Optimizing the Design and Implementation of the Linux ARM Hypervisor

Christoffer Dall, Shih-Wei Li, Jason Nieh

Computer Science Department



Virtualization

- Essential technology for servers and cloud computing
- Support running multiple virtual machines on a piece of hardware
- Efficient partitioning and utilization of hardware resources

Virtualization on ARM









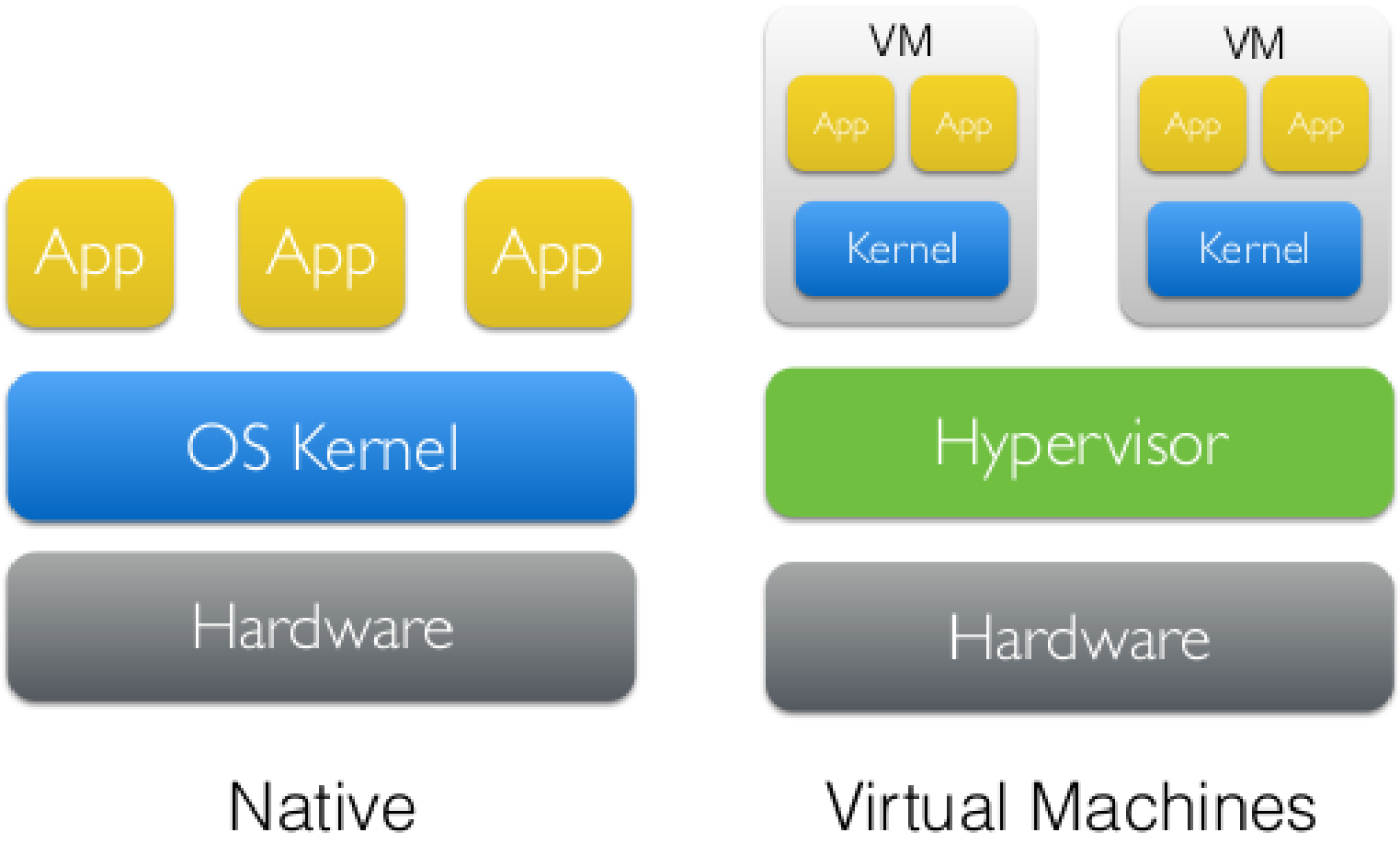


ARM Servers

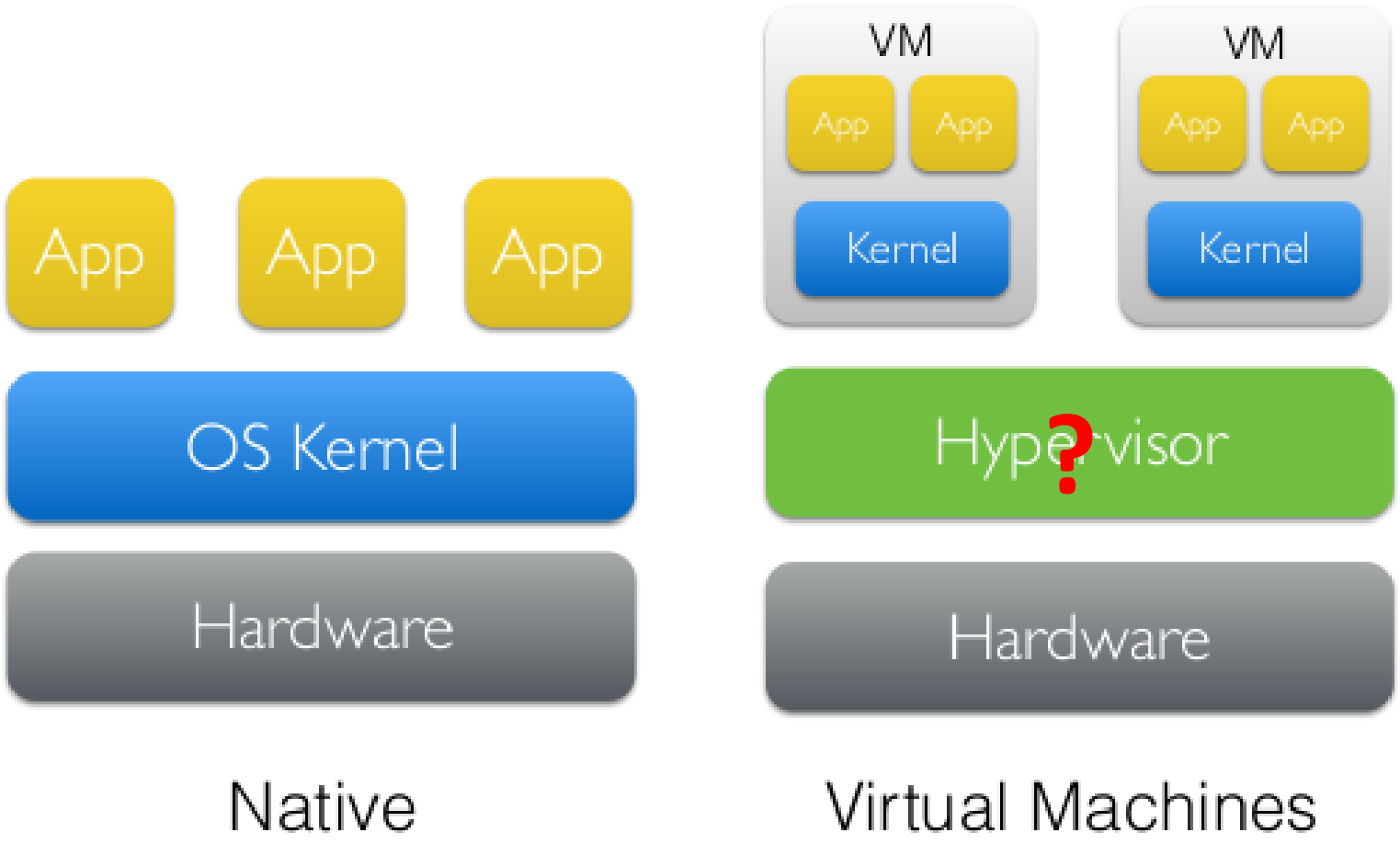
ARM®

Virtualization Extensions

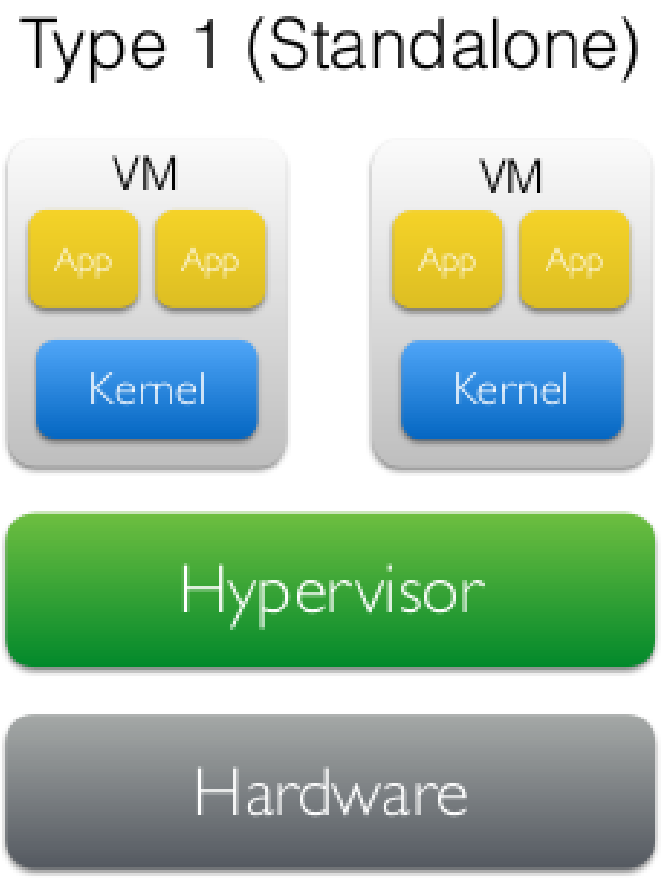
Virtualization



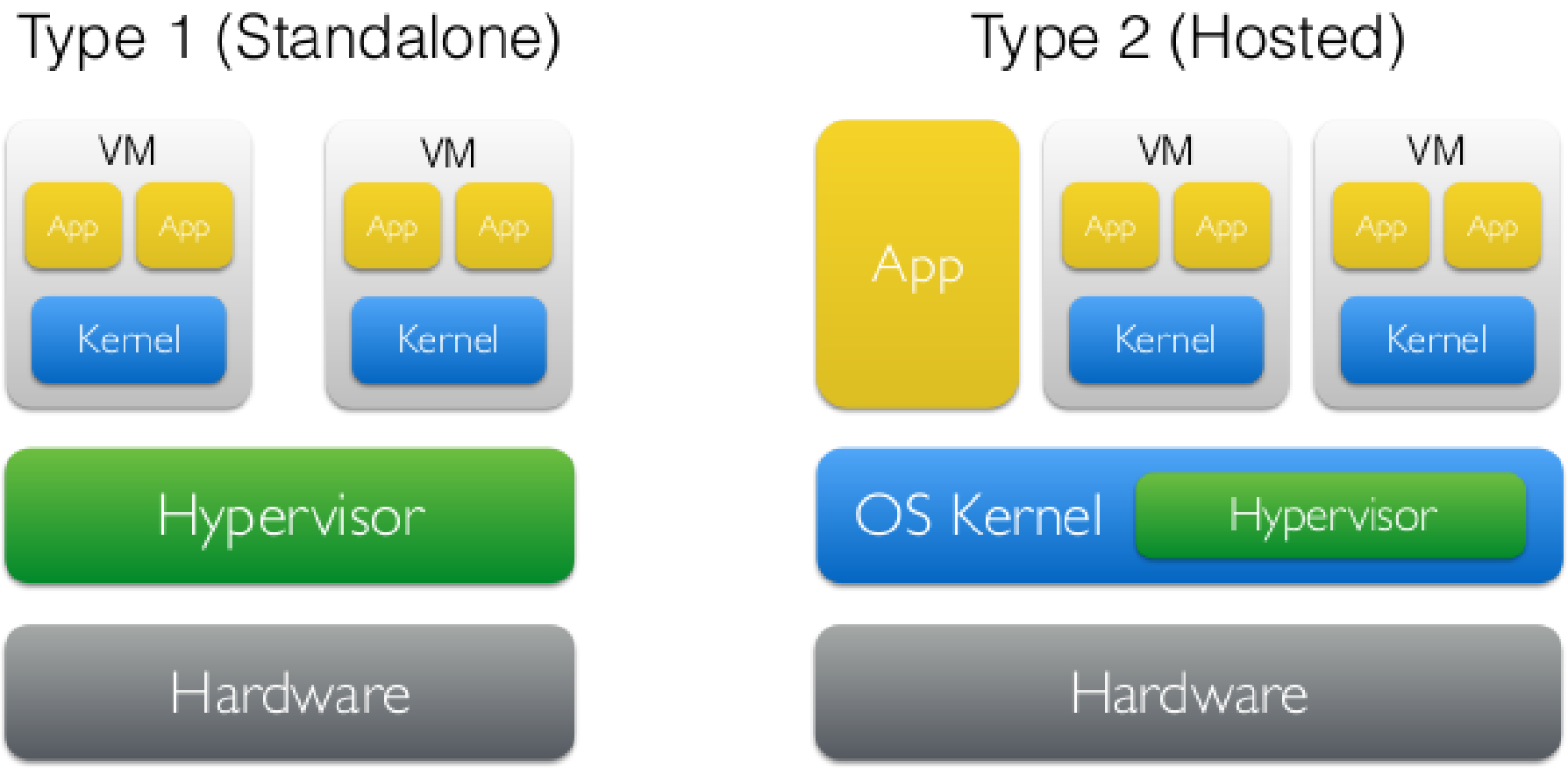
Virtualization



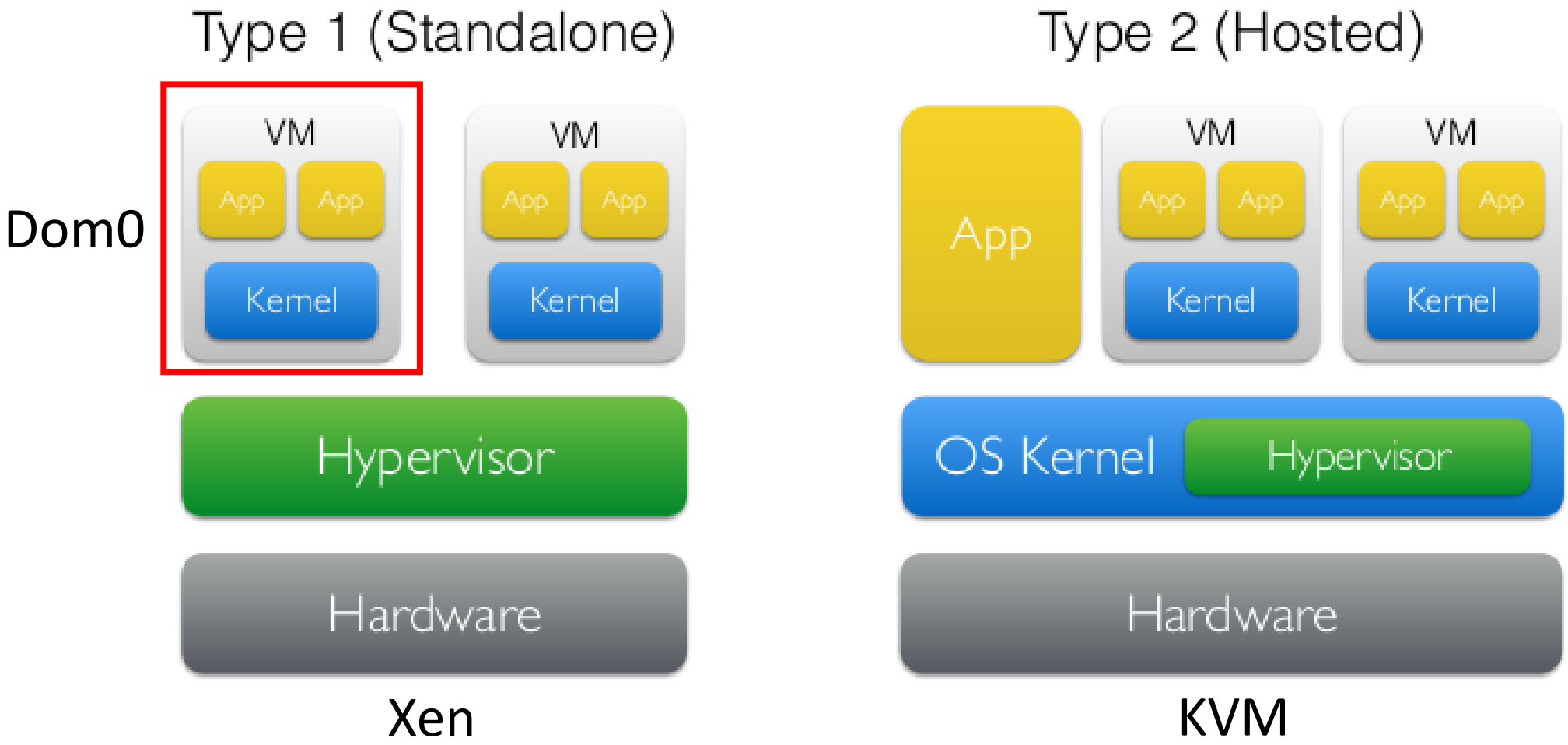
Hypervisor Design



Hypervisor Design

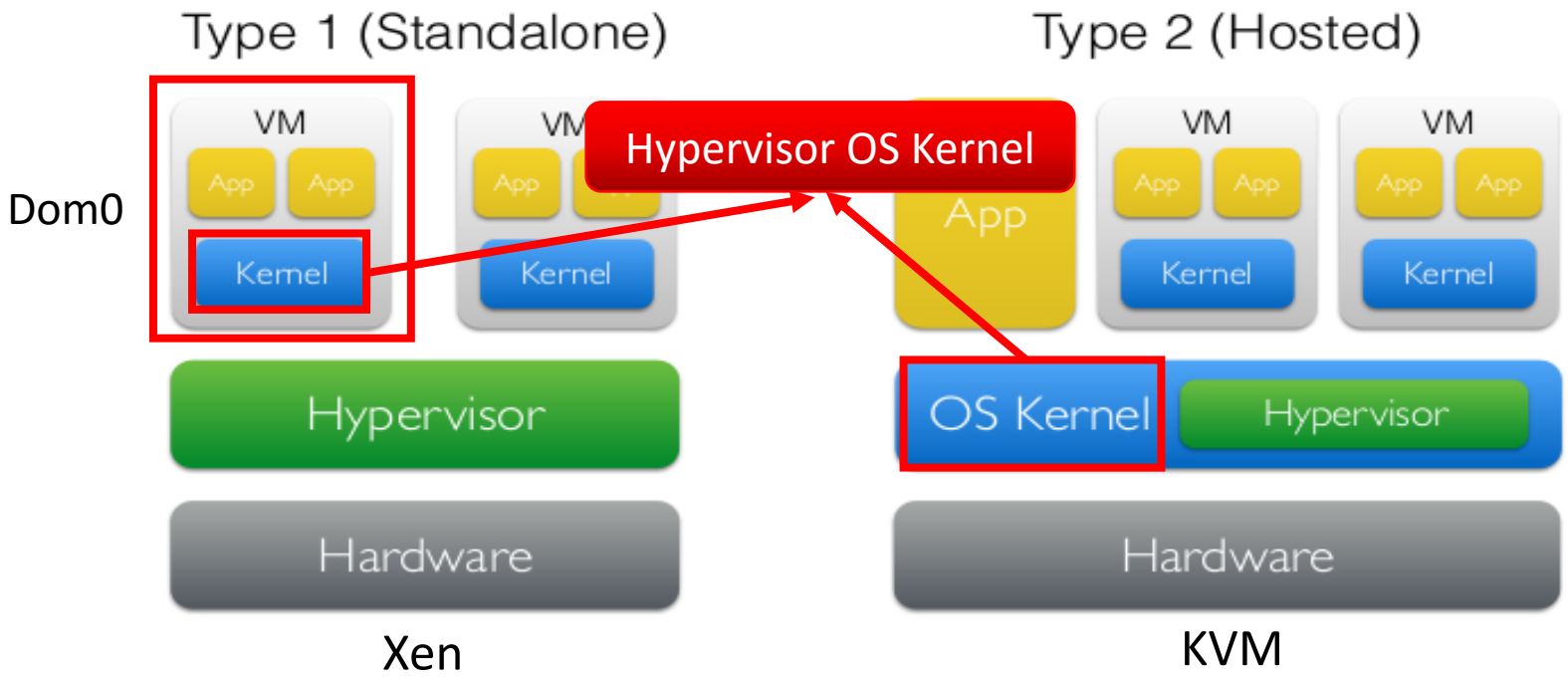


