# COS3043
# System Fundamentals

Lecture 3

# List of Discussion

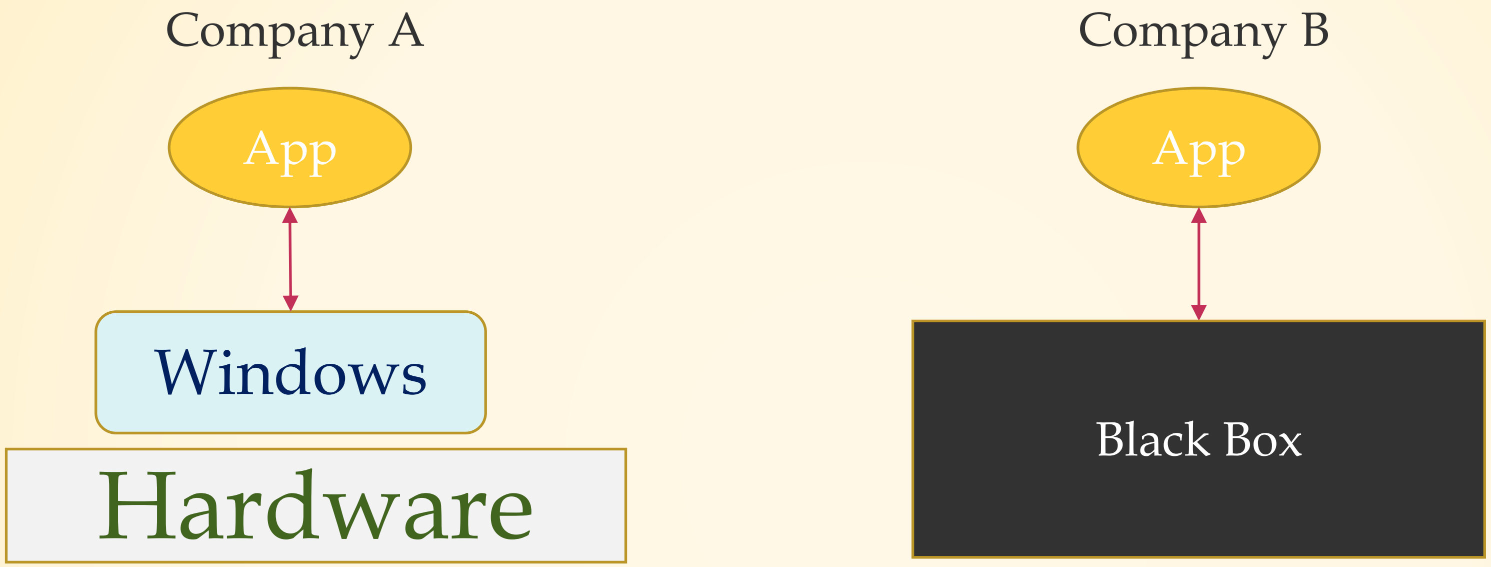
- Virtualization Basics
- Memory Virtualization
- CPU Virtualization
- Device Virtualization

# Virtualization Basics

# Question?

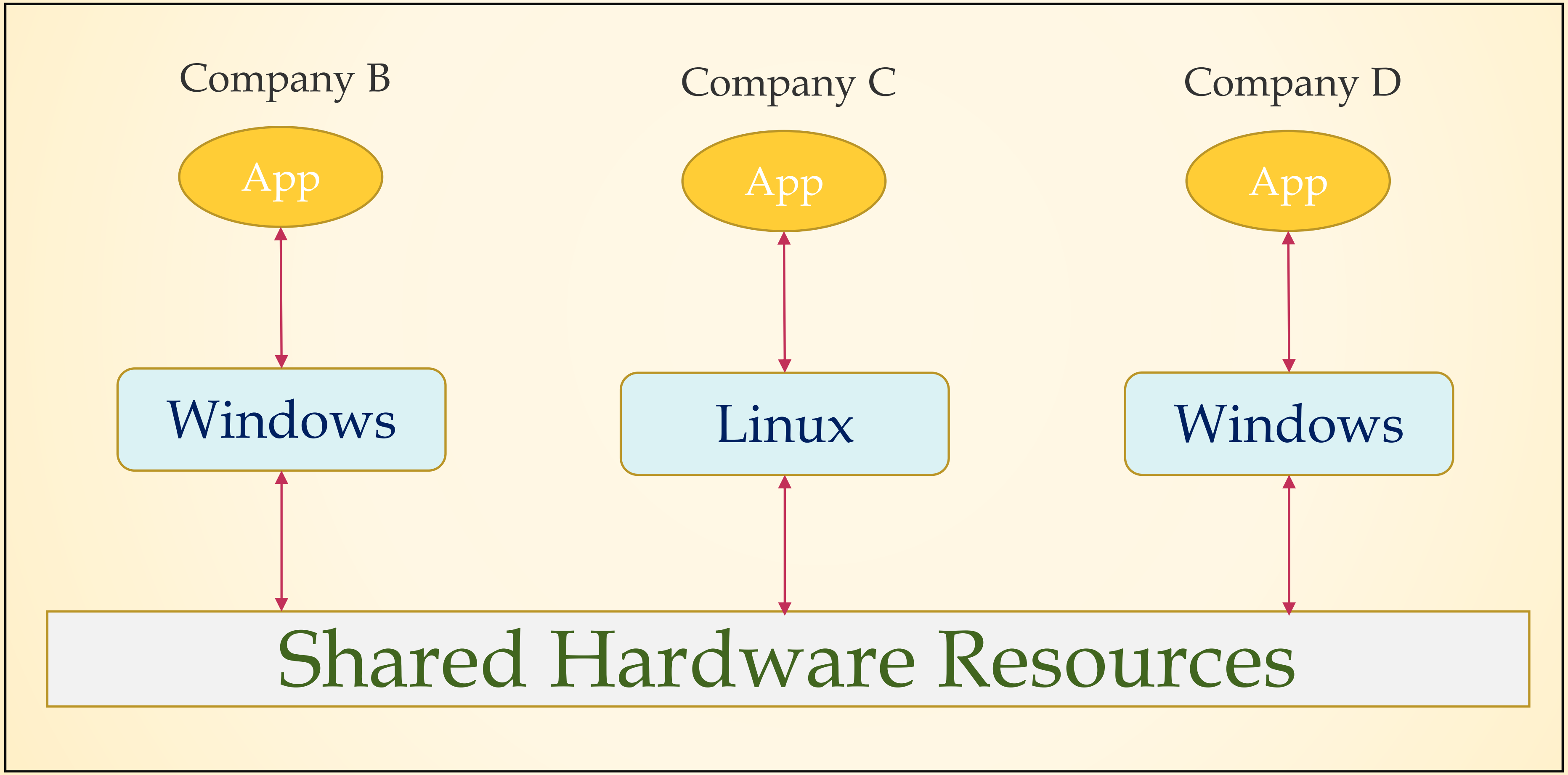
- Are the below concepts/items related to virtualization?
  - ➢Memory System?
  - ➢JVM?
  - ➢Virtual Box?
  - ➢Cloud Computing?
  - ➢VM Ware Workstation?
  - ➢The Movie “Inception”?
  - ➢Data Centres?

