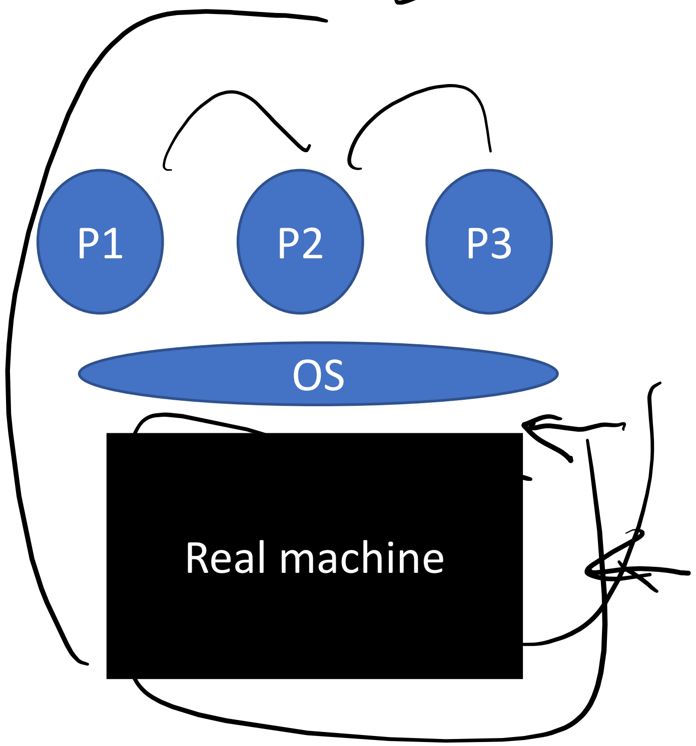
Virtual Machines (VM)

