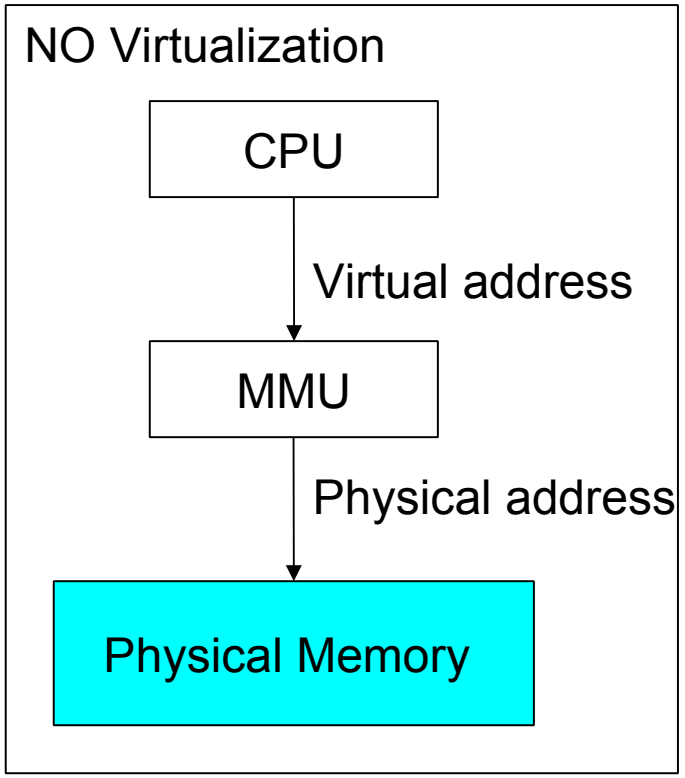
KVM MMU Virtualization

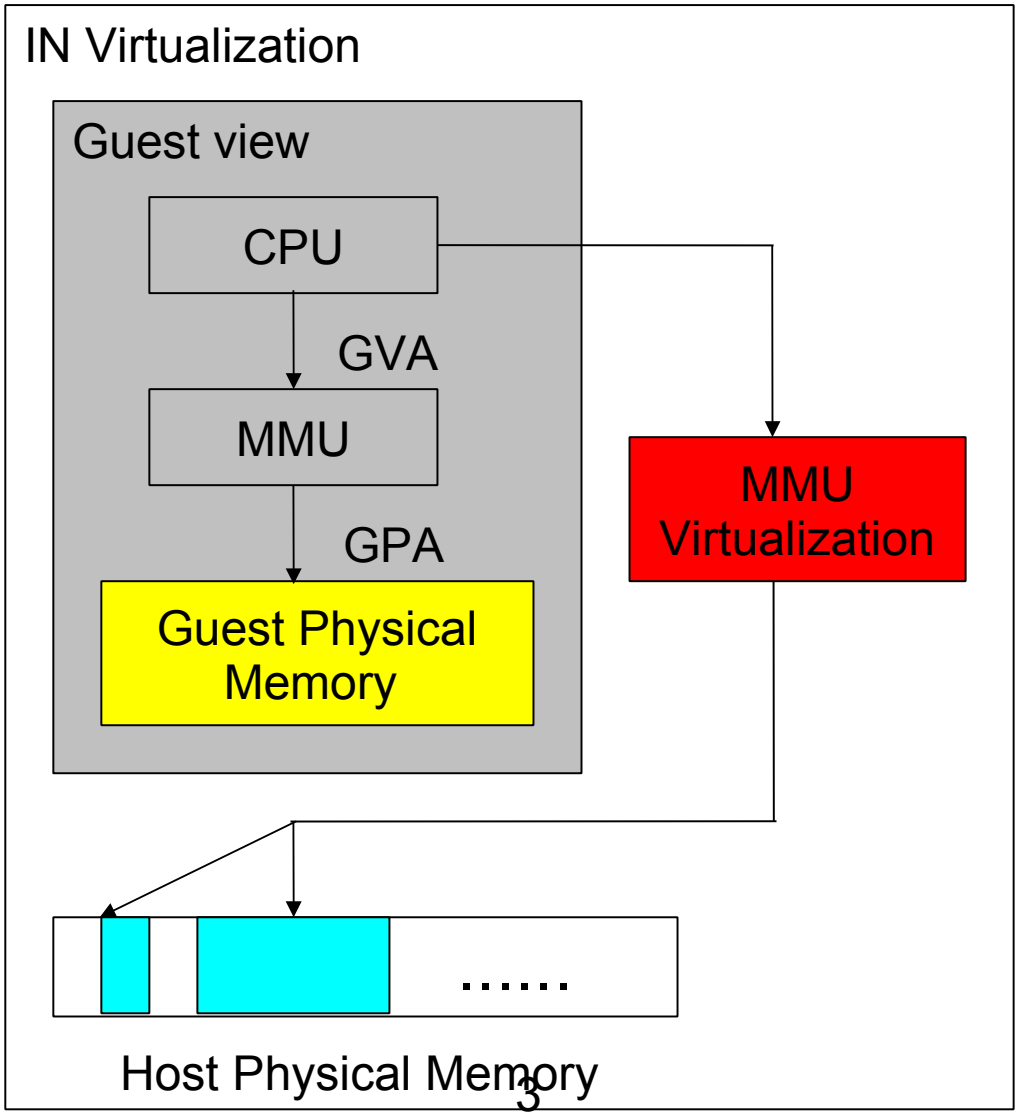
Xiao Guangrong
<xiaoguangrong@cn.fujitsu.com>

Index

- What is MMU Virtualization?
- How to implement MMU Virtualization?
- How to optimize MMU Virtualization?
- What will I do?