Hypervisor Design



Hypervisor OS Kernel

- A full OS to support hypervisor functionality



Overview

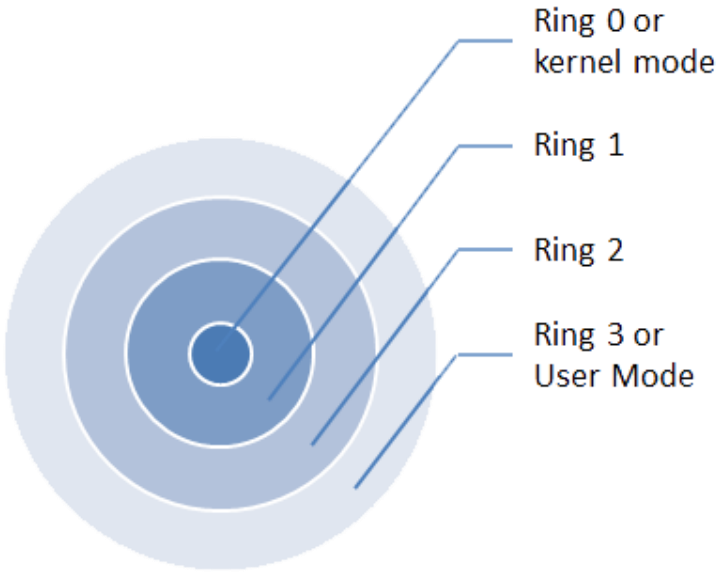
- Cost in transitions from the VM to the Hypervisor OS Kernel is critical to virtualization performance
- Optimize the design and performance of KVM/ARM
  - Take advantage of unique features in ARM Virtualization support