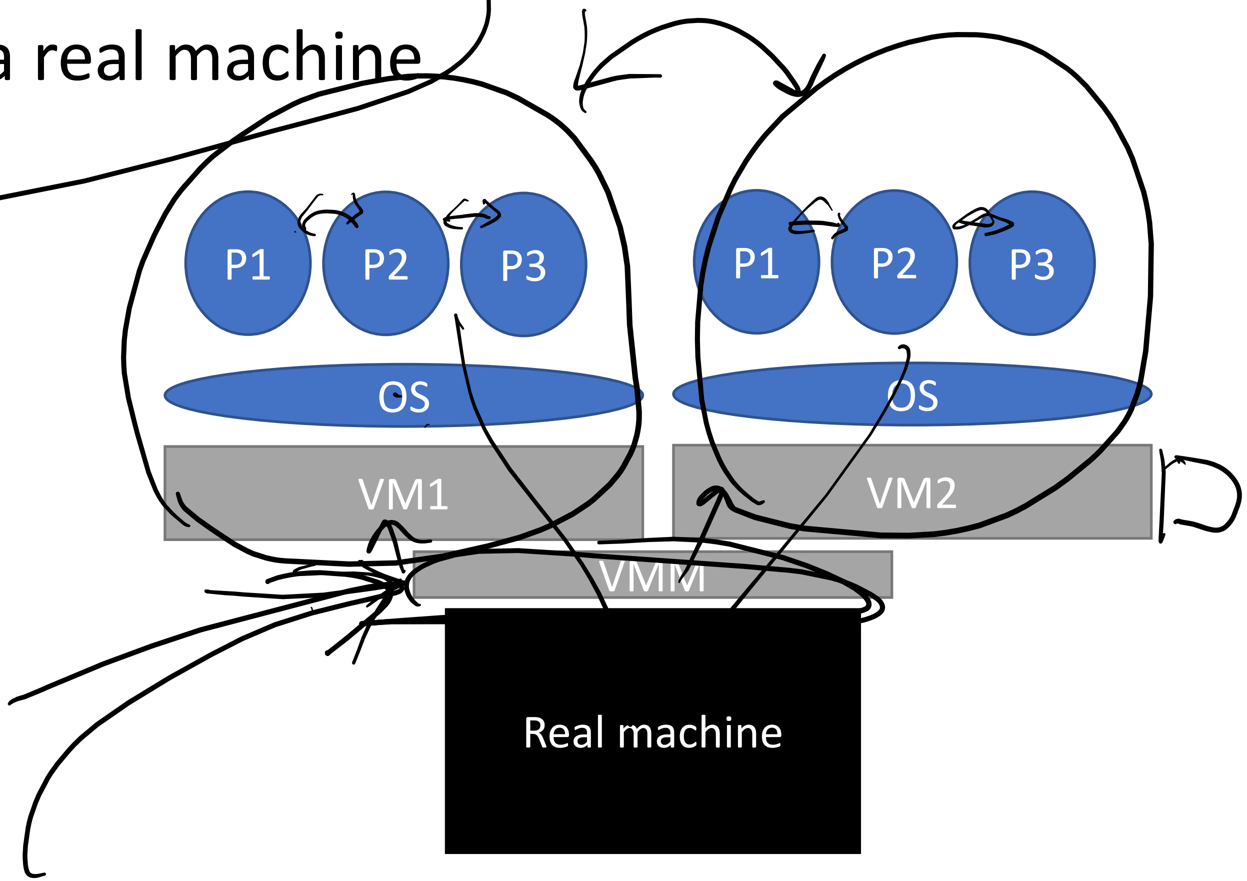
Background of Disco and its authors

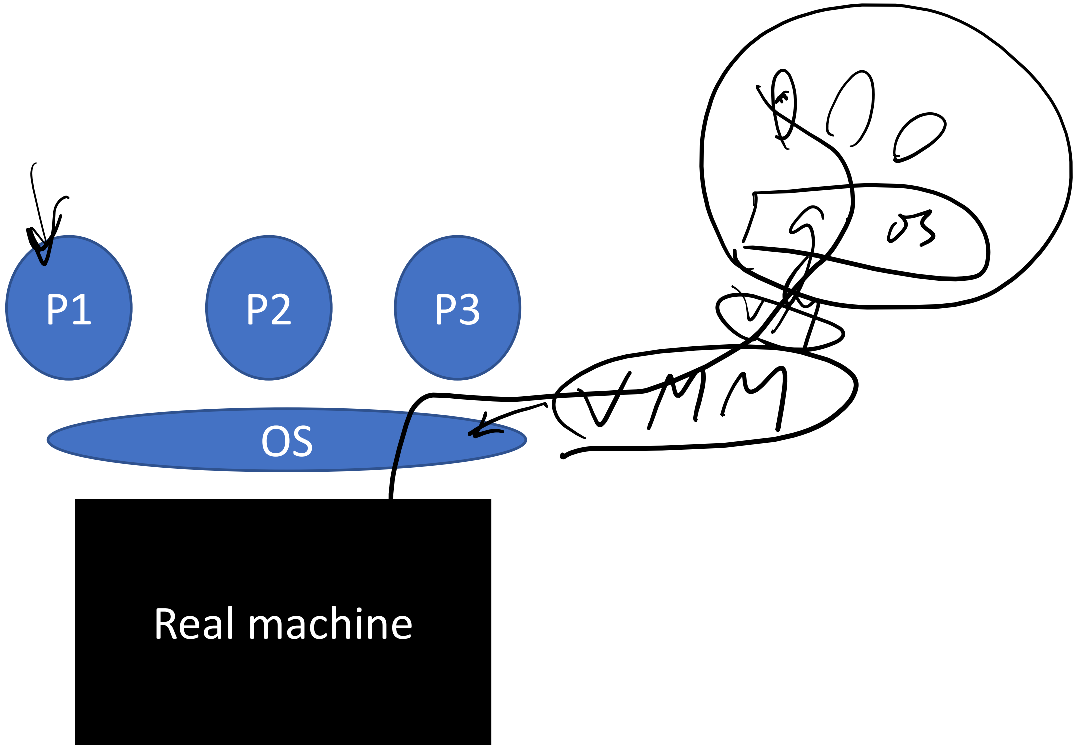
Background of Xen

Definition of VM

- An isolated, efficient duplicate of a real machine





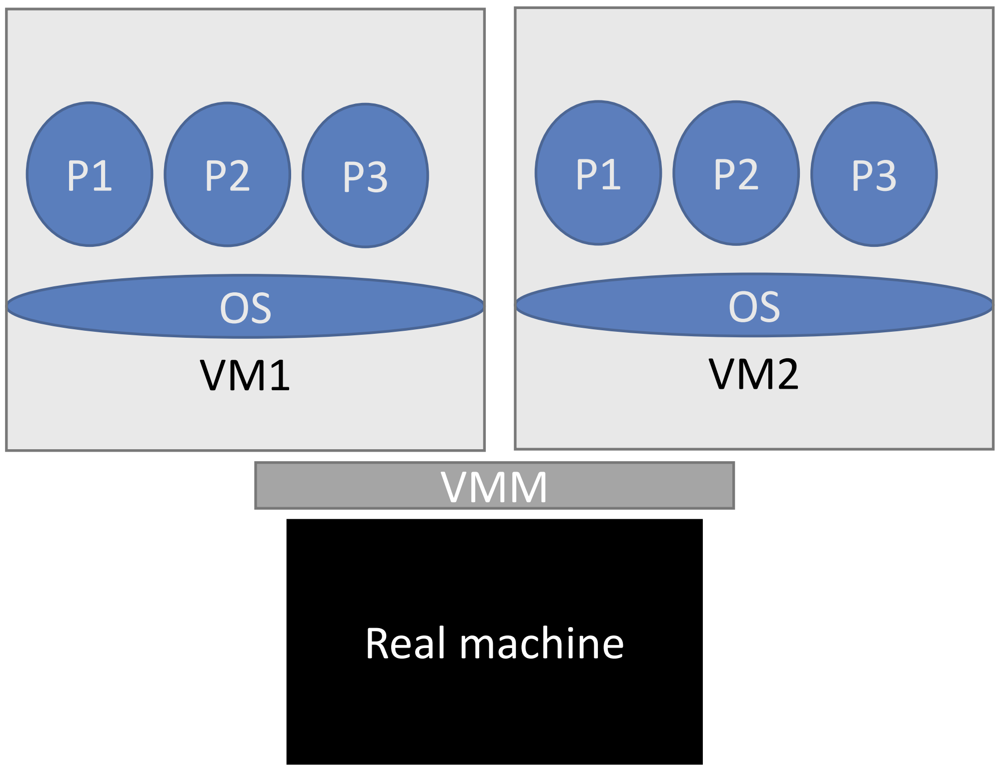

P1
P2
P3
OS
Real machine
OS
VMM

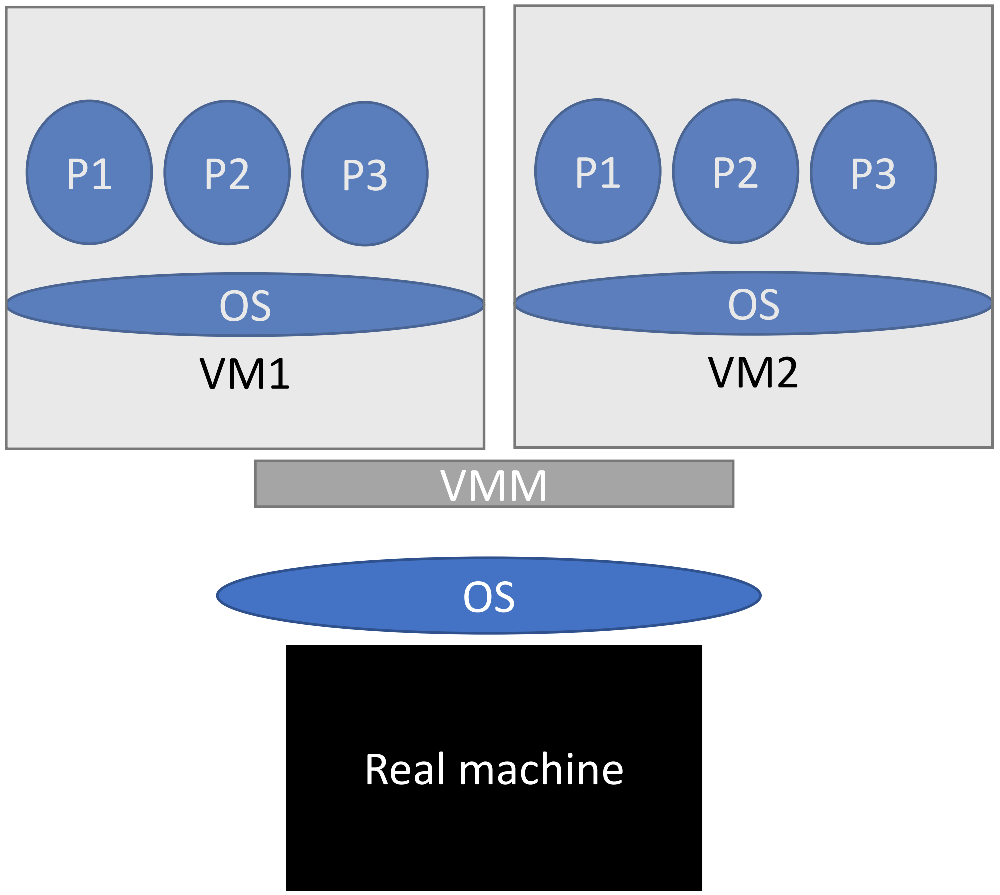
Why do we use VM now?

