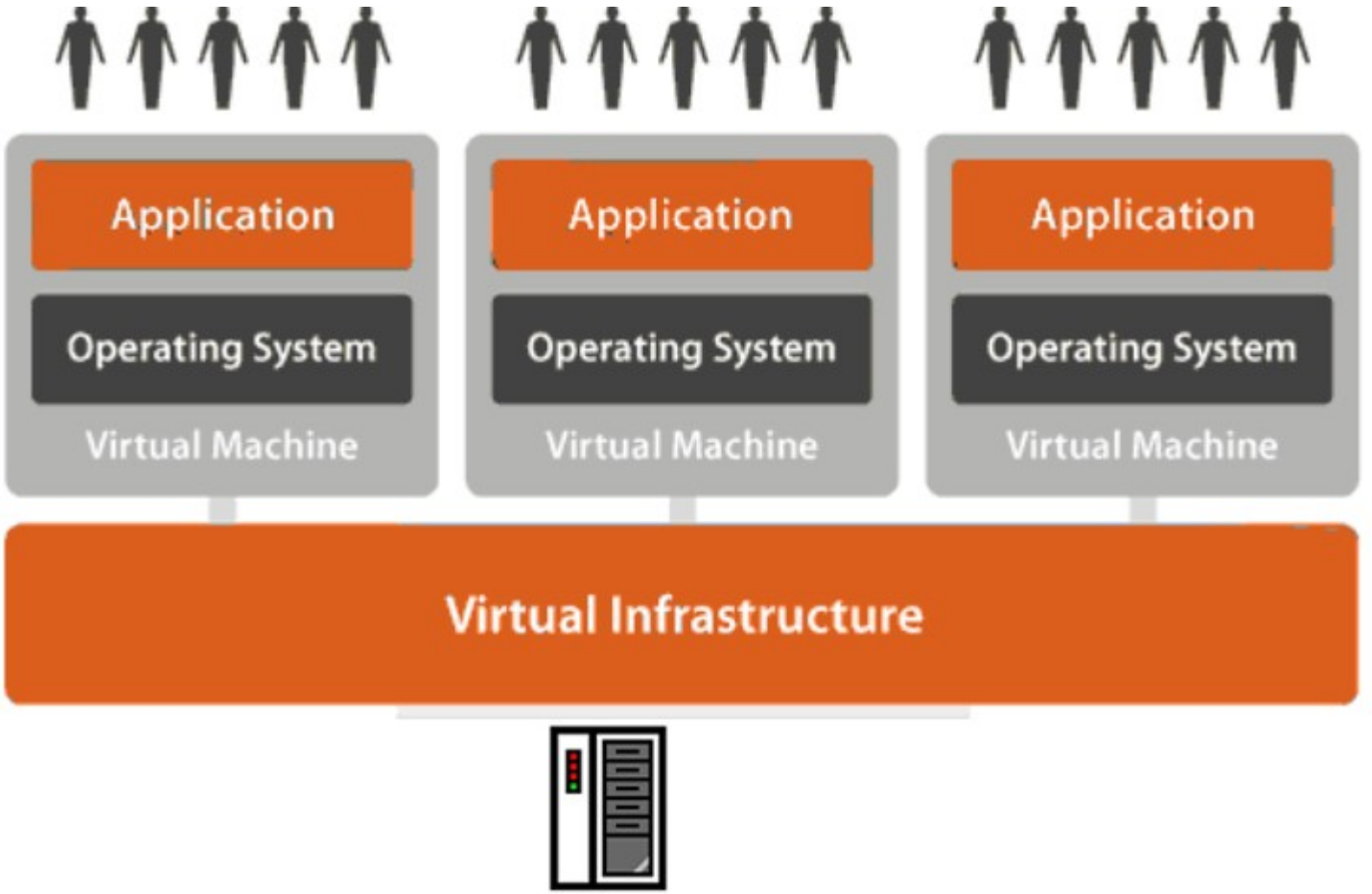
KVM/ARM: The Design and Implementation of the Linux ARM Hypervisor

Fall 2014

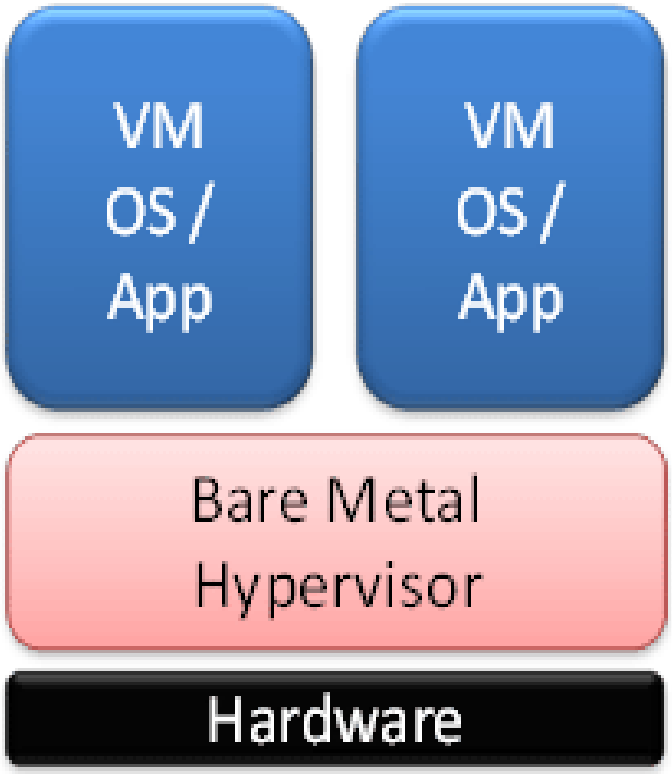
Presented By: Probir Roy

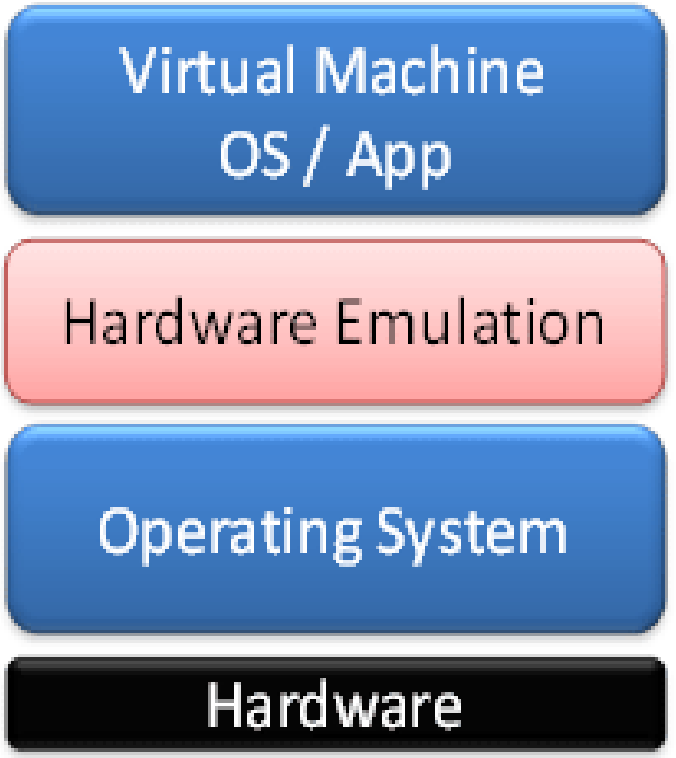
Virtualization & Hypervisor



Virtualization & Hypervisor (cont.)

Which type of Hypervisor is better?