  - Reduce the cost of running Hypervisor OS Kernel

Hardware virtualization support

- Modern hypervisors leverage hardware virtualization support in their design
- Different hardware features provided by the architectures
- Different approach to run Hypervisor OS Kernel

x86 Virtualization Support (Intel VMX)

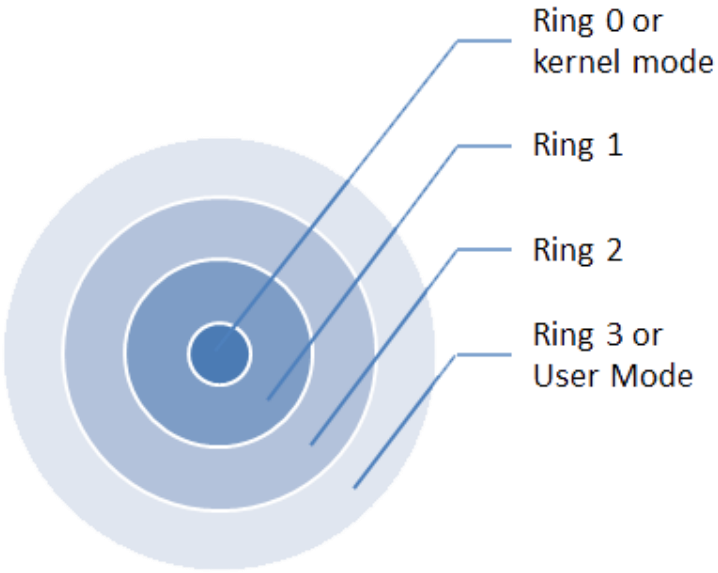
Root (Hypervisor)



