Lecture 6 — CS 695

(17) reminder: Assignment 1 due 29th Jan. Monday.

① p → v ~ page walk, / or in-built Kernel fn's.

② LKMs — KVM

debugfs
sysfs
procfs
} — cgroups config & mgmt.

---

memory virtualization with VMs.

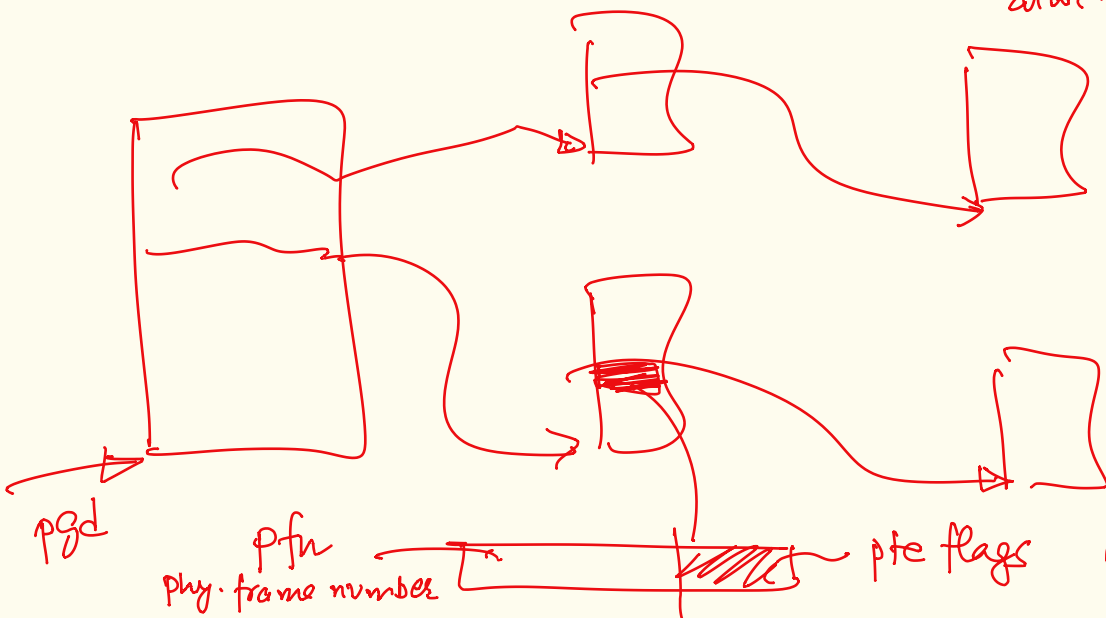
native: virtual memory / address space ← abstraction