What is MMU Virtualization





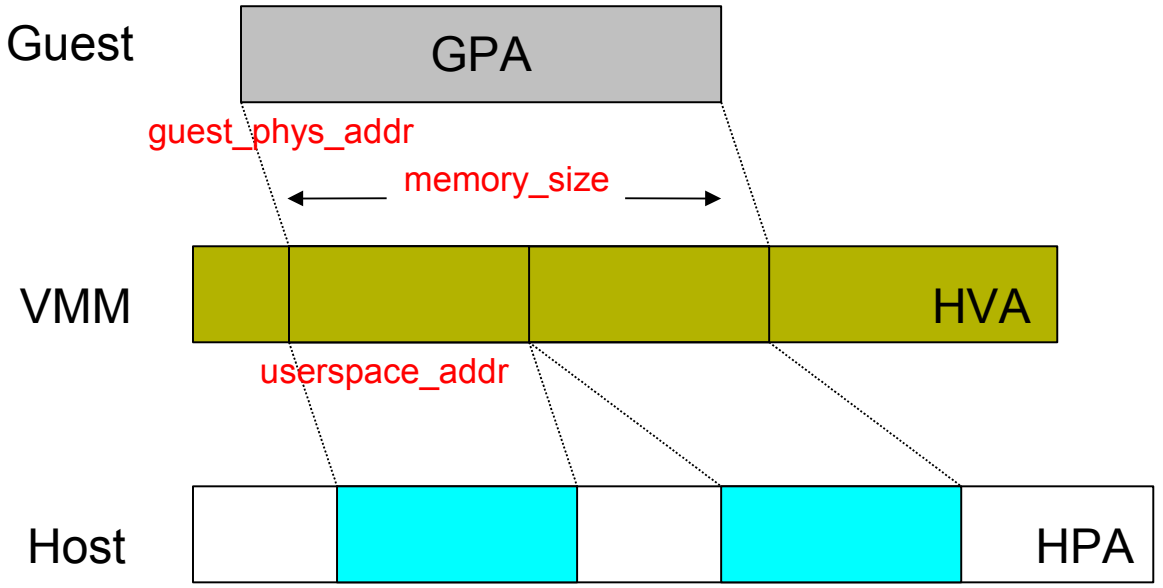
GVA: guest virtual address
GPA: guest physical address

The functions of MMU Virtualization

- Translate guest physical address to the specified host physical address
- Control the memory access permission
  - R/W, NX, U/S
- Track Accessed/Dirty bits of guest page table

GFN to PFN in KVM

- Use ioctl(fd, KVM_SET_USER_MEMORY_REGION, kvm_userspace_memory_region) to register guest physical memory
  - guest_phys_addr, memory_size, userspace_addr