# Platform Virtualization

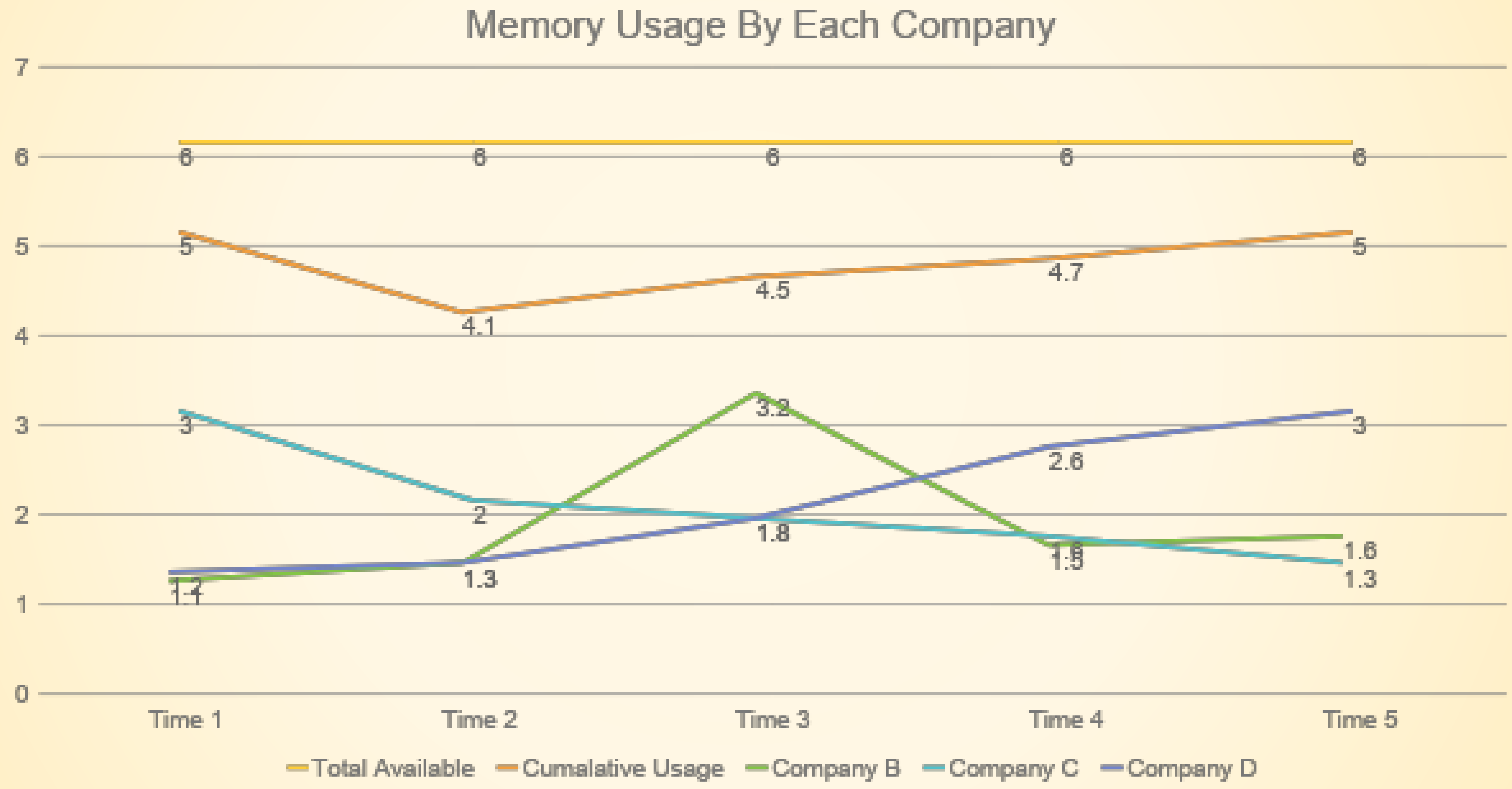


- Are the below concepts/items related to virtualization?
  - Memory System?

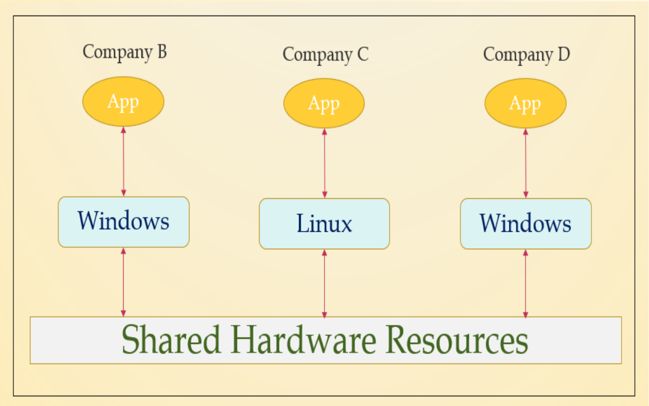
# Utility Computing



# Utility Computing

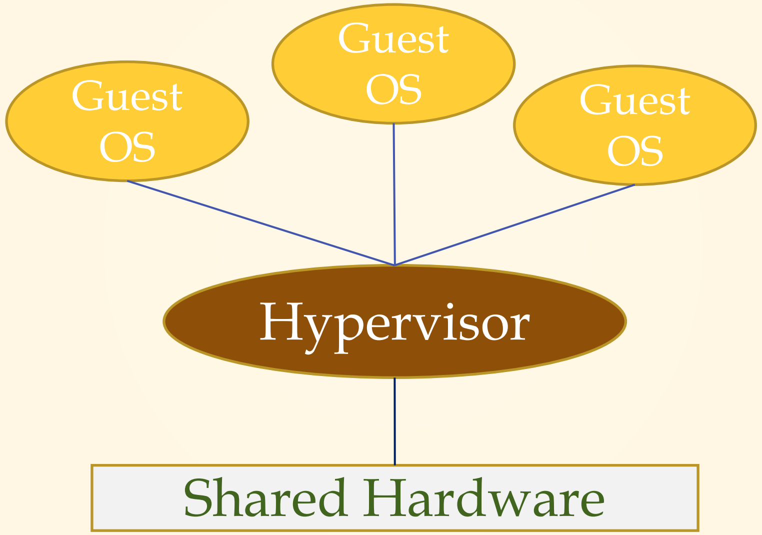


# Hypervisors



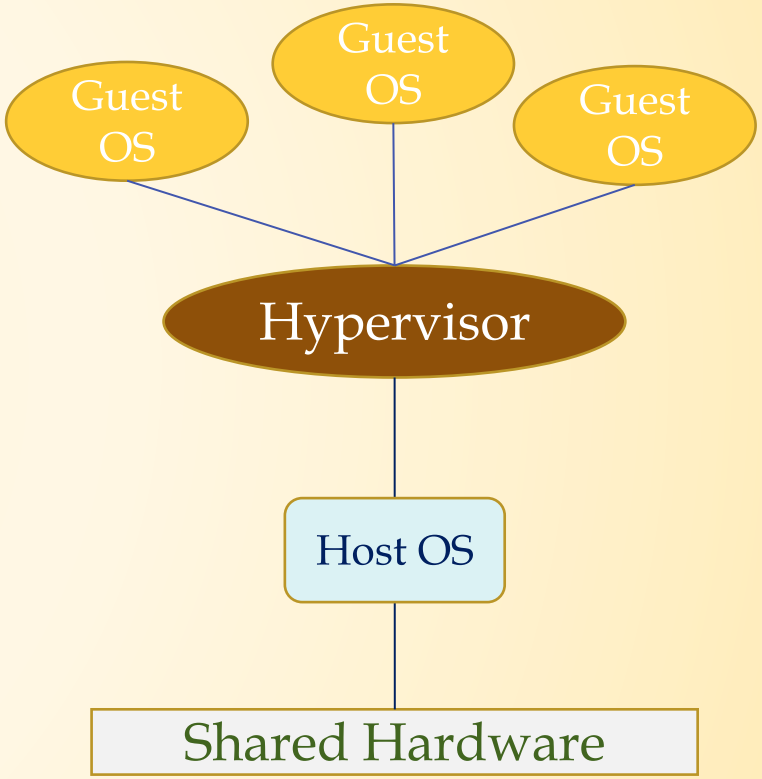
**Type 1 Hypervisor (Bare-Metal Hypervisor):**

Native
(bare metal)

