Nested Virtualization

Dongxiao Xu, Xiantao Zhang, Yang Zhang

May 9, 2013

Agenda

- Nested Virtualization Overview
- Dive into Nested Virtualization Details
  - Nested CPU Virtualization
  - Nested MMU Virtualization
  - Nested I/O Virtualization
- Optimization with Intel HW Feature
  - Nested CPU Virtualization: VMCS shadowing
  - Nested MMU Virtualization: Virtual EPT
  - Nested I/O Virtualization: Virtual VT-d

Nested Virtualization Overview

What is Nested Virtualization

Concept

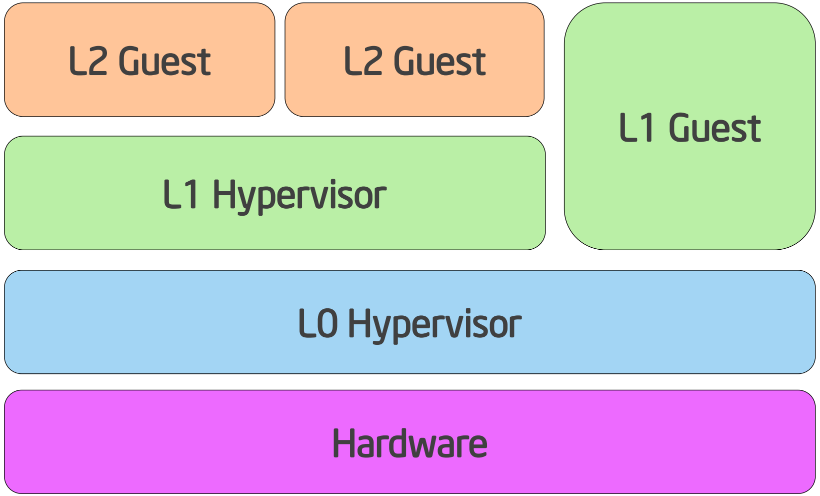
- Running virtual machines inside virtual machine

Objectives of Nested Virtualization

- Unmodified Hypervisor/Guest
- Isolation and security
- Efficiency in performance

Approaches

- Software Solutions (Para-Virtualization, BT, etc.)
- Hardware Solution: Nested VMX



Nested Virtualization -- A key feature towards cloud computing

- Virtualization in daily life
  - Windows 7 compatibility mode
  - Windows 8 with Hyper-V
  - Linux with KVM
- Run virtualization apps/solutions by cloud user
  - Solution 1: Allocate a physical machine with virtualization capability
  - Solution 2: Nested Virtualization capable VMMs

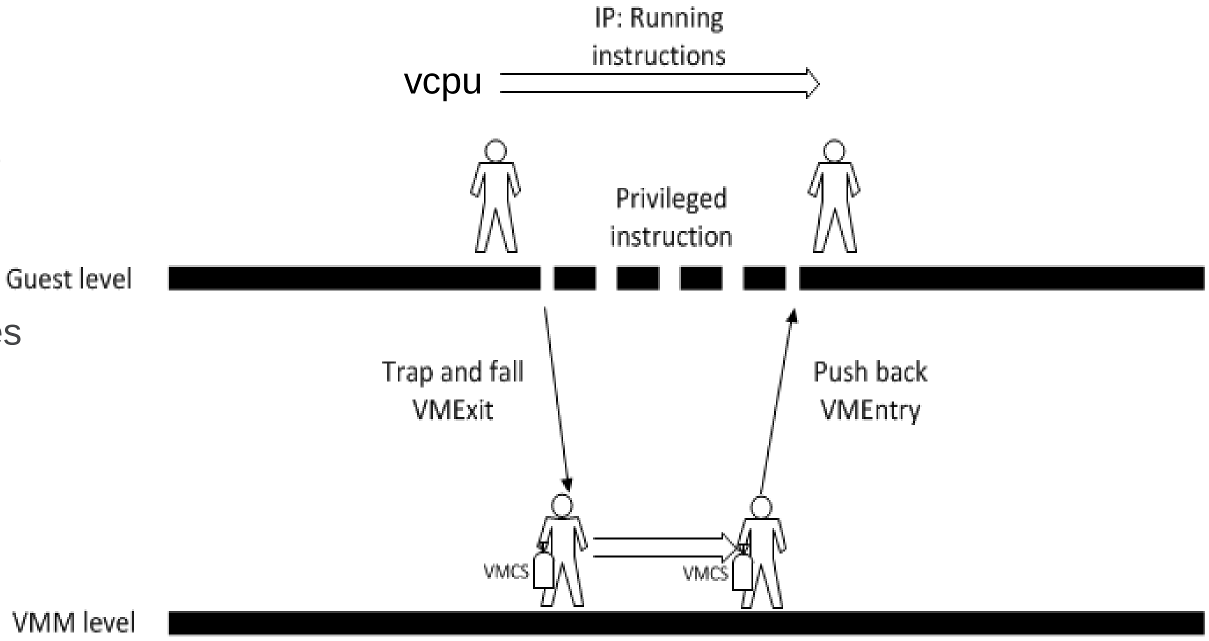
Nested Virtualization – Other usage models

- Hypervisor level of Anti-Virus
- Facility for investigating VMM behavior
- New HW feature emulation
- …

Dive into Nested Virtualization Details

CPU Virtualization

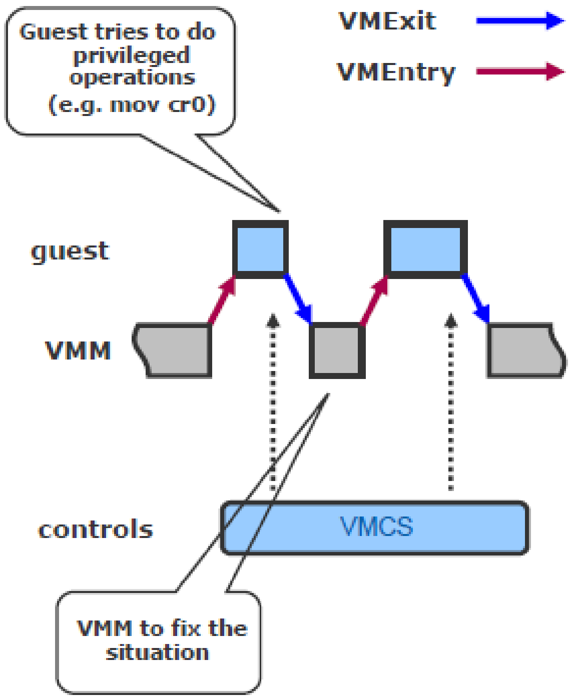
- Guest vcpu is running privileged instruction
- Guest traps into VMM by VM exit
- VMM emulates the instruction on behalf of guest
- VMM updates guest EIP and goes back to guest
- Guest vcpu continues running