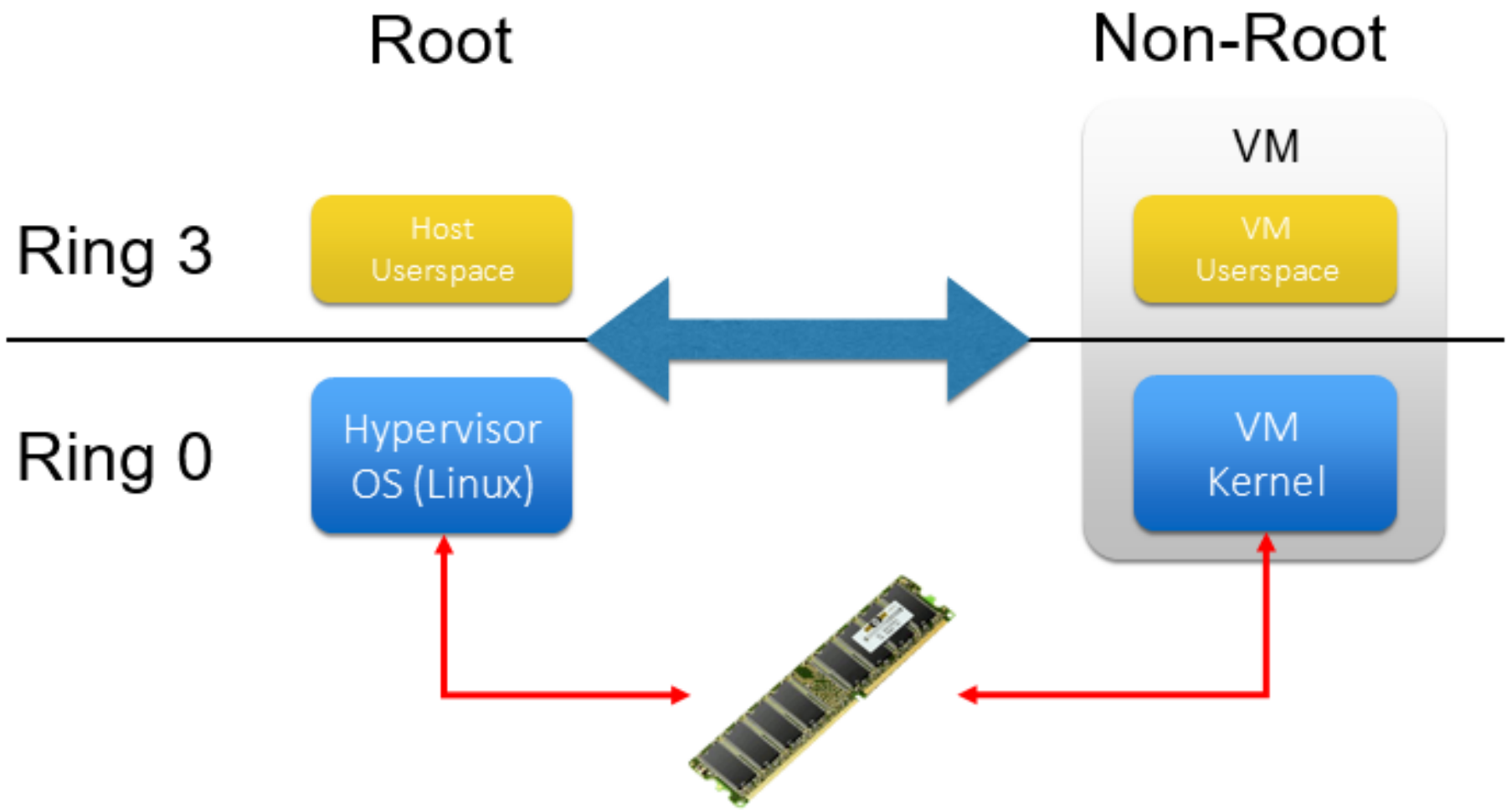




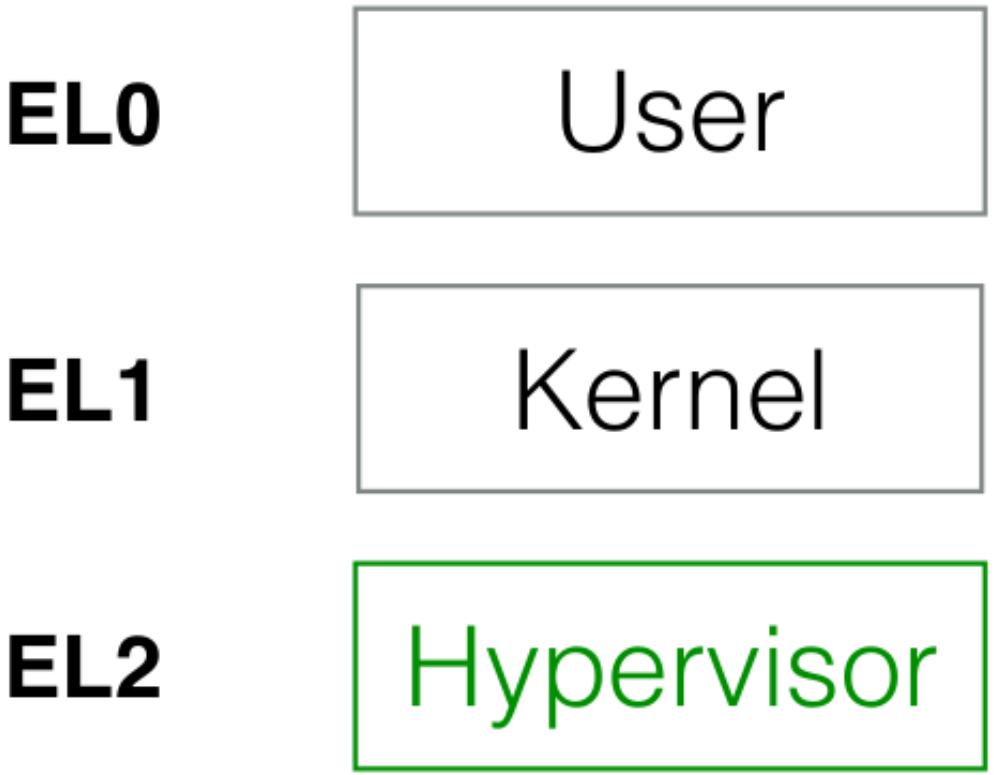
Non-Root (VM)