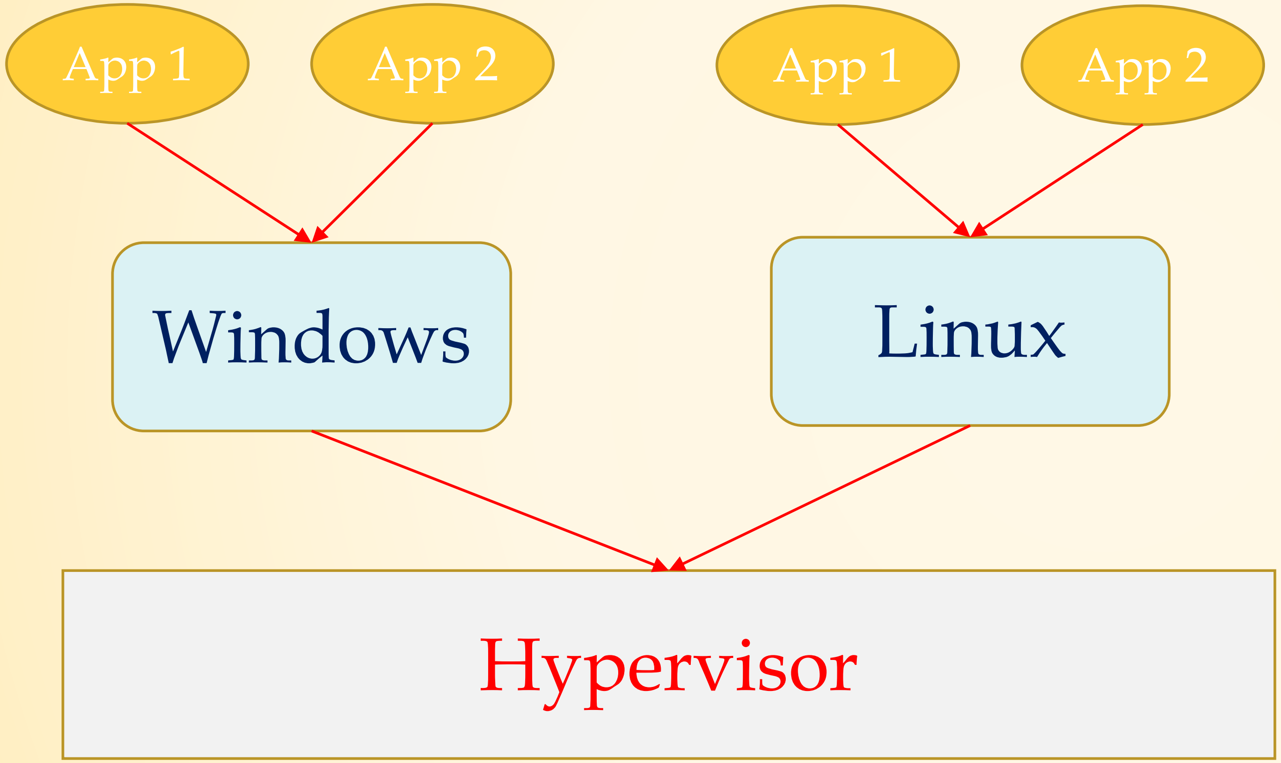


**Type 2 Hypervisor (Hosted Hypervisor):**

Hosted

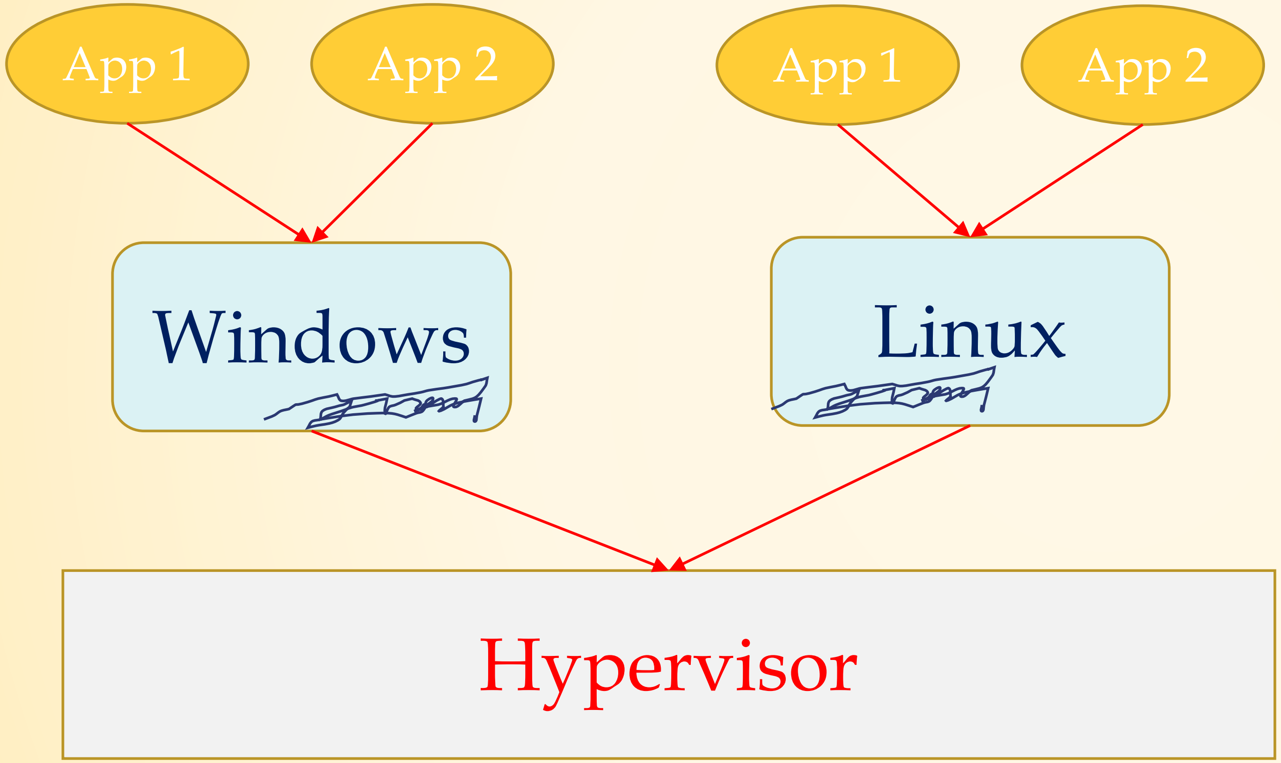


# Full Virtualization



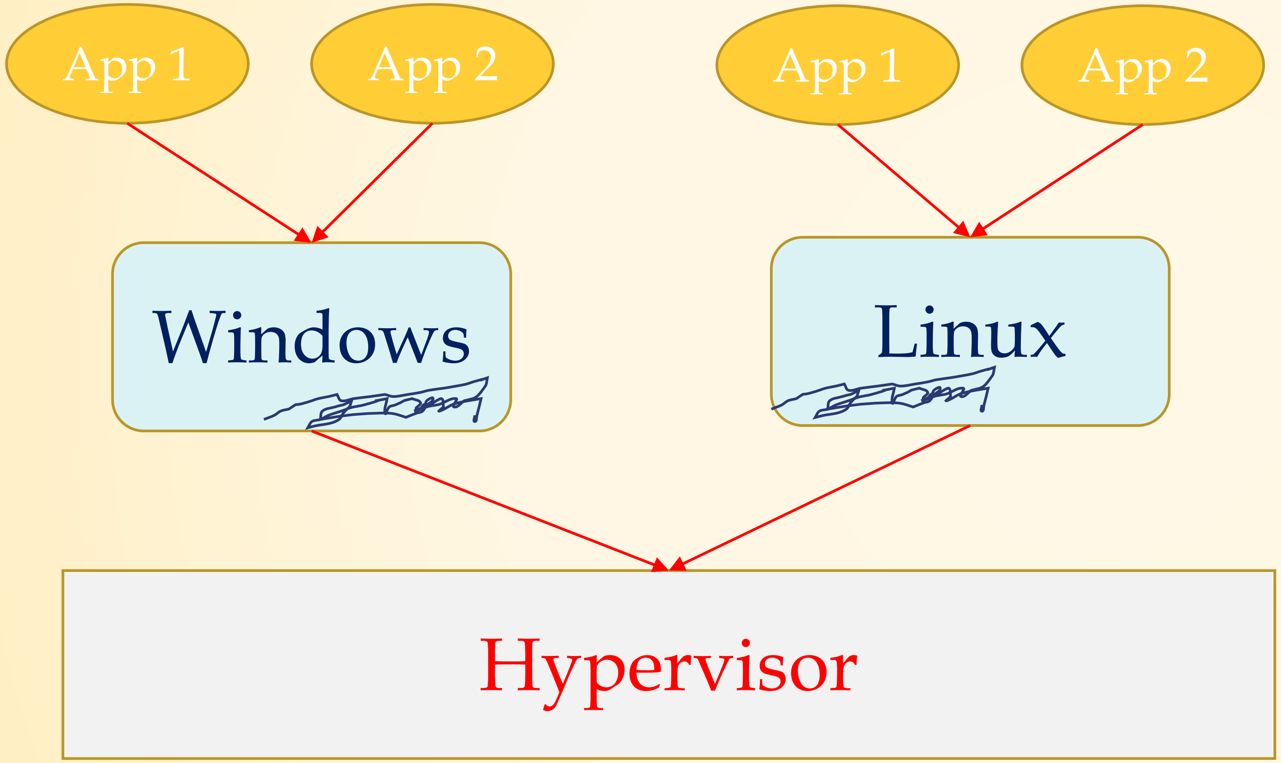
- The key idea is to leave the guest OS 'untouched'.
- The binary of the OS are unchanged => in 'full' mode – no single line of the OS codes has been changed.
- But the OS are running on the users level, which means both OS are running with different privilege.
- It adopts "trap + emulate" strategy – when an OS has to run an privilege operation which is higher than an user level, then it will resolve a trap in the Hypervisor. Then the Hypervisor emulate the intended functionality.