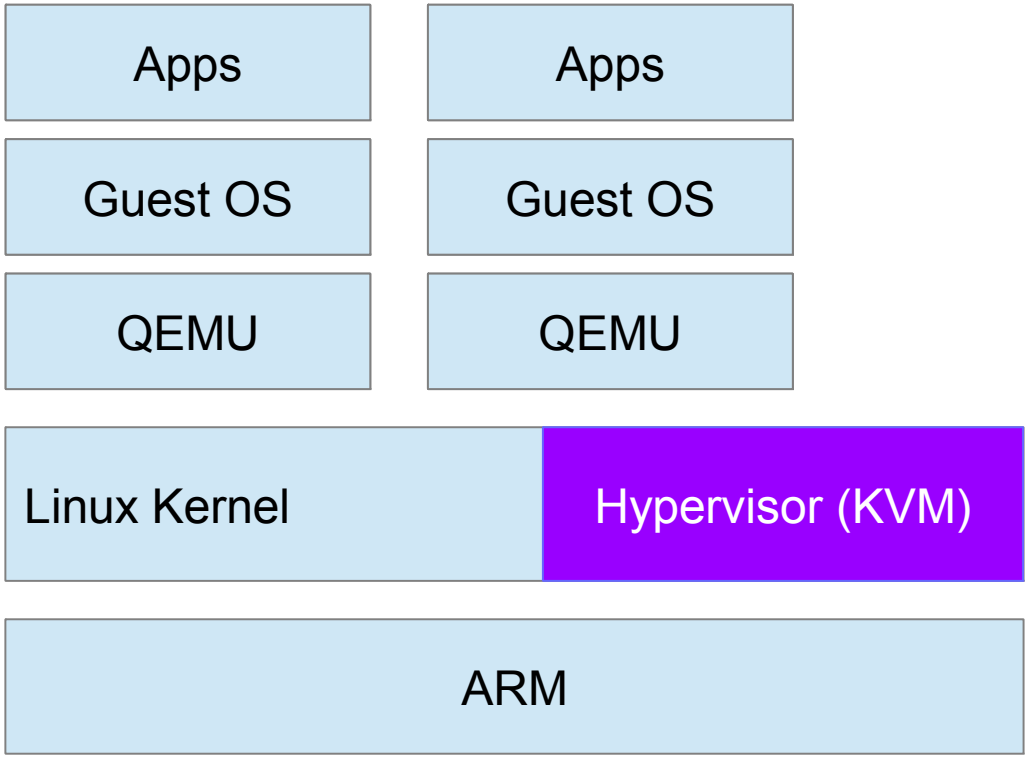
Bare Metal / Native Hypervisor

Hosted Hypervisor

Xen, VMWare ESX

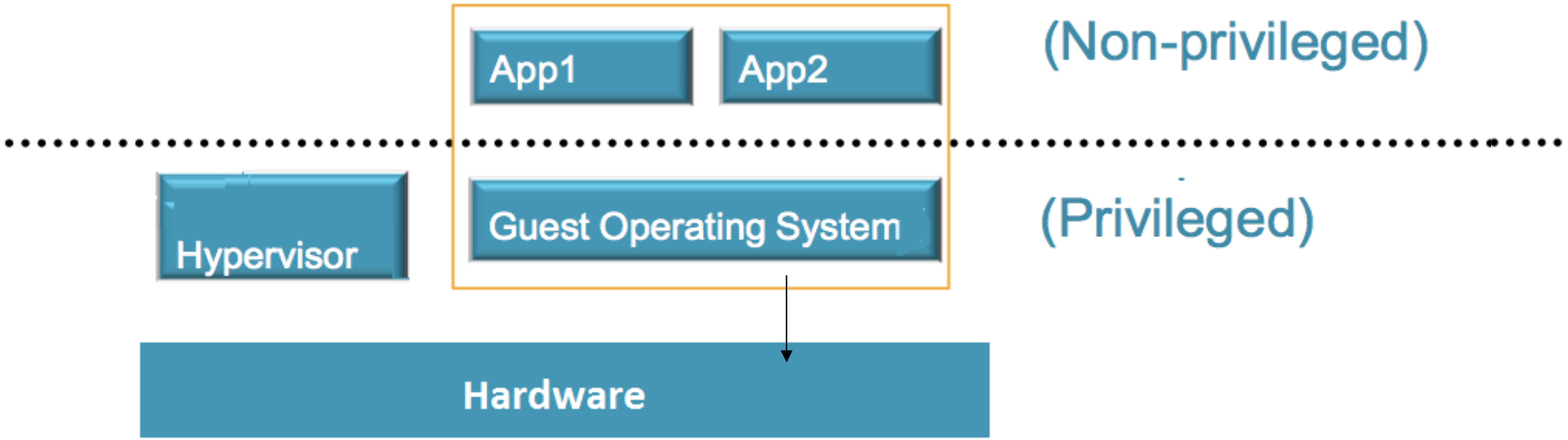
VMWare Workstation

KVM/ARM: The Design and Implementation of the Linux ARM Hypervisor



Full Virtualization

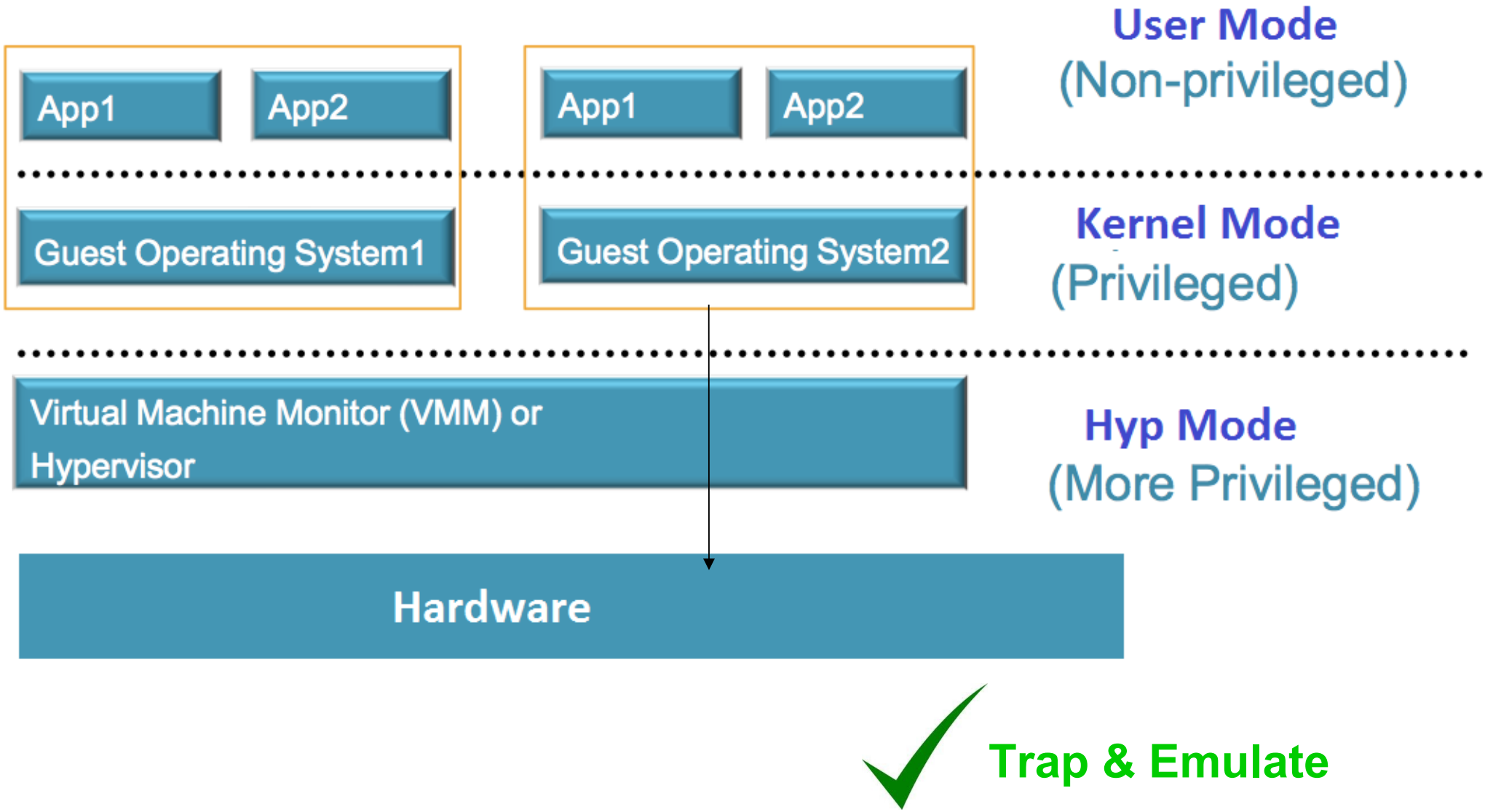
Prior ARMV7



Not Classically Virtualizable

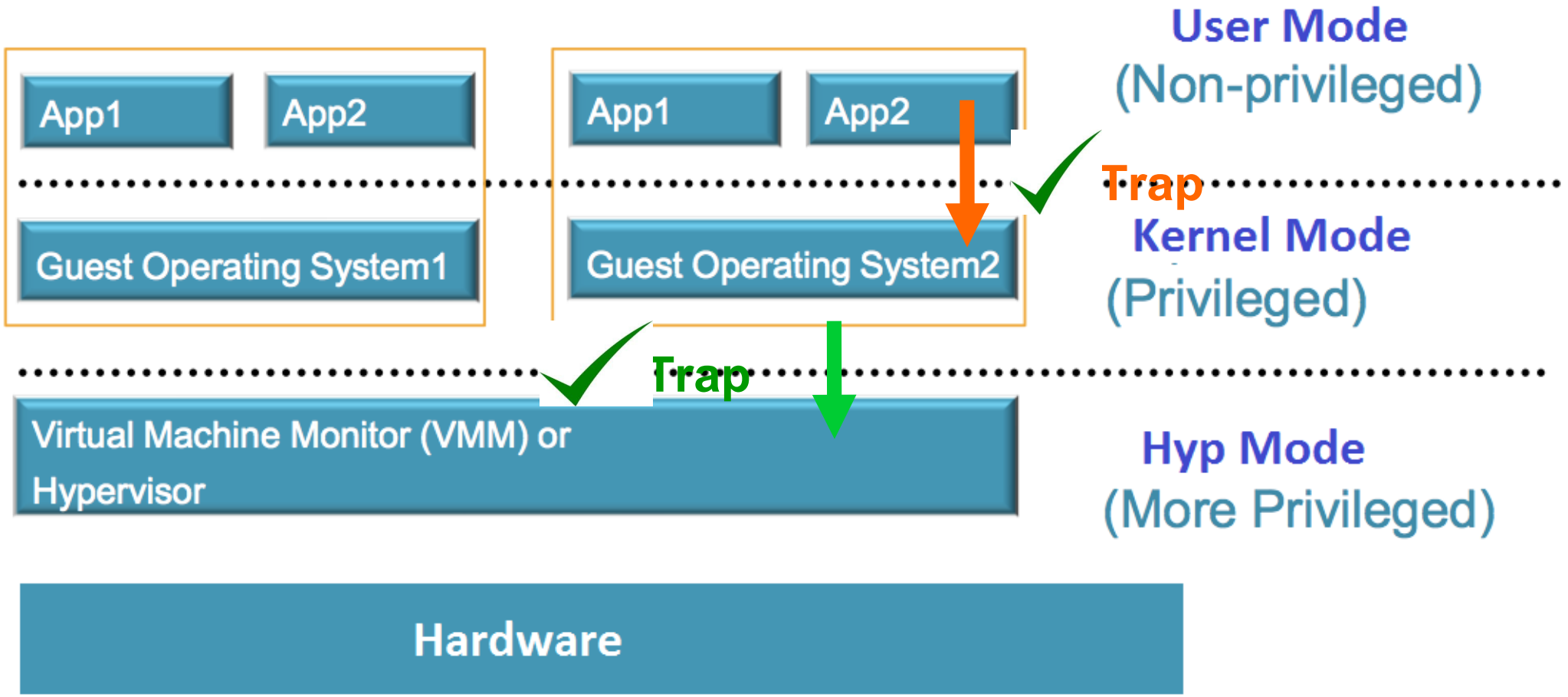
Trap & Emulate

ARM V7 Virtualization Extension



CPU Virtualization

Why trapping at Kernel Mode is useful?

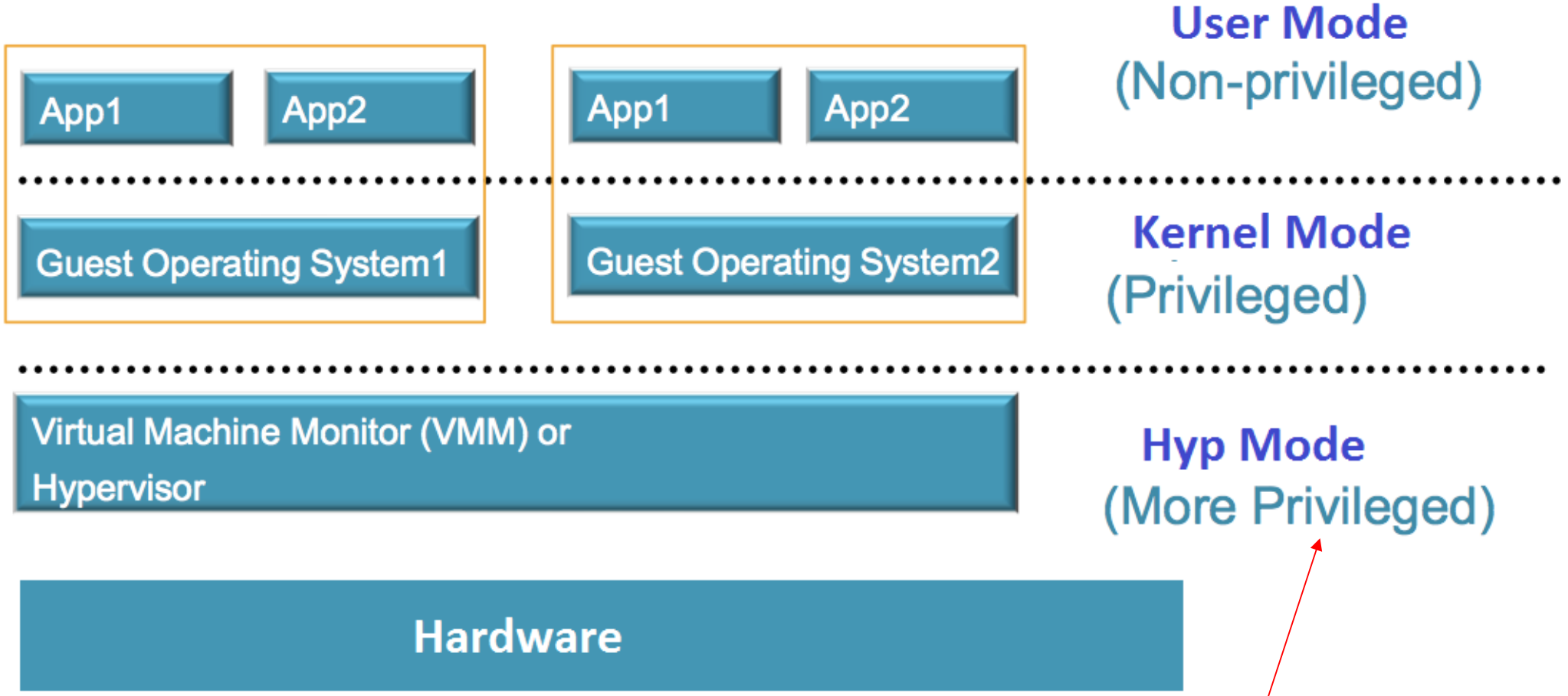


From Hyp Mode, hardware Configurable to trap sensitive instructions and interrupt to Hyp Mode

Hardware Configurable to trap sensitive instructions and interrupt directly to VM's Kernel mode

CPU Virtualization

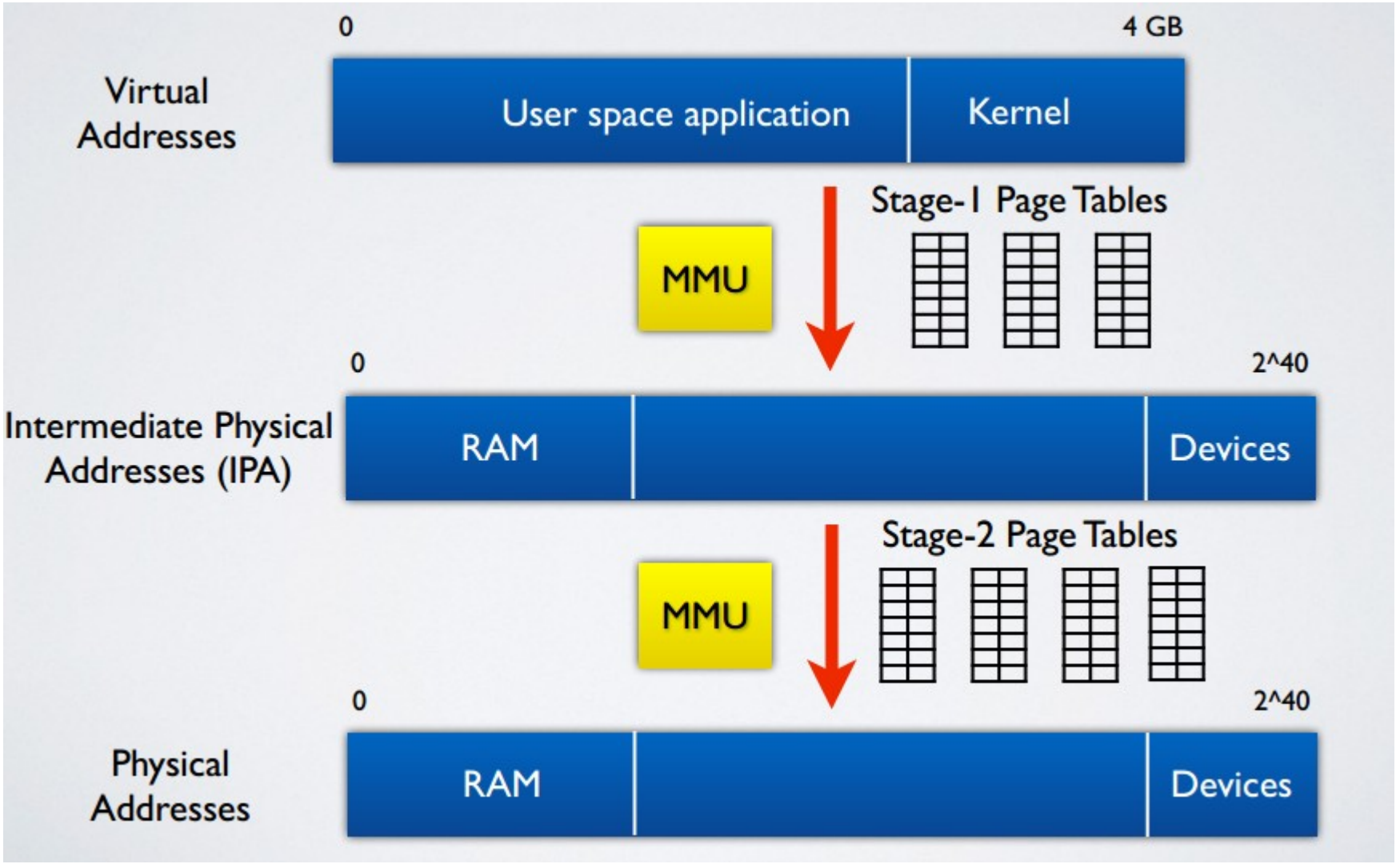
Hypervisor should be lightweight and simple to program



Hyp mode has its own separate address space

Page tables entries is protected from user mode, as they should not be shared with user mode

Memory Virtualization

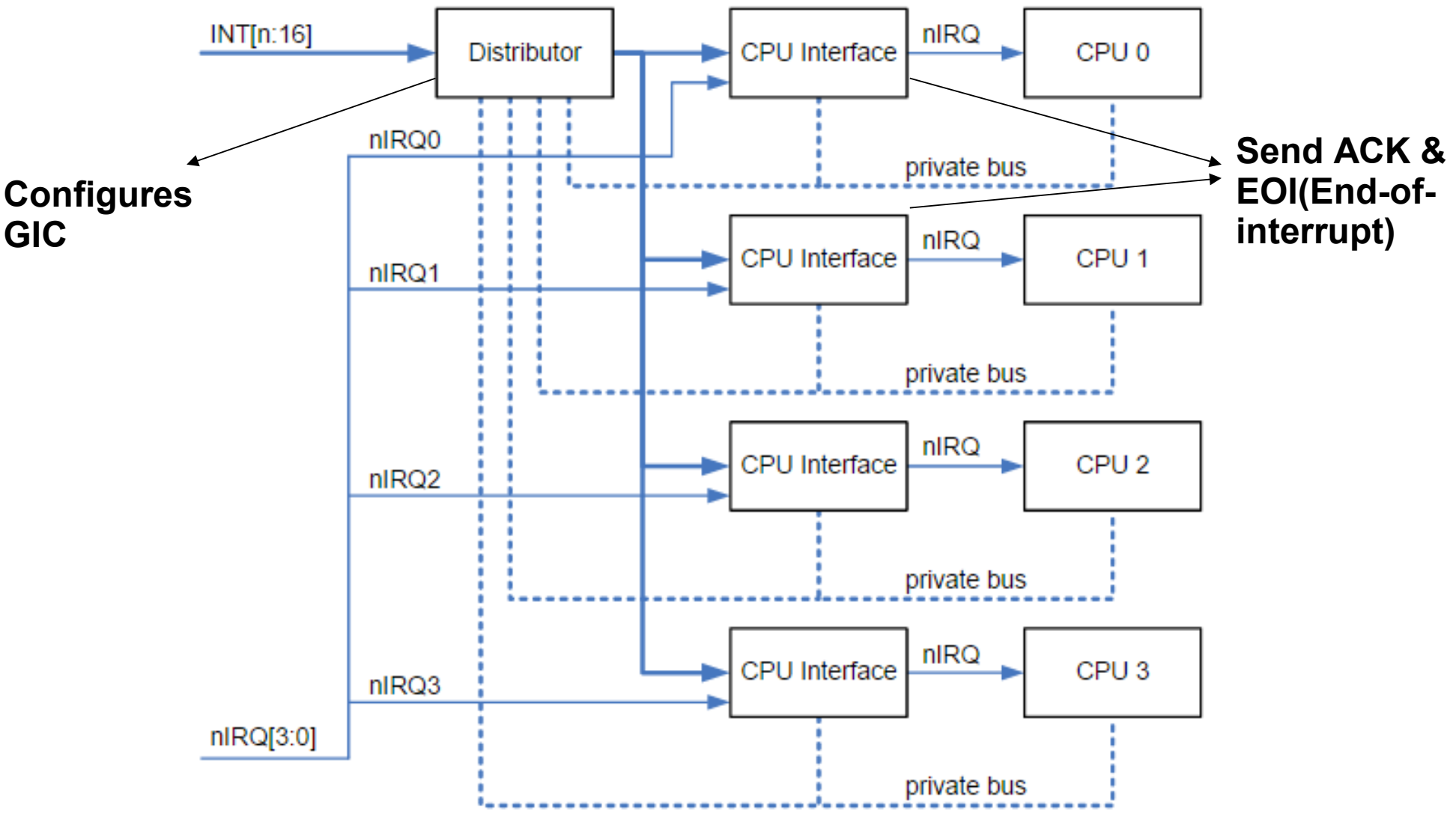


Hardware support to virtualize physical memory: Stage 2 Page Tables Enabled/Disabled from Hyp Mode