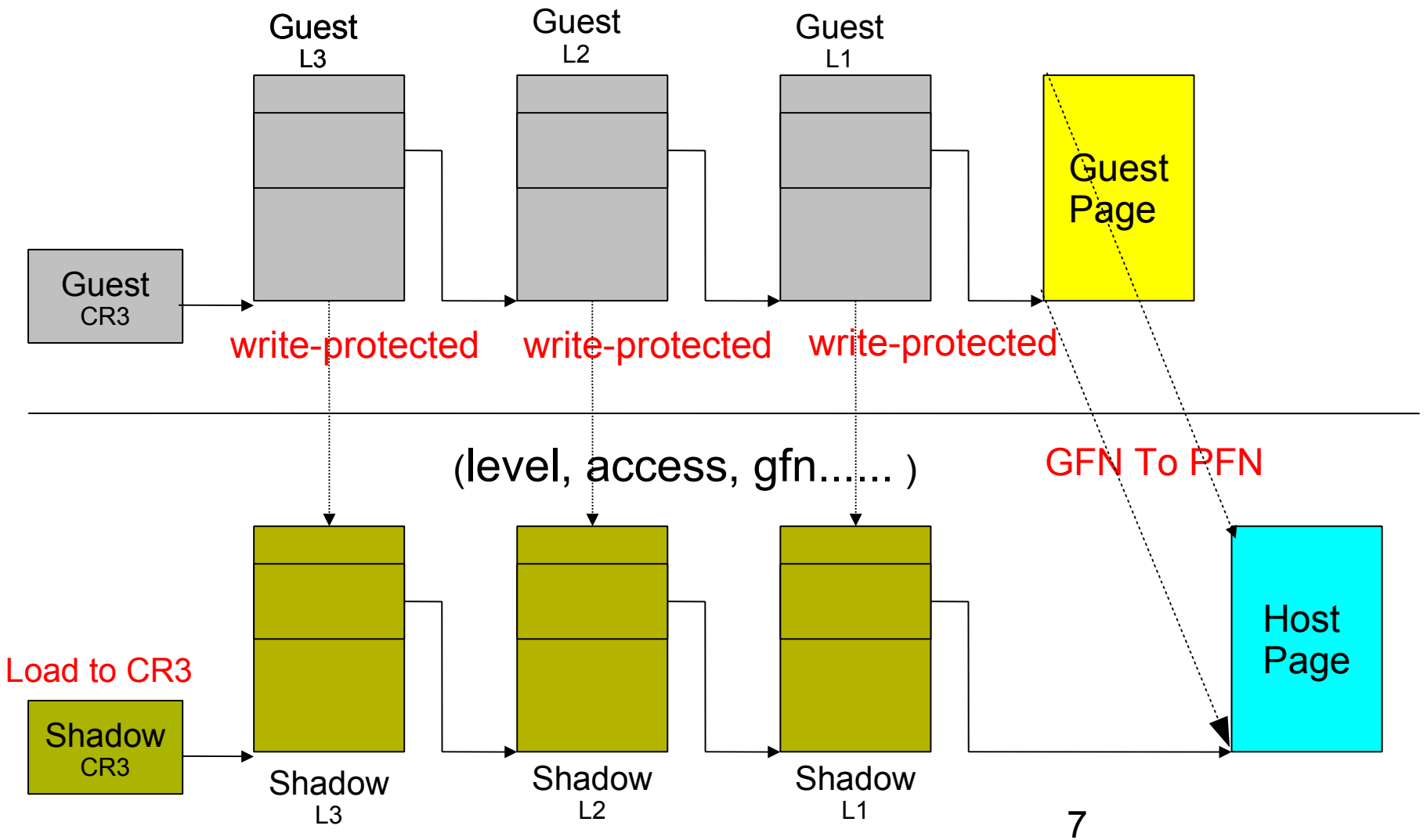


GFN: Guest Frame Number
PFN: Host Page Frame Number
GPA: Guest Physical Address
HVA: Host Virtual Address
HPA: Host Physical Address

How to implement MMU Virtualization?

- Soft MMU
  - Implemented by software
- Hard MMU
  - Supported by hardware
    - NPT on SVM from AMD
    - EPT on VMX from Intel

Soft MMU: overview



Soft MMU: how to implement (1)

- Lower the access permission to keep consistency:
  - Guest pages table are write-protected to keep the consistency between shadow page table and guest page table
  - Remove the writable bit of shadow PTE if the Dirty bit of guest PTE is not set

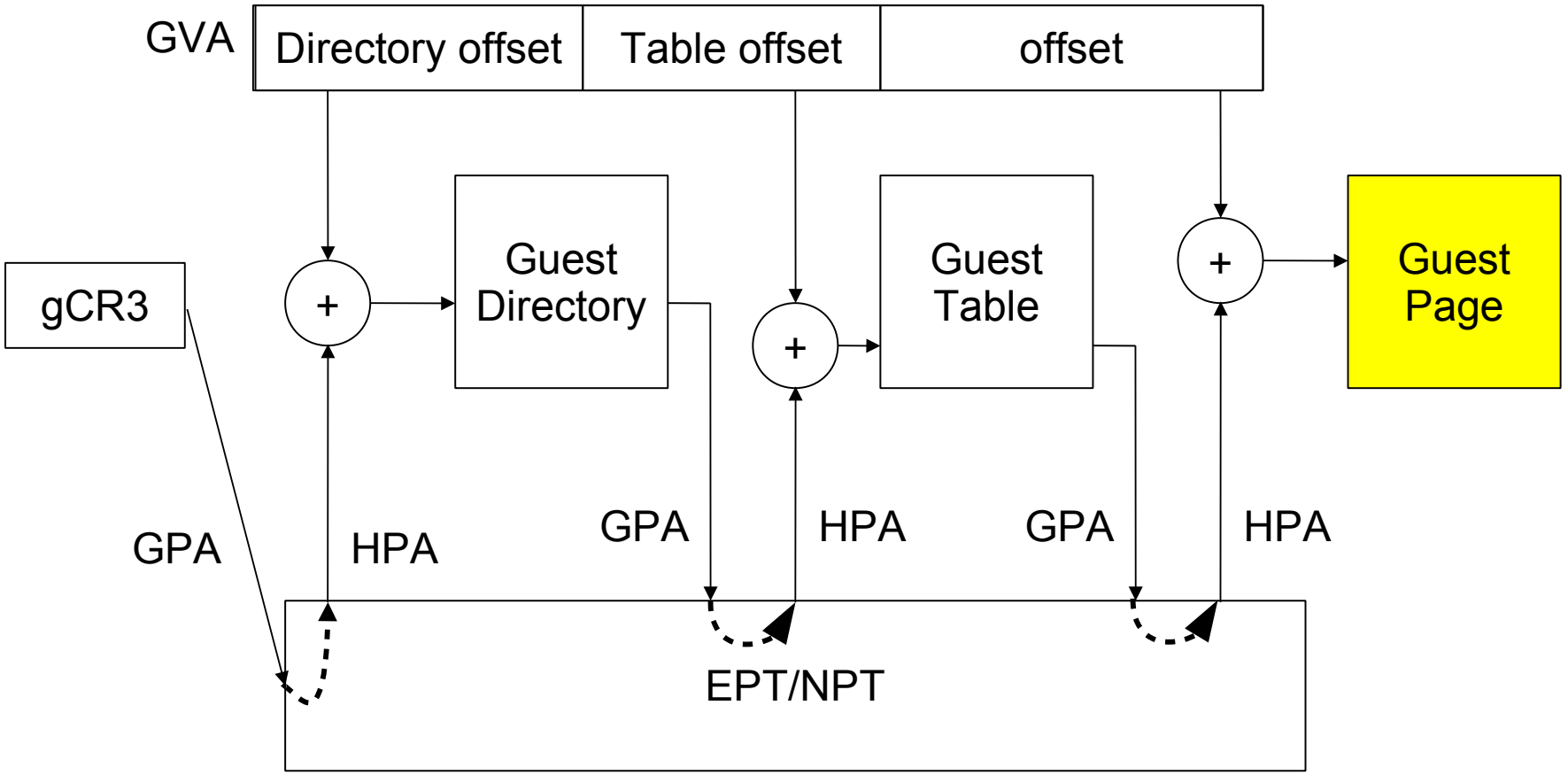
Soft MMU: how to implement (2)