# Para Virtualization



- The other approach is to modify some or partial of the source code in each OS.
- If we can do that, not only we can avoid problematic instructions, it also increase optimization – to let OS seeing the hardware status.
- As far as application is concerned, nothing is changed as application only access to the OS.

# Para Virtualization



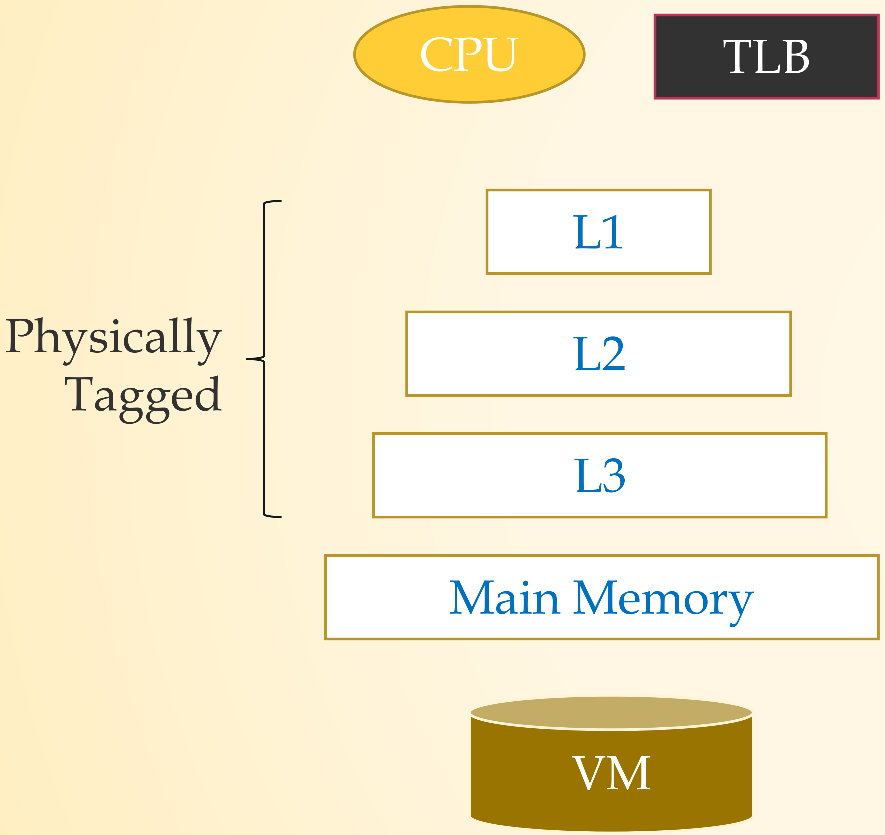
| Sub System | Linux | MS XP |
|---|---|---|
| Architecture Ind. | 78 | 1299 |
| Virtual Net. Dr. | 484 | - |
| Virtual Block Dev. | 1070 | - |
| Xen Specific | 1363 | 3321 |
| **Total** | **2995** | **4620** |
| **% Code Base** | **1.36** | **0.04** |

# Looking Forward

- What needs to be done?
  - ➢ Virtualize Hardware
    - ✓ Memory Hierarchy
    - ✓ CPU
    - ✓ Devices
  - ➢ Effect data and control between guests and Hypervisor

