CS 695: Virtualization and Cloud Computing

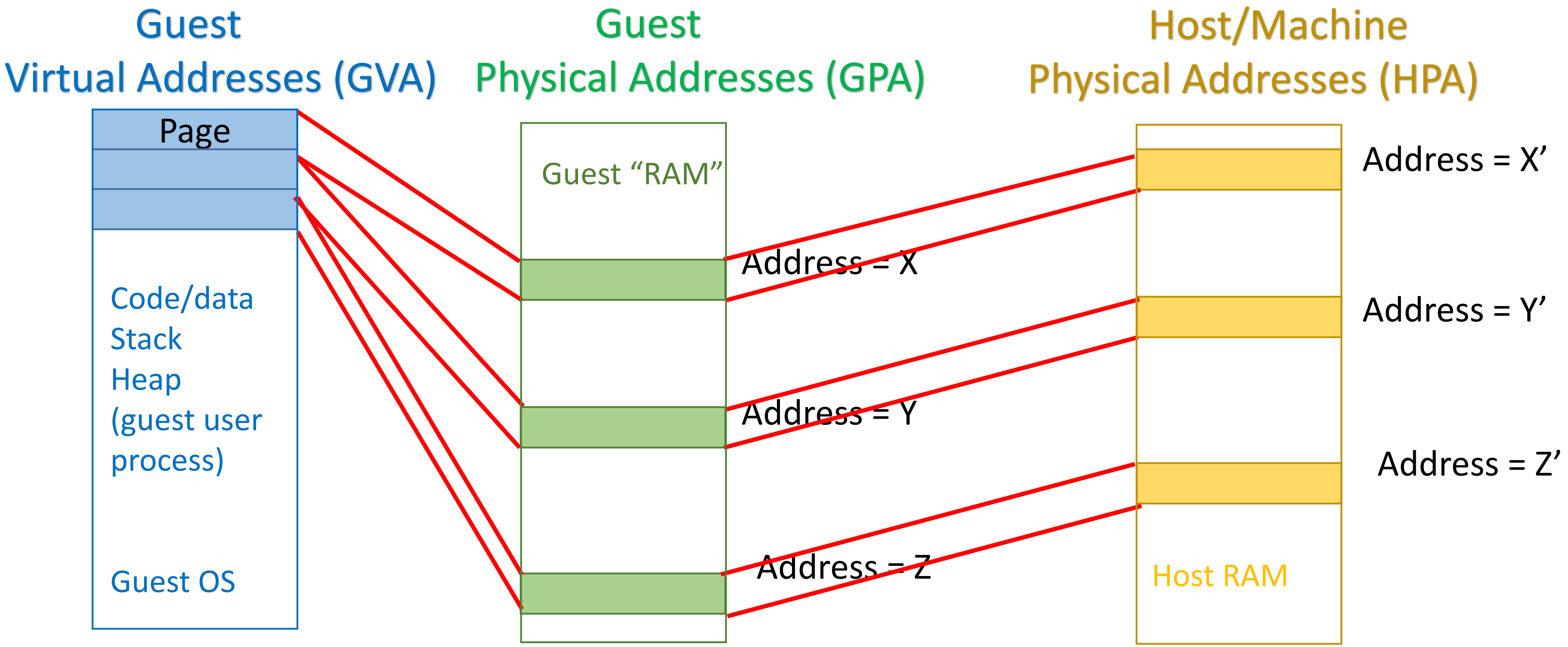
Lecture 6: Memory Virtualization

Mythili Vutukuru

IIT Bombay

Spring 2021

Recap: Memory virtualization problem