http://www.cs.columbia.edu/~cdall/pubs/KVMARM_talk.pdf

Interrupt Virtualization

Generic Interrupt Controller

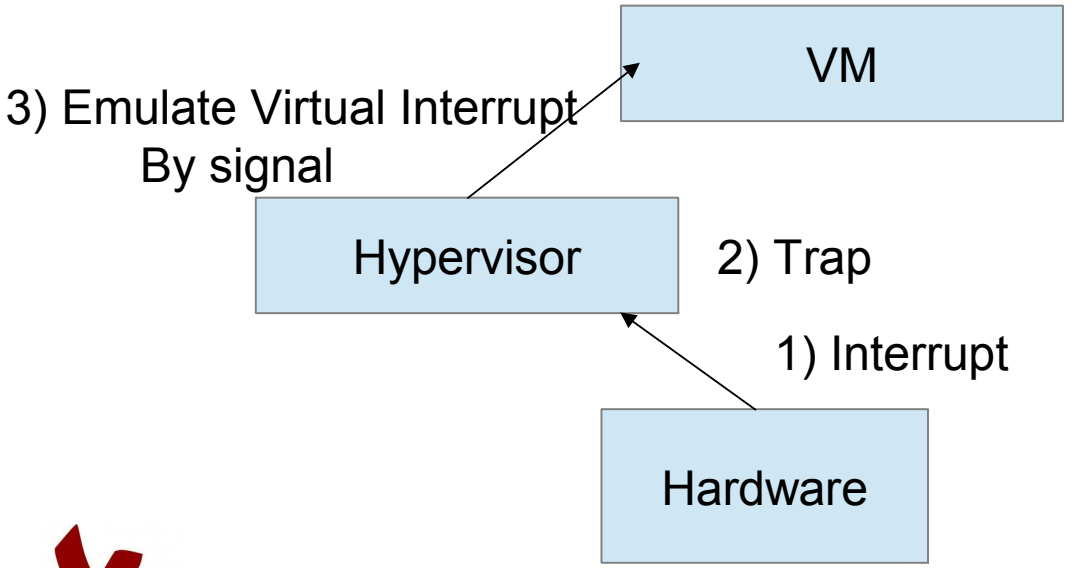


http://infocenter.arm.com/

Interrupt Virtualization

Generic Interrupt Controller

Trapping Interrupt in Hyp Mode



Cumbersome & Expensive

http://infocenter.arm.com/

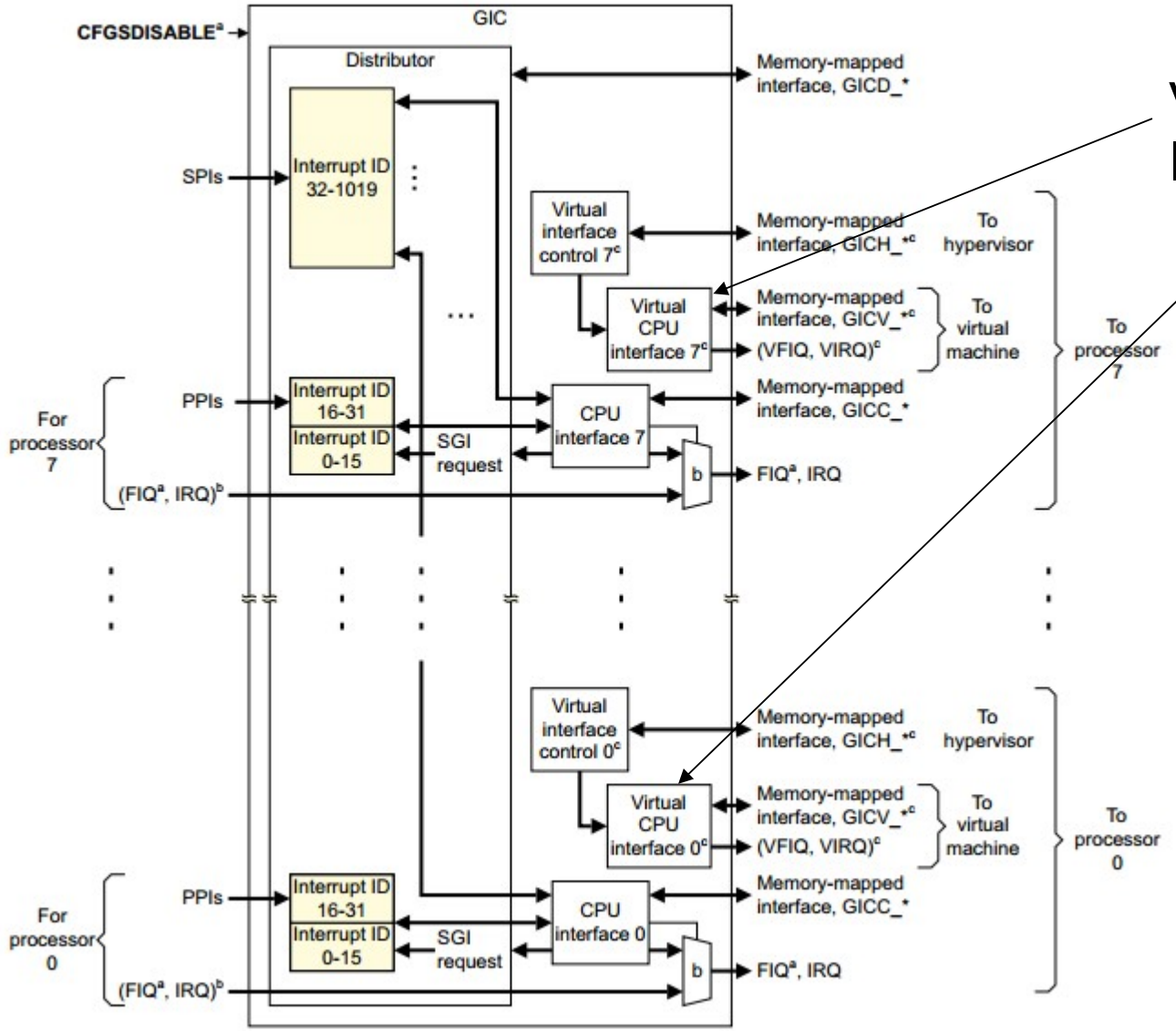
Interrupt Virtualization

Generic Interrupt Controller (V2.0)

Virtual GIC

Why Virtual Distributor is required?

But No Virtual Distributor



Virtual CPU Interfaces

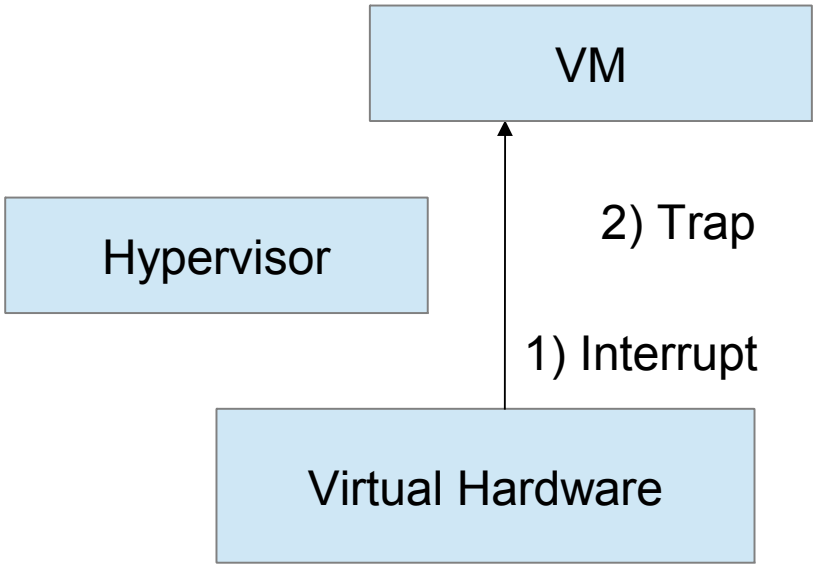
http://infocenter.arm.com/

Interrupt Virtualization

Generic Interrupt Controller (V2.0)

Virtual GIC

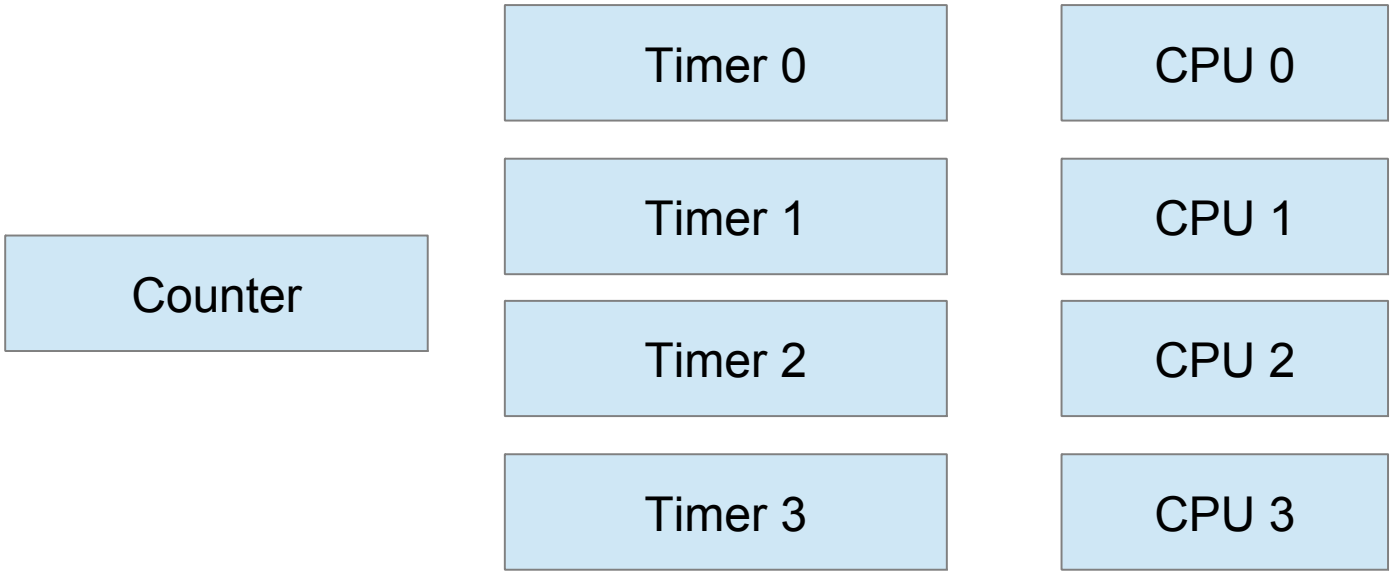
Trapping Interrupt in Kernel Mode



http://infocenter.arm.com/

Timer Virtualization

Generic Timer support



Can be configured from Hyp Mode

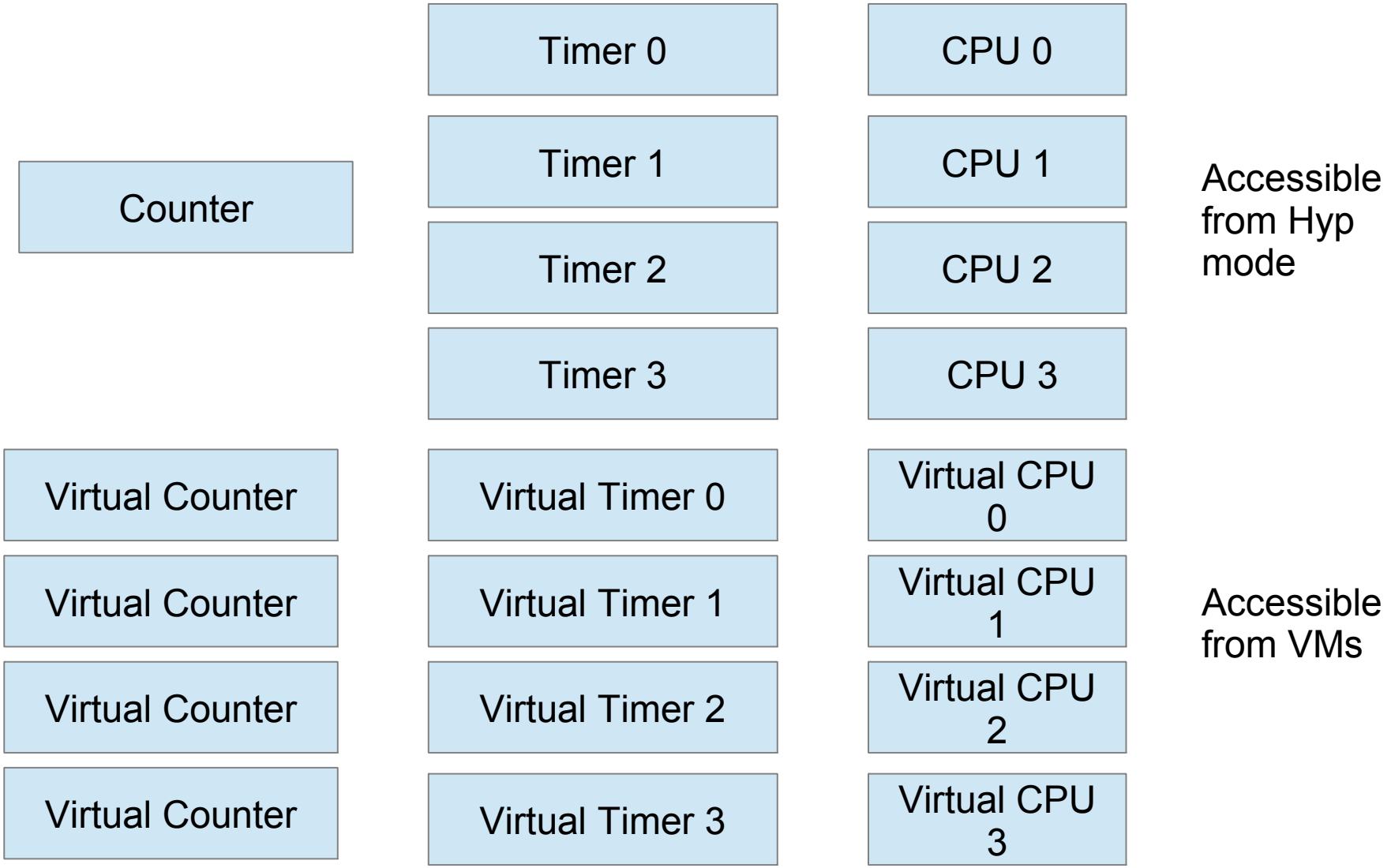
But requires Timer operations to be performed by Hyp

ARM introduces Virtual Timer and Counter

http://infocenter.arm.com/

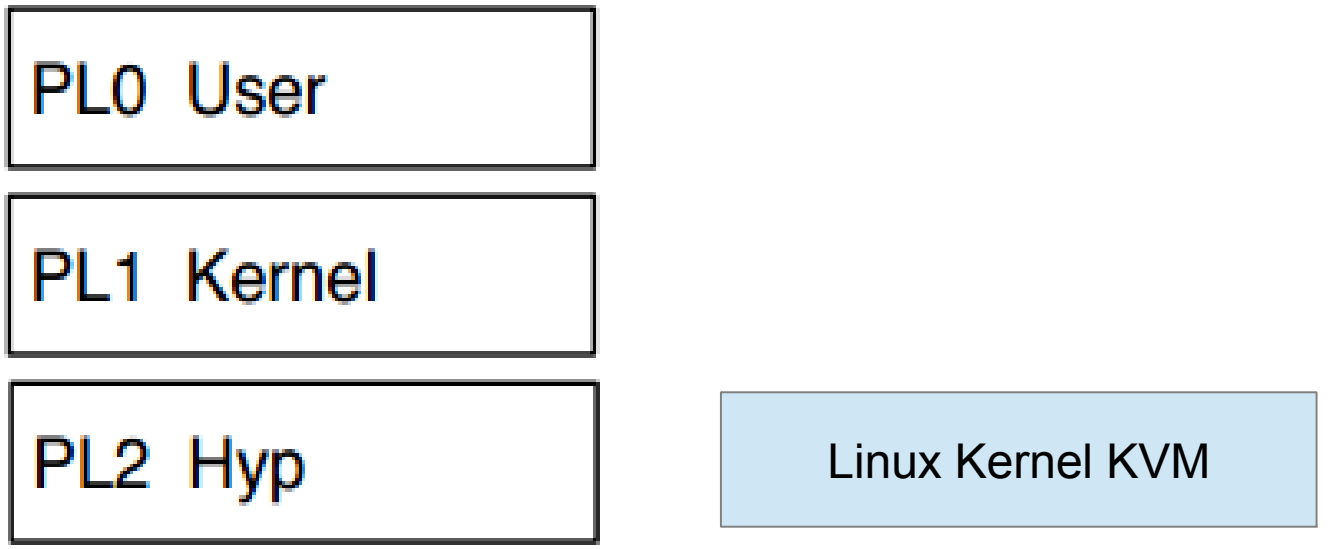
Timer Virtualization

Generic and Virtualized Timer support



Why Virtual Timer support is required?

Hypervisor Architecture



Firstly, Linux is written to work in kernel mode and would not run unmodified in Hyp Mode

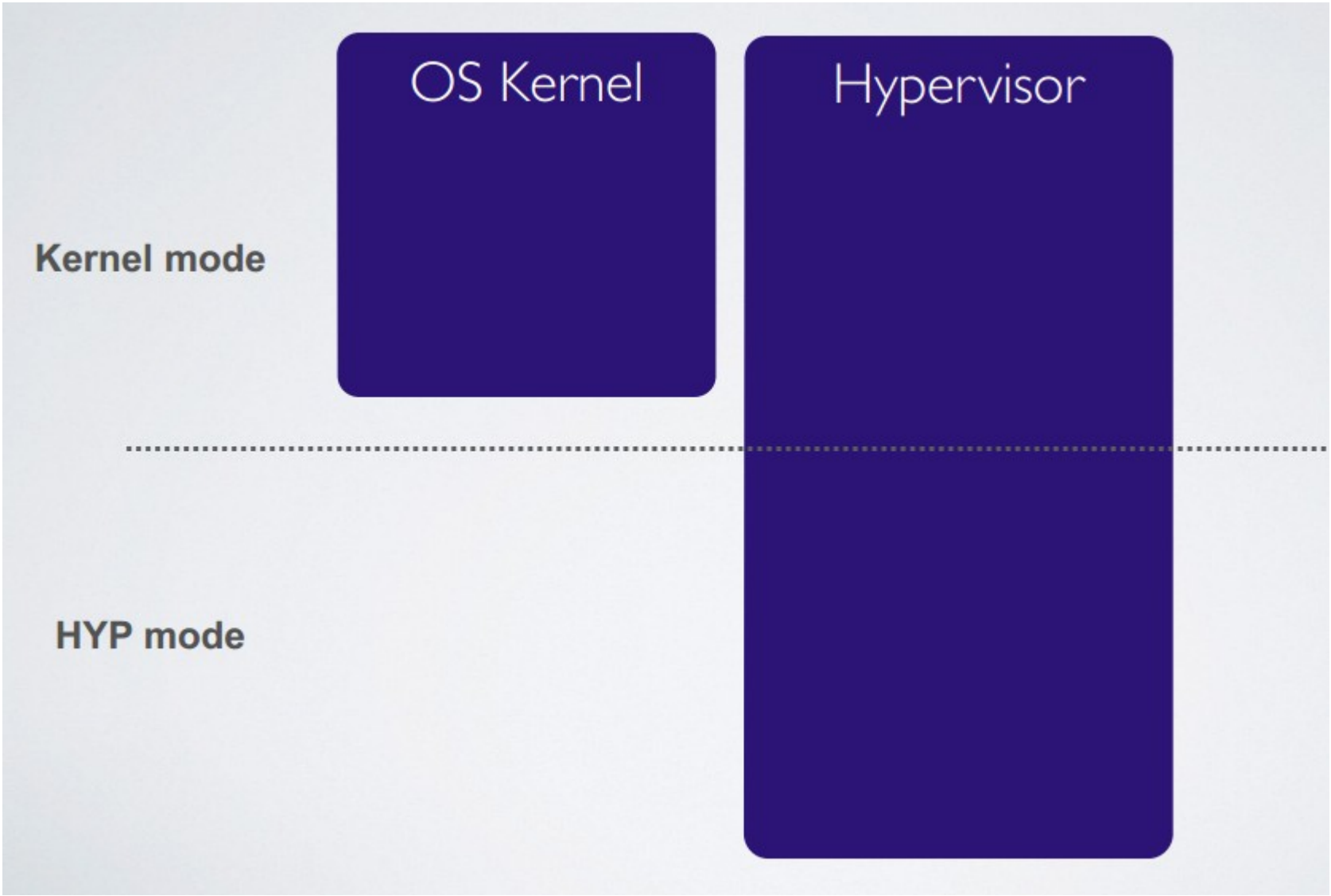
Some registers and table formats are different in Hyp mode than in kernel mode