CPU Virtualization

VMX key concepts

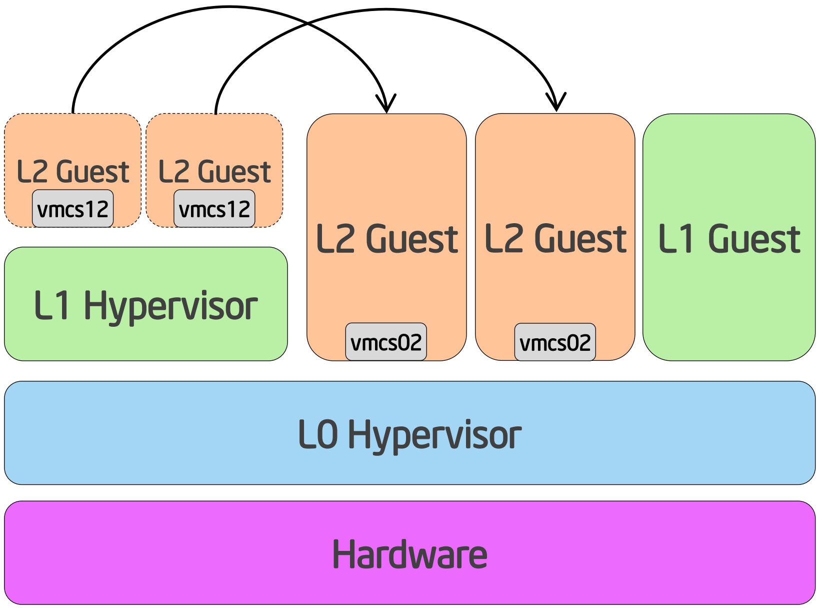
- New running modes: root and non-root modes.
- The info bag: VMCS (4k page)
  - Saving guest running status
  - Control when guest exists
  - Information exchange
- Guest → VMM: VM Exit
  - Guest traps into VMM context
  - VMM helps to emulate the instruction
- VMM → Guest: VM Entry
  - VMLAUNCH or VMRESUME
  - Returning from VMM back to guest



Nested CPU Virtualization

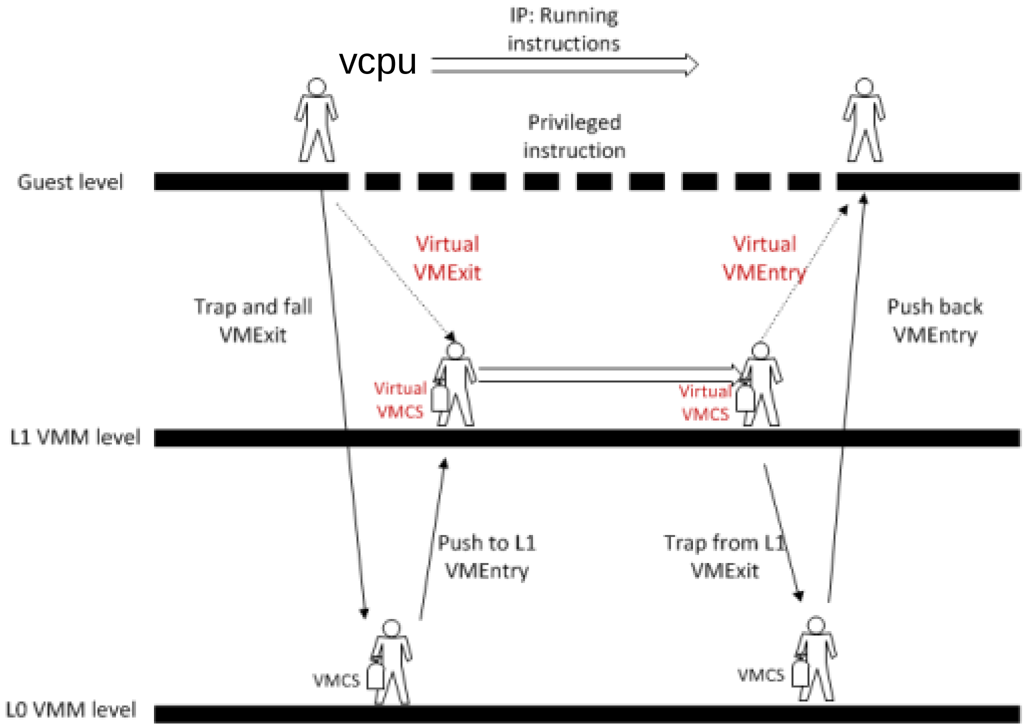
Nested virtualization guidelines:

- L0 emulates virtual VMCS (vmcs12), virtual VM exit and virtual VM entry for L1 VMM so that L1 regards L2 as its guest.
- L0 provides the actual runtime environment for L2 guest directly by constructing vmcs02 and load it into hardware.



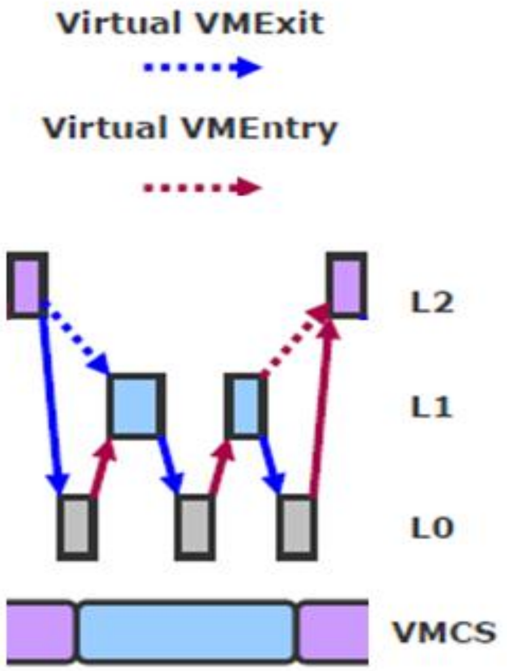
Nested CPU Virtualization

- Guest vcpu encounters a privileged instruction when running
- Guest traps into L0 VMM by VM exit
- L0 VMM checks the status, prepare the virtual VMCS, and inject the VM entry into L1 VMM
- L1 VMM emulates this instruction
- When finished, L1 VMM call VMRESUME trying back to guest
- L0 VMM get the VM exit (due to VMRESUME executed in non-root mode), and issue real VMRESUME
- Guest continues to run instructions