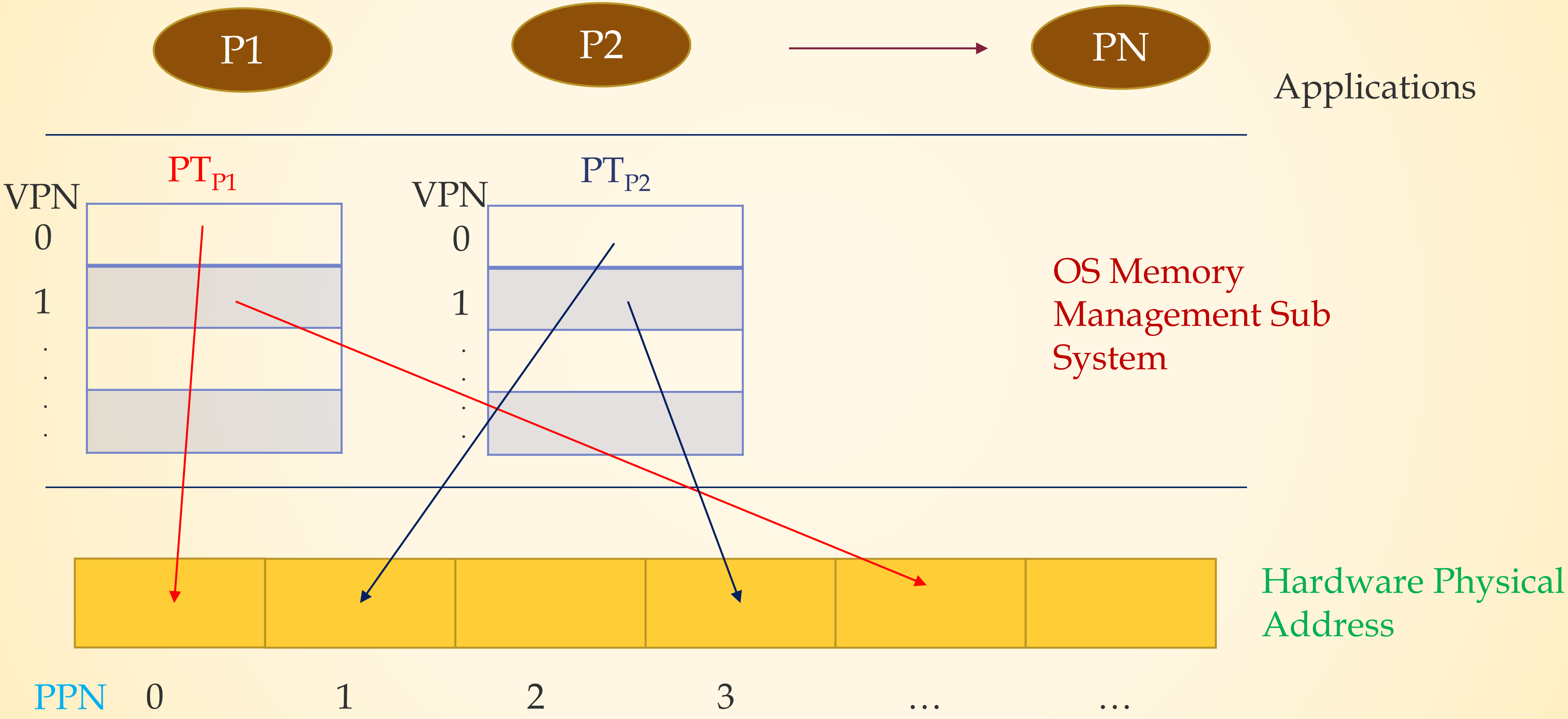
# Memory Virtualization

# Memory Hierarchy

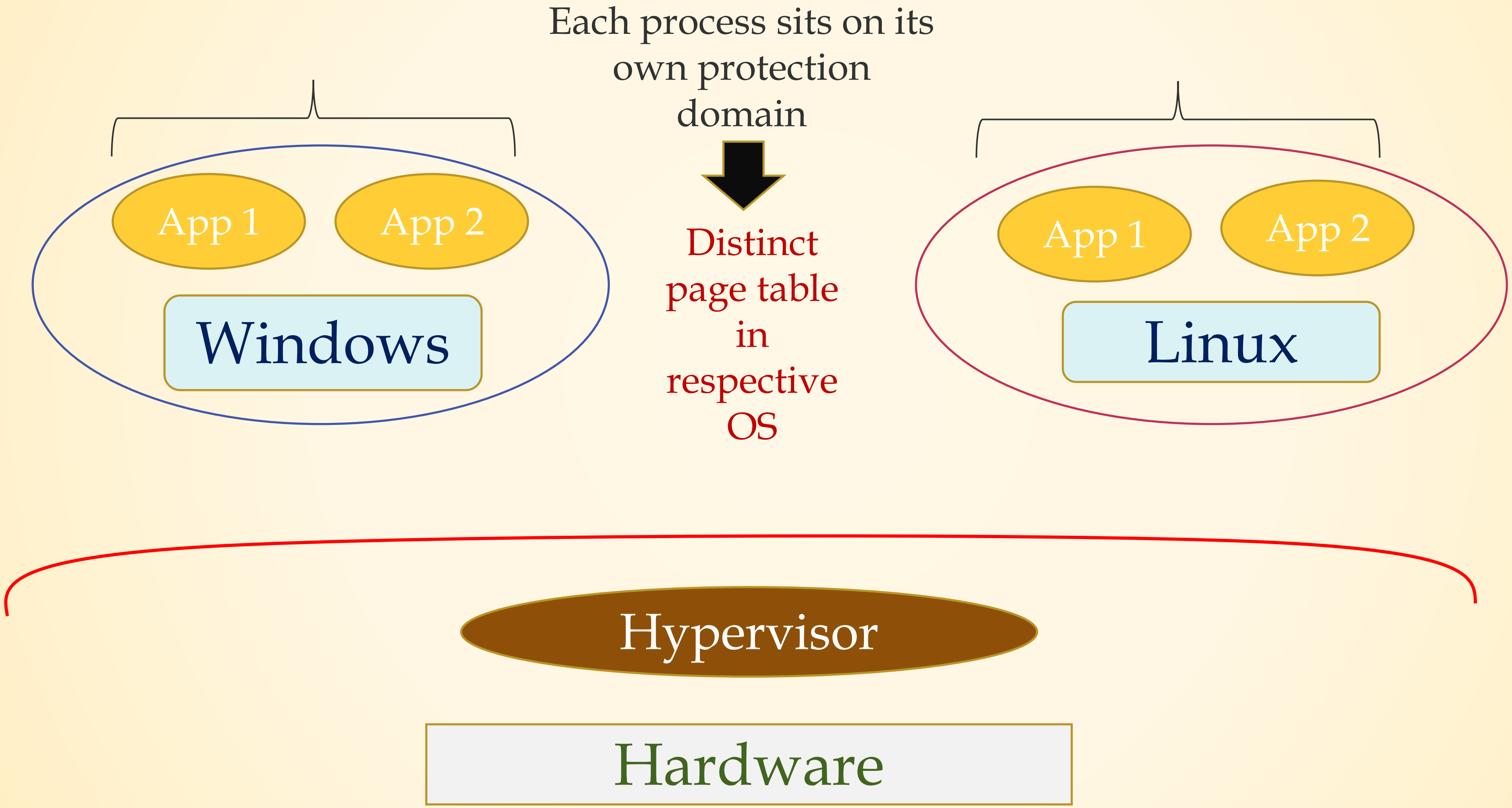


- What's the thorny issue?
  - ➢Handling virtual memory => key functionality => the virtual memory to the physical address mapping.