V2P

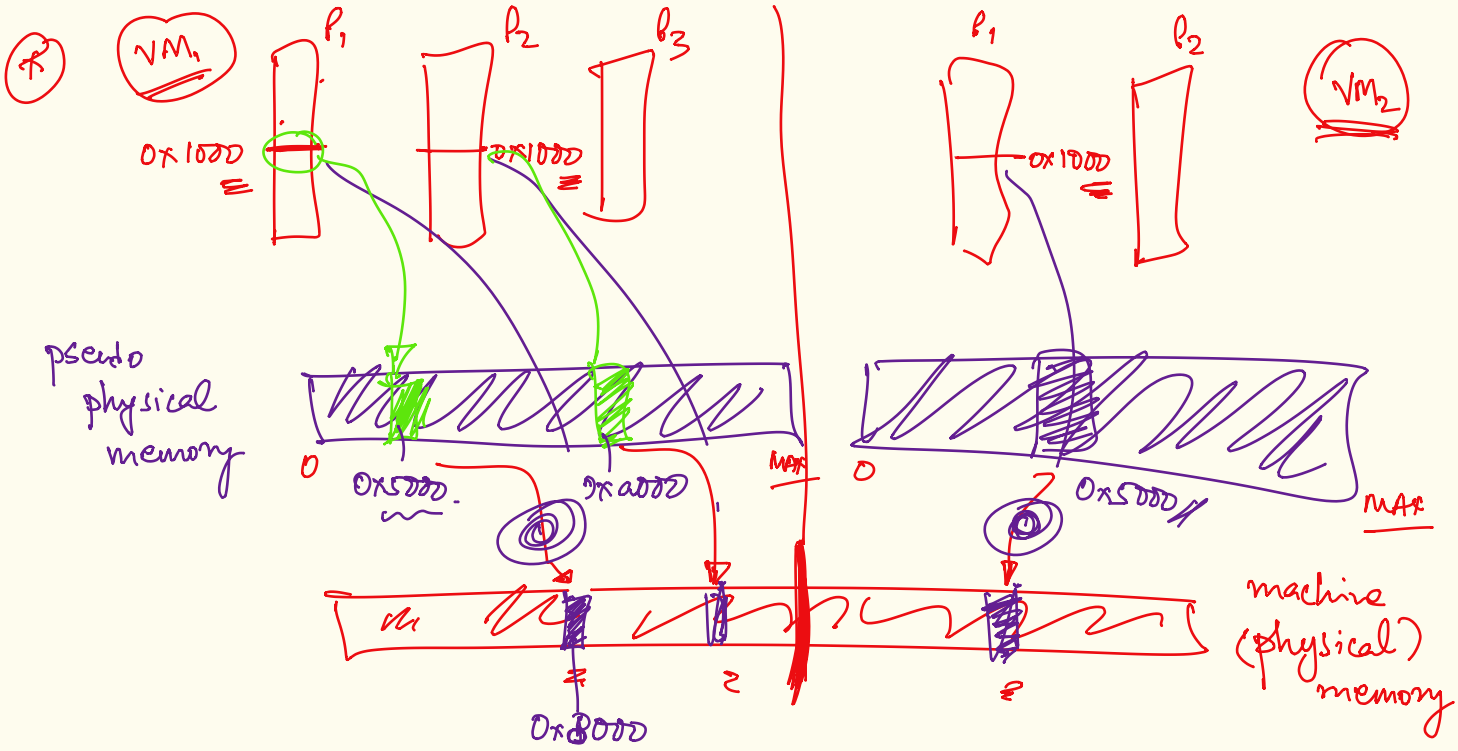
- 0-starting
- linear & contiguous
- per-process
- "full" addressability

through V2P — isolation — permissions

via page tables. — elastic provisioning of memory alloc.



- 2-level
- 4-level page tables.
- 4KB
- 4MB pages.



gva → ppa → mpa

guest virtual address

V2P — guest-managed

p2m — hypervisor managed

if. Static cannot generalize

(1) to on-demand & elastic provisioning.

(2) nested page tables — two

Constraint : MMU single page table walk.

VM1: gva_1, gva_2, gva_3 → ppa_1 → mpa

VM2: gva_1, gva_2 → ppa_2 → mpa

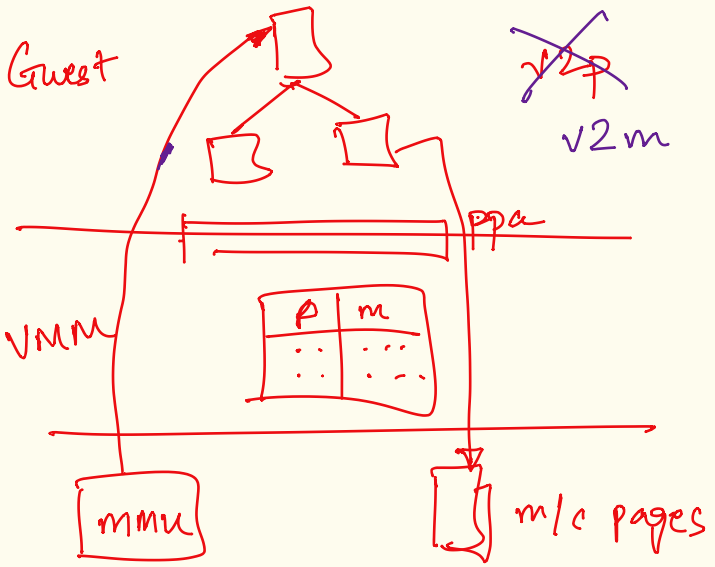
paging — paging

gva - guest virtual addr.