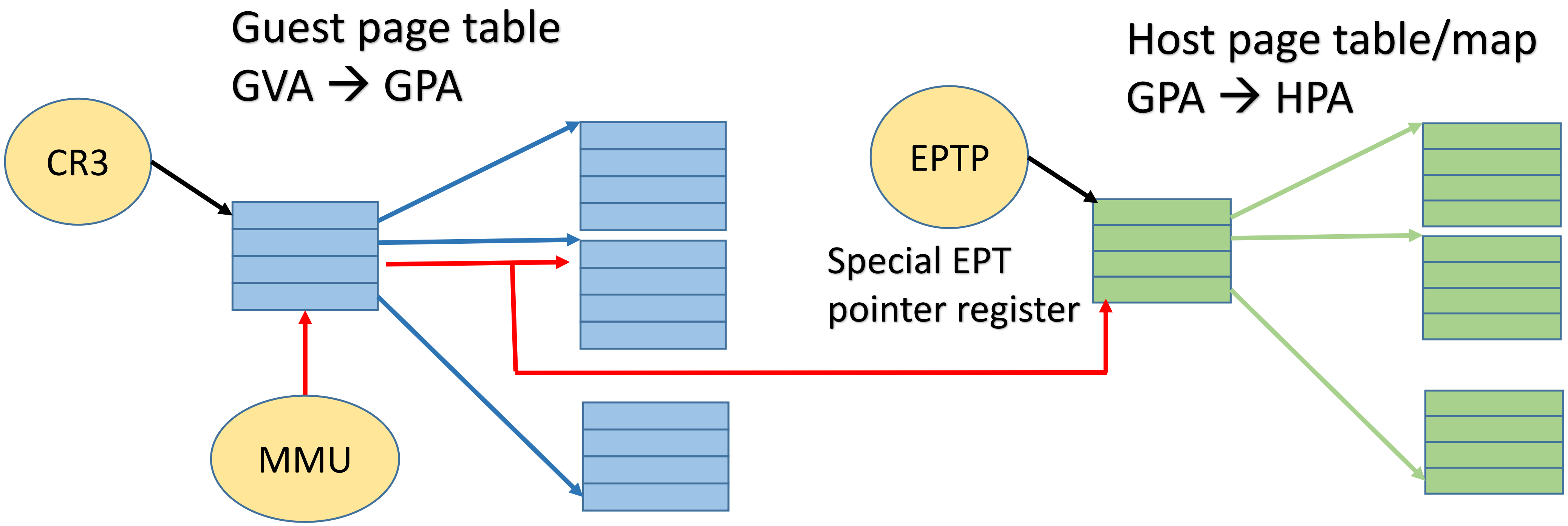


Guest "RAM" is actually memory of the userspace hypervisor process running on the host, which is mapped to host memory by the host's page table