# Memory Sub System Recall



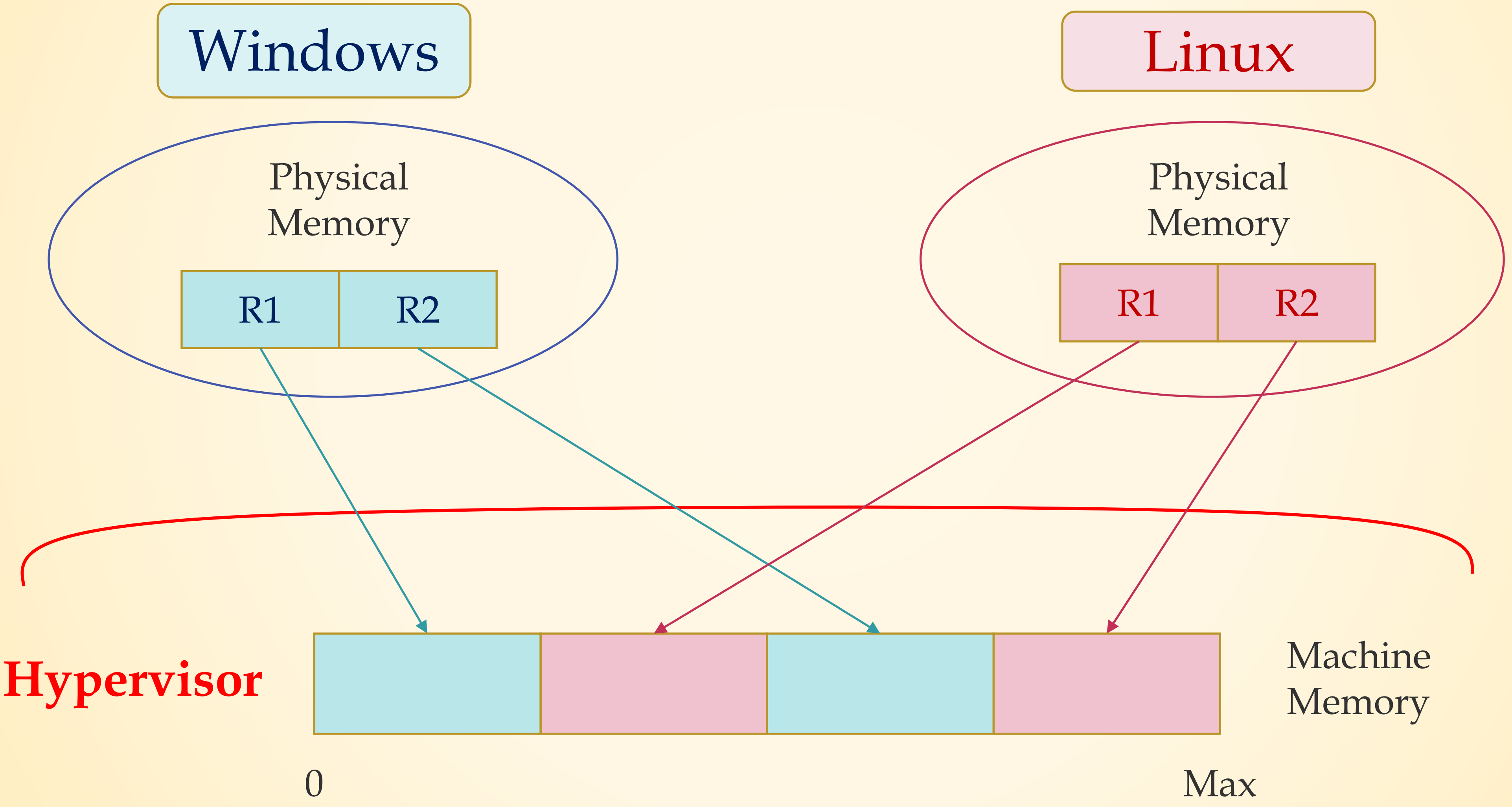
# Memory Management and Hypervisor



Each process sits on its own protection domain

Distinct page table in respective OS

# Memory Manager Zoomed Out



Windows

Physical Memory

R1

R2

Linux

Physical Memory

R1

R2

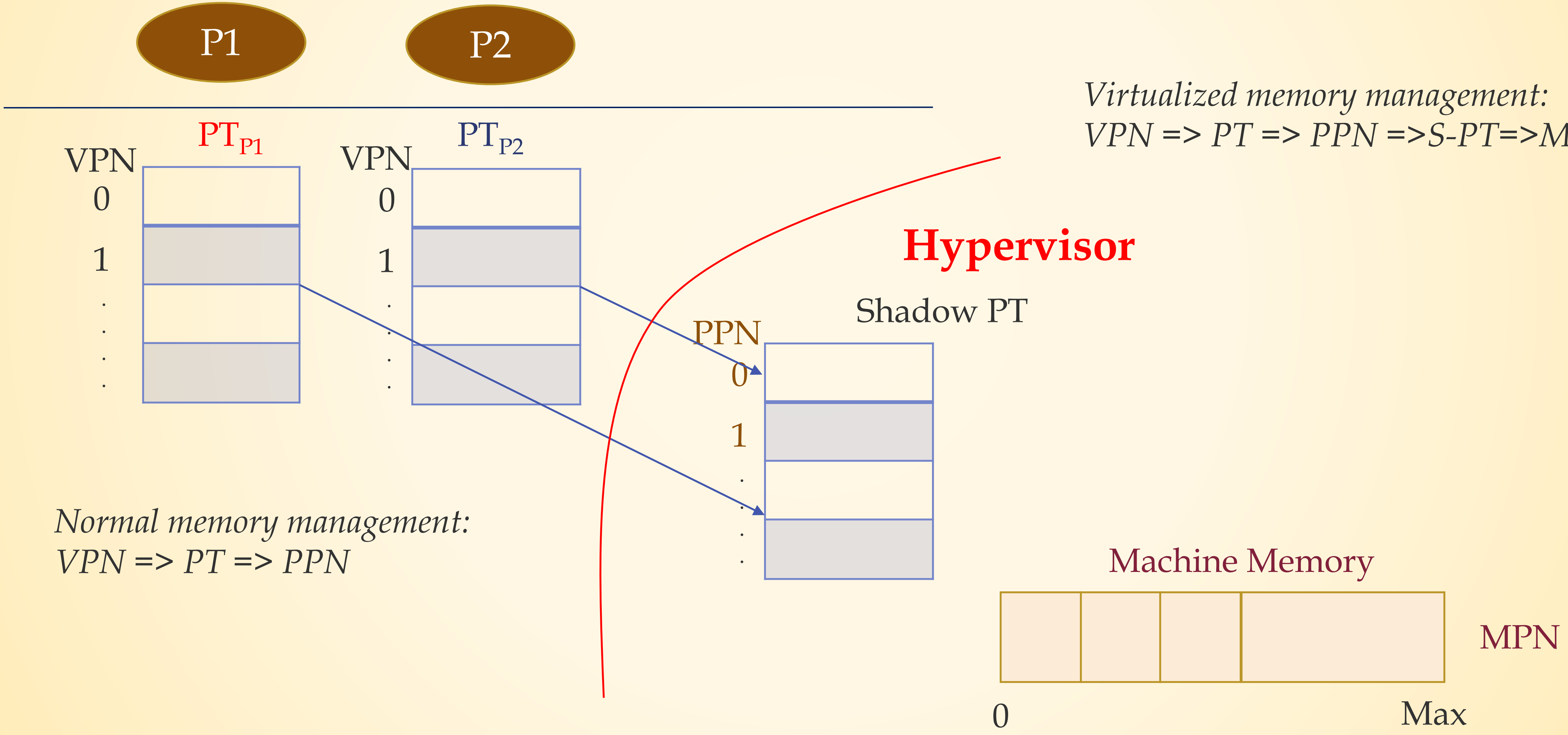
Hypervisor

Machine Memory

0

Max

# Memory – Zooming Back



*Virtualized memory management:*
*VPN => PT => PPN =>S-PT=>MPN*

*Normal memory management:*
*VPN => PT => PPN*

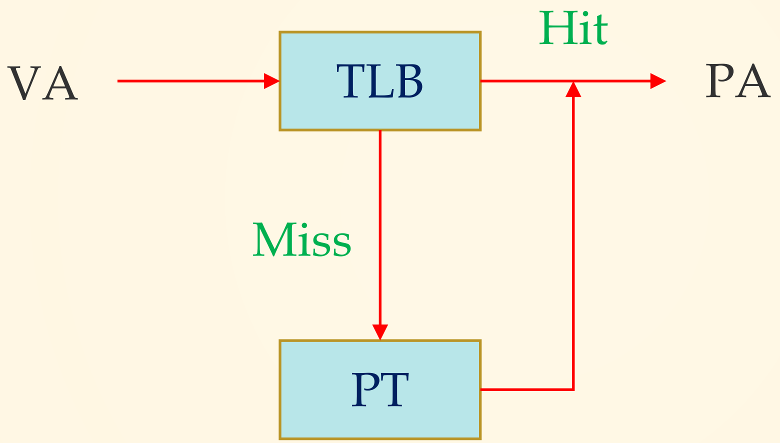
# Question?

- Who keeps PPN=>MPN in fully virtualized environment?
  - Guest OS or Hypervisor?

- Who keeps PPN=>MPN in para virtualized environment?
  - Guest OS or Hypervisor?

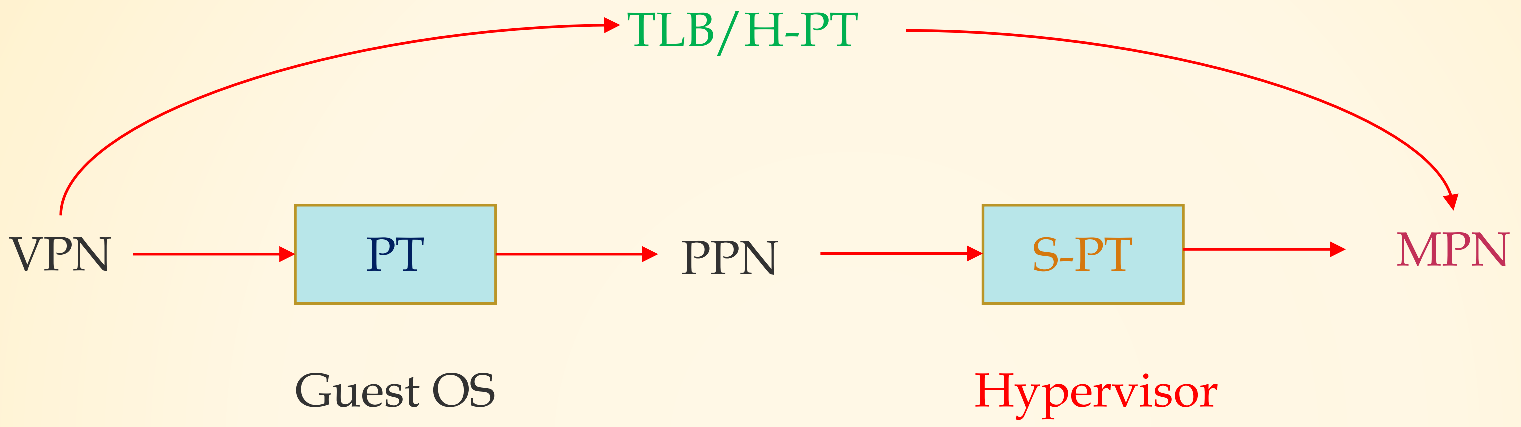
# Shadow Page Table

- In many architectures (ex: Intel X86)
  - ➢CPU uses page table for address translation.



=> Hardware PT is really the S-PT in virtualized setting.

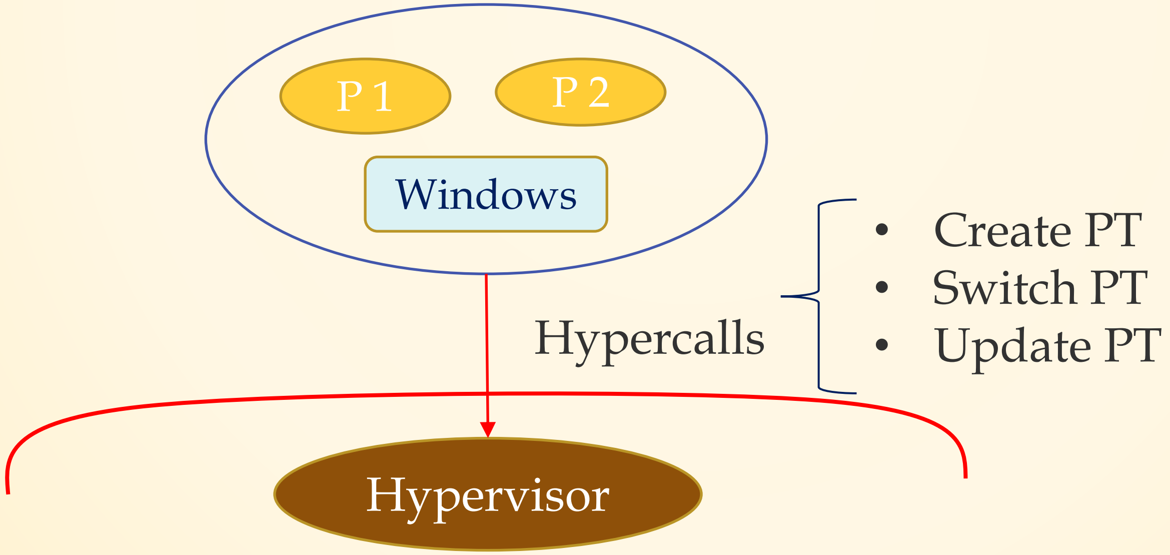
# Efficient Mapping in Full Virtualization



- How to make the above operation efficient?
  - ➢PT/TLB updates => guest OS trapped
  - ➢S-PT updated by Hypervisor
  - ➢Translations installed into TLB/hardware PT