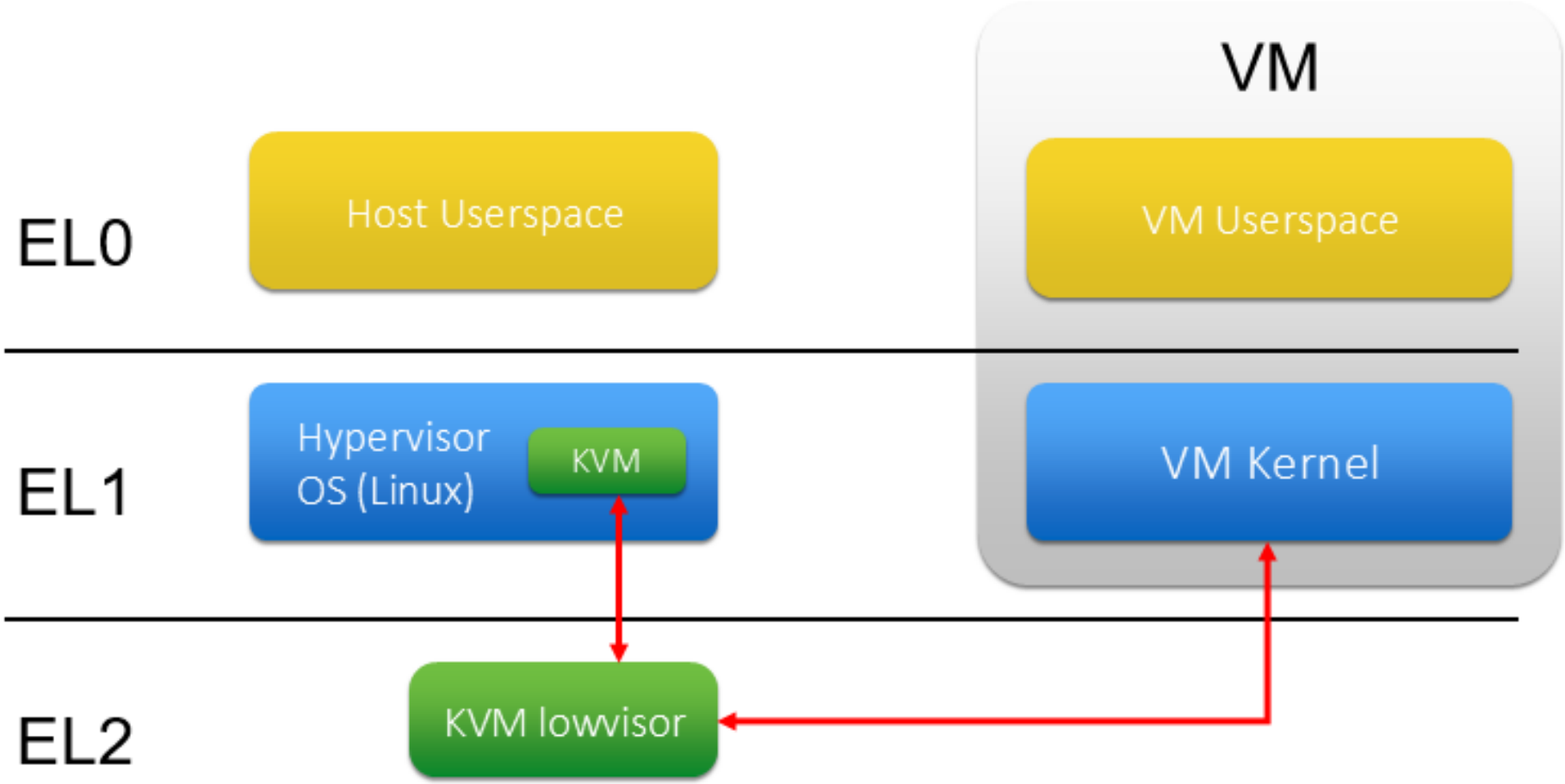
KVM x86


Root
Non-Root
VM
Ring 3
Host
Userspace
VM
Userspace
Ring 0
Hypervisor
OS (Linux)
VM
Kernel

ARM Virtualization Extensions



KVM/ARM

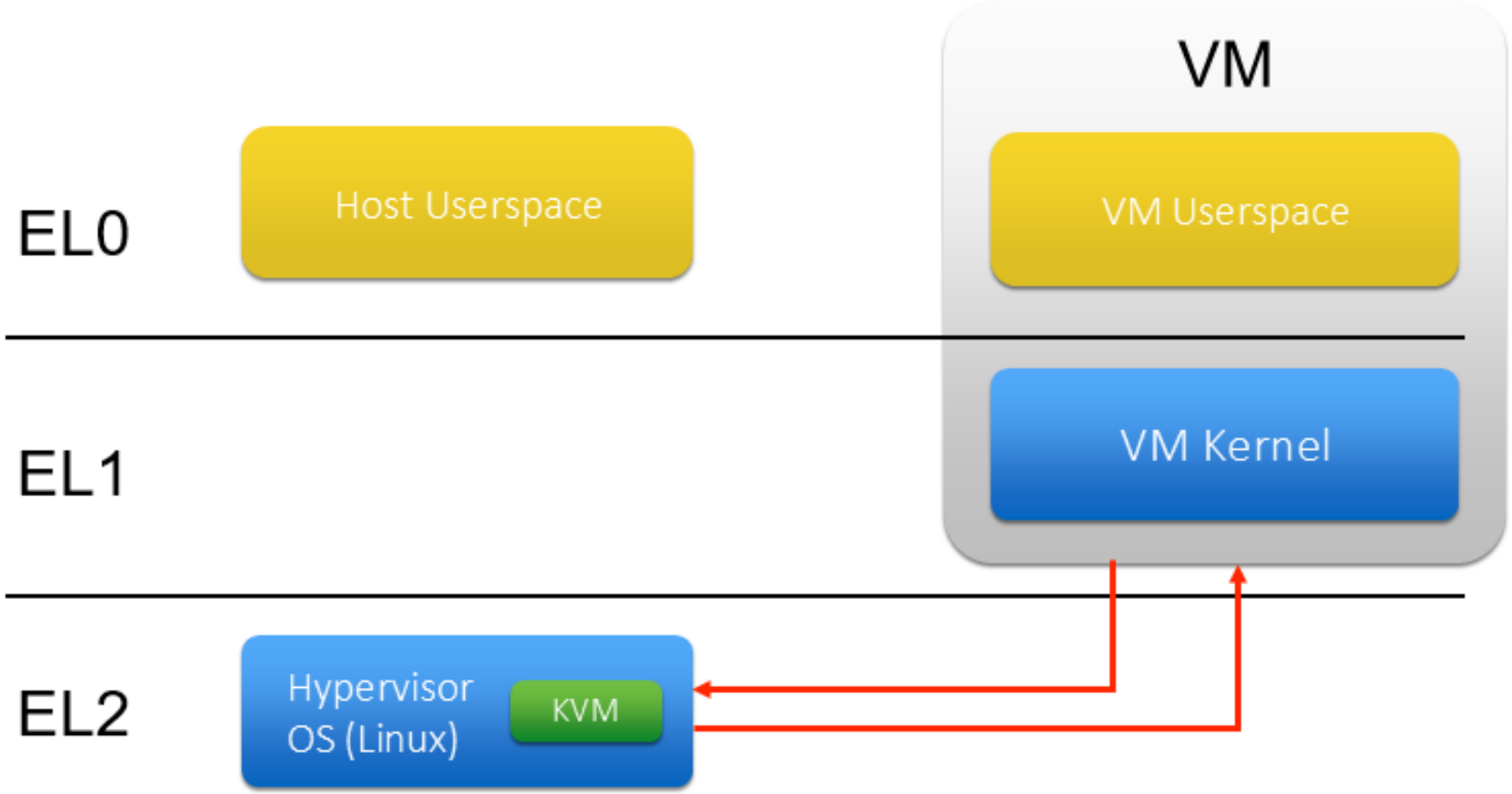

VM
EL0
Host Userspace
VM Userspace
EL1
Hypervisor
OS (Linux)
KVM
VM Kernel
EL2
KVM lowvisor

Sharing kernel mode state

- Multiplexing kernel mode is expensive!