Secondly, Running entire kernel in Hyp Mode would adversely affect native performance

Hyp mode has its own separate address space. This requires explicitly map user space data while accessing user space memory

Hypervisor Architecture

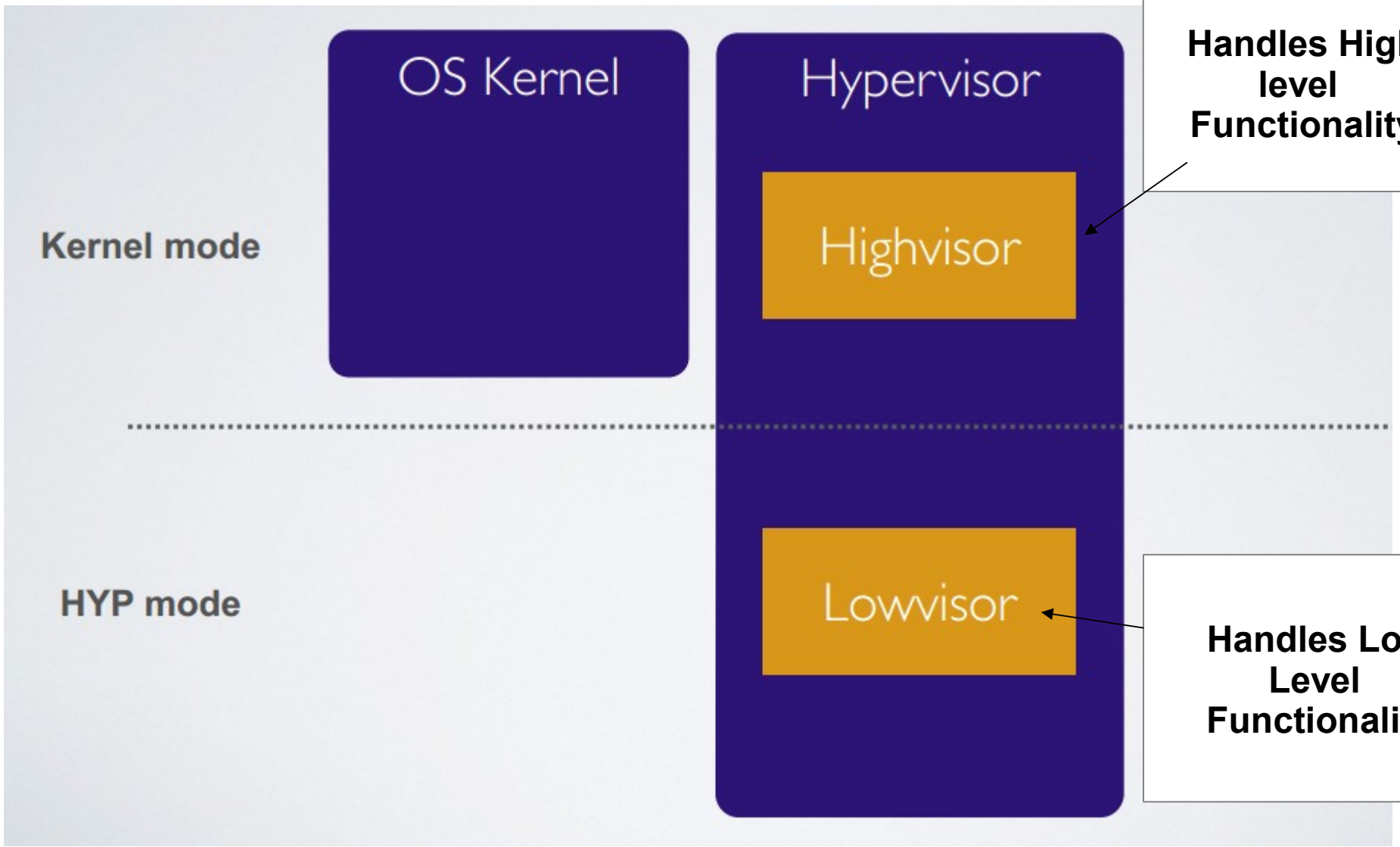
Split-Mode Virtualization



http://www.cs.columbia.edu/~cdall/pubs/KVMARM_talk.pdf

Hypervisor Architecture

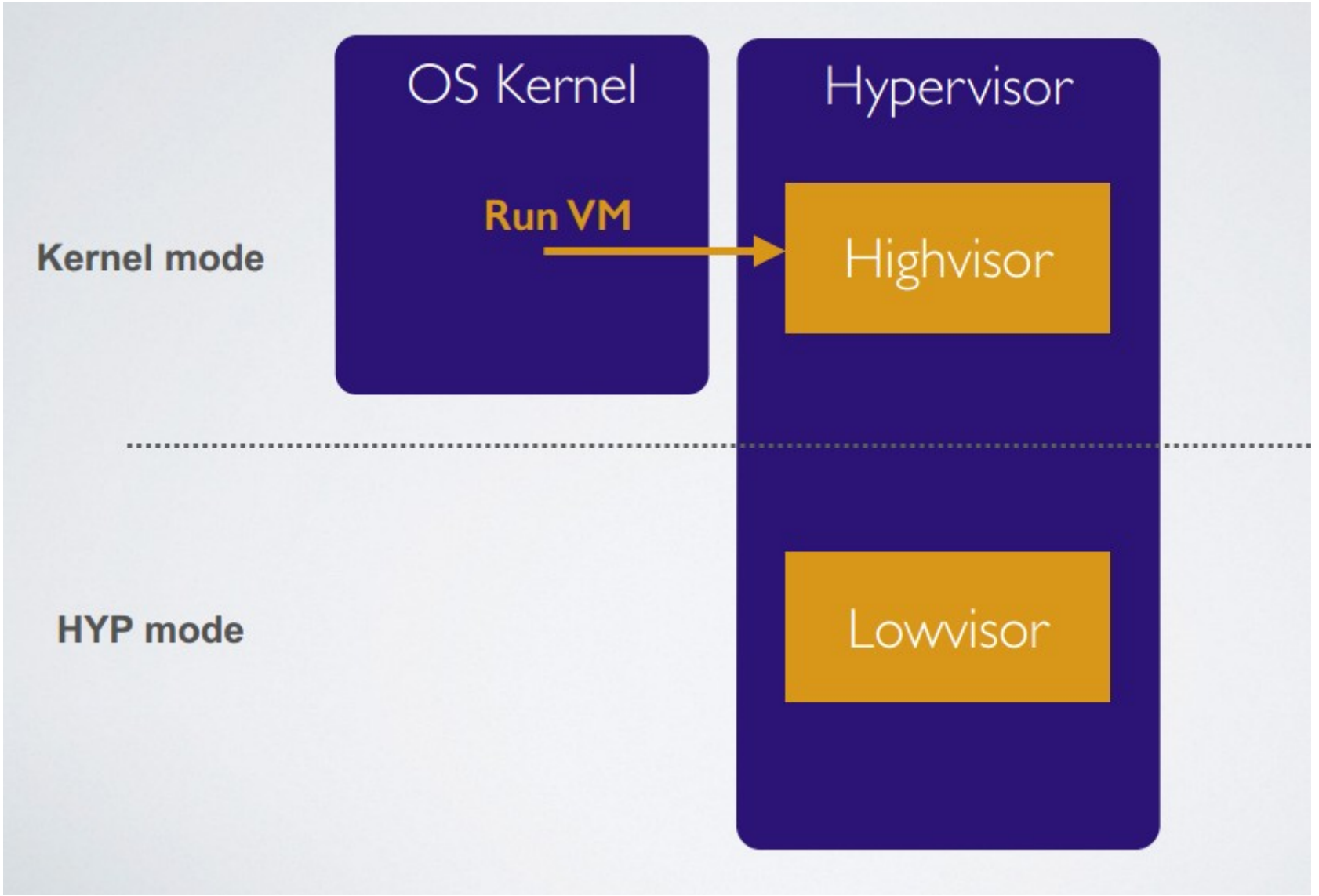
Split-Mode Virtualization



http://www.cs.columbia.edu/~cdall/pubs/KVMARM_talk.pdf

Hypervisor Architecture

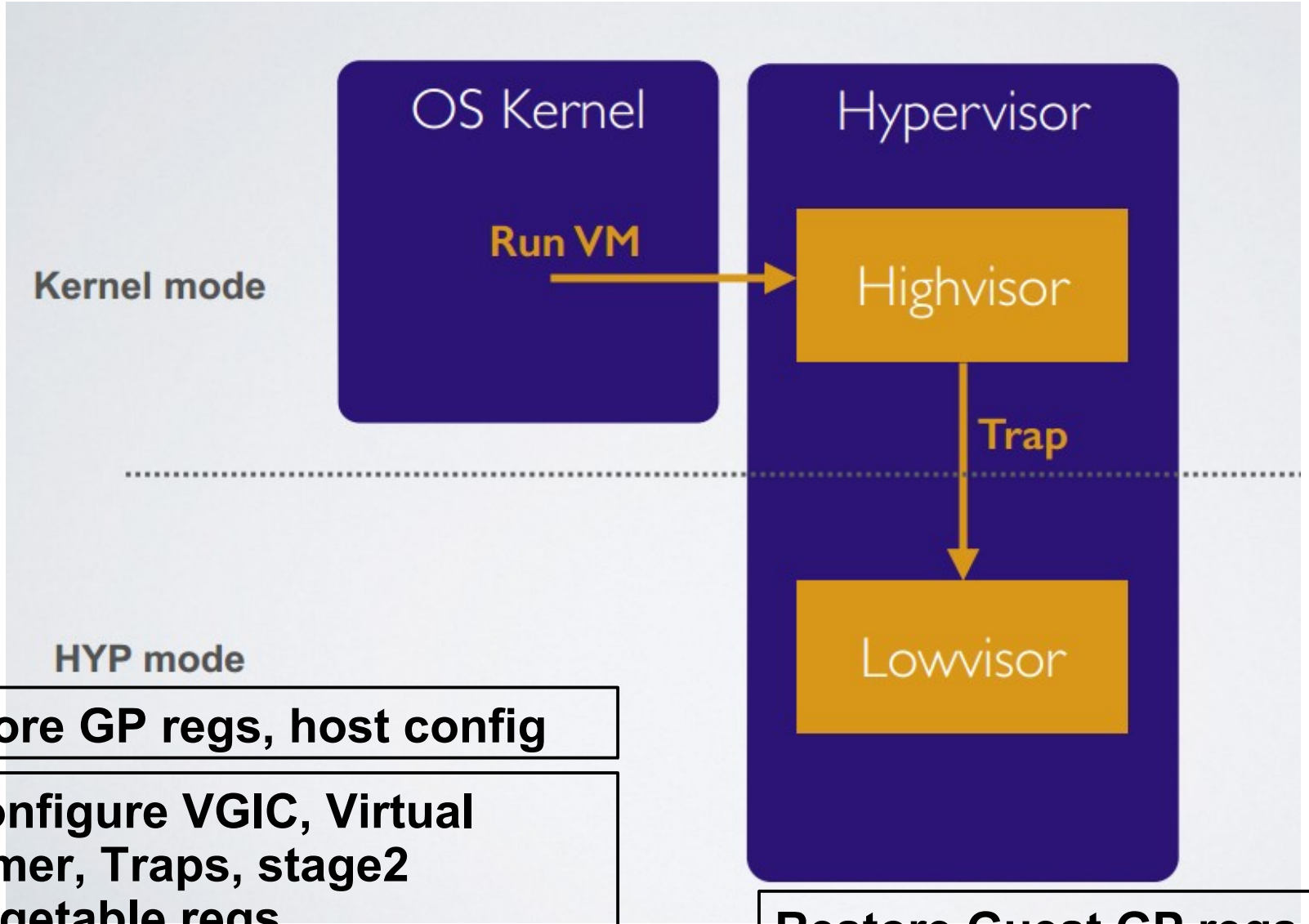
Split-Mode Virtualization



http://www.cs.columbia.edu/~cdall/pubs/KVMARM_talk.pdf

Hypervisor Architecture

Split-Mode Virtualization



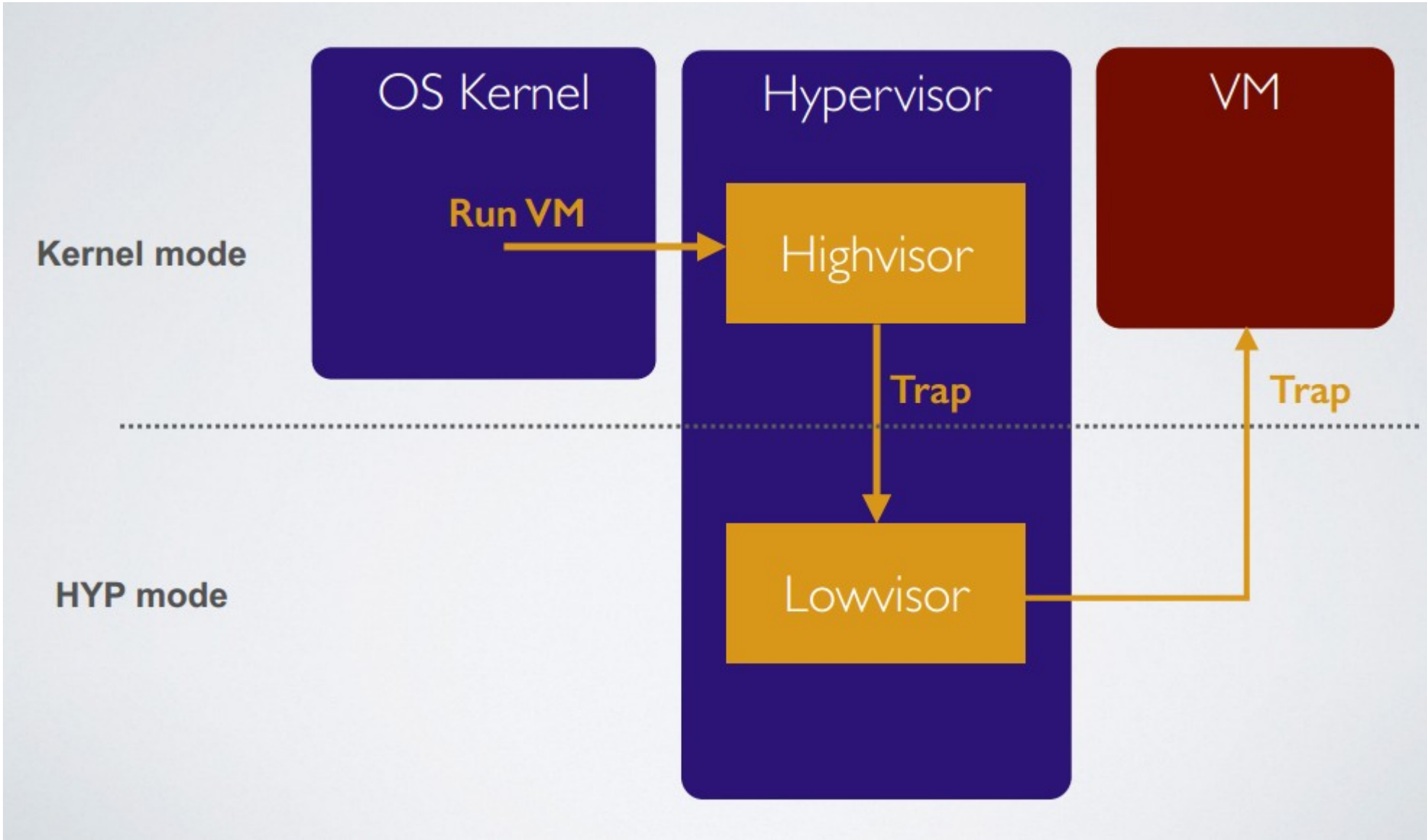
Store GP regs, host config