- The guest events we should intercept:
  - Page fault
    - If guest page table is invalid, inject this page fault to guest
    - Atomically set A/D bit of guest page table
    - Setup/fix the mapping of shadow page
    - Sync shadow page table and guest page table if it is a write page fault
  - Load CR3
    - Flush TLB
    - Load the new shadow page
  - INVLPG
    - Flush TLB

Hard MMU

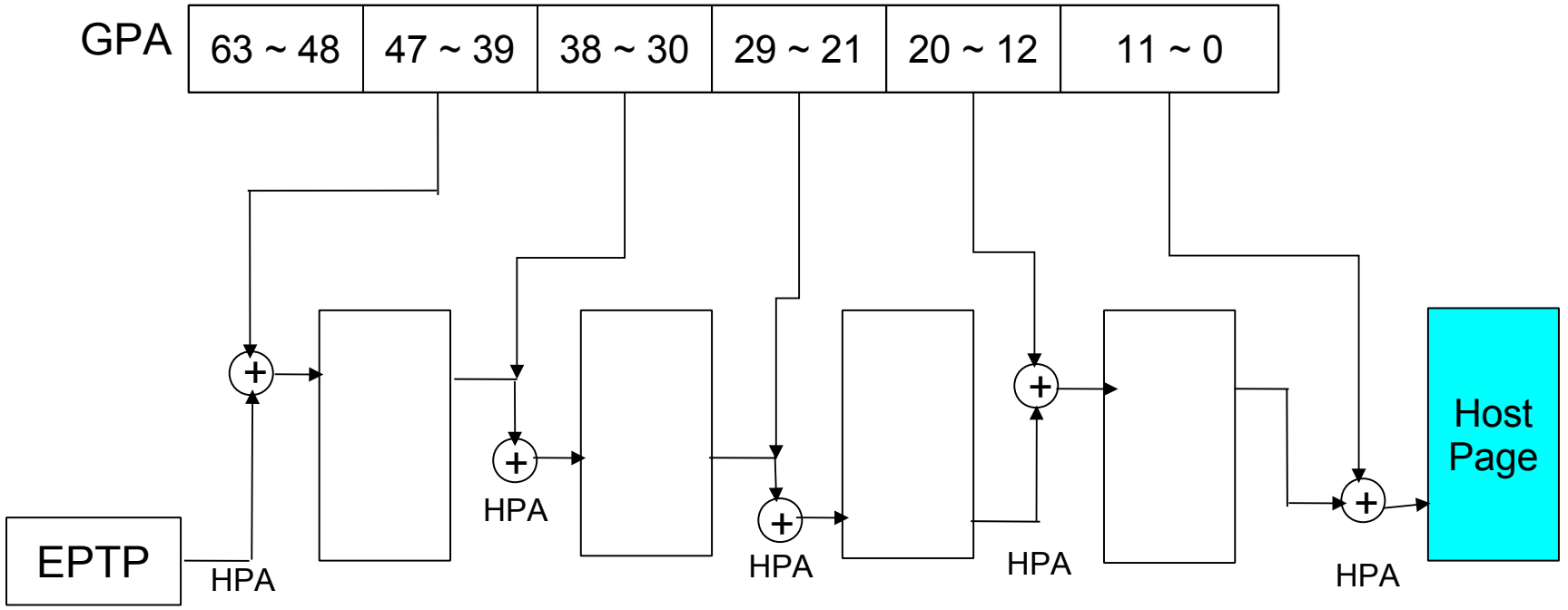
- EPT/NPT functions:
  - The new layer to translate guest physical address to host physical address
  - Use EPT/NPT for all guest physical address accesses, including MMIO and guest page table walking
- Comparing to Soft MMU:
  - It's simple and reduces lots of VM exits...

Hard MMU: overview



Hard MMU: translate GPA to HPA

EPT/NPT



How to optimize MMU Virtualization?