Recap: Techniques for memory virtualization

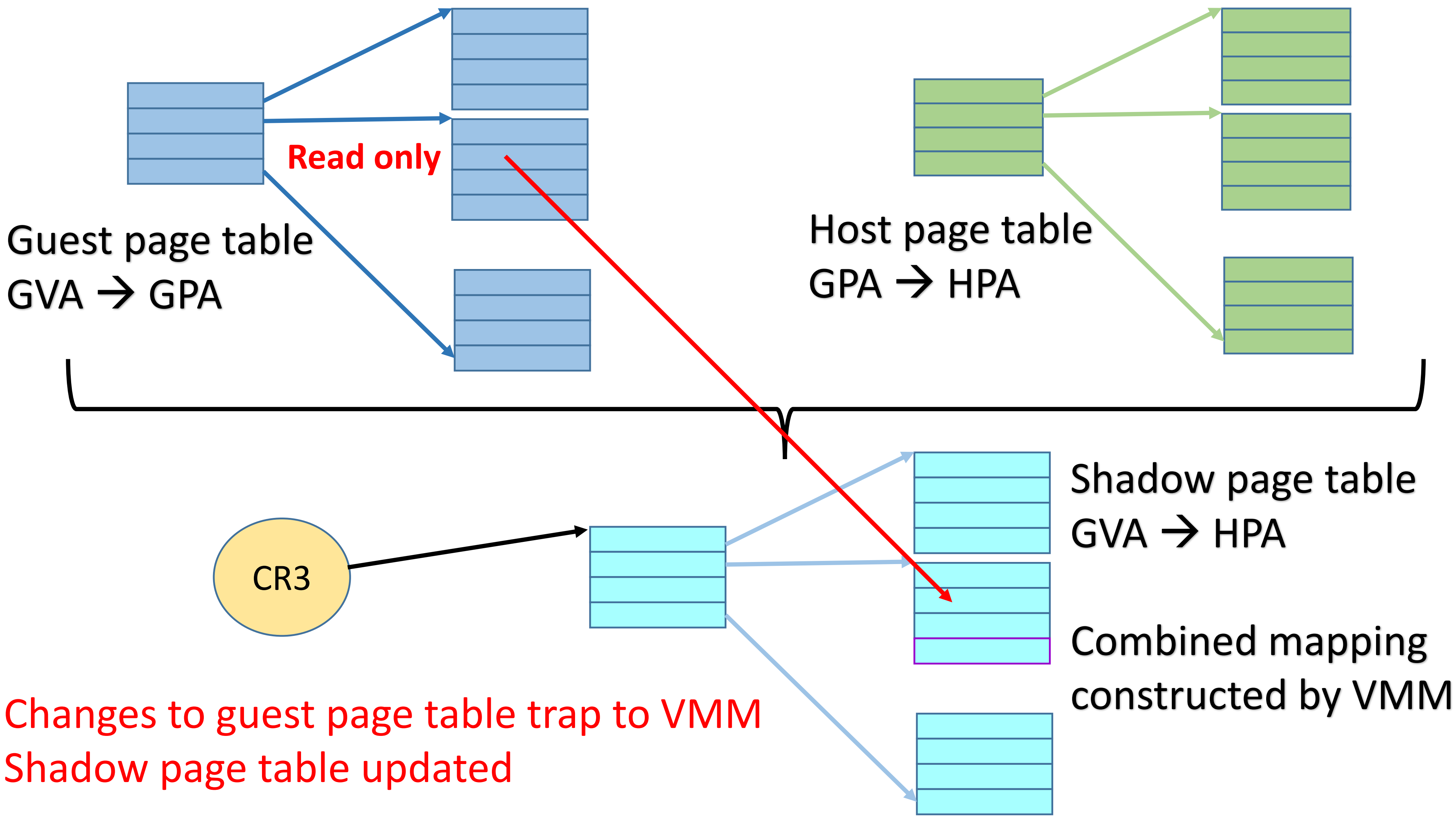
- Guest page table has GVA→GPA mapping
  - Each guest OS thinks it has access to all RAM starting at address 0
- VMM / Host OS has GPA→HPA mapping
  - Hypervisor knows mapping between host virtual address (HVA) and guest physical address (GPA), because it has setup its userspace memory as guest RAM
  - Host OS knows mapping from HVA to HPA for all processes, so it knows GPA→HPA
  - That is, guest "RAM" pages are userspace process pages that are distributed across host memory and host OS knows the physical locations of these guest "RAM" pages
- Which page table should MMU use?
- Shadow paging: VMM creates a combined mapping GVA→HPA and MMU is given a pointer to this page table
  - Used in VMWare workstation (full virtualization)
- Extended page tables (EPT): MMU hardware is aware of virtualization, takes pointers to two separate page tables
  - Used in QEMU/KVM

Extended page tables



- Page table walk by MMU: Start walking guest page table using GVA
- Guest PTE (for every level page table walk) gives GPA (cannot use GPA to access memory)
- Use GPA, walk host page table to find HPA, then access memory page, then next level access
- Every step in guest page table walk requires walking N-level host page table
- N-level page tables in guest/host result in page table walk of NXN memory accesses