Configure VGIC, Virtual Timer, Traps, stage2 pagetable regs

Restore Guest GP regs, VM config

Hypervisor Architecture

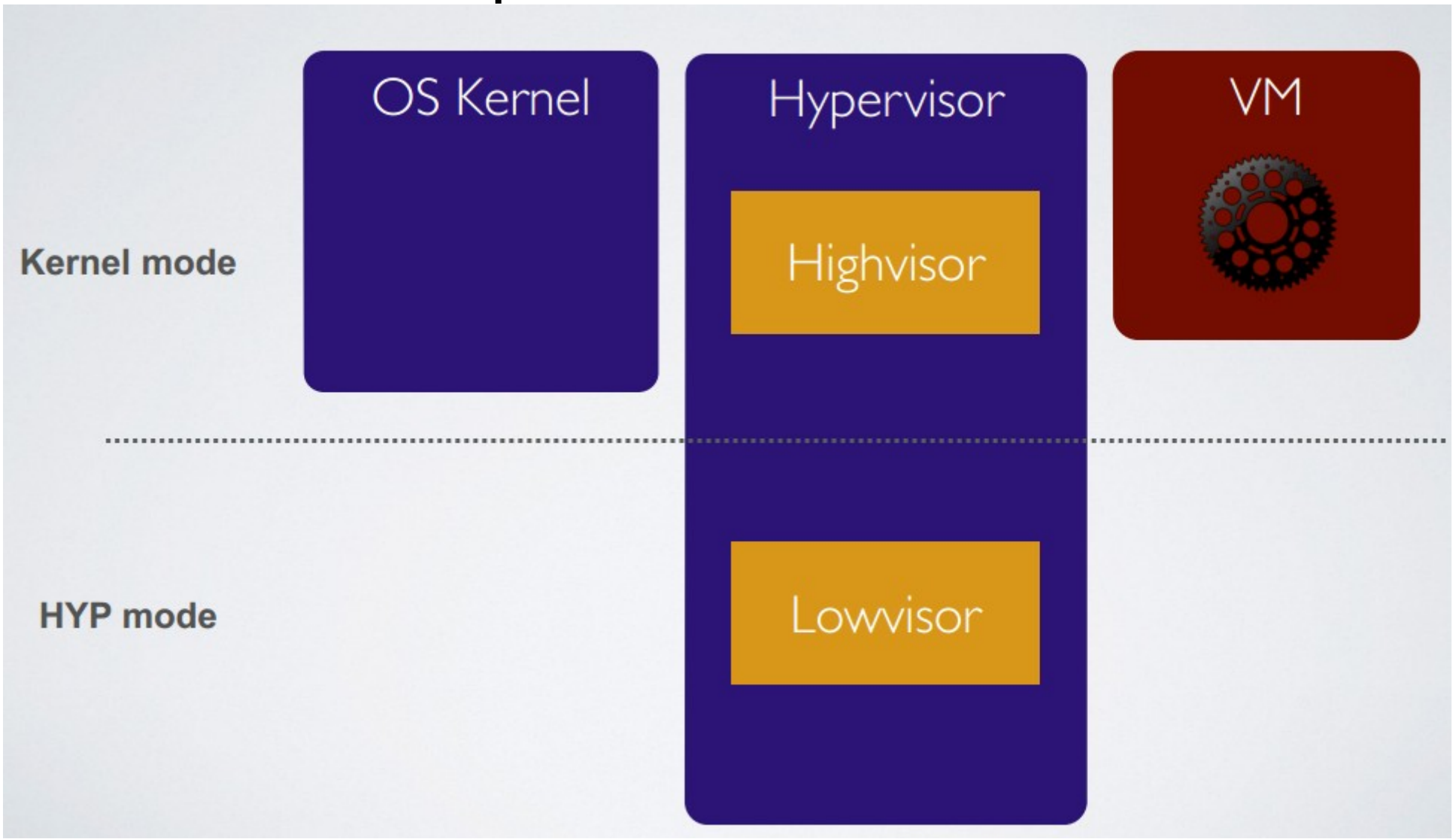
Split-Mode Virtualization



Lowvisor creates correct execution context by configuring hardware

Hypervisor Architecture

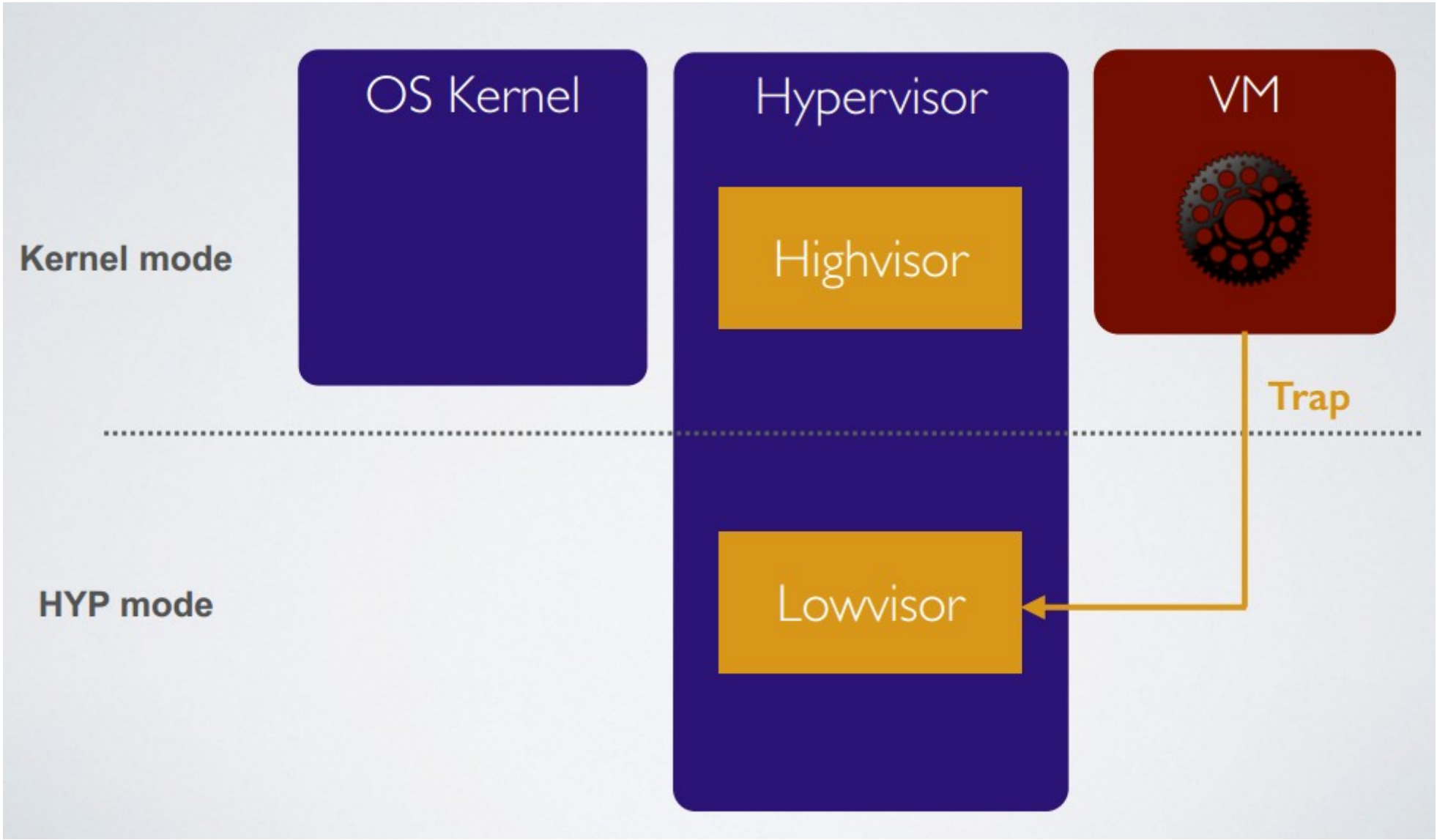
Split-Mode Virtualization



http://www.cs.columbia.edu/~cdall/pubs/KVMARM_talk.pdf

Hypervisor Architecture

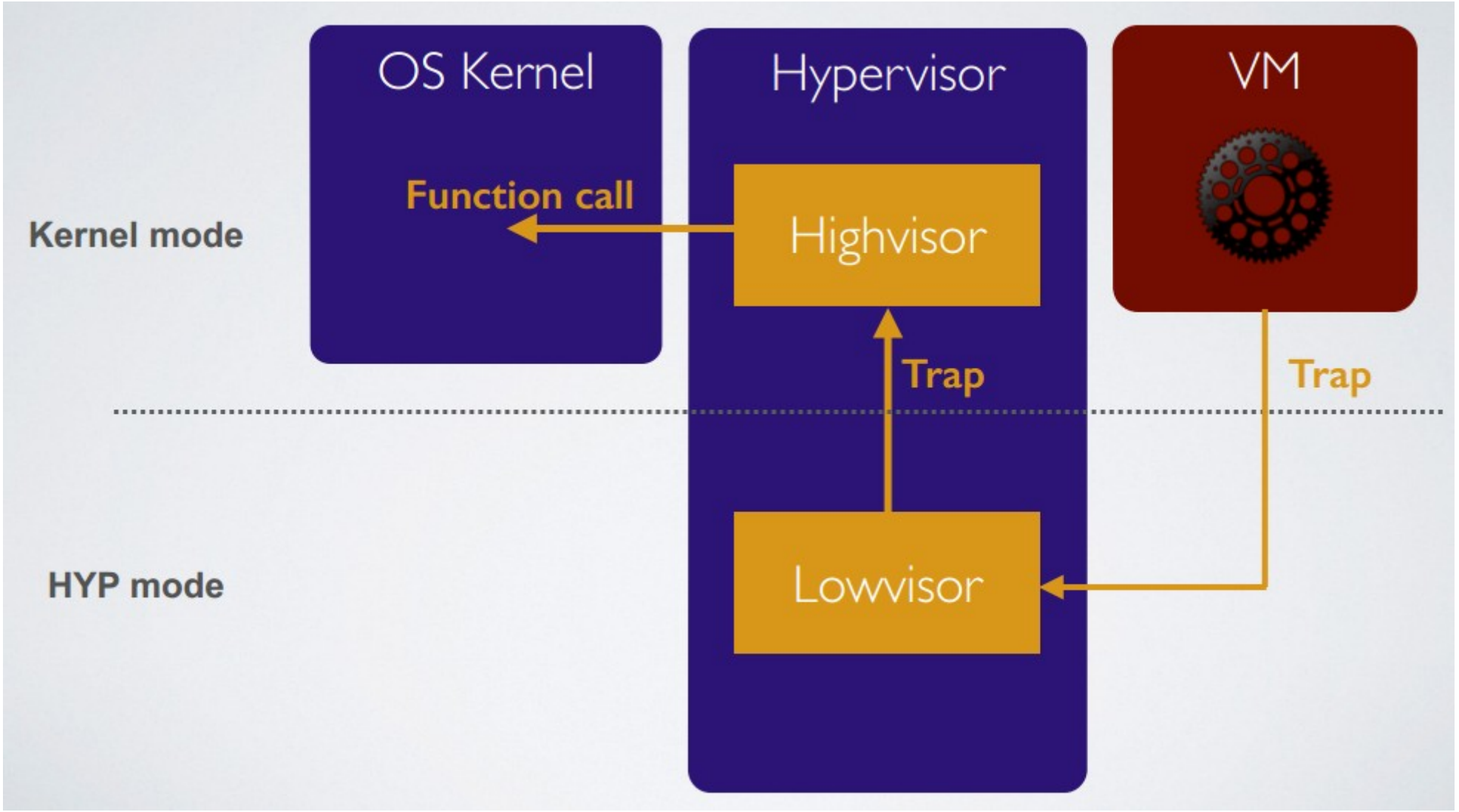
Split-Mode Virtualization



Lowvisor: Handles Interrupts and Exceptions

Hypervisor Architecture

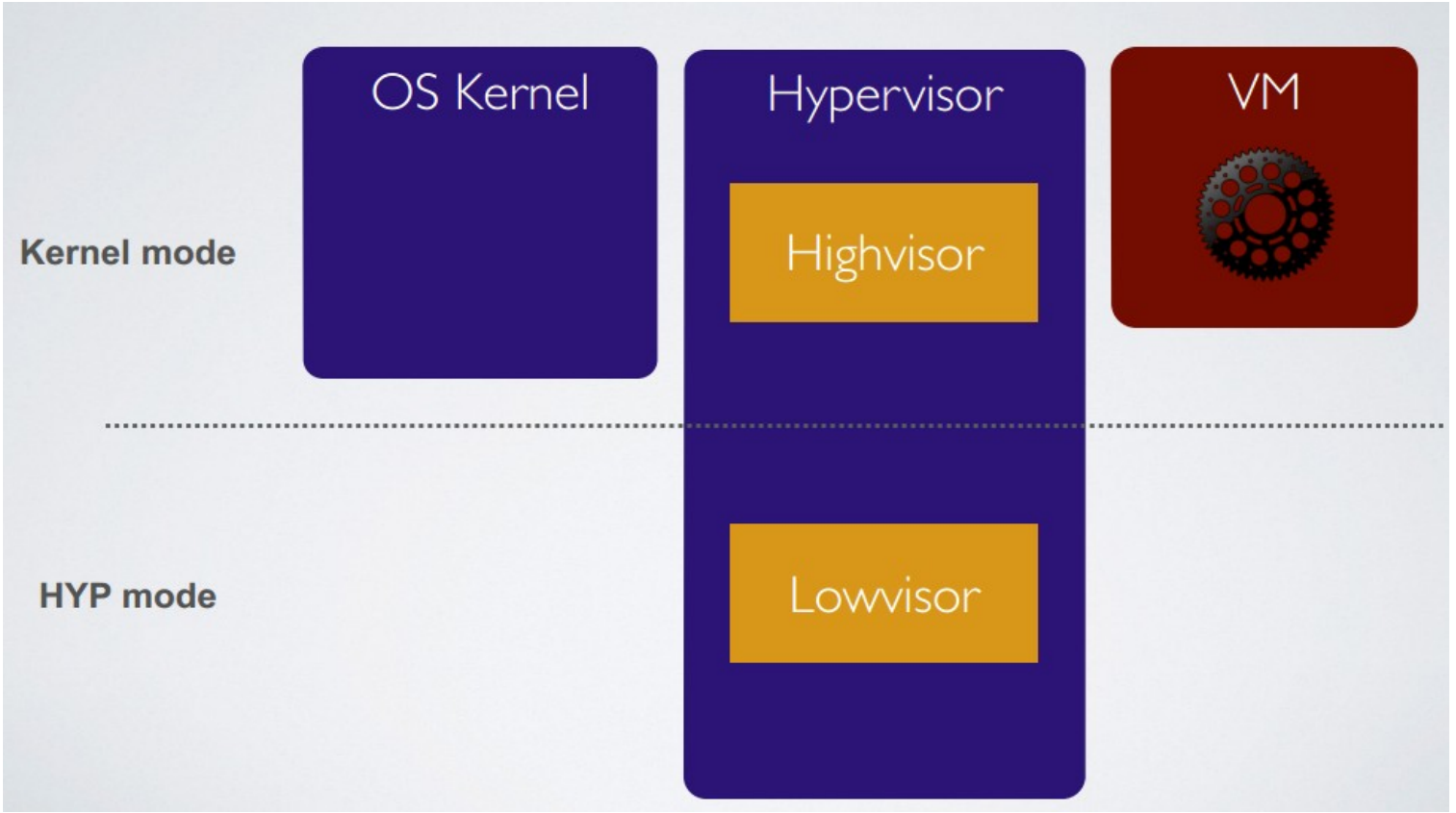
Split-Mode Virtualization



http://www.cs.columbia.edu/~cdall/pubs/KVMARM_talk.pdf

Hypervisor Architecture

What are the Functionalities of LowVisor?