- 1. No-trap for non-present PTEs
- 2. Unsync shadow pages
- 3. PTE prefetch
- 4. KSM
- 5. THP

1. No-trap for non-present PTEs

- Objective
  - Reduce VM exits
- In soft MMU, if the page-fault is caused by PTE not present(PFEC.P = 0), it is not intercepted by the host
- It only works on VMX
  - VMCB.PFEC_MASK = 1, MCB.PFEC_MATCH = 1, VMCB.EXCEPTION_BITMAP.#PF = 1

PTE: Page Table Entry
FPEC: Page Fault Error Code

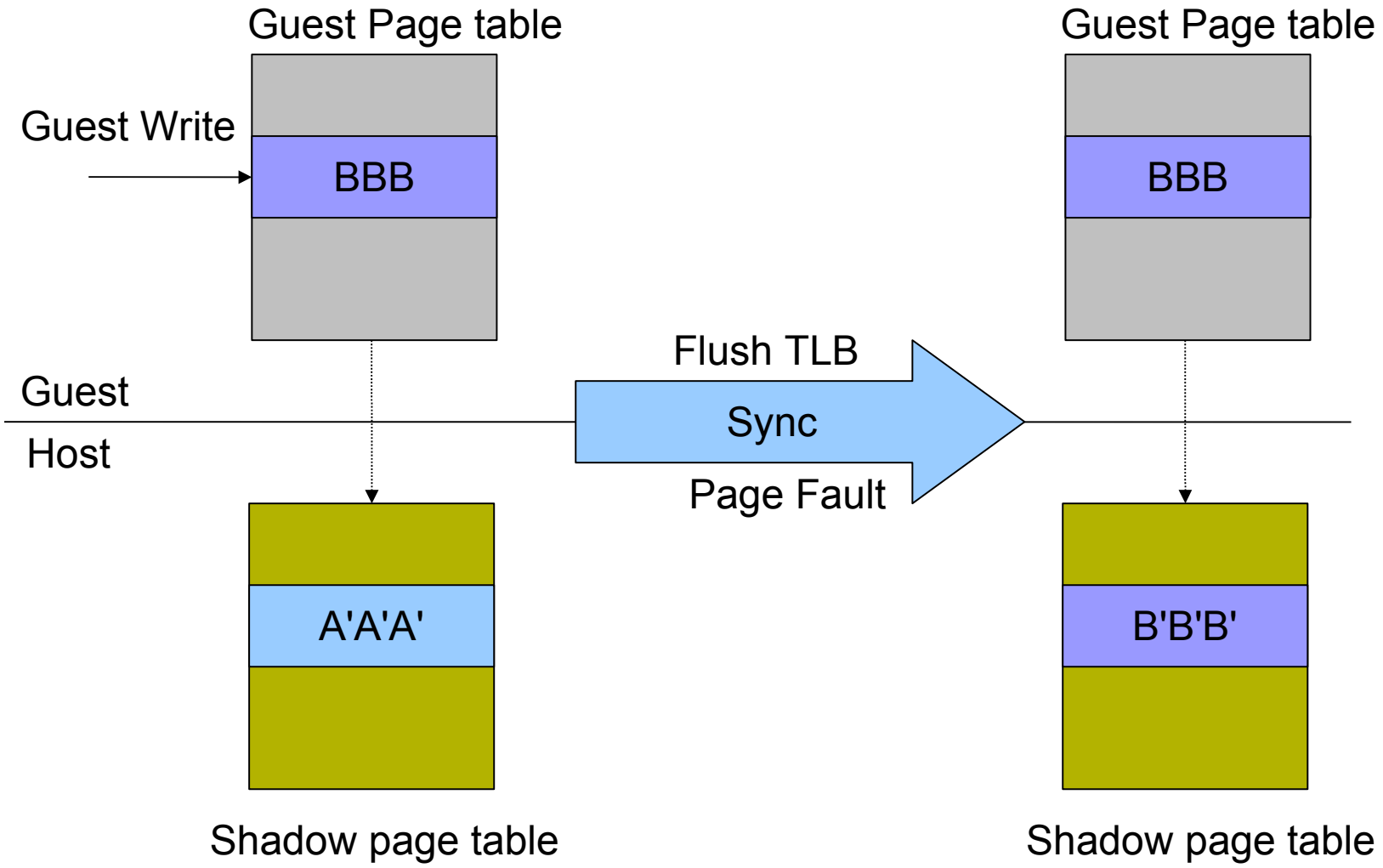
2. Unsync shadow pages

- Objective
  - Reduce VM exits
- Background:
  - In soft MMU, in order to keep consistency, we need to write-protect the guest page table
- This mechanism can let guest writes these pages directly

2. Unsync shadow pages

- For the performance reason, we allow the guest page table to be writable if and only if the page is the last level page-structure (Level 1)
- Base on TLB rules
  - We need to flush TLB to ensure the translation use the modified page structures, then we can Intercept the TLB flush operations and sync shadow pages
  - Sometimes, TLB is not need to be flushed(old PTE.P = 0, PTE.A=0, raise access permission), then it can be synced though page fault