- OS flexibility
- Data center
  - Resource utility
  - Security and performance isolation

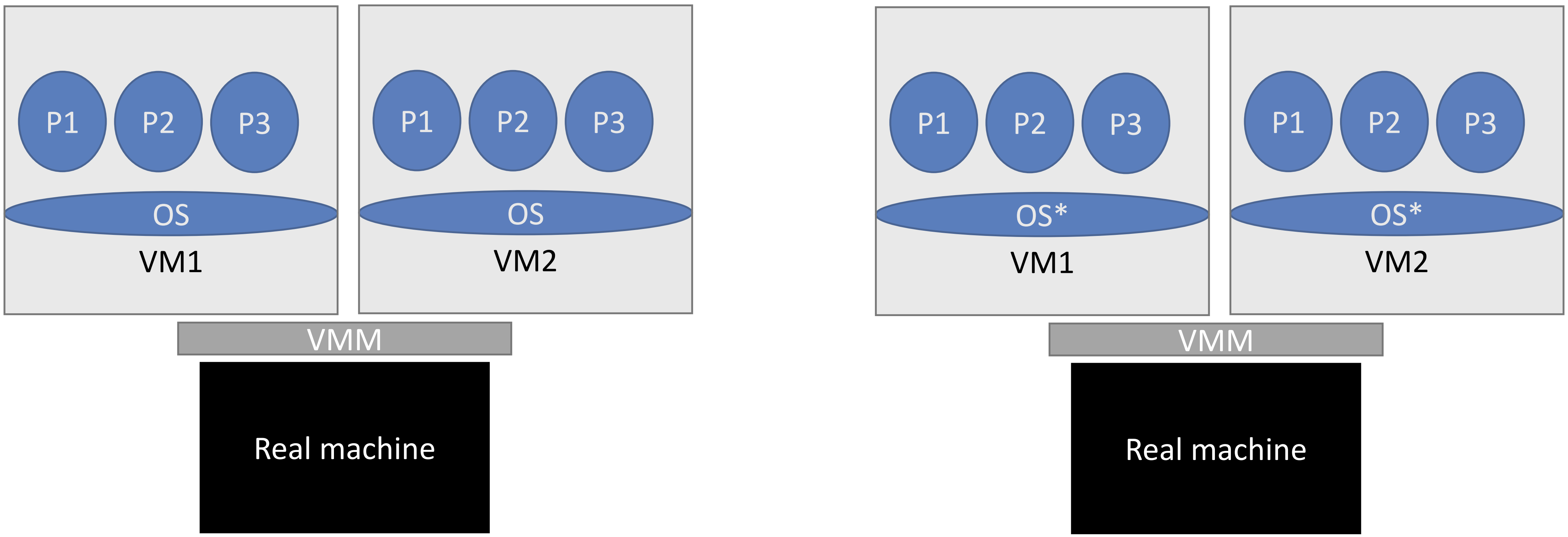
Why for Disco?

Different types of virtual machine

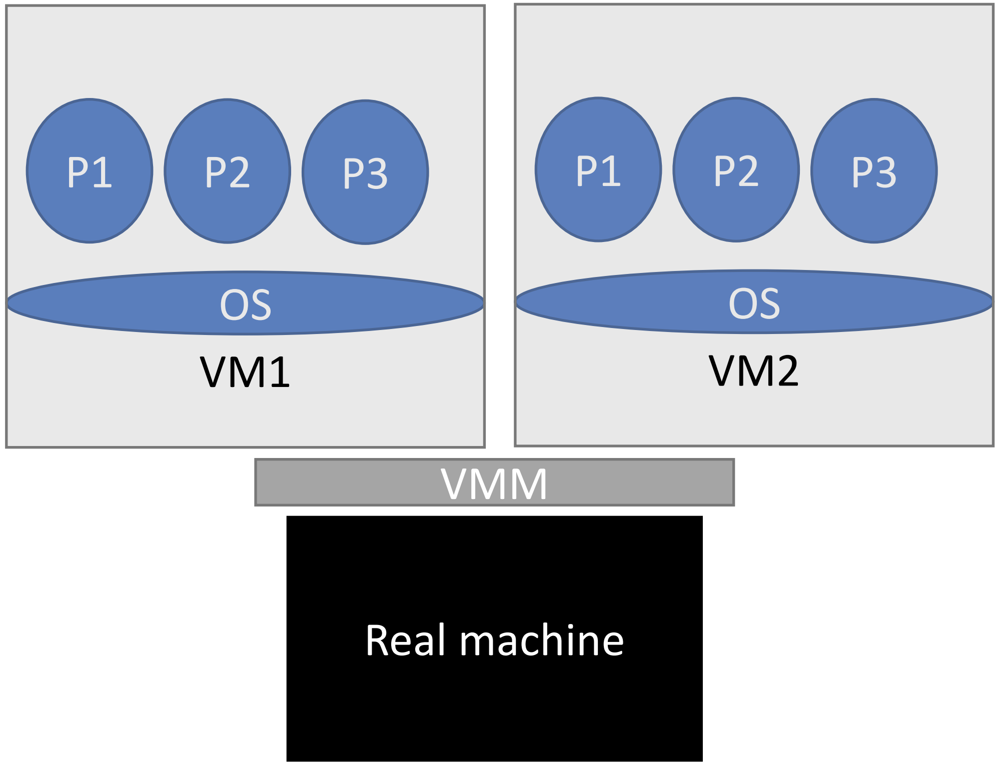




Different types of virtual machine

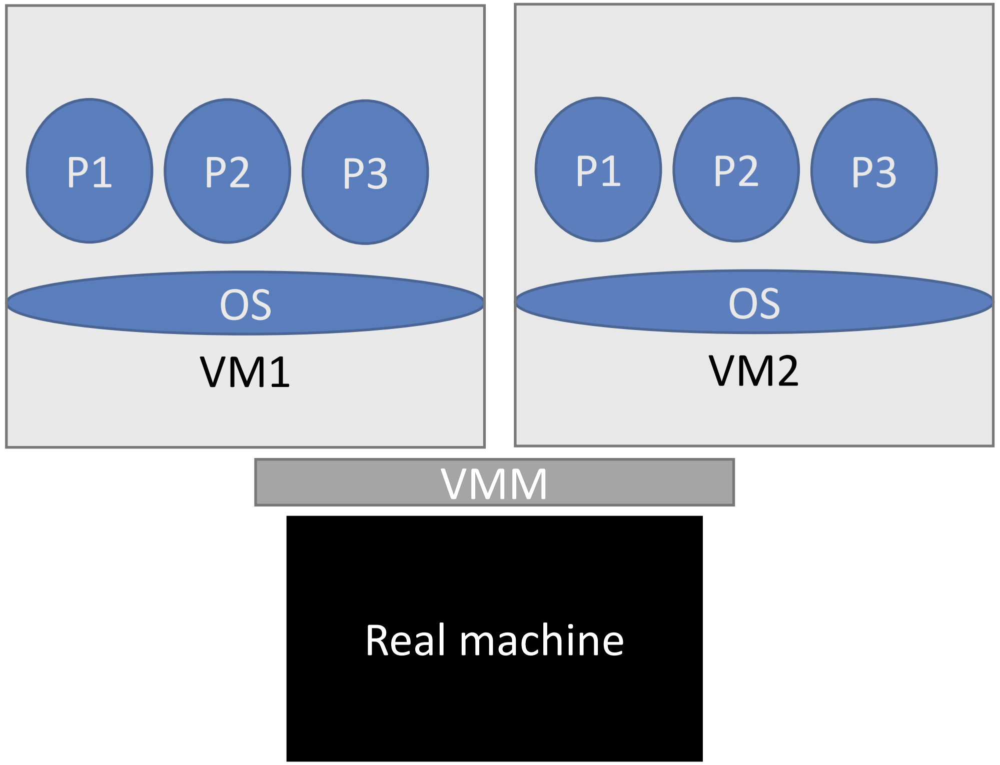


Challenge of virtualization



- How to isolate?
  - What if OS in VM1 get all CPU time?
  - What if OS in VM2 get all physical memory?
- How to be efficient?