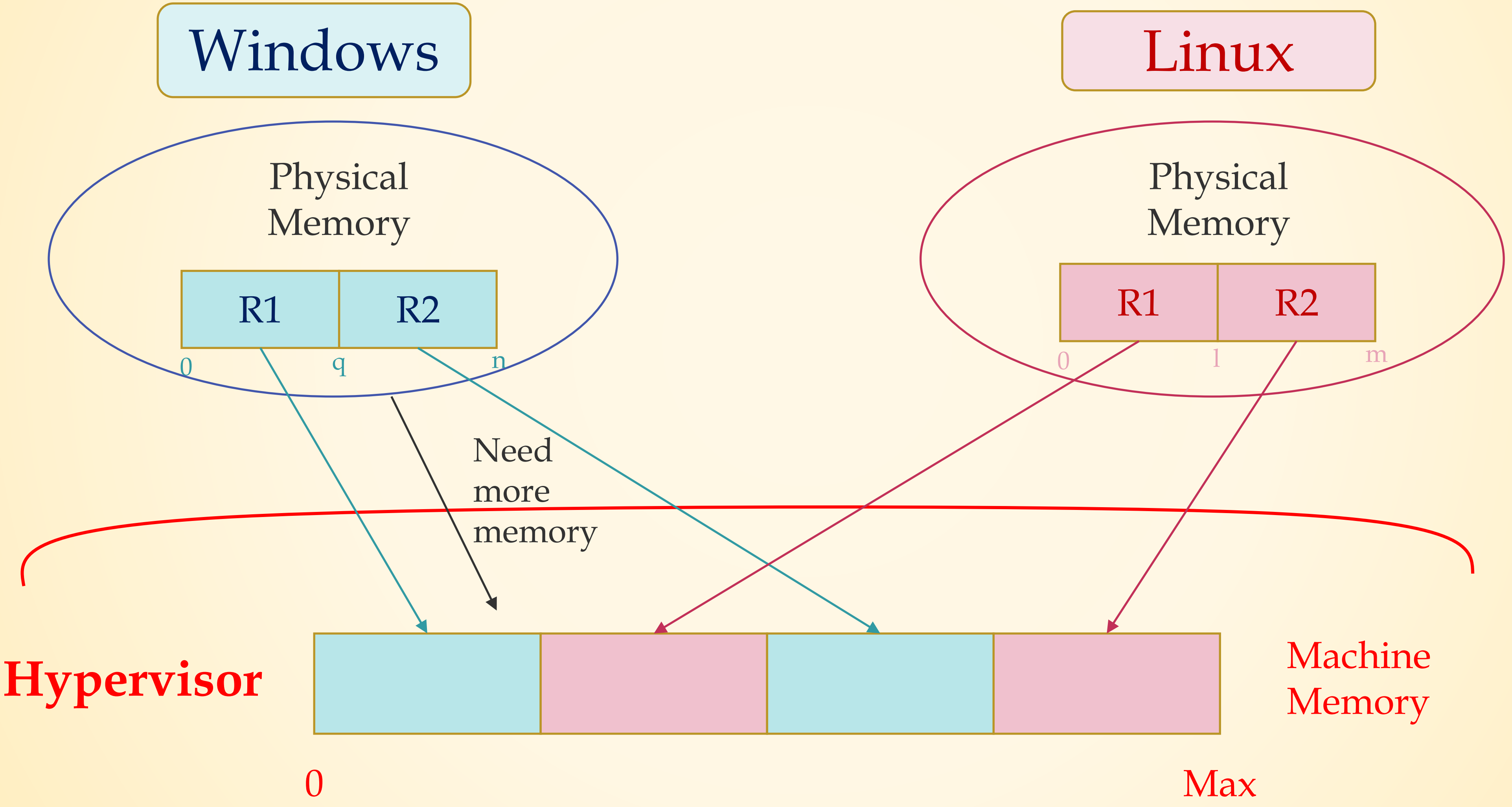
# Efficient Mapping in Para Virtualization

- Shift the burden (PPN=>MPN) to guest OS
  - Maintain contiguous “physical memory”
  - Map to discontinuous hardware pages.
  - The operation of managing, allocating, mapping can be done at guest OS.



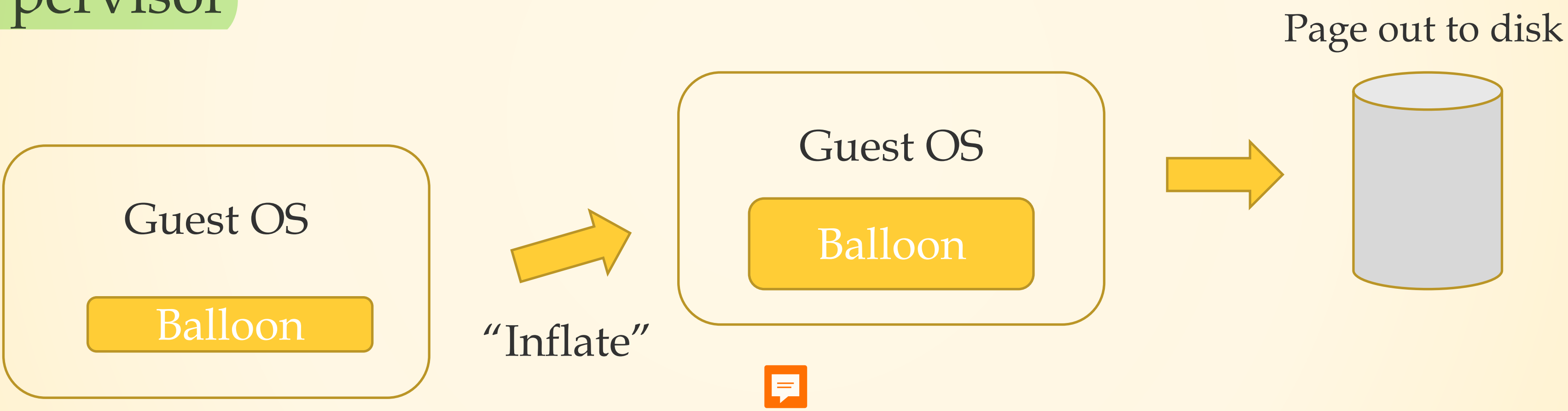
- Create PT
- Switch PT
- Update PT

# Dynamically Increasing Memory



# Ballooning

- Thus here it comes, the other technique to address the previous mentioned issue – ballooning.
- It's a special device driver installed inside every guest OS by Hypervisor



1. House needs more memory.
2. Then Hypervisor contacts one of the guest OS (which is not actively using all its memory) Balloon via private channel => to inflate the Balloon
3. Then the Balloon returns the memory to Hypervisor.

# Sharing Memory Across Virtual Machines

- Memory is precious resource, thus whenever we can share it, we will share.
- But, can it be done across virtual machines?
    - ➢ *The answer is "YES" provided, the it's same environment: OS + Application (same version)*
    - ➢ *Example: A similar Firefox process running on Linux on both VM1 and VM2*