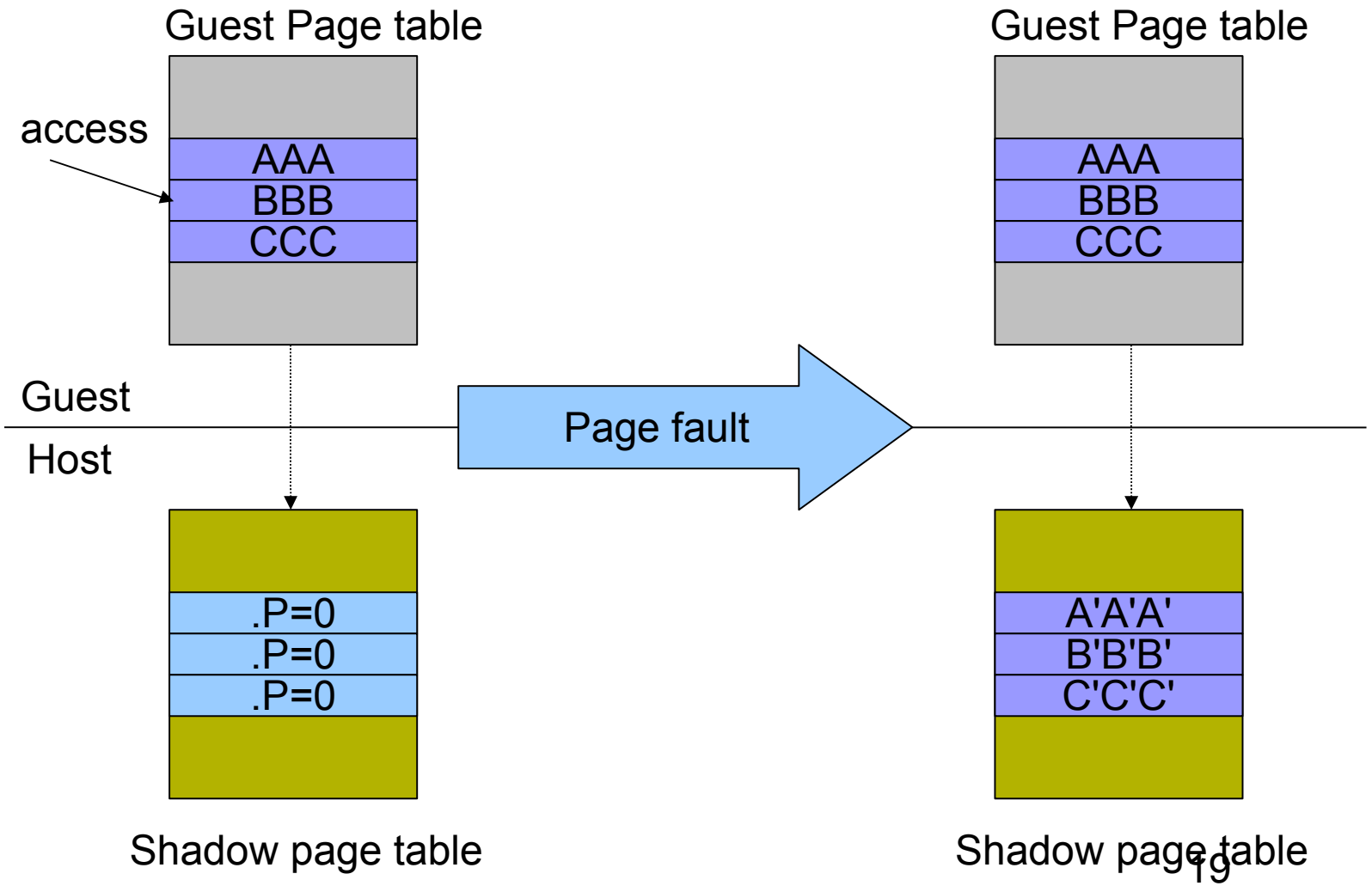
2. Unsync shadow pages



3. PTE prefetch

- Objective
  - Reduce VM exits
- When #PF occurs, we prefetch other invalid shadow PTEs, so if these PTEs are accessed, the #PF can be avoided