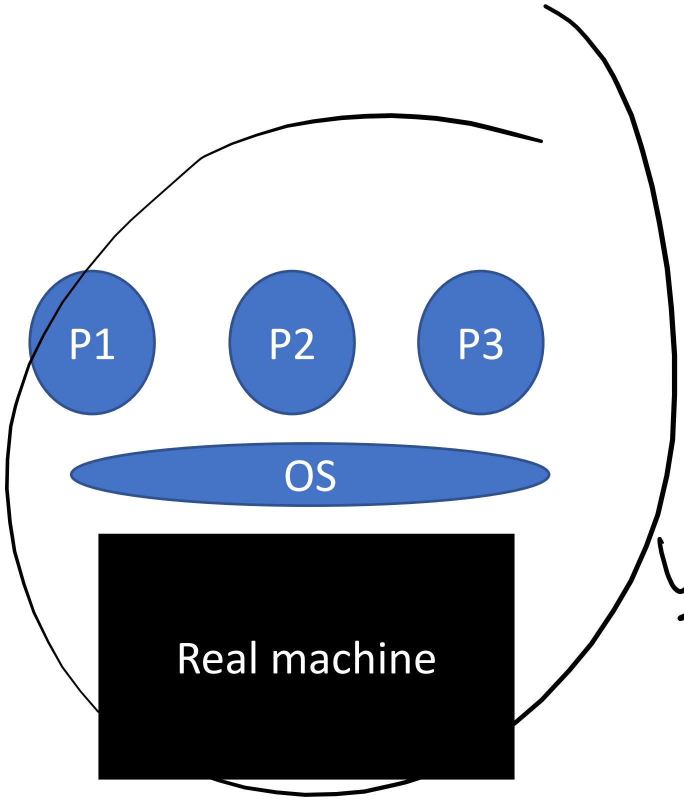
Challenge of virtualization



- How to isolate?
  - What if OS in VM1 get all CPU time?
  - What if OS in VM2 get all physical memory?
- How to be efficient?
  - All but privileged instructions are directly executed on real machine without VMM stepping in

How to virtualize CPU in VMM

CPU usage isolation non-virtualized world



Non-preemptive CPU scheduling

yield()

Preemptive

timer interrupt

user

OS

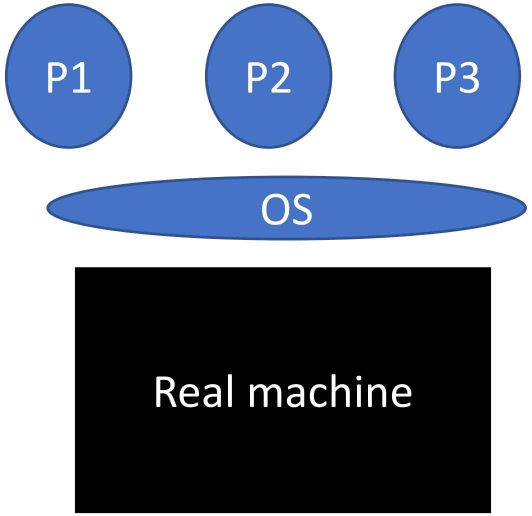
Kernel

Register

hello

timer

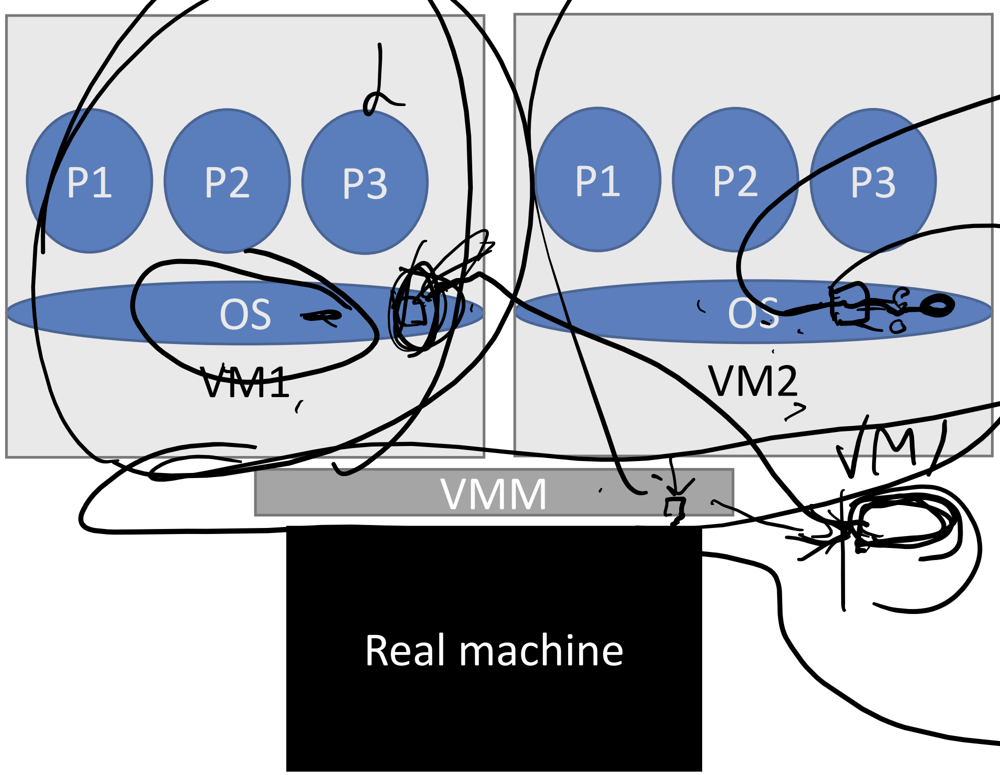
How did OS manage CPU across processes?



- Timer interrupt!

- At a timer interrupt
  - Hardware checks IHTB register
  - Interrupt handler table is looked up
  - OS's interrupt handler is executed
  - CPU switch …

How can VMM manage CPU across VMs?



- What would happen at a timer interrupt?
  - Hardware checks IHTB register
  - Interrupt handler table is looked up
  - ~~OS's interrupt handler is executed~~
  - CPU switch …

Whose table?
OS1? OS2? Or …?

Solution

- Accessing interrupt handler table base register is a privileged instruction!
  - Executing this instruction from non-kernel mode will raise exceptions!

- Can we run OS outside kernel mode, so that …?