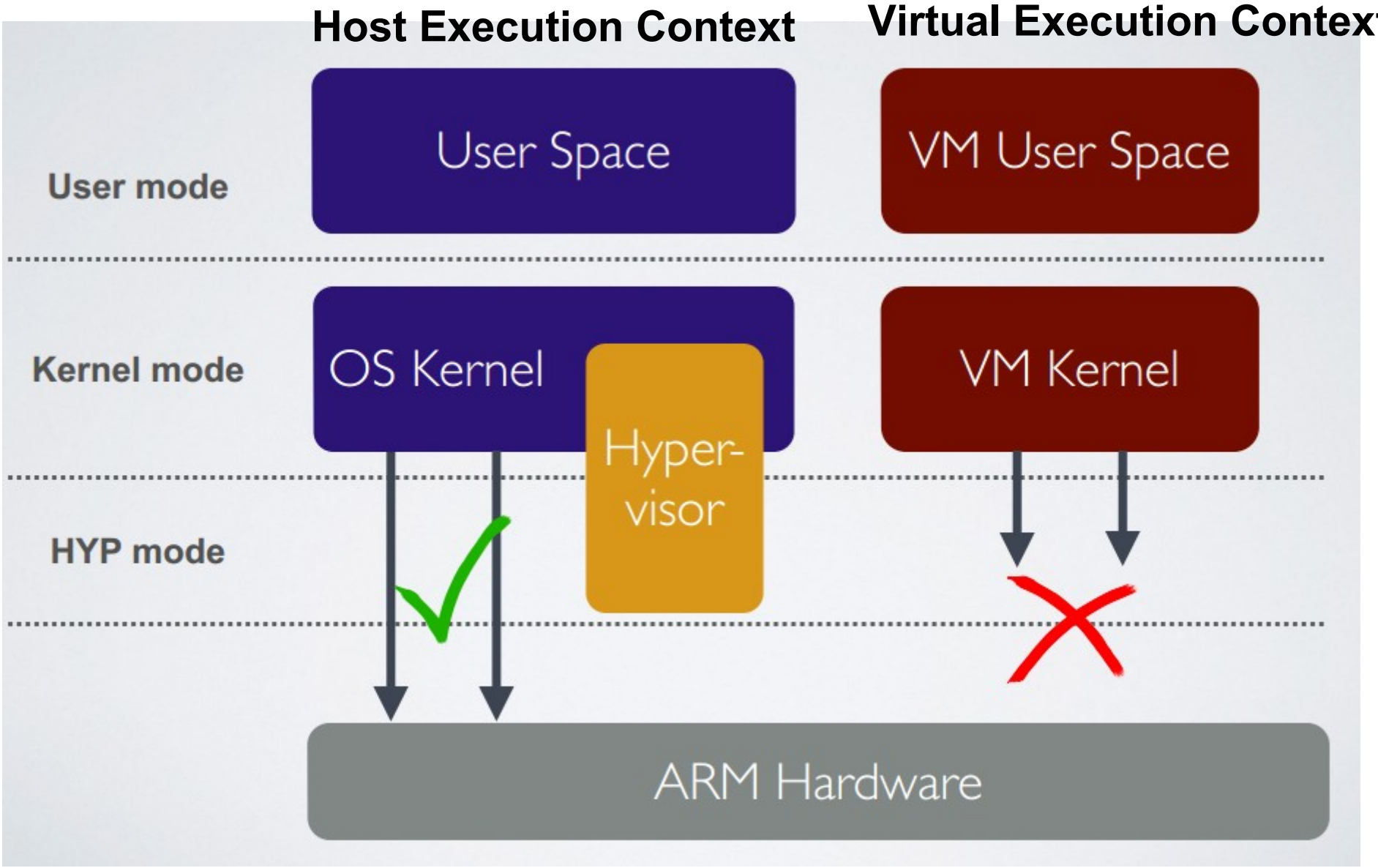


Trap handler: Handles Interrupt and Exception

Set Correct Execution context by configuring hardware

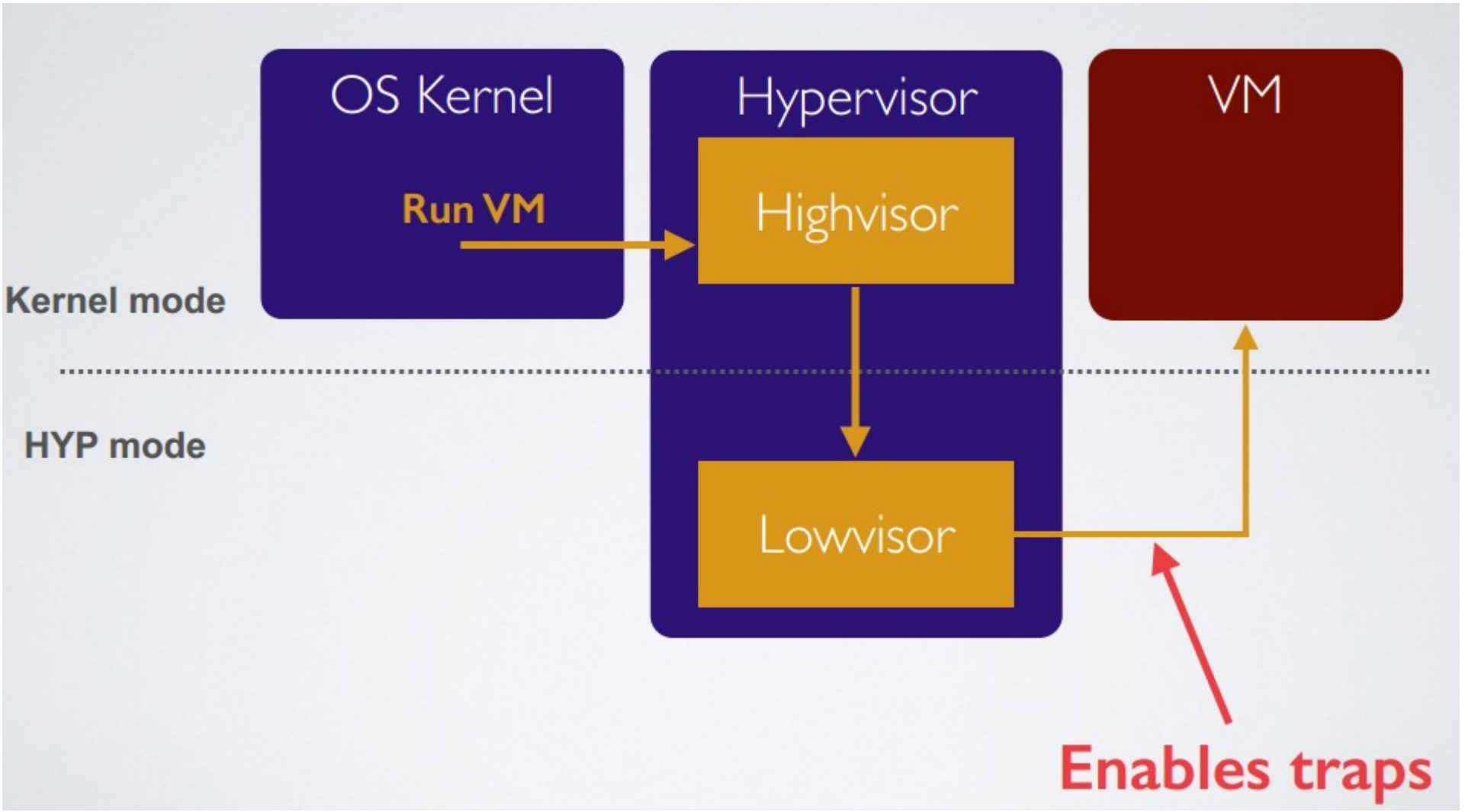
Perform Context Switch between Host and VM execution context

Hypervisor Architecture



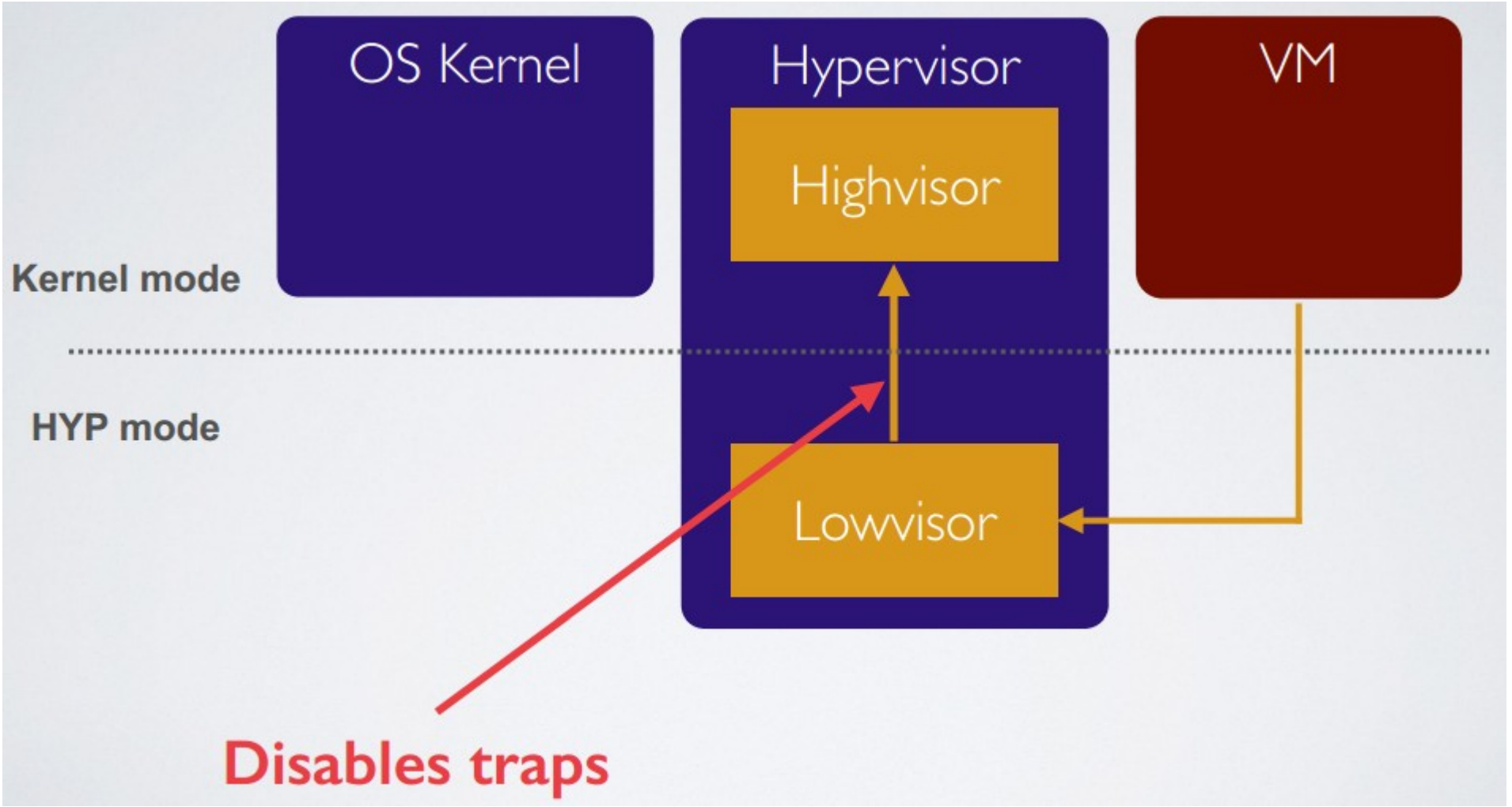
Perform Context Switch between Host and VM execution context

CPU Virtualization



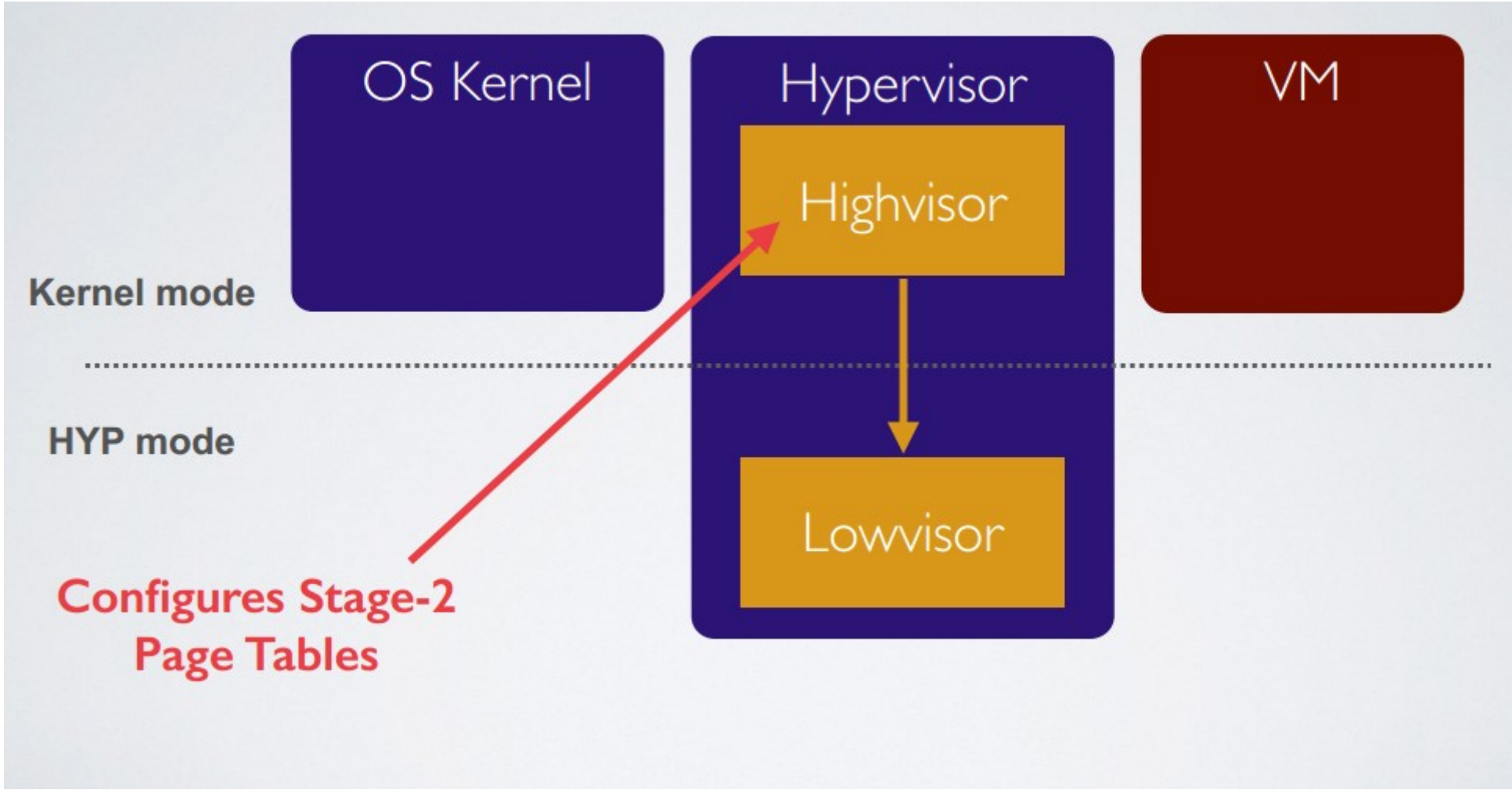
Perform Context Switch between Host and VM execution context