3. PTE prefetch

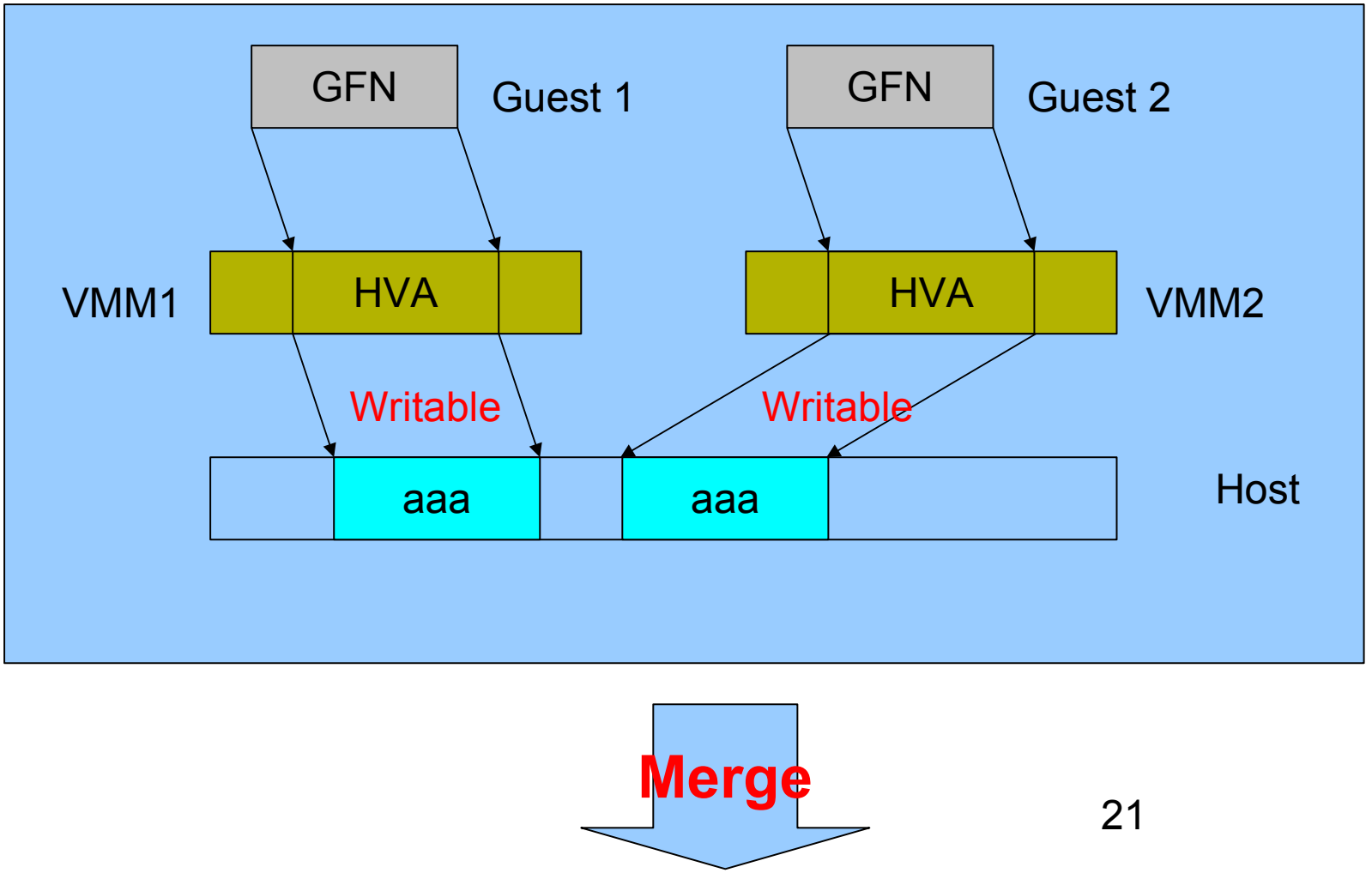


4. KSM

- Objective
  - Saving memory
- Kernel Shared Memory
- It's a memory-saving de-duplication feature

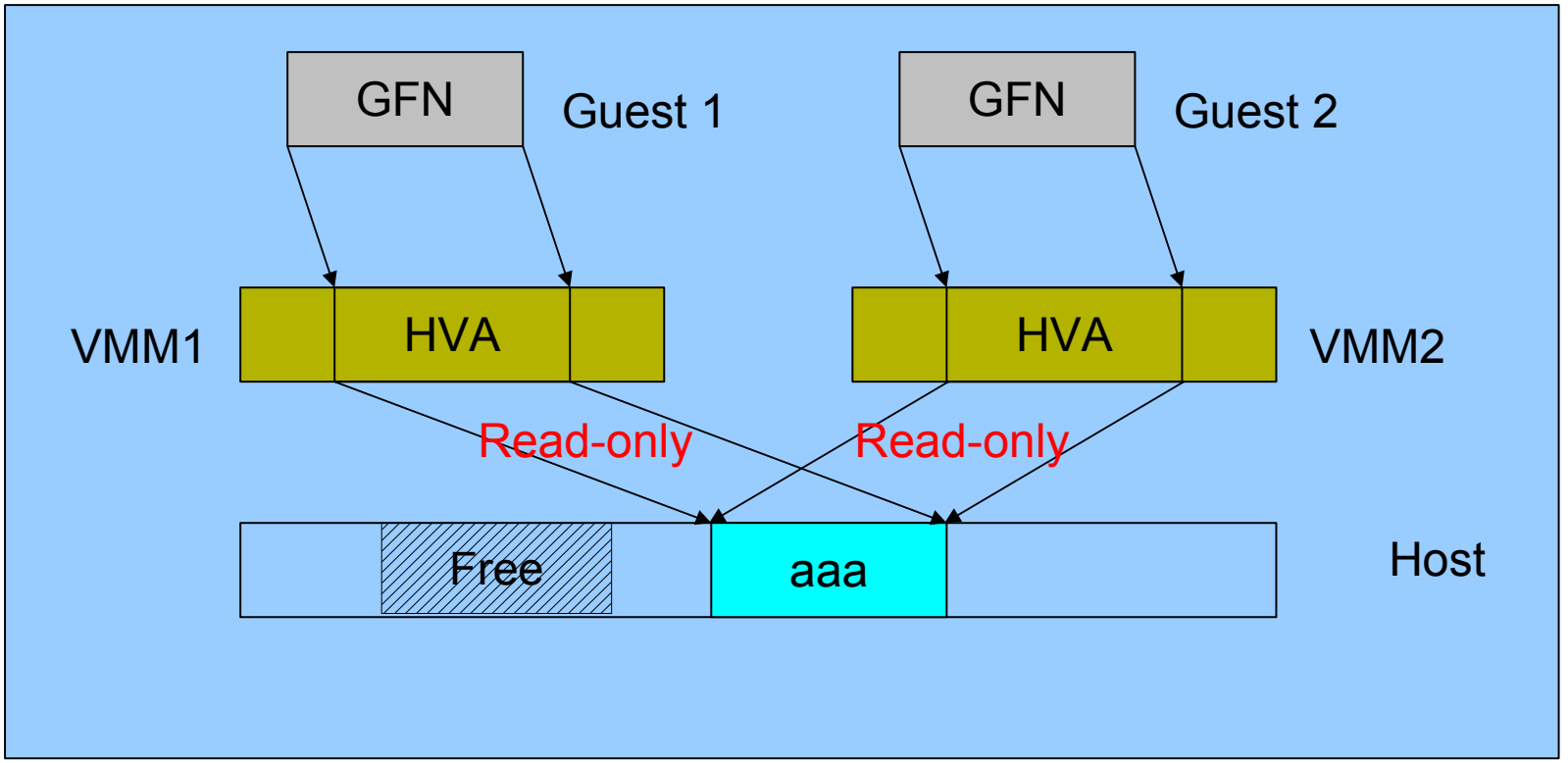
4. KSM

Origin state:



4. KSM

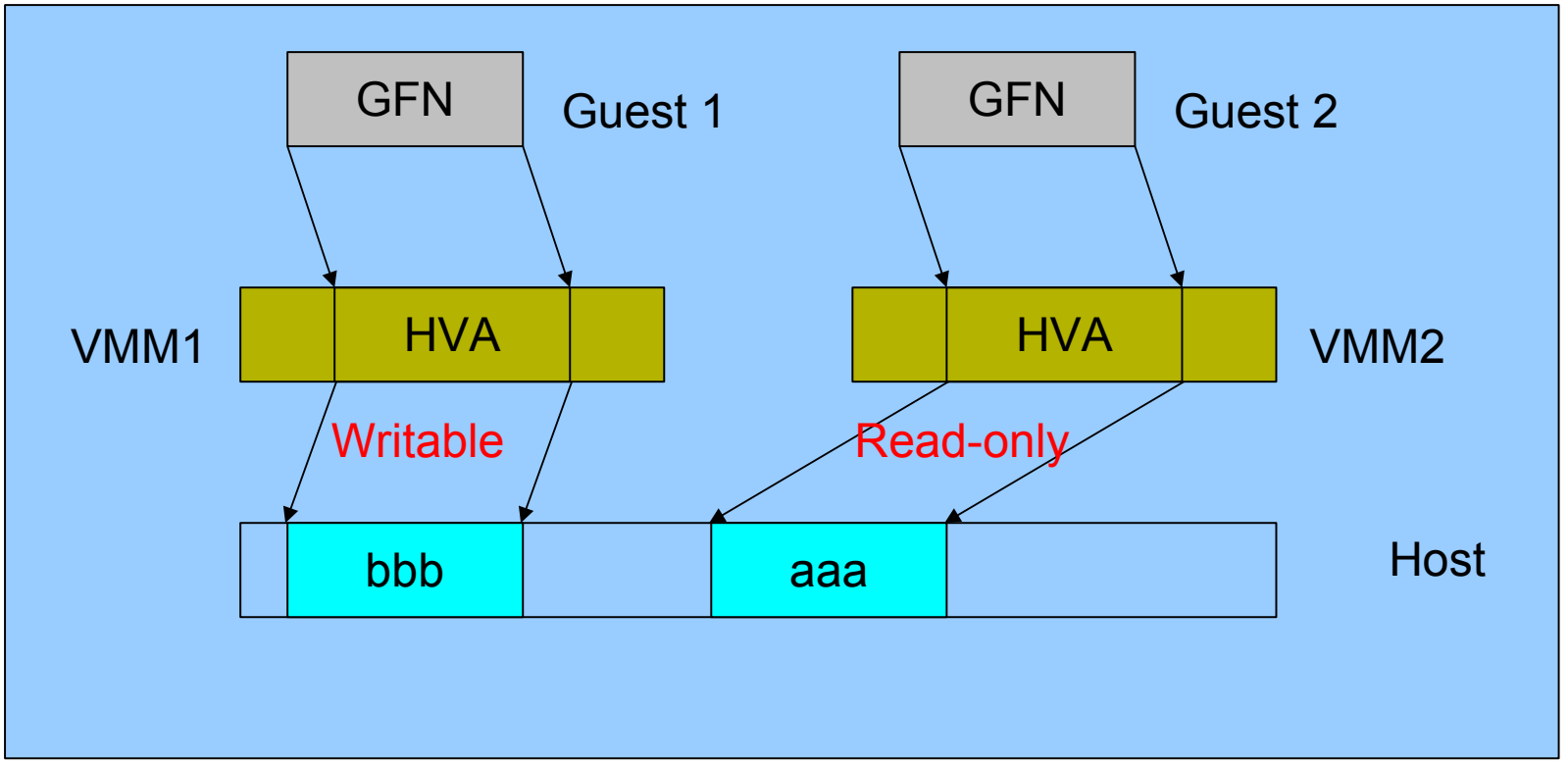
Merge state:



If Guest 1 writes the page **COW**

4. KSM

Cow state:



5. THP

- Objective
  - Reduce memory accesses while guest and EPT/NPT page table walking
  - Improve TLB usage
- Transparent Hugepage
- Both host and guest can use huge page automatically

What will I do

- Lockless MMU
  - Feature:
    - Avoid big lock for the whole MMU
      - Lock shadow page instead of whole MMU
    - Lockless to walk shadow page
      - Use RCU to avoid shadow page to be freed
    - Lockless to update shadow PTE
      - Cmpxchg...

What will I do

- Lockless MMU
  - Advantage:
    - Allow VCPU to run concurrently on MMU path
    - Good preparing work for KSM to track dirty page which is mapped by shadow page table
    - Good preparing work for LRU algorithm of MMU page eviction

Questions?

Thanks!