Solution

PC add r1, r2, r3

VA → PA

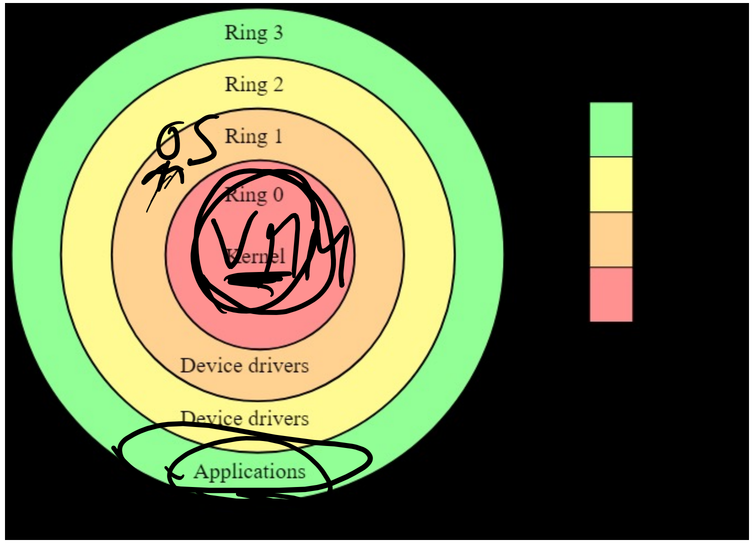
- Run VMM in the most priviledged mode (ring 0)
- Run kernel in ring 1
- Run applications still in ring 3

0010

10

- VMM register its interrupt handler table
- When OS tries to register …

- At a timer interrupt …



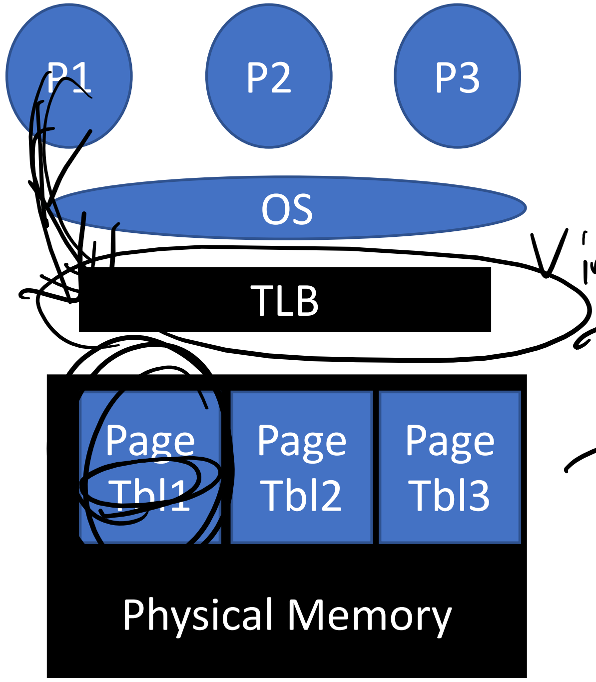
How to virtualize memory in VMM

How did OS manage memory across processes?

MIPS

- Virtual Address, Physical Address

- How to translate virtual address to phy. address?
  - TLB look up
  - At TLB miss — OS Handler
    - Page table look up
    - (if valid & present) load page table entry to TLB — Vir A → phy A
  - Redo TLB look up



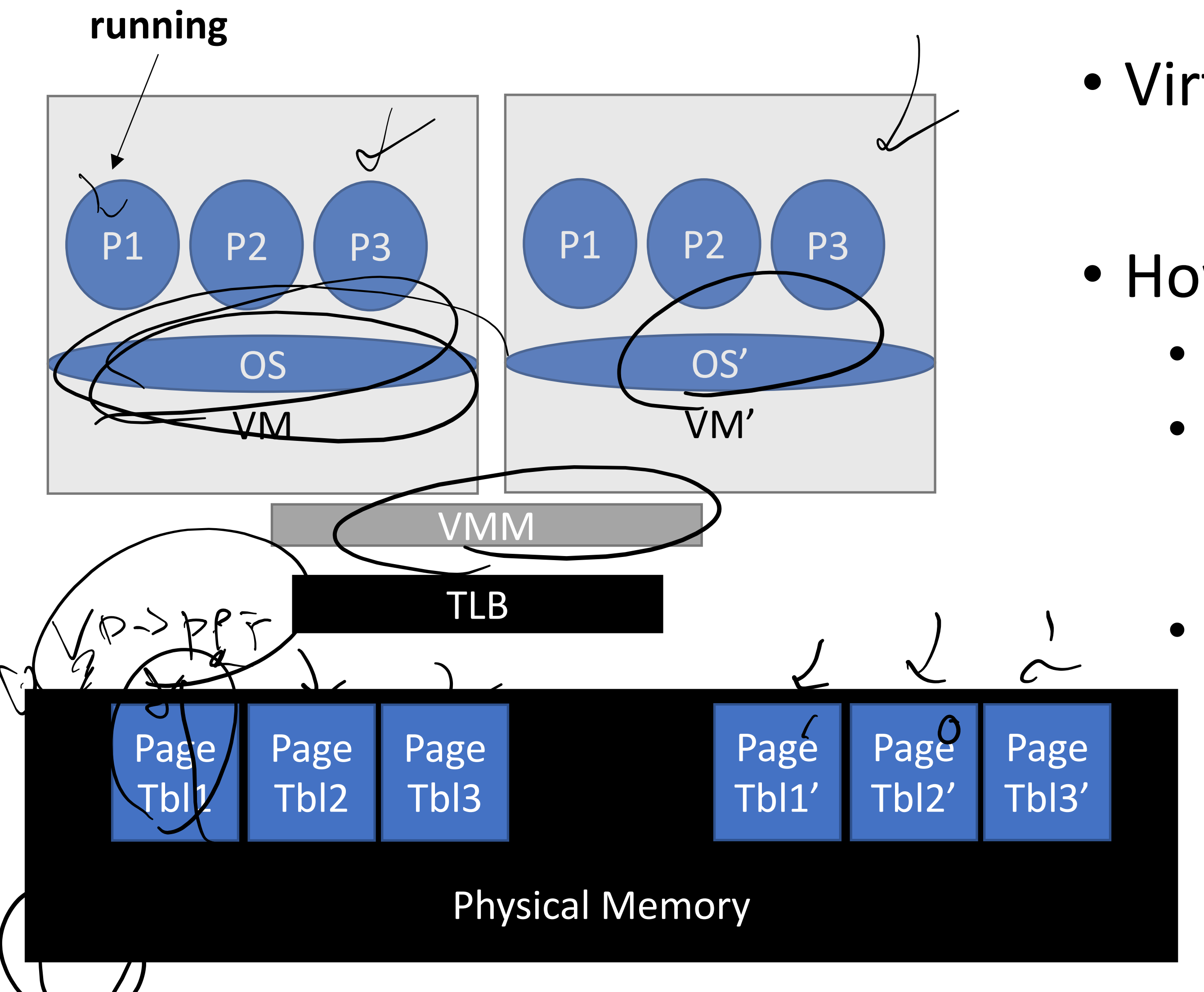
VP → P Pram

int A[0, ... N]

A[1]

4K

How did VMM manage memory across OS?