Nested CPU Virtualization

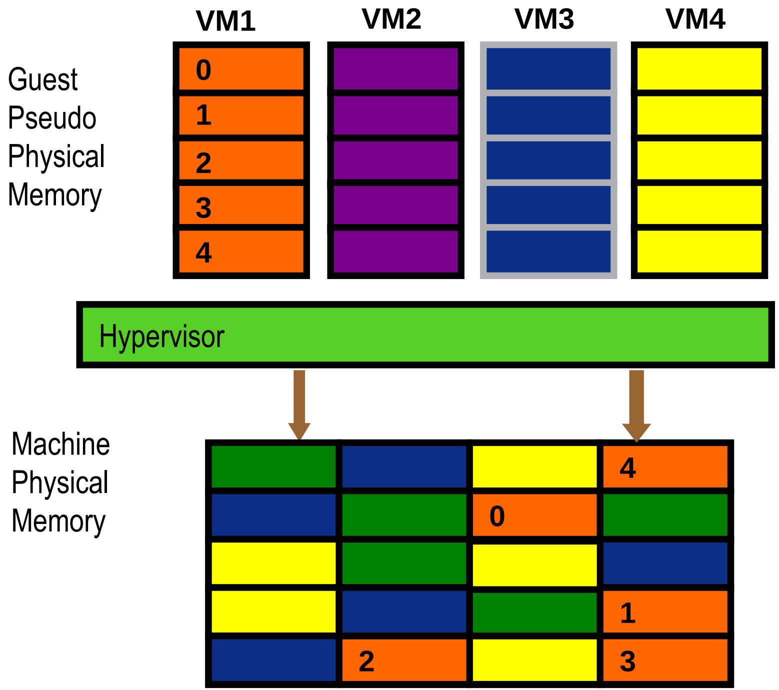
- Nested VMX key concepts
  - Virtualize VMCS
    - Shadow VMCS and virtual VMCS
  - Virtualize VMExit
    - Guest trap into L0 VMM
    - L0 inject to L1 VMM
  - Virtualize VMEntry
    - L1 VMM trap into L0 VMM
    - L0 VMM return back to L2 guest



MMU Virtualization

What memory virtualization needs to do?

- Present guest OS memory resource it expects
- Isolate memory among guests/HV from guests
- Share memory resources whenever possible



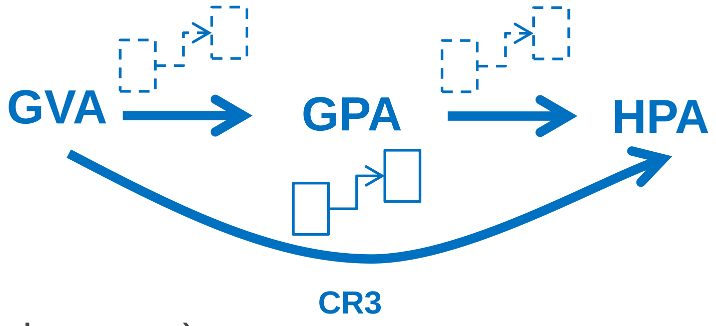
MMU Virtualization

Software solution: Shadow Page Tables

- Utilize the original CPU paging mechanism (CR3).
- Guest OS maintains gva to gpa mappings.
- Hypervisor maintains gpa to hpa mappings.
- Hypervisor establishes shadow page tables (gva → hpa).

Pros:

- Support unmodified guests.
- No specific hardware required.



Cons

- Performance issue (trap guest PT operations, etc).
- Complex software implementation.

MMU Virtualization

Hardware solution: Extended Page Tables (EPT)

- Bring in another paging dimension in hardware.
- Guest OS maintains gva to gpa mappings, loaded in CR3.
- Hypervisor maintains gpa to hpa mappings, loaded in EPTP.

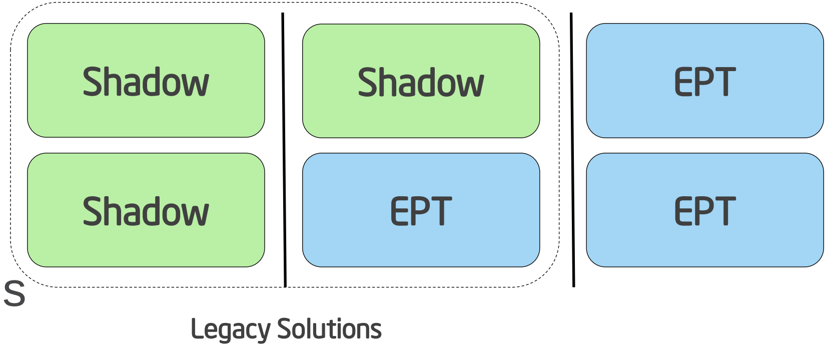
Pros:

- Guest owns CR3 page table.
- Performance is highlight.
- Simplified software.

Nested MMU Virtualization

Challenges

- Three dimensions of paging
  - ✓ L2 gva → L2 gpa
  - ✓ L2 gpa → L1 gpa
  - ✓ L1 gpa → L0 hpa
- CPU only provides two dimension paging with EPT



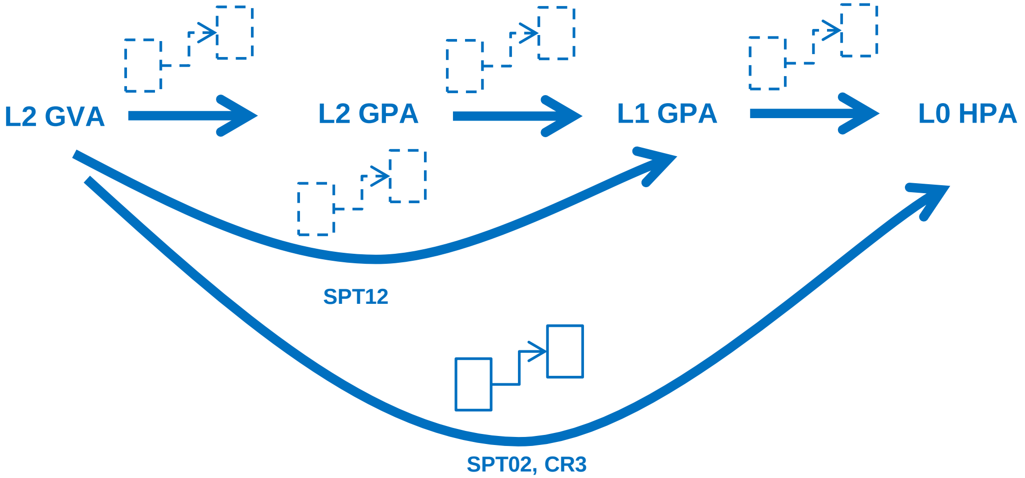
Nested MMU virtualization combinations

- shadow on shadow
- shadow on EPT
- EPT on EPT
- EPT on shadow (No valuable significance)