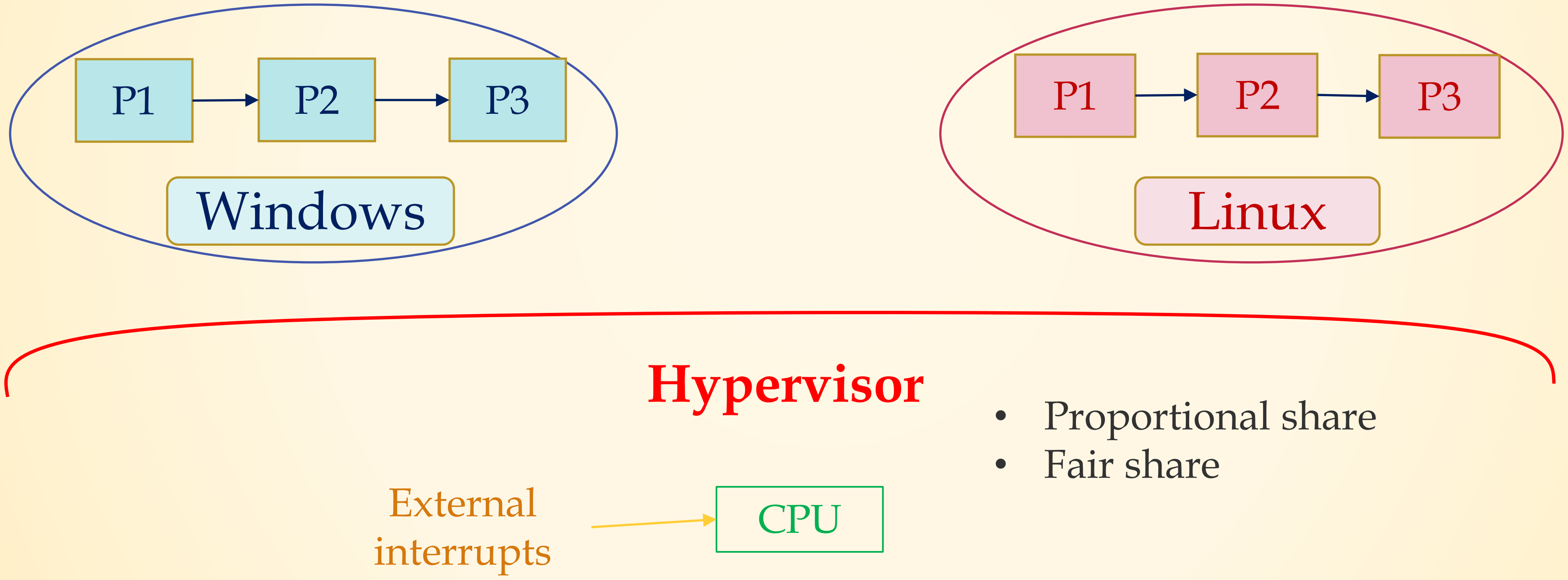
# Memory Allocation Policies

- Pure Share Approach
  - “pay less, get less”, could lead to ‘holding’.
- Working Set Approach
  - Allocate and reclaim as and when needed.
- Dynamic Idle-Adjusted Share Approach
  - Because of the notion of “pay more, get more”, it comes mix pure share + working set.
  - ‘Tax’ idle page more than active page. ‘Tax’ means take back the resource.
  - Reclaim most idle memory.
  - Allow for sudden working set increases.

# CPU Virtualization

# First Part

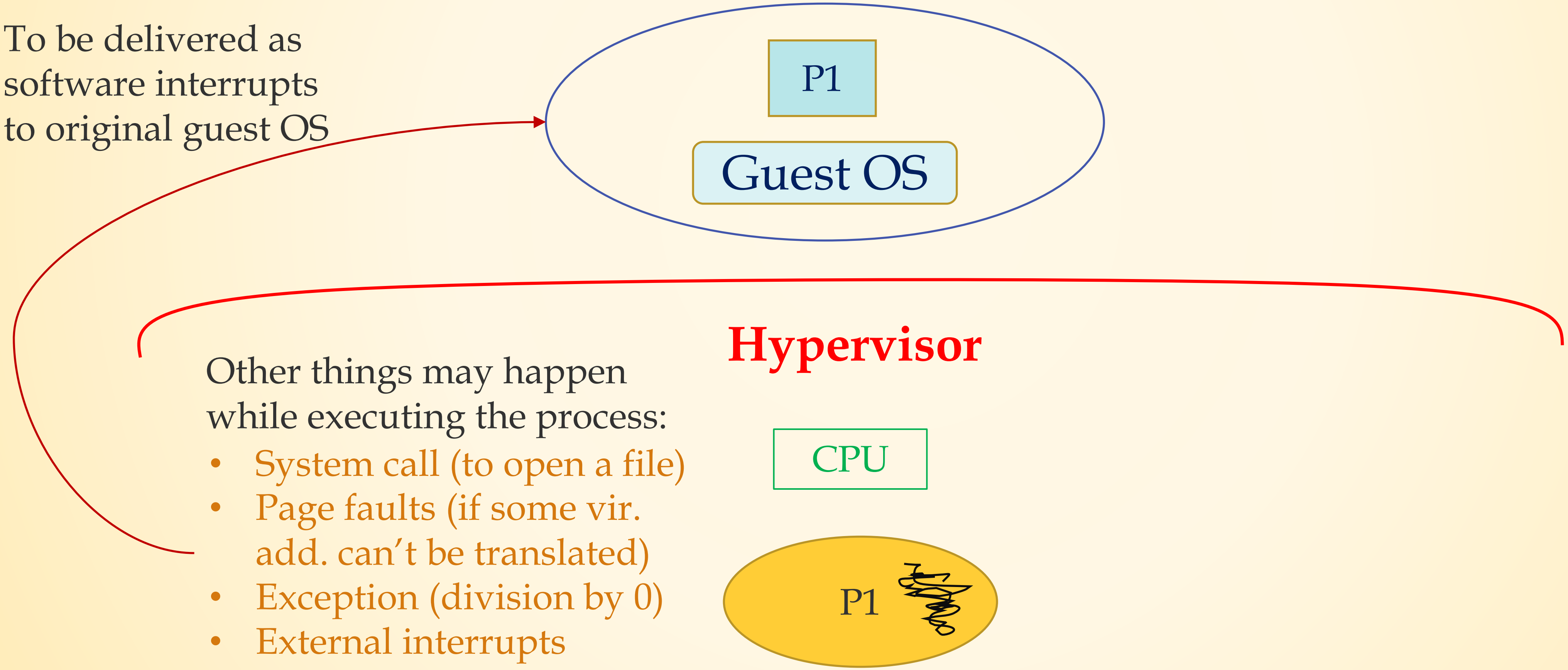
- Illusion of ownership of CPU for each guest OS



- Proportional share
- Fair share

# Second Part

- Deliver events to original guest OS



To be delivered as software interrupts to original guest OS

Other things may happen while executing the process:

- System call (to open a file)
- Page faults (if some vir. add. can’t be translated)
- Exception (division by 0)
- External interrupts

# Device Virtualization

# Introduction

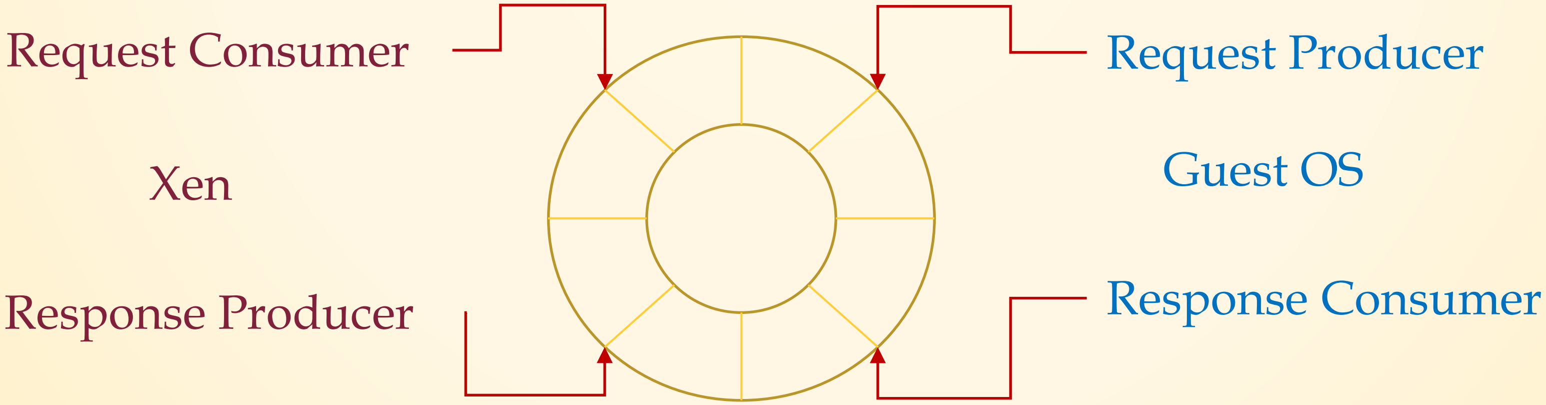
- Full virtualization
  - “trap and emulate” – illusion for guest OS thought it owns the device.
  - No room for innovation
- Para virtualization
  - More opportunity for innovation
  - Interaction between device and guest OS: control and data transfer.

# Control Transfer

- Full virtualization
  - Implicit (traps) by guest => Hypervisor.
  - Software interrupts (events) by Hypervisor => guest OS
- Para virtualization
  - Explicit (hypercalls) by guest => Hypervisor.
  - Software interrupts (events) by Hypervisor => guest OS
  - Guest OS has control in hypercalls on when event notifications to be delivered.

# Data Transfer

- Full virtualization
  - Implicit.
- Para virtualization
  - Explicit - using data structure with pointer.
  - Example: Xen's Asynchronous I/O Rings.
  - Each guest has an I/O ring for communication.



# Measuring

- CPU Usage
- Memory Usage
- Storage Usage
- Network Usage

# Conclusion

- Difference from Extensible OS
  - Focus on protection and flexibility
- Virtualization is a big trend that every players from different domain are trying to compete and advance in the same pool.