- Virtual A, Physical A

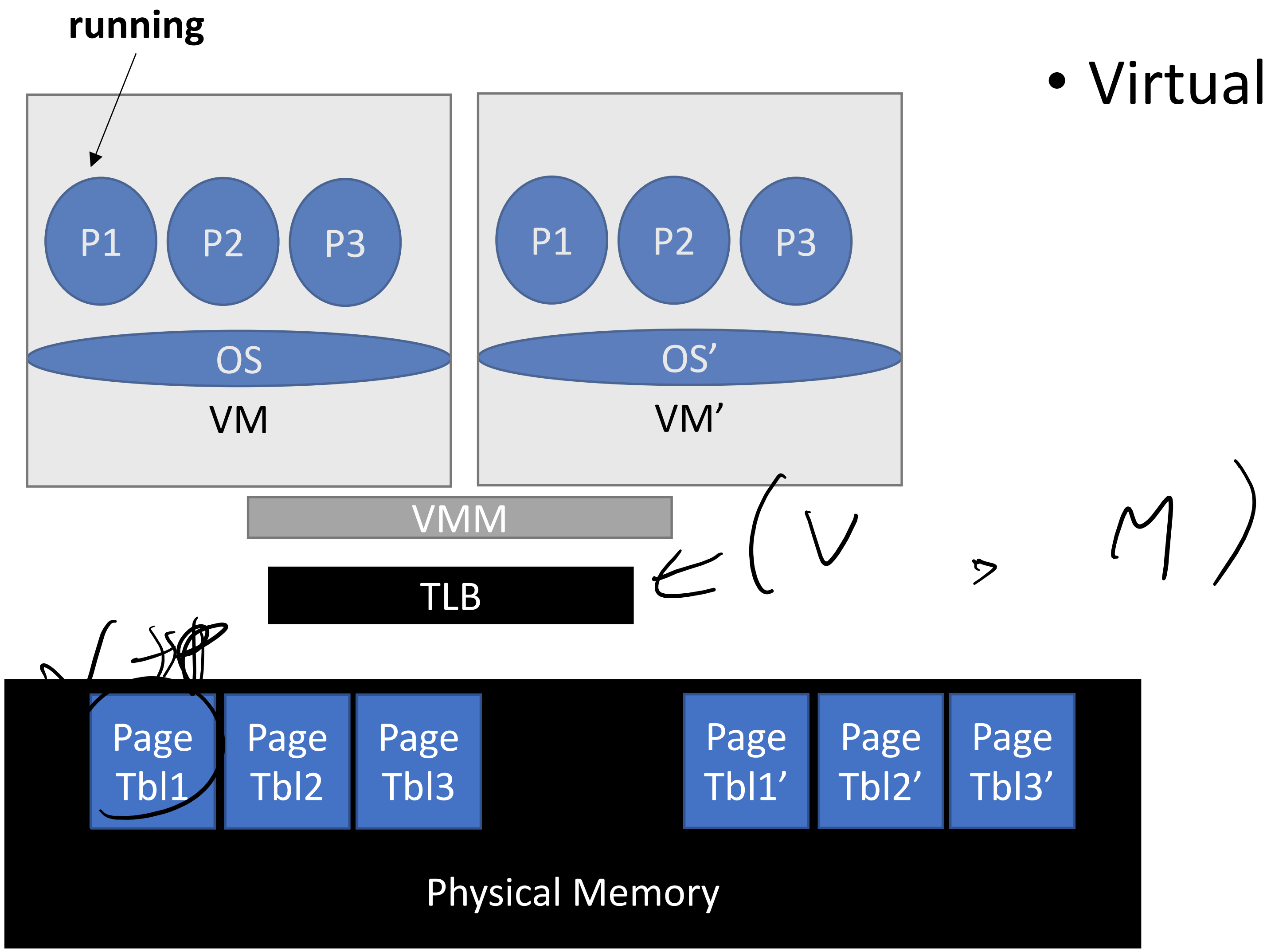
What is the problem?

- How to translate VA?
  - TLB look up
  - At TLB miss
    - Page table look up
    - (if valid & present) load page table entry to TLB
  - Redo TLB look up