Nested MMU Virtualization

Shadow on Shadow

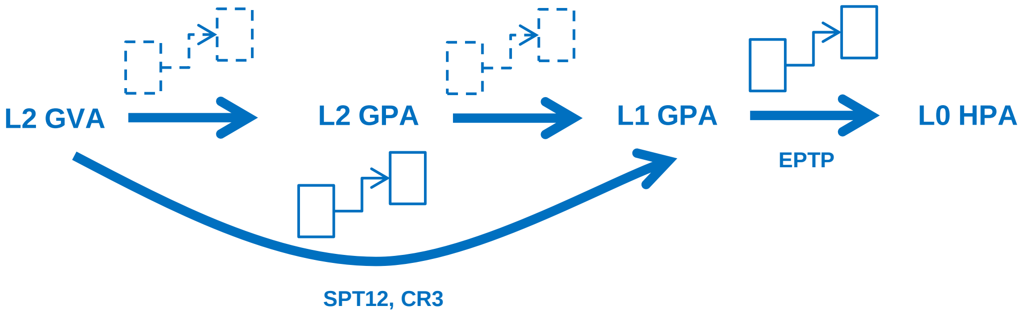
- Use one dimension page table to emulate three.
- No specific hardware requirement, can use in old platforms.
- Performance is not good.



Nested MMU Virtualization

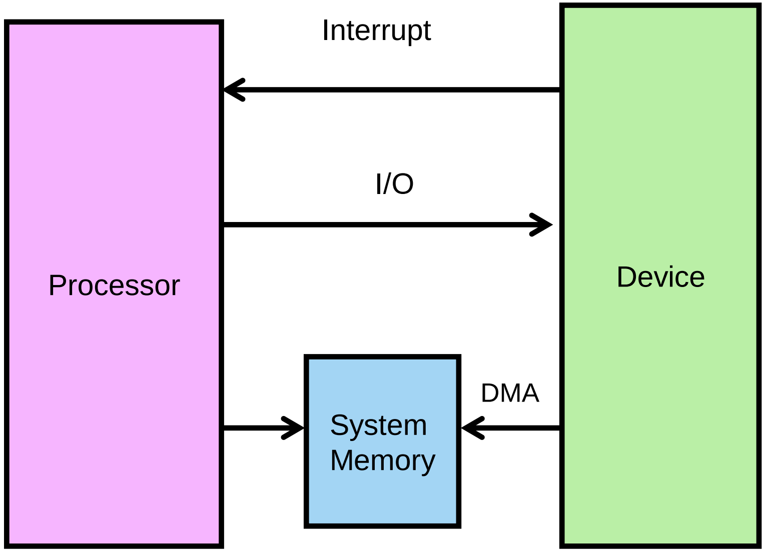
Shadow on EPT

- Use two dimension page tables to emulate three.
- Need EPT involvement.
- L2 page faults will trigger a lot of emulation effort.



I/O Virtualization

- Software communicate with device
  - Port I/O
  - MMIO
- Device transfer data to and from system memory
  - DMA access
- Events notification from device
  - Interrupt
- Device discovery and configuration
  - Configuration space access in PCI device



I/O Virtualization

Software solution: Device emulation

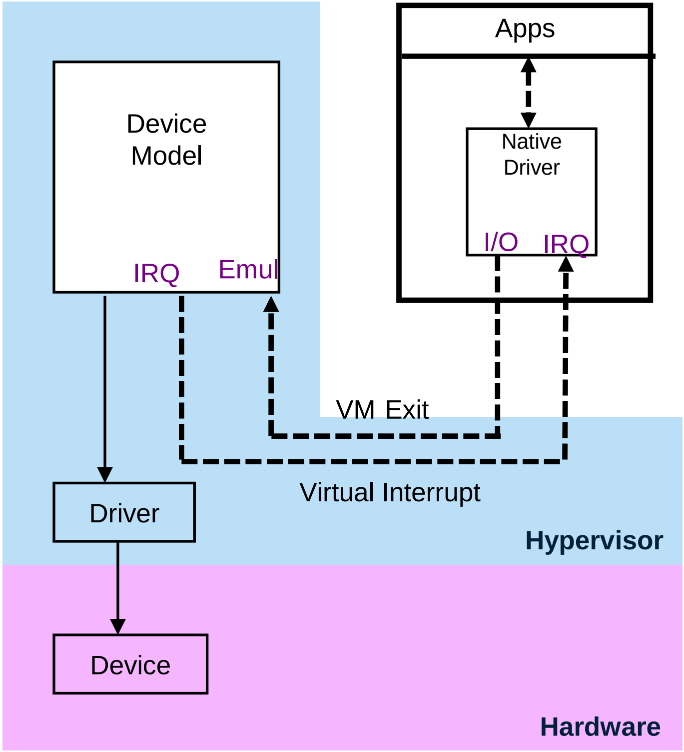
- Maintain same SW interface (I/O, MMIO, INTR, DMA)
- Use arbitrary media to emulate virtualized device

Pros

- Transparent to VM software stack
- Agnostic to physical device in the platform. Thus
  - Legacy SW can still run, even after HW upgrade
  - Smooth VM migration across different platforms
- Good physical device sharing

Cons

- Un-optimum performance
- Cannot enjoy latest & greatest HW
  - Lack of modern device emulation, since too complex
- Poor scalability
- Isolation and stability depends on implementation



I/O Virtualization

Software solution: Para-virtualized I/O

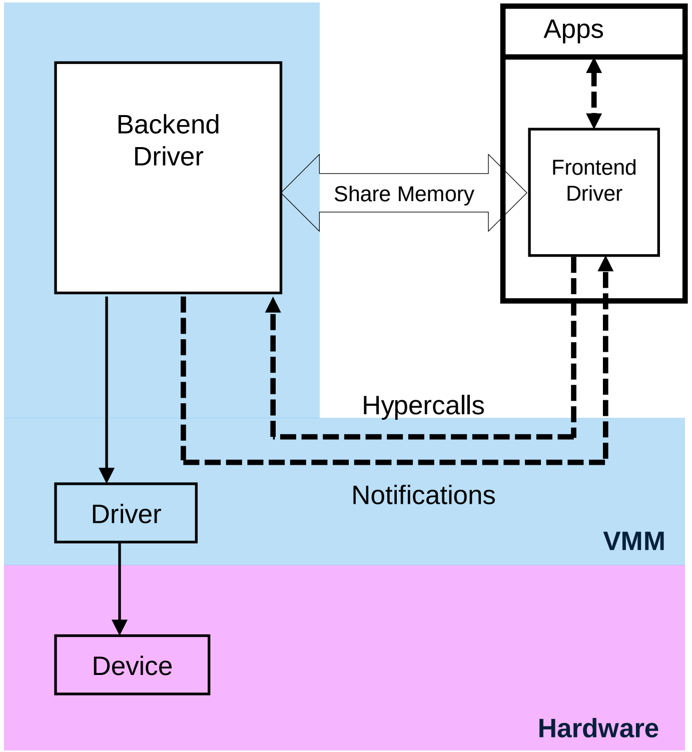
- VMM presents a specialized virtual device to VM
- A new & efficient interface between VMM & VM driver
  - Usually high-level abstraction
  - Requires a specialized driver in VM