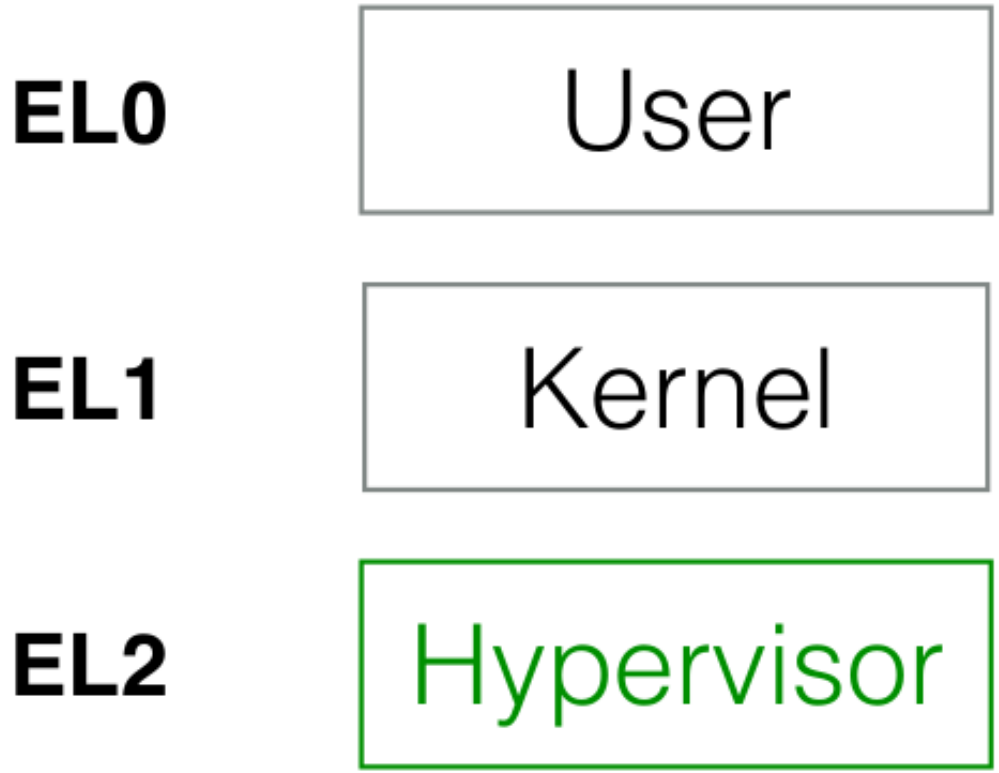
- Multiplexing the shared kernel mode in software is more expensive than hardware

Separate mode for Hypervisor OS kernel



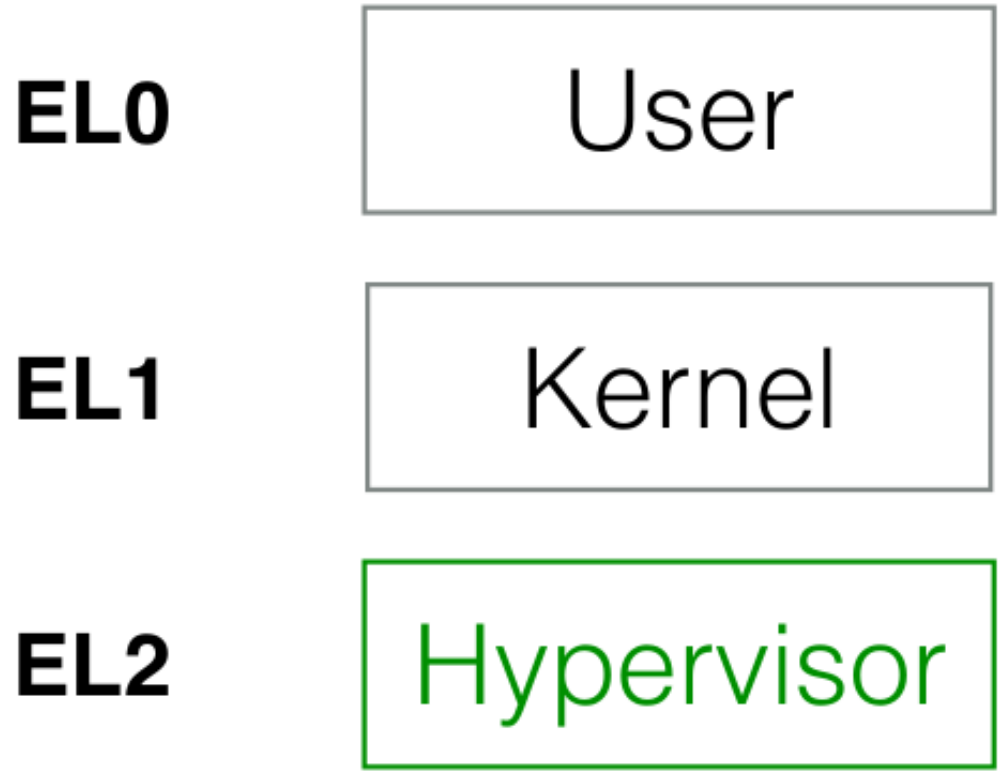
Challenges

- EL2 :
  - Separate CPU mode
  - Different page table formats
  - Limited capabilities compared to EL1



Challenges

- EL2 :
  - Separate CPU mode
  - Different page table formats
  - Limited capabilities compared to EL1
  - Cannot run unmodified OS in EL2!



Running Linux and KVM in EL2

- Hardware: VHE (Virtualization Host Extensions)
  - Hardware is not available yet!

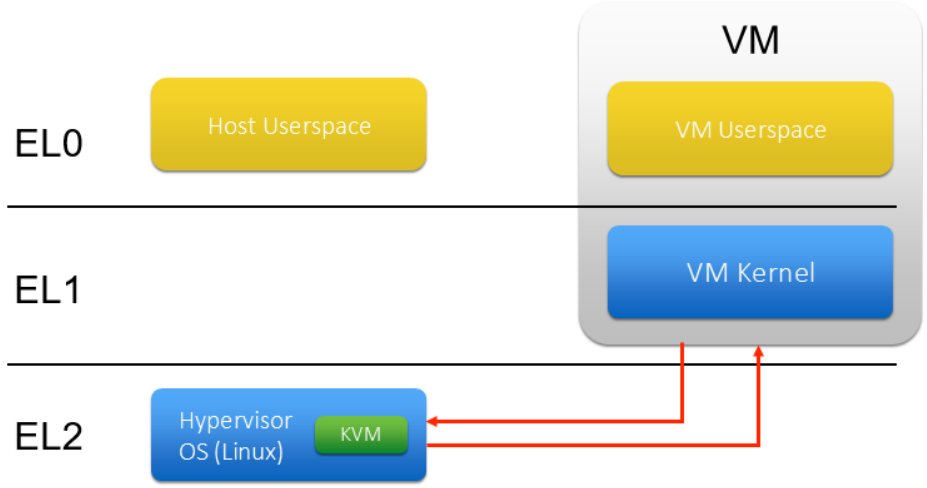
Running Linux and KVM in EL2

- Hardware: VHE (Virtualization Host Extensions)
  - Hardware is not available yet!