CPU Virtualization



Perform Context Switch between Host and VM execution context

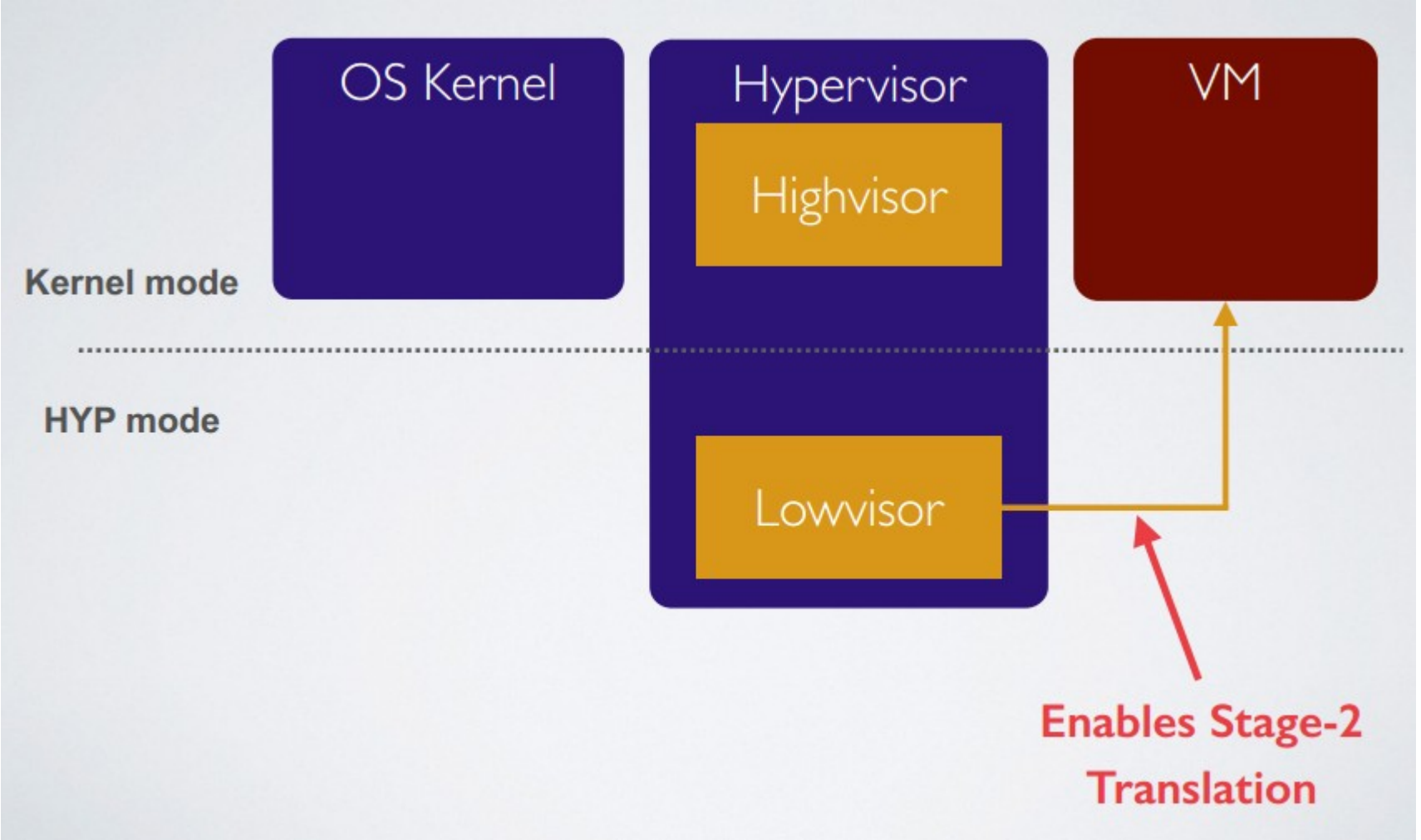
Memory Virtualization



Configuring page tables is a high level Functionality

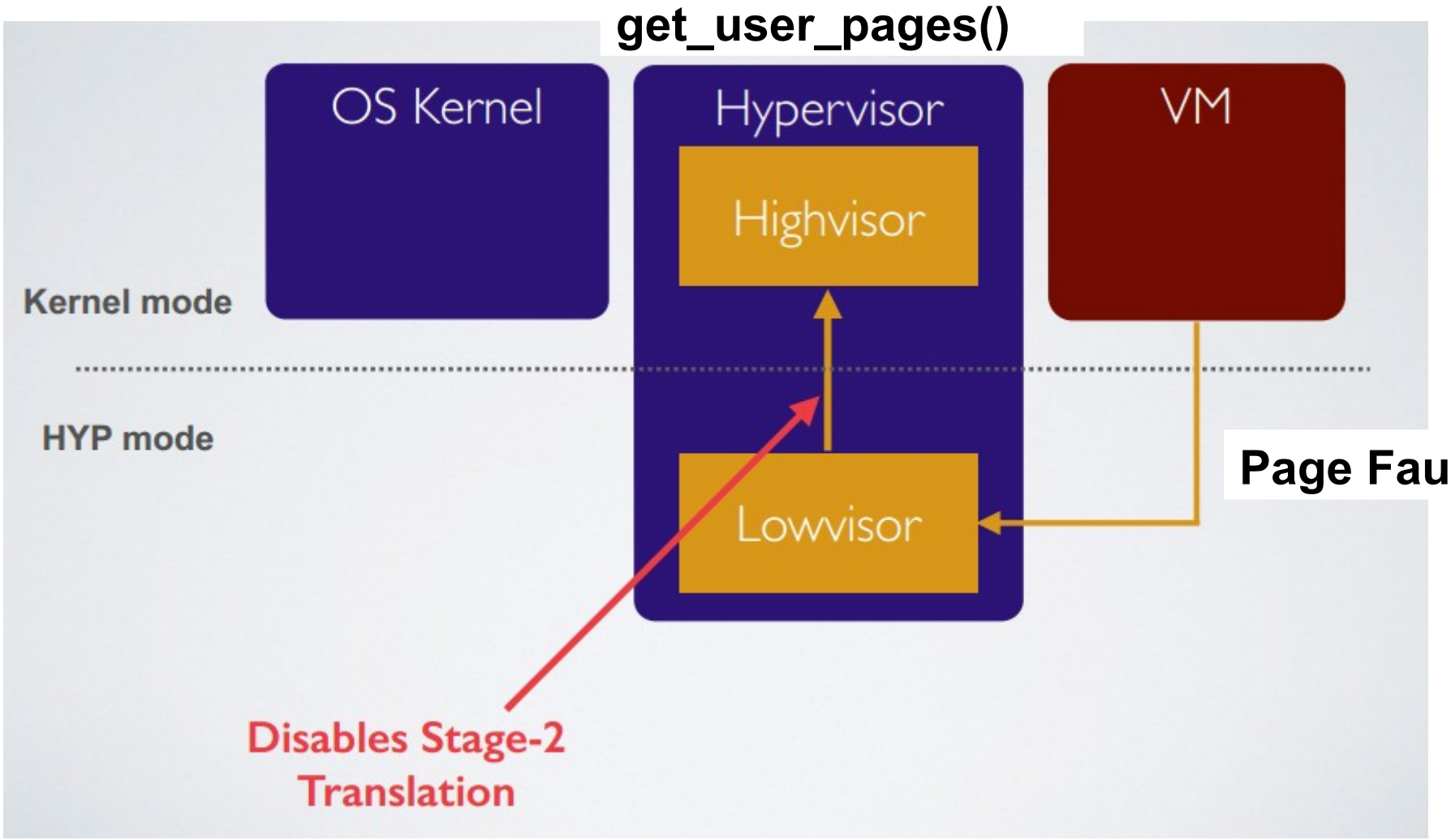
http://www.cs.columbia.edu/~cdall/pubs/KVMARM_talk.pdf

Memory Virtualization



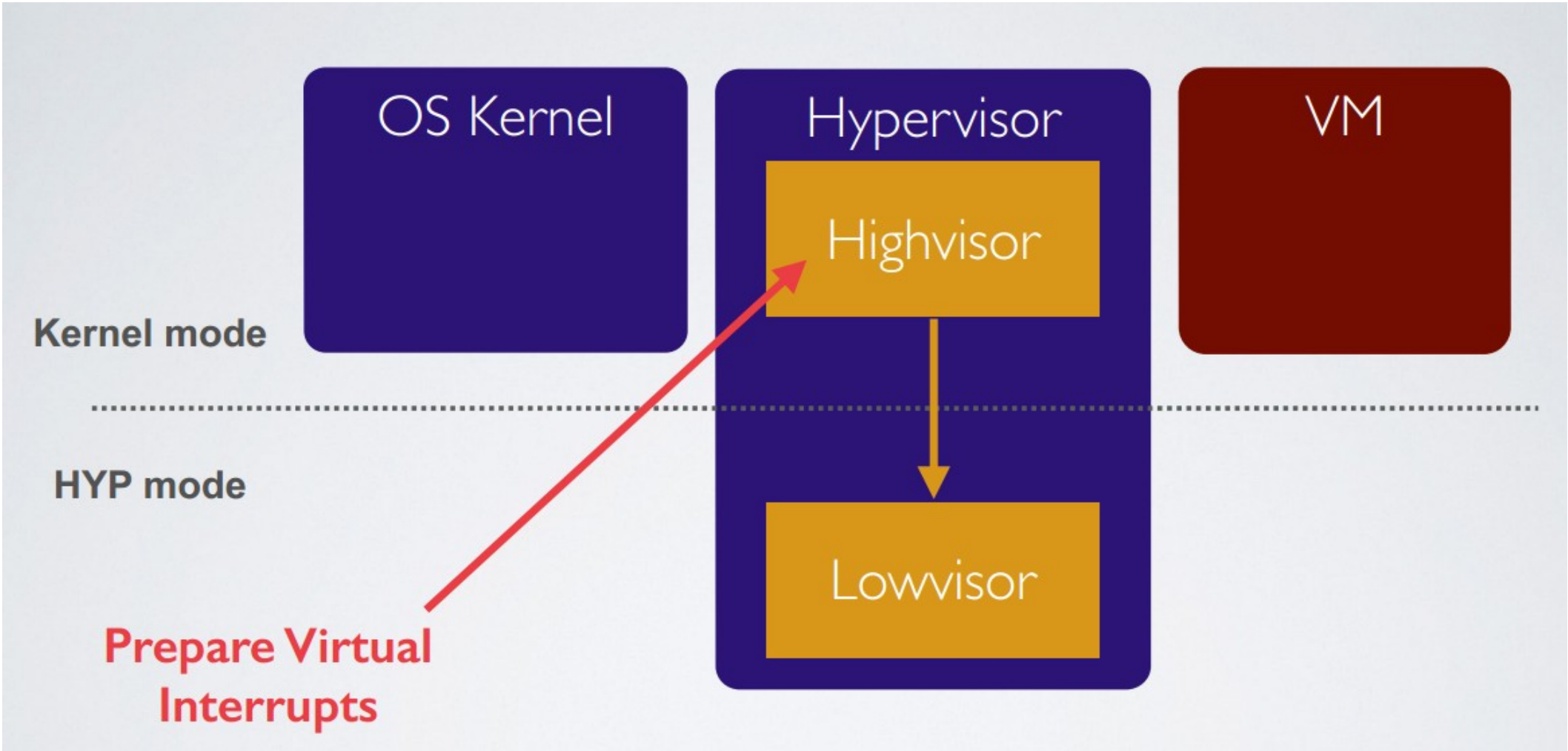
But LowVisor has hardware access as it runs in Hyp Mode

Memory Virtualization



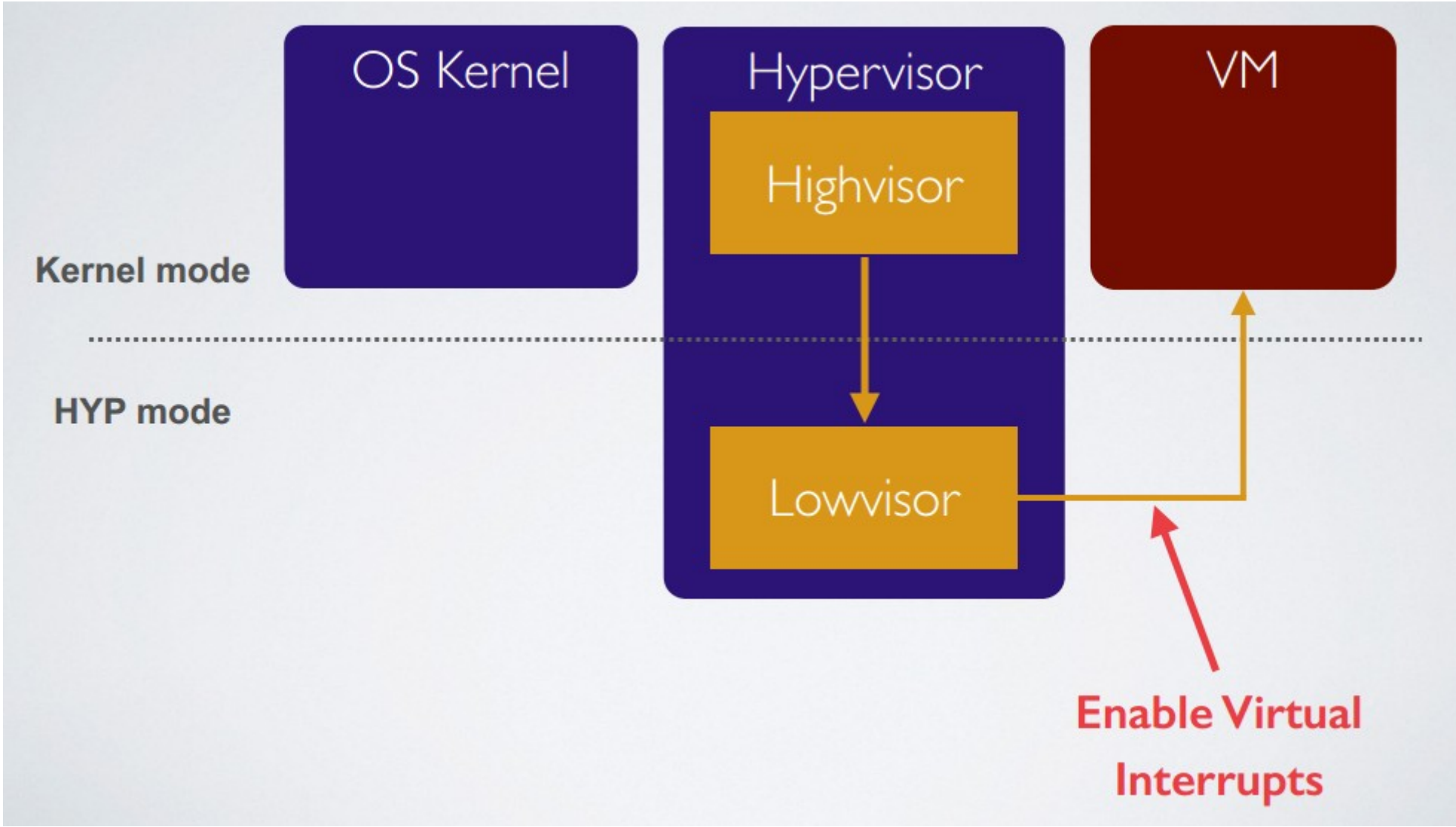
http://www.cs.columbia.edu/~cdall/pubs/KVMARM_talk.pdf

Interrupt Virtualization



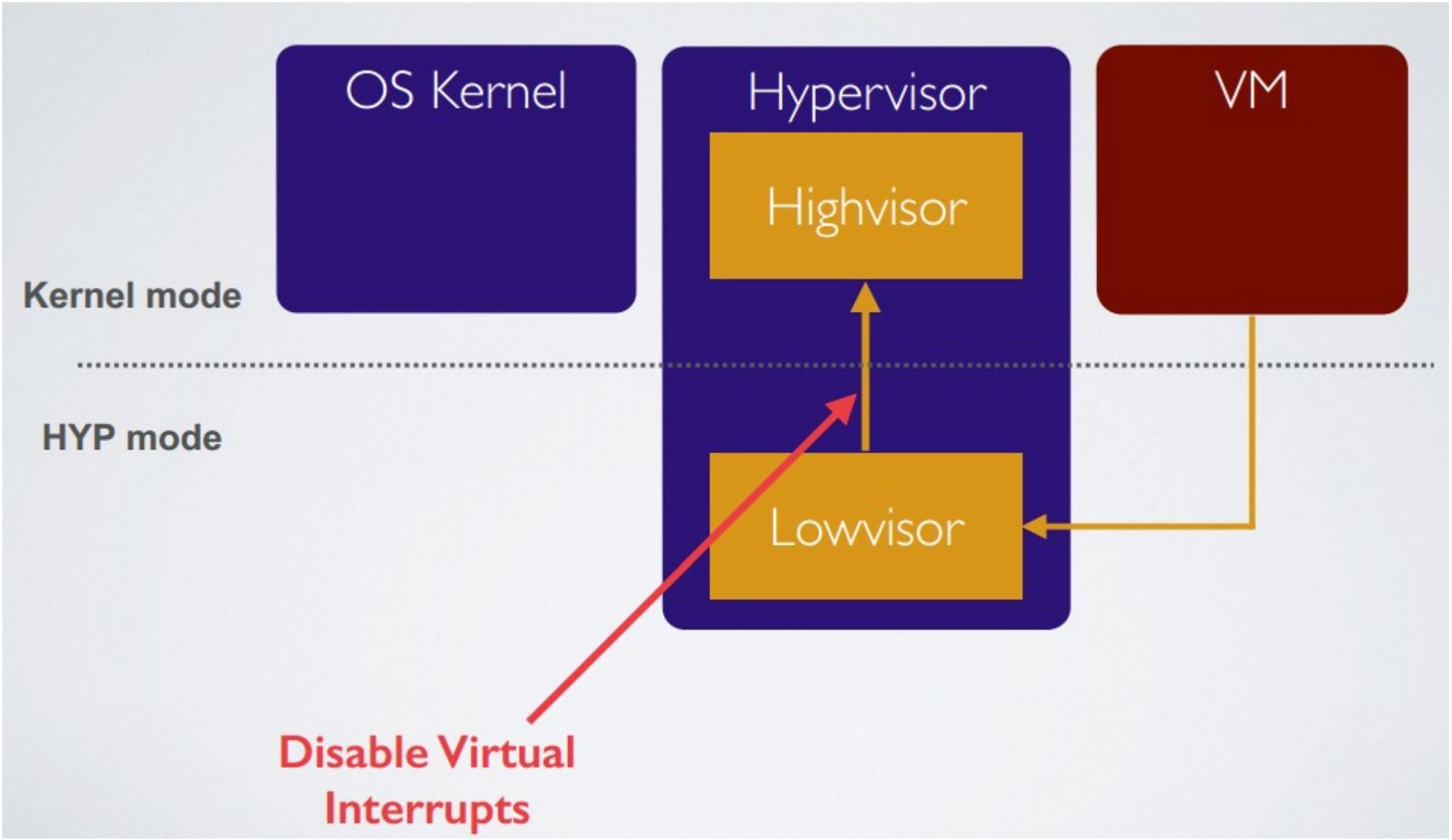
http://www.cs.columbia.edu/~cdall/pubs/KVMARM_talk.pdf