Pros

- Better performance
- Agnostic to physical device in the platform
  - Legacy SW can still run, even after HW upgrade
  - Smooth VM migration across different platforms
- Good physical device sharing

Cons

- Need install specialized driver in VM
- High CPU utilization (I/O interface and memory copy)
- Not so good scalability because of CPU utilization



I/O Virtualization

Hardware solution: VT-d

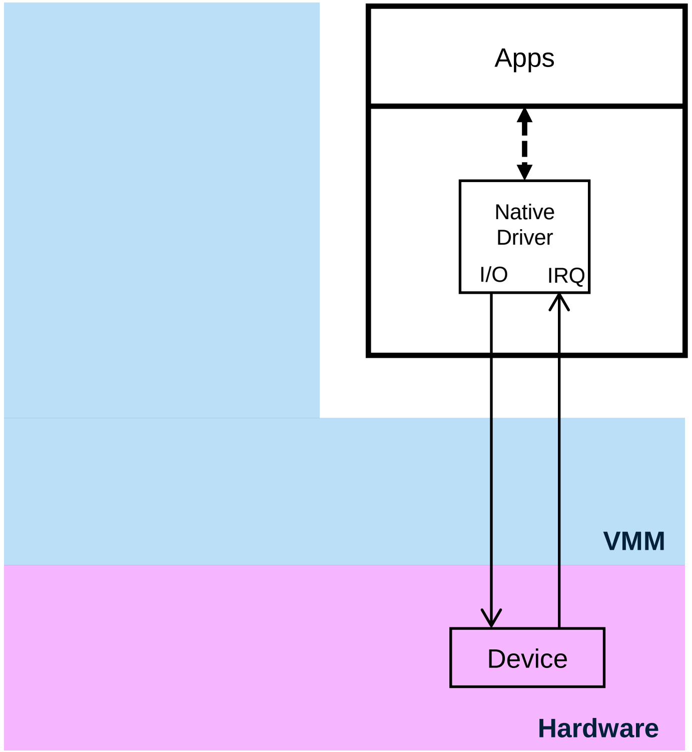
- Assign a physical device to a VM directly
- VM access the device directly, w/ VMM intervention reduced to minimum
- Physical device access guest's memory directly with help of Intel's VT-d technology

Pros

- Near-to-native performance
- Minimum VMM intervention, thus low CPU utilization
- Good isolation

Cons

- Exclusive device access
- PCI slots in system is limited (SR-IOV)



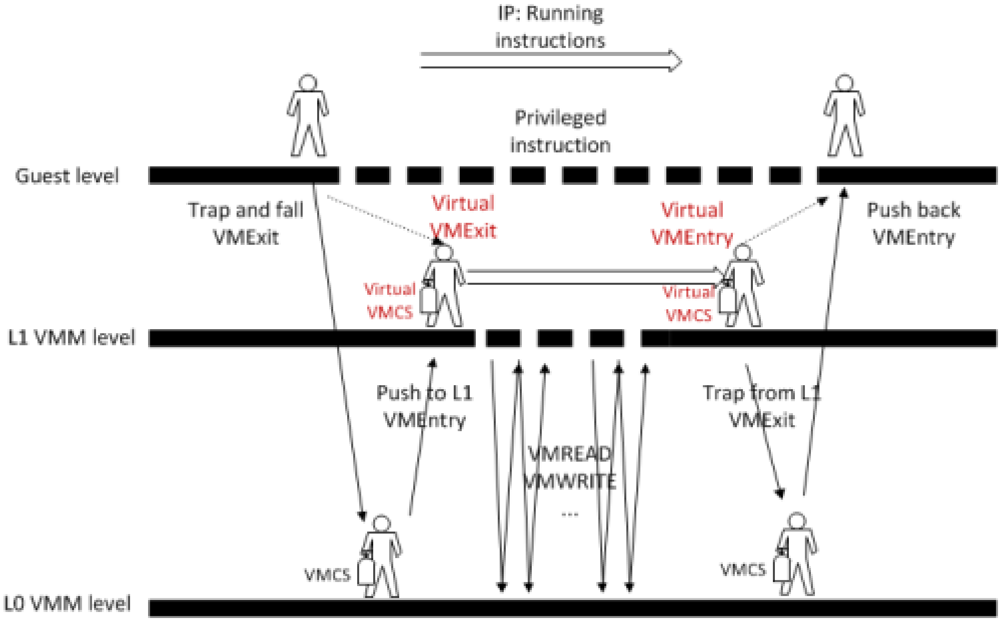
Nested I/O Virtualization

- Nested I/O Virtualization Combinations:
  - L2 software emulation + L1 software emulation
  - L2 software emulation + L1 PV
  - L2 software emulation + L1 VT-d
  - L2 PV + L1 software emulation
  - L2 PV + L1 PV
  - L2 PV + L1 VT-d
- Mixed Pros and Cons
  - But, still need software intercept.