- Software: Change Linux to run in EL2

KVM/ARM Redesign

- Avoid multiplexing kernel mode states
- Avoid operating across different CPU modes
  - Allow the Hypervisor OS kernel to program the EL2 hardware
  - Reduce code complexity



Experimental Setup

ARM Hardware

- AMD Seattle Rev B0
- 2 GHz AMD A1100 CPU
- 8-way SMP
- 16 GB RAM
- 10 GB Ethernet

x86 Hardware

- Dell PowerEdge r320
- 2.1 GHz Intel Xeon E5-2450
- 8-way SMP
- 16 GB RAM
- 10 GB Ethernet

VM

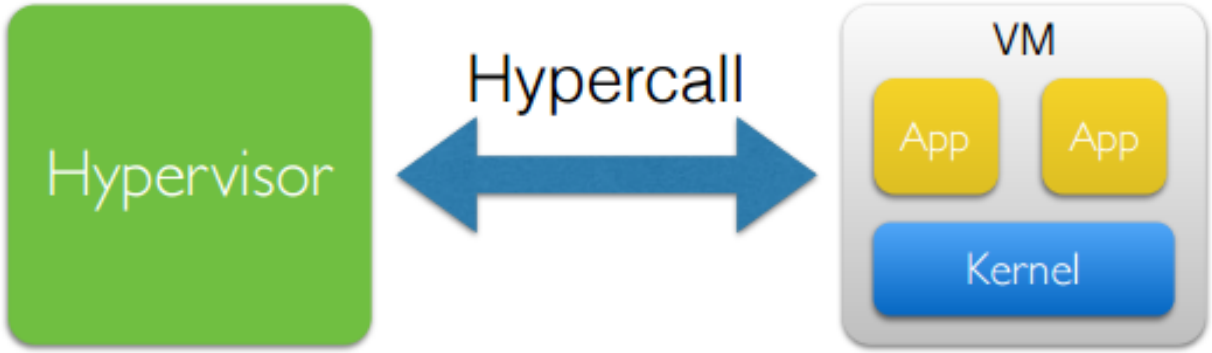
- 4-way SMP
- 12GB RAM
- Passthrough NIC



Hypercall benchmark

- No-op Hypercall: VM traps to the hypervisor and returns immediately
- Base cost for hypervisor to serve the virtual machines
- Include the cost of transitioning between the VM and the hypervisor

  OS kernel in KVM



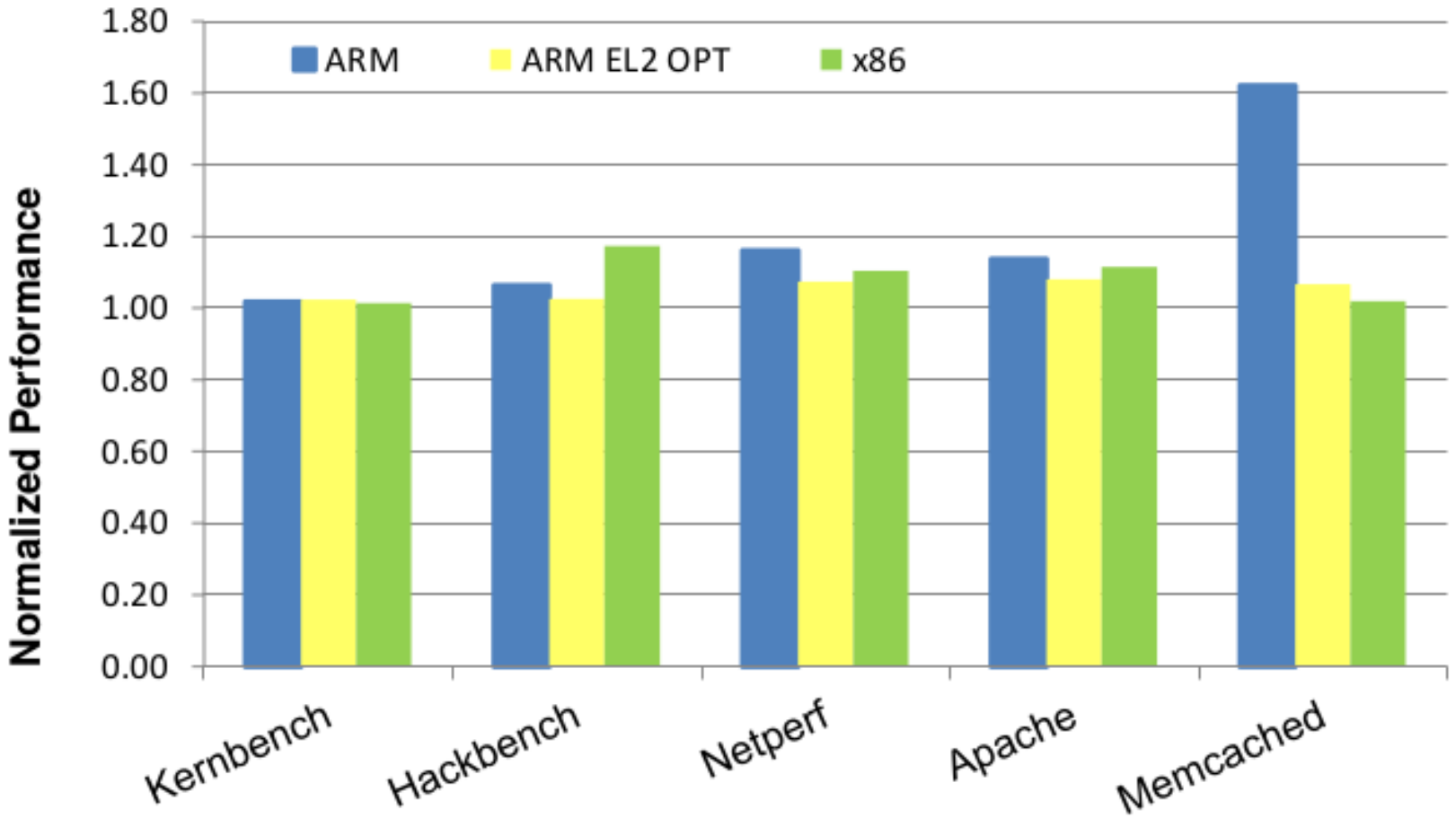
Hypercall benchmark

CPU cycles	ARM	ARM EL2 OPT	x86
Hypercall	6,413	752	1,437

Application Workloads

Applications	Description
Kernbench	Kernel compile
Hackbench	Scheduler stress
Netperf (TCP_RR)	Network performance
Apache	Web server stress
Memcached	Key-Value store

Application Results



Conclusion

- Optimized the design and performance of KVM/ARM
  - Improved its performance by roughly an order of magnitude
  - Accomplished faster performance than KVM on x86
- Benefit from future ARM hardware with VHE support
- Many of the proposed changes have been merged to mainline Linux

Q&A