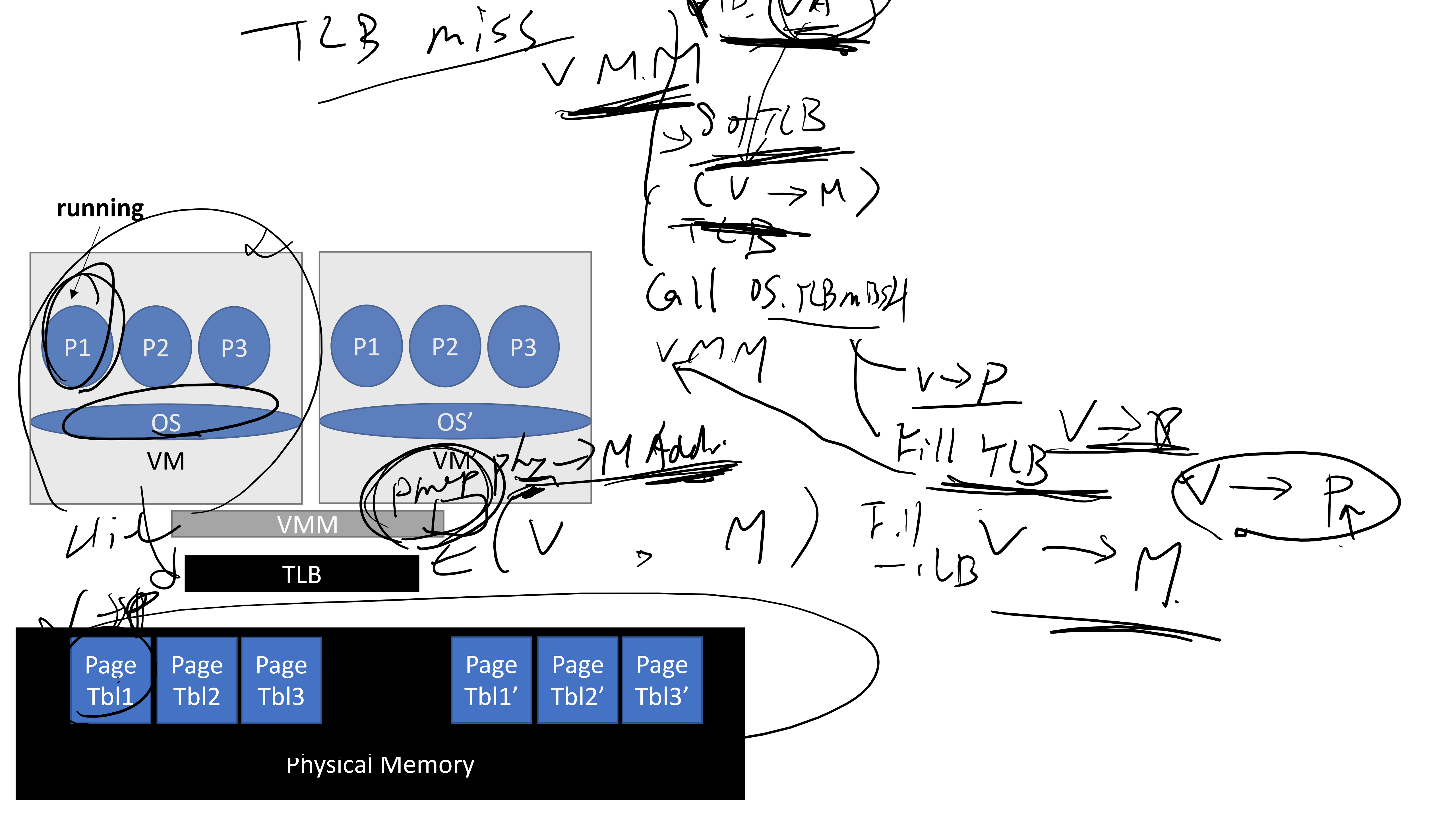
1G

Madhu Adhikari

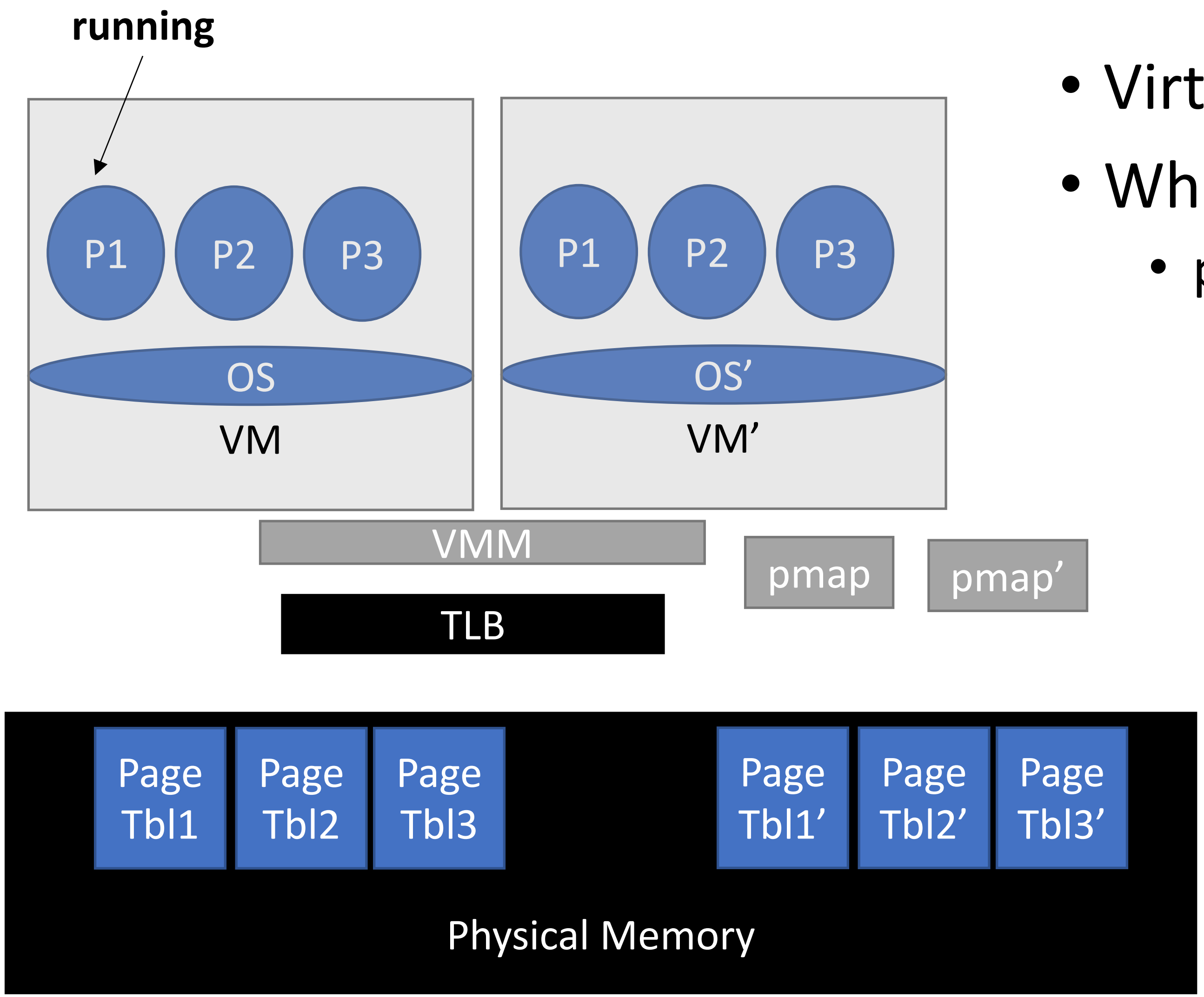
How did VMM manage memory across OS?