Optimizations with Intel HW Feature

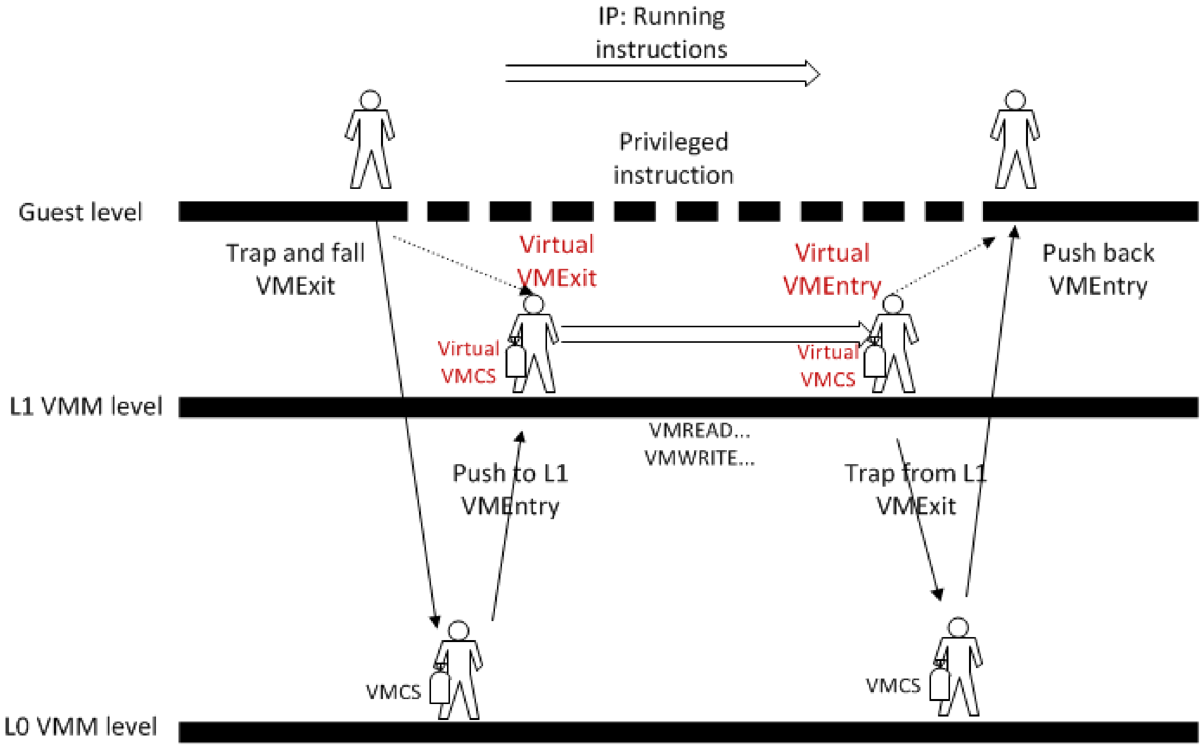
Nested CPU Optimization: Reduce VM Exits

- L1VMM operates VMCS in non-root mode
- VM exit will be triggered to emulate the non-root VMREAD/VMWRITE
- For a single virtual VM exit, L1 will trigger ~20 VMREAD/VMWRITE.



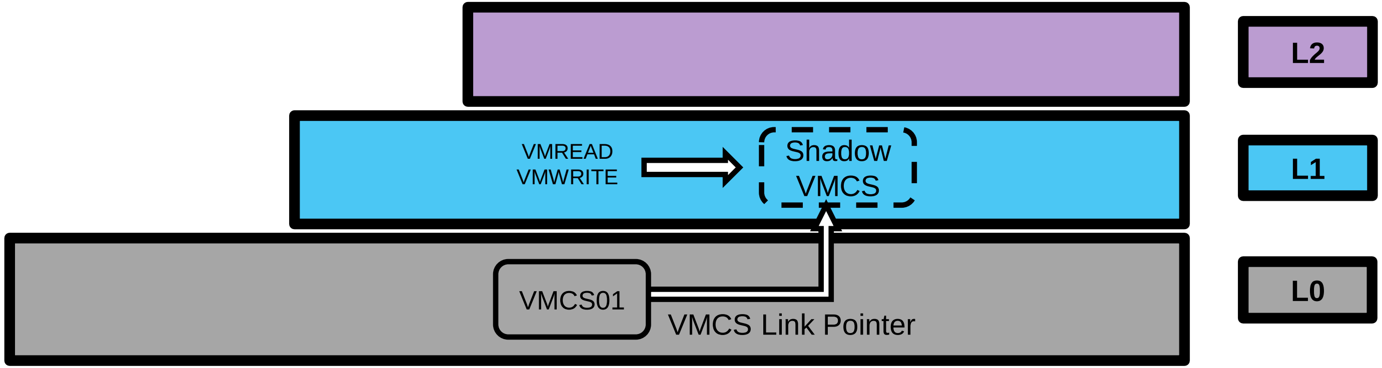
Goal: Reduce heavy VM exits due to non-root VMREAD/VMWRITE

Nested CPU Optimization: Ideal Case



Nested CPU Optimization: VMCS Shadowing

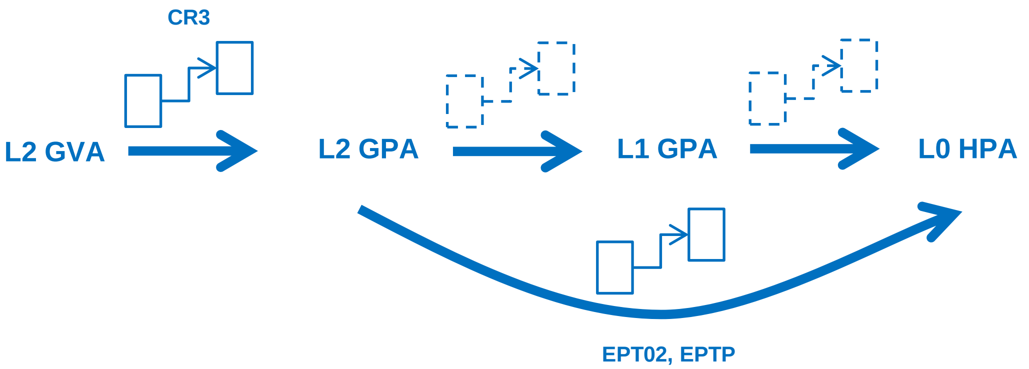
- Hardware assisted non-root VMREAD/VMWRITE
- Shadow VMCS is linked to L1's VMCS by VMCS Link Pointer
- No VMExit happened for non-root VMREAD/VMWRITE