ppa - pseudo physical addr.

mpa - machine phy. / real address.

~~Soln~~ Soln 1: Direct Mapping (Para-virtualized approach)



v2p p2m

v2m

(1) maintain a single v2m mapping

(2) mmu walks v2m for Consistent translations.

(Q) how to build the v2m table?

v2p p2m

(*) eg: page fault in the guest

(1) gva find ppa for page table update.

- update mapping via a hypercall

update_pgtable (gva, ppa, pgd)

VMM: ppa → mpa pgd ↓ mpa

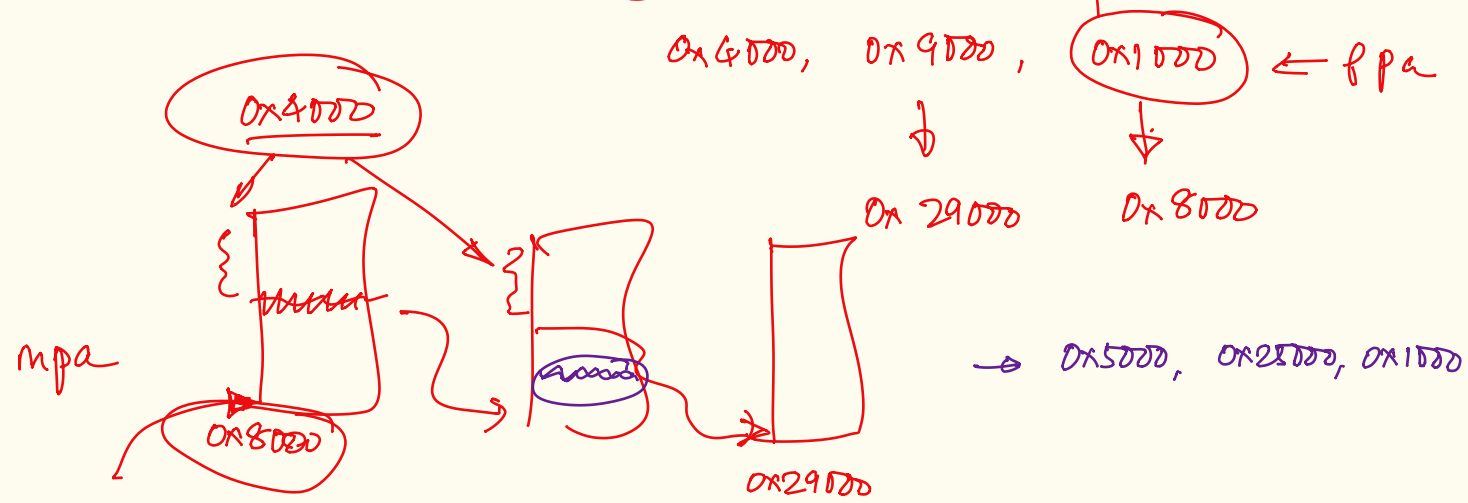
use pgd_{mpa} to add gva → mpa entry.

(2) load pg-table of a process

update_CR3 (pgd) ← ppa ← hypercall

vmm: CR3 ← mpa $_{pgd}$ (p→m)

③ update_pgtable (gva, ppa, pgd)



Key steps: ① guest does not update pg tables directly. only via hypercalls.

② all page table pages are marked read-only.

Solution 2: Shadow Paging (full virtualized VMs)