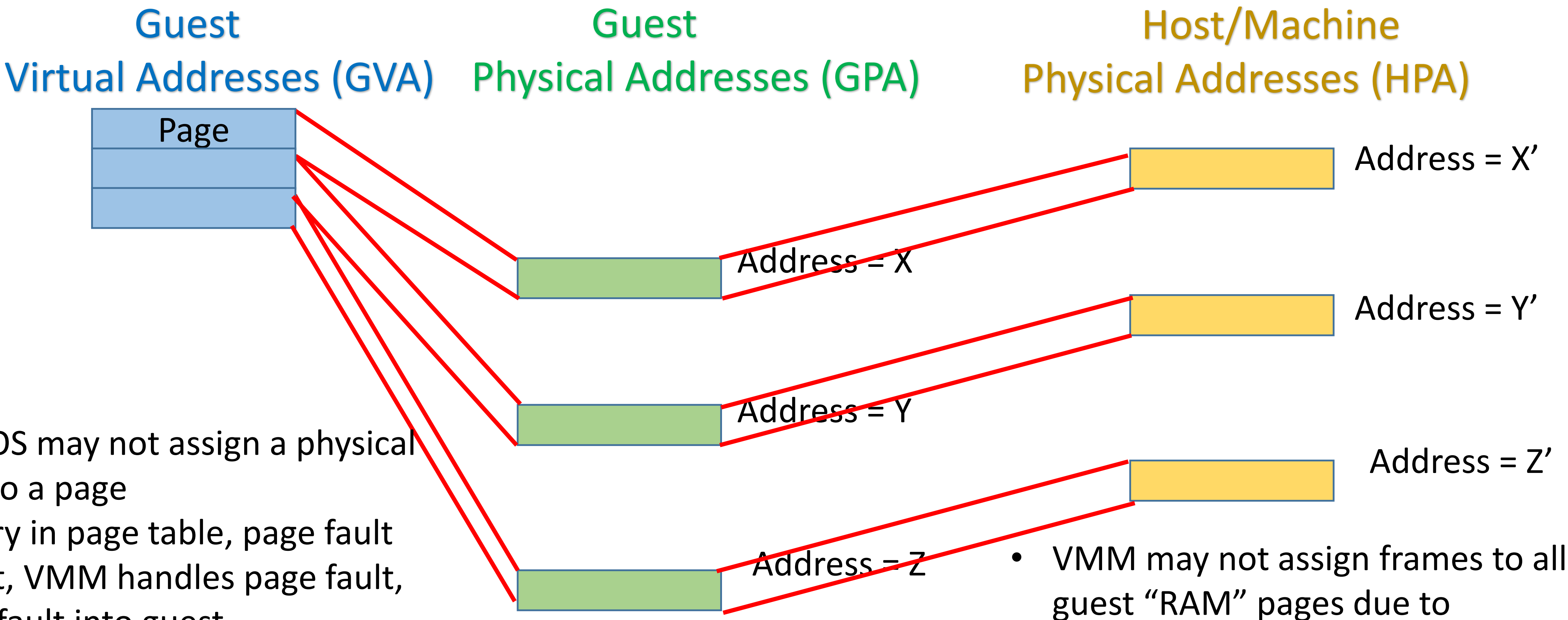
Shadow page tables



Maintaining shadow page tables

- Guest writes to CR3, privileged operation traps to VMM
  - VMM marks the guest page table pages as read-only
  - VMM constructs shadow page table, sets CR3 to it
- Shadow page table can be built on demand
  - Start with empty page table, add entries on page faults
- Guest changes page table, traps to VMM, shadow entry updated
- Guest OS keeps multiple page tables of active processes in memory
  - On context switch, new page table used, but old page table still in memory
  - What about shadow page tables? How many in memory?
- Many design choices exist
  - VMM can discard old shadow page table on context switch, and rebuild it later (overhead during context switch)
  - VMM can maintain multiple shadow page tables of active processes (overhead to track changes to all page table pages)

Demand Paging and Page Faults



- Guest OS may not assign a physical frame to a page
- No entry in page table, page fault
- VM exit, VMM handles page fault, injects fault into guest
- Guest assigns physical frame

- VMM may not assign frames to all guest "RAM" pages due to memory pressure
- Page fault is handled by VMM
- Guest OS not aware of this fault
- "Hidden" page faults

Memory reclamation technqiues

- VMM reclaims memory from guest "RAM" under memory pressure
- Uncooperative swapping: VMM reclaims some guest "RAM" pages and swaps them to disk
  - Page fault and memory assigned when guest accesses the page
  - May hurt performance, important pages may be swapped to disk
- Ballooning: VMM opens dummy device, requests pages from VM ("inflating the balloon"), then swaps these pages out
  - Since pages assigned to a device, pages not used by other processes in guest
  - Cooperative, guest can assign free pages
  - Lesser impact on performance, but reclamation takes time, requires guest changes
- Memory sharing: memory pages with identical content across VMs (OS images, zero pages etc.) shared between VMs
  - One physical frame mapped into page tables of multiple guests
  - Scan pages periodically to compute and match hash-based similarity of pages

Summary

- Memory virtualization
  - Extended page tables
  - Shadow page tables
- Memory reclamation techniques
  - Uncooperative swapping
  - Ballooning
  - Memory sharing

- "Memory Resource Management in VMware ESX Server", Carl A. Waldspurger.