Interrupt Virtualization



http://www.cs.columbia.edu/~cdall/pubs/KVMARM_talk.pdf

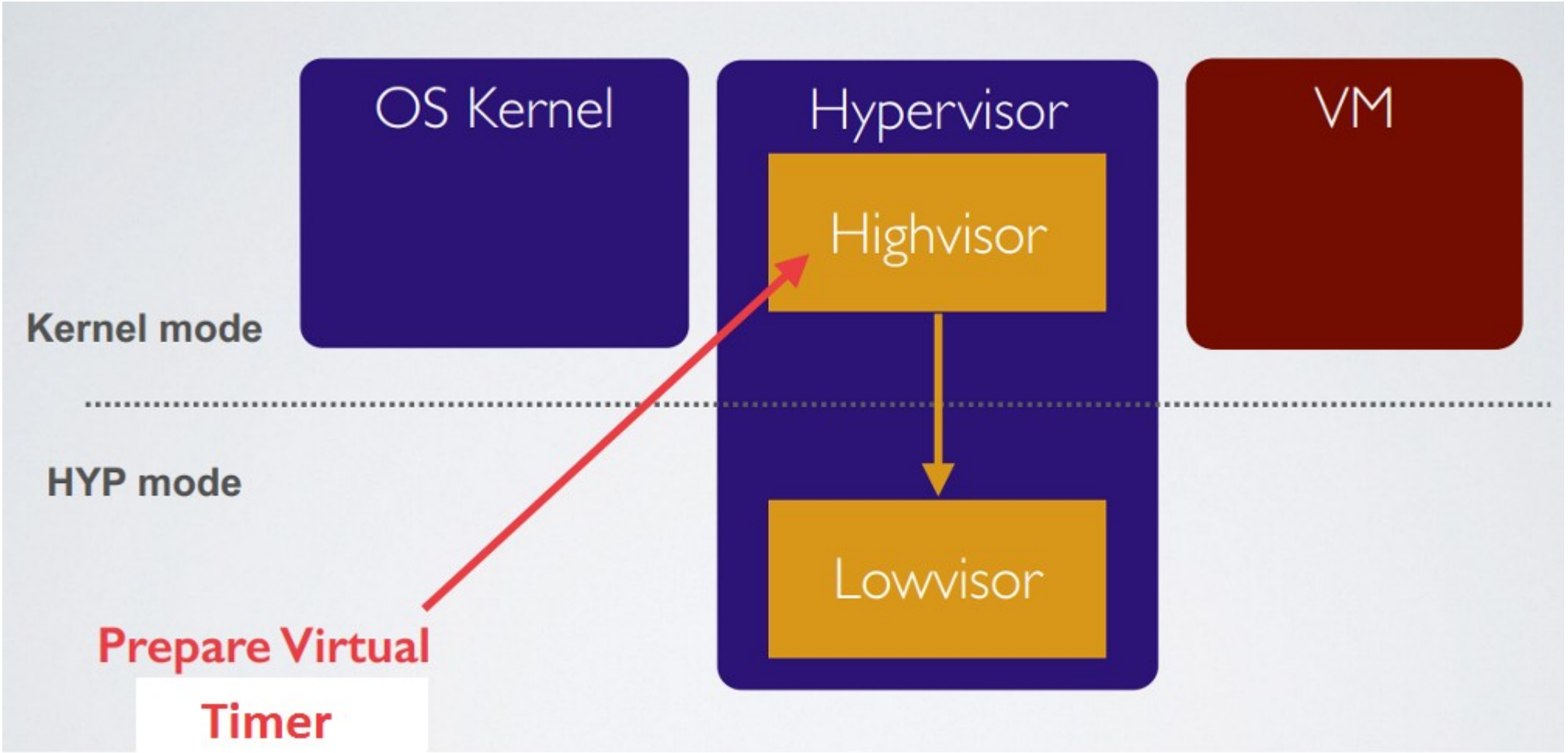
Interrupt Virtualization



KVM/ARM implements virtual GIC Distributor in software

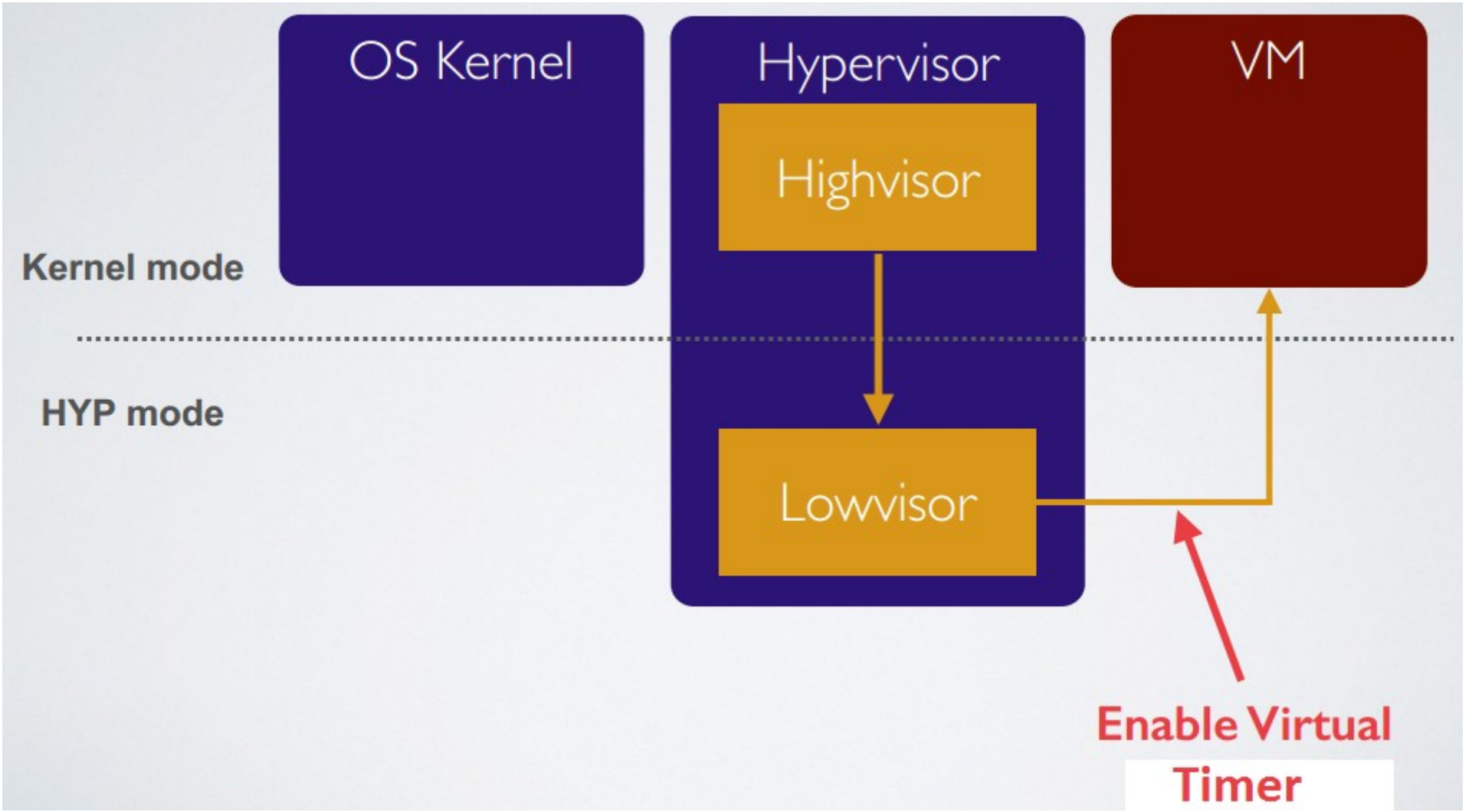
What happens when a virtual interrupt occurs?

Timer Virtualization



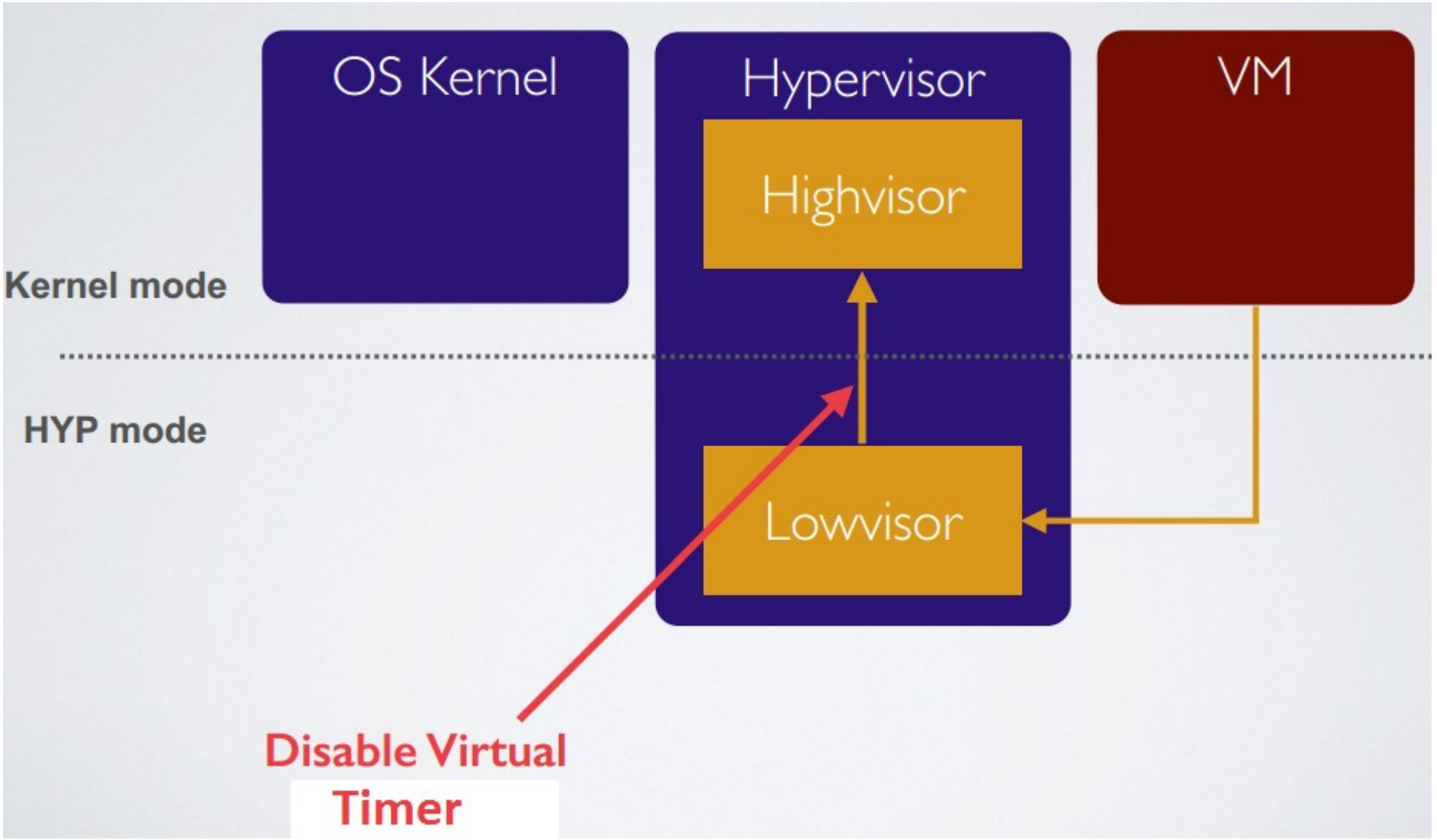
http://www.cs.columbia.edu/~cdall/pubs/KVMARM_talk.pdf

Timer Virtualization



http://www.cs.columbia.edu/~cdall/pubs/KVMARM_talk.pdf

Timer Virtualization



Virtual timers cannot directly raise virtual interrupts, but always raise hardware interrupts

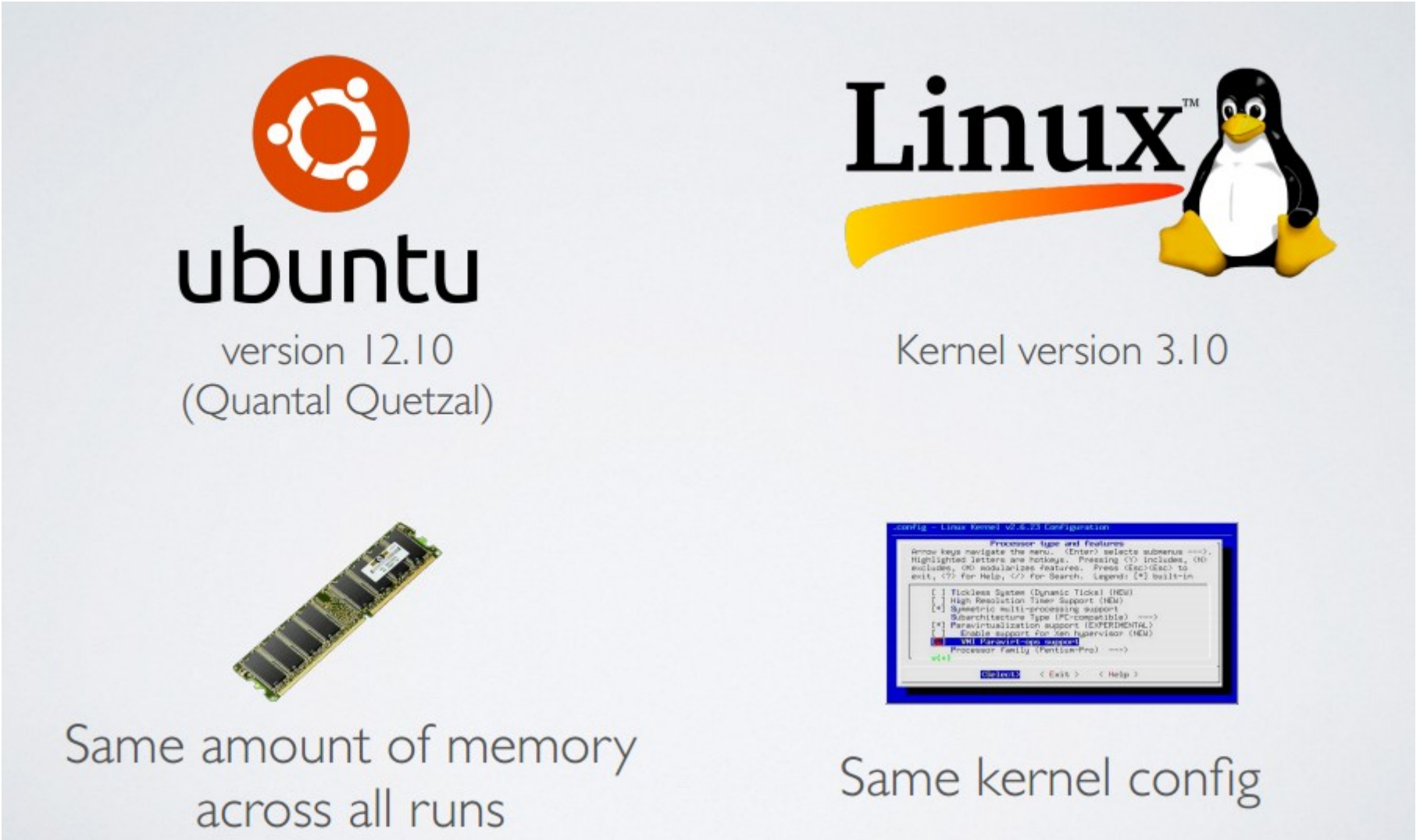
Implementation & Experimental Setup

Arndale Board - Exynos 5250
Dual-Core Cortex A-15 @ 1.4 GHz
eSATA SSD

MacBook Air
Dual-Core Intel i7 @1.8 GHz
SATA SSD

http://www.cs.columbia.edu/~cdall/pubs/KVMARM_talk.pdf

Implementation & Experimental Setup



http://www.cs.columbia.edu/~cdall/pubs/KVMARM_talk.pdf

Performance on Micro Benchmark

Micro Test	ARM	ARM no VGIC/vtimers	x86 laptop	x86 server
Hypercall	5,326	2,270	1,336	1,638
Trap	27	27	632	821
I/O Kern	5,990	2,850	3,19	3,291
I/O	10,119	6,704	10,9	12,218
	14,366	32,951	17,1	21,177
OI+ACK	427	13,726	2	2,305

Table 3: Micro-Architectural Cycl nts

Two world switches

Cost of saving and restoring VGIC states

hardware support to save and restore state

Performance on Micro Benchmark

Micro Test	ARM	ARM no VGIC/vtimers	x86 laptop	x86 server
Hypercall	5,326	2,270	1,336	1,638
Trap	27	27	632	821
I/O Kernel	5,990	2,850	3,190	3,291
I/O Use	10,119	6,704	10,9	12,218
IPI	14,366	32,951	17,	21,177
OI+ACK	427	13,726	2	2,305

Table 3: Micro-Architectural Cycl unts

Switching the hardware mode

Manipulate two registers to trap

hardware supported save and restore state

Performance on Micro Benchmark

Micro Test	ARM	ARM no VGIC/vtimers	x86 laptop	x86 server
Hypercall	5,326	2,270	1,336	1,638
Trap	27	27	632	821
I/O Kernel	5,990	2,850	3,190	3,291
I/O User	10,119	6,704	10,985	12,218
IPI	14,366	32,951	17,138	21,177
I+ACK	427	13,726	2,043	2,305

Table 3: Micro-Architectural Cycle Cou

cost of an I/O operation from the VM to a device

saving and restoring VGIC

x86 KVM saves and restores additional state lazily

hardware supported save and restore state

Performance on Micro Benchmark

Micro Test	ARM	ARM no VGIC/vtimers	x86 laptop	x86 server
Hypercall	5,326	2,270	1,336	1,638
Trap	27	27	632	821
I/O Kernel	5,990	2,850	3,190	3,291
I/O User	10,119	6,704	10,985	12,218
IPI	14,366	32,951	17,138	21,177
EOI+ACK	427	13,726	2,043	2,305

Table 3: Micro-Architectural Cycle[illegible]unts

cost of Inter-processor Interrupt

Hardware supported VGIC

EOI/ACK must be handled in Root mode in x86

Recommendation to Hardware Designers

Share Kernel Mode Memory Model in Hyp Mode

- Current implementation is different and require extra effort for memory management

Make VGIC state access fast, or at least infrequent

- Cause overhead because of slow MMIO access. Summary registers describing virtual interrupt state could be an improvement

Completely avoid IPI traps

- As Virtual Distributors are implemented in software, this cause some overhead.

Summary

KVM/ARM was the first system to run unmodified guests on hardware

Facilitated by split-mode virtualization

Comparison with KVM

Questions?