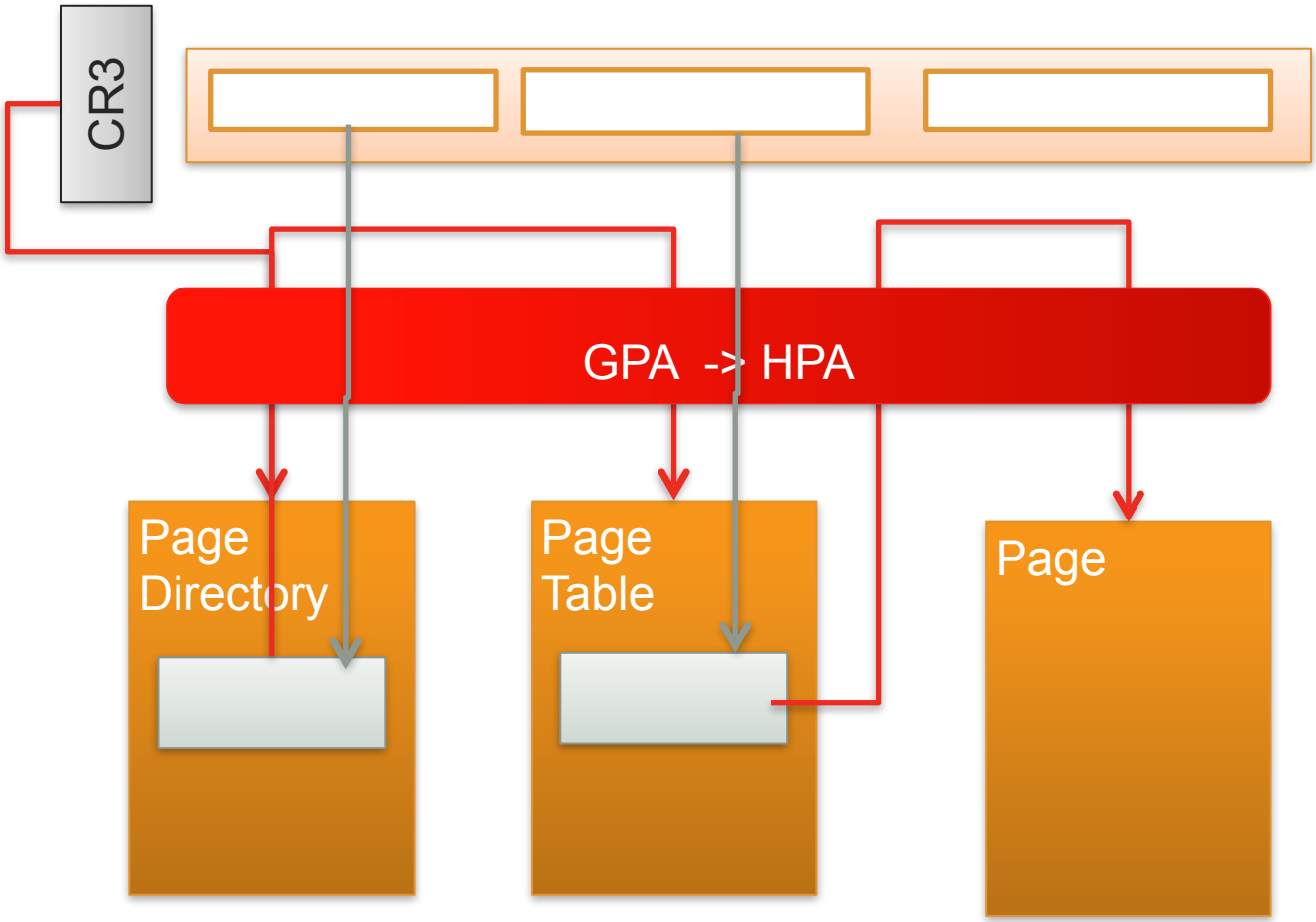
Virtualizing paging: challenges

- ✓ Performance loss due to #PFs
- ✓ Dangerous hide-and-seek
- ✓ Bugs!
- ✓ Extra TLB flushes on guest<->vmm switches

Virtualizing paging: NPT (RVI/EPT):



Virtualizing paging: NPT (RVI/EPT):



Virtualizing paging: NPT (RVI/EPT)

Pros:

- ✓No shadow paging
- ✓#PF only on (GPA->HPA miss + memory mapped I/O) -> performance
- ✓No TLB errors & hide-find games

Cons:

- ✓Possible performance loss on stable WS
- ✓Extra #PF on guest code access

Conclusions

Paging is a hierarchy data structure that implements a compromise between fast access and meta-data volume. Time in computer systems is never reliable enough

Questions?