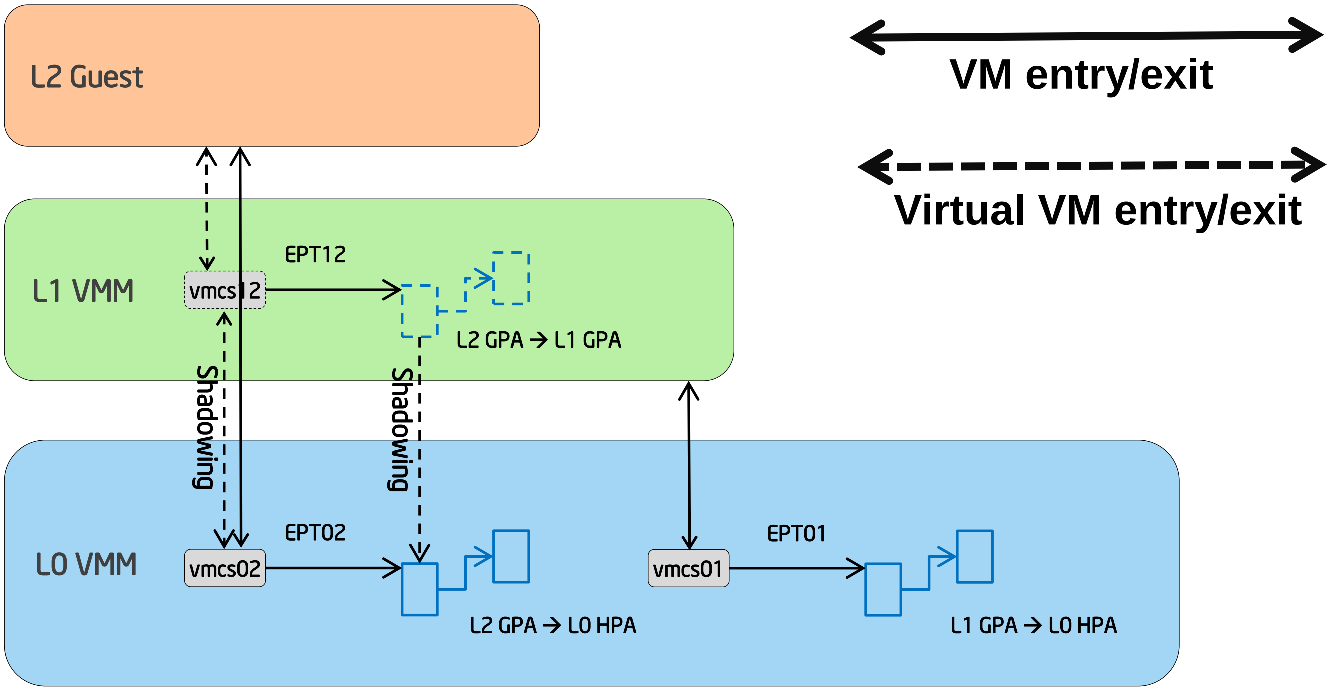
Nested MMU Optimization

EPT on EPT

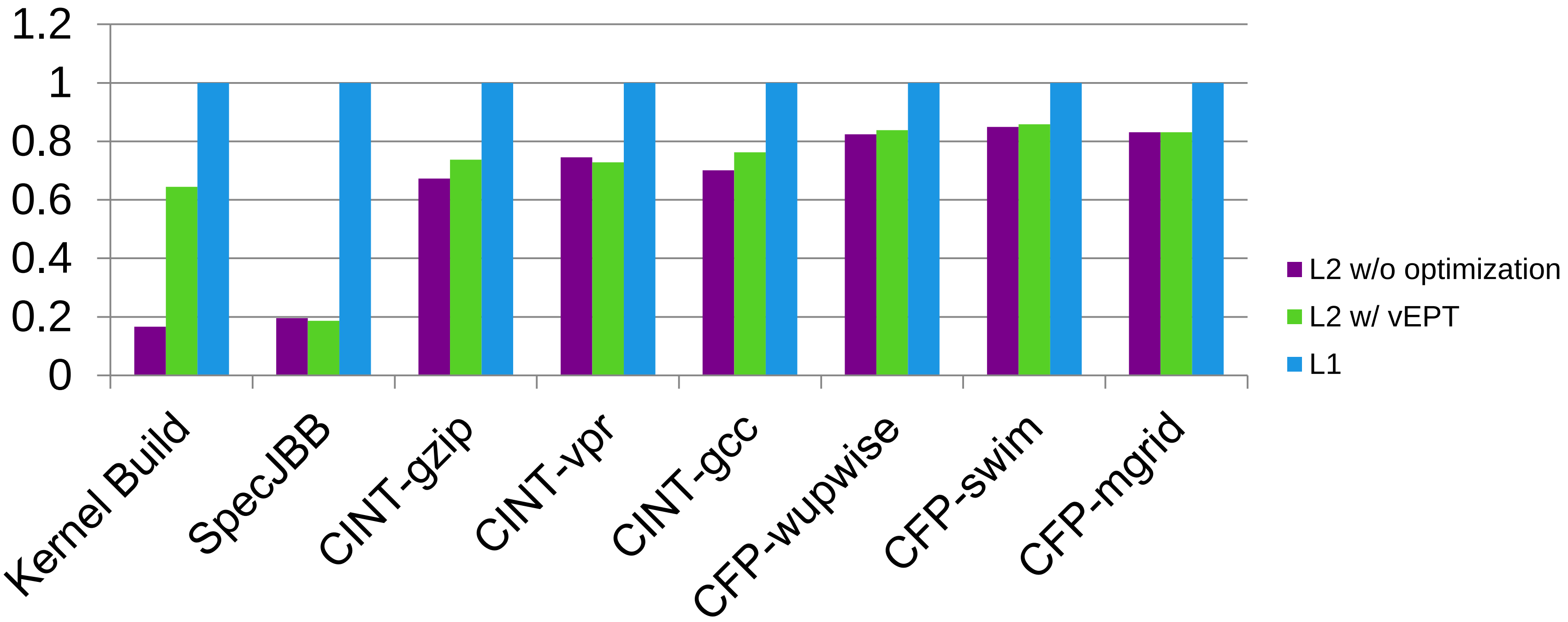
- Use two dimension page tables to emulate three.
- Need EPT involvement.
- L2 fully owns its page tables.
- Best performance among the three solutions.