- Virtual A, Physical A, and Machine A

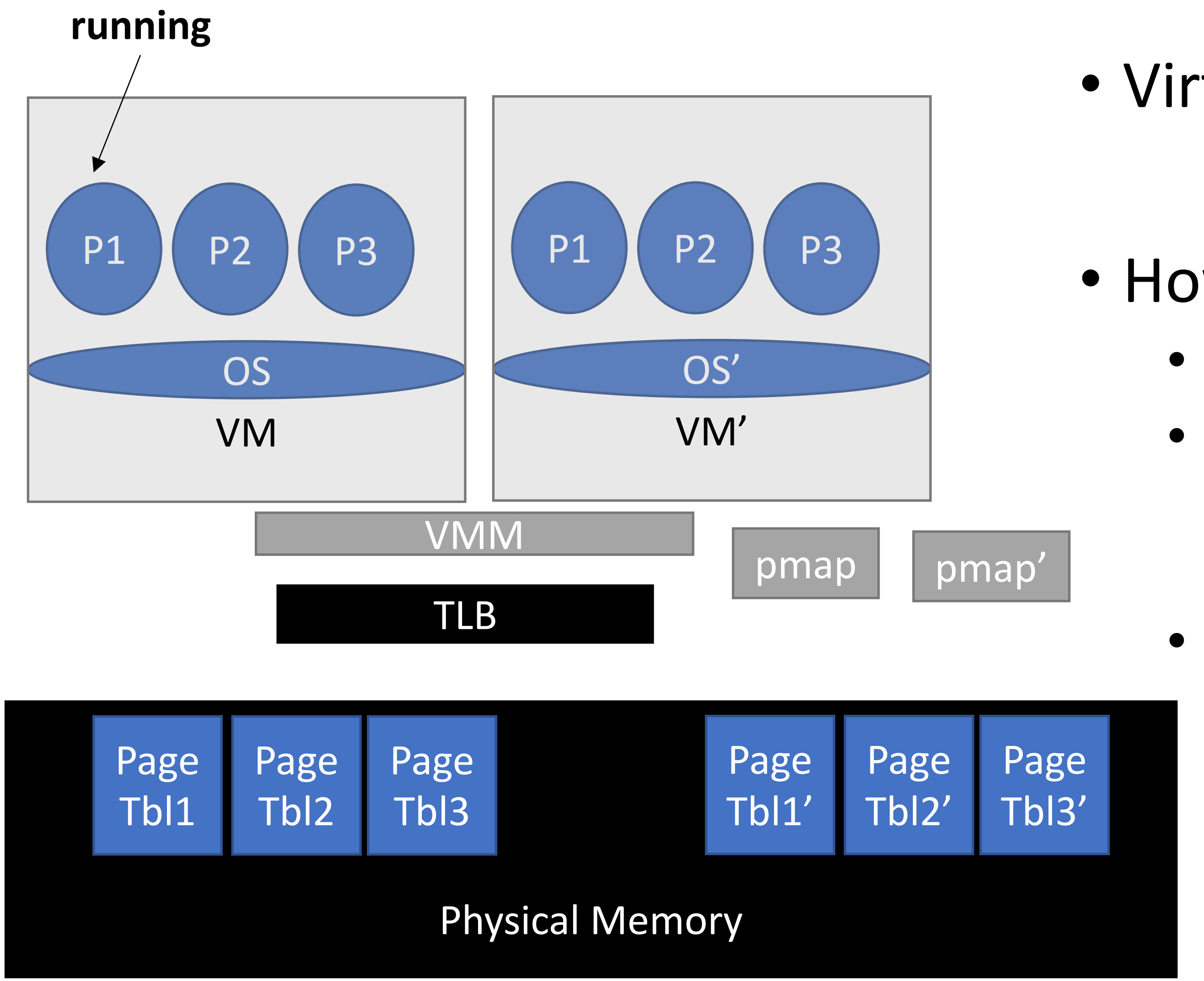

TLB miss
VMM
TLB
(V → M)
Call OS.TLBmissH
VMM
V→P
Fill TLB
V→P
V → P
Pmap
→ M Addr.
(V → M)
Fill TLB
V → M.
Hit
running
P1
P2
P3
OS
VM
P1
P2
P3
OS'
VM
VMM
TLB
Page Tbl1
Page Tbl2
Page Tbl3
Page Tbl1'
Page Tbl2'
Page Tbl3'
Physical Memory

How did VMM manage memory across OS?



- Virtual A, Physical A, and Machine A
- Where is the physical – machine mapping?
  - pmap

How did VMM manage memory across OS?



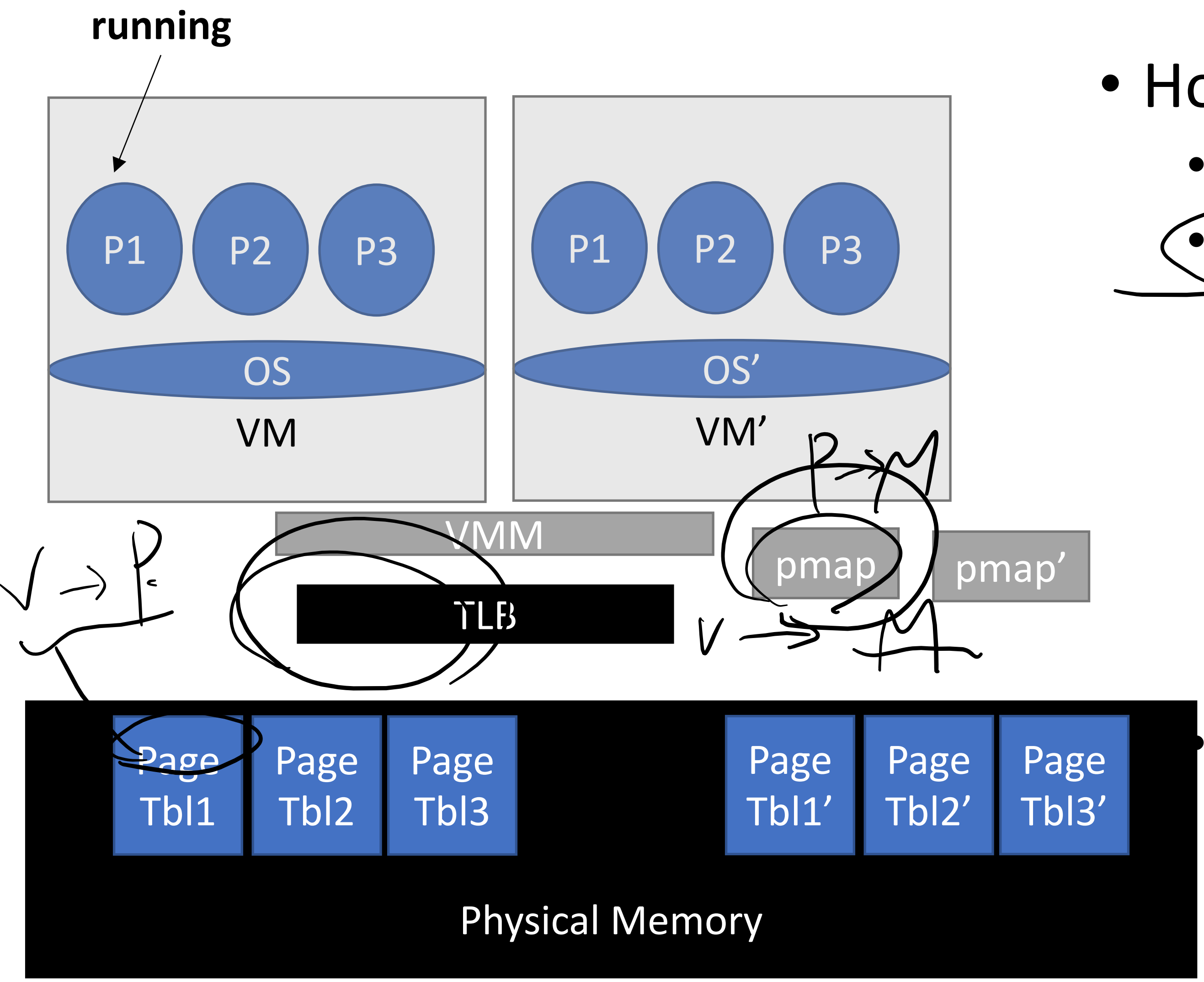
- Virtual A, Physical A, Machine A
- How to translate VA?
  - TLB look up
  - At TLB miss
    - Page table look up
    - (if valid & present) load page table entry to TLB
  - Redo TLB look up

What is in TLB?

What is in Page Tbl?

Who will handle miss (interrupt)?

How did VMM manage memory across OS?



- How to translate VA?
  - TLB look up (if hit, finish)
  - if TLB miss, trap to VMM
    - Check software TLB in VMM (if hit, fill TLB, …)
    - If sTLB miss, call into OS
      - OS page table look up
      - (if valid & present) load page table entry to TLB

        ➔privileged instruction exception

        🡐trap to VMM

        VMM does PA→MA, fill TLB with VA→MA
  - Redo TLB look up

Virtualization on x86 machines is much harder!

- On X86, TLB miss does not trigger any exception
  - Hardware automatically looks up the page table, and fills TLB

  ➔There is no opportunity for VMM to jump in during this process

  ➔Xen requires changes to guest OS, so that guest OS' page table contains virtual to machine address translation