Virtual EPT Architecture



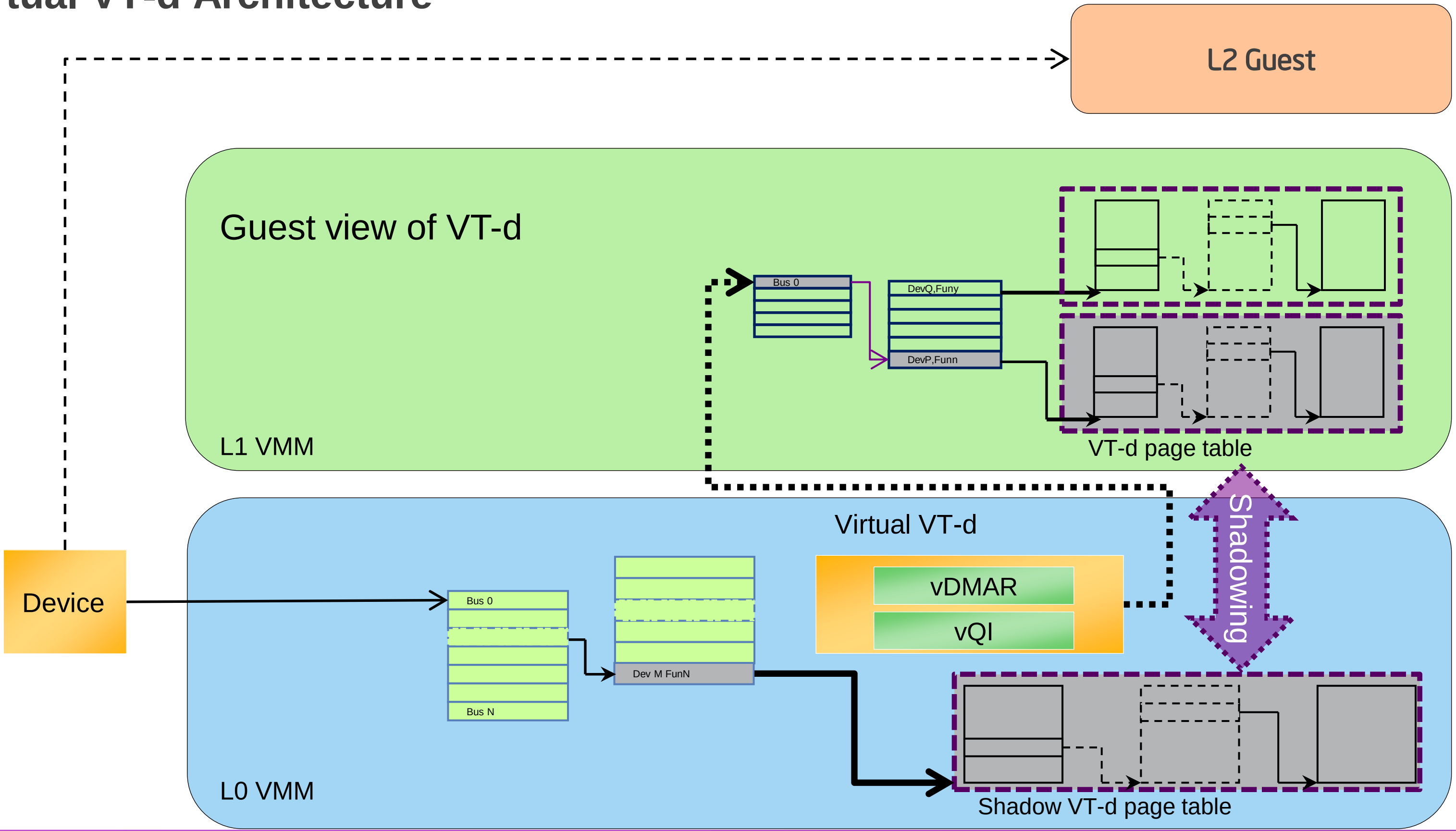
Virtual EPT Performance



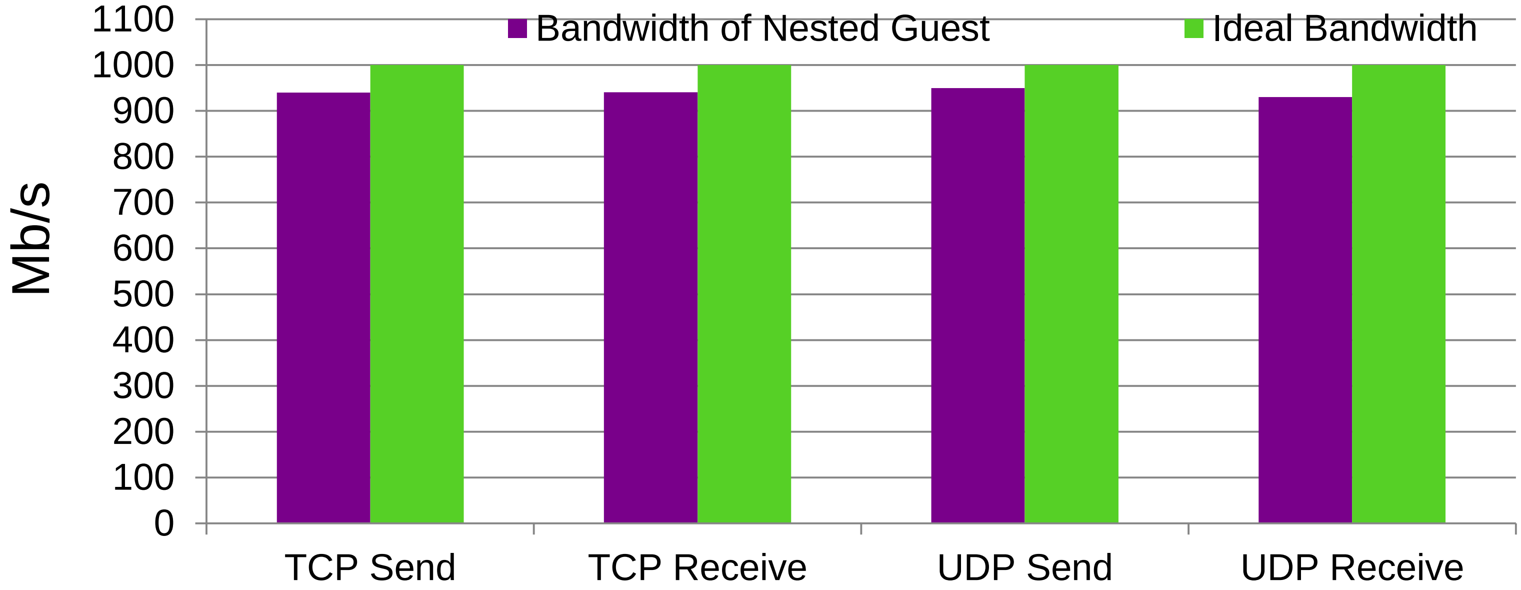
Nested I/O Optimization:

- I/O performance for L2 guest is very slow
  - Due to extremely long device emulation path through all the way to L1 & L0 VMMs
- How to fix that?
  - Present virtual VT-d engine to L1 VMM
  - Device can be directly assigned to L2 guest
    - High I/O performance, because of minimum VMM intervention.
- Must-to-have features in Virtual VT-d
  - DMA Remapping & Queue Invalidation: Exposed
  - Interrupt remapping: Not Exposed

Virtual VT-d Architecture



Performance Evaluation of virtual VT-d



Iperf testing with the assigned NIC to nested Guest

Bandwidth is good enough!

Thank You!

Q & A?