Taking it to the Nest Level

Nested KVM on the POWER9 Processor

Suraj Jitindar Singh - IBM Australia

Disclaimer

This work represents the view of the authors and does not necessarily represent the view of IBM.

Who am I?

- Live in Canberra, Australia
- Work at Ozlabs, IBM Australia
- Virtualisation on Power
  - Linux/KVM
  - QEMU
- Ride Motorbikes

This is going to go by quick

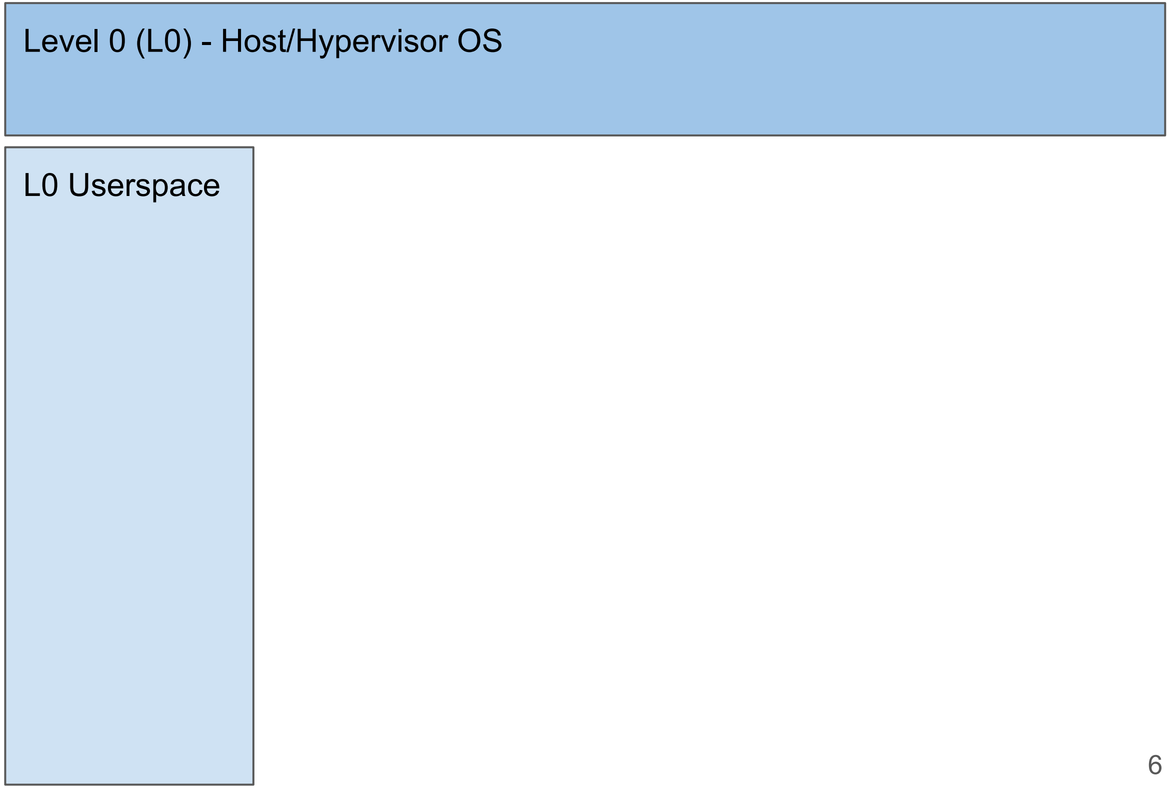
- If possible please keep questions to the end

Some Terminology

- What is KVM?
- What is Nested KVM?
    - L0 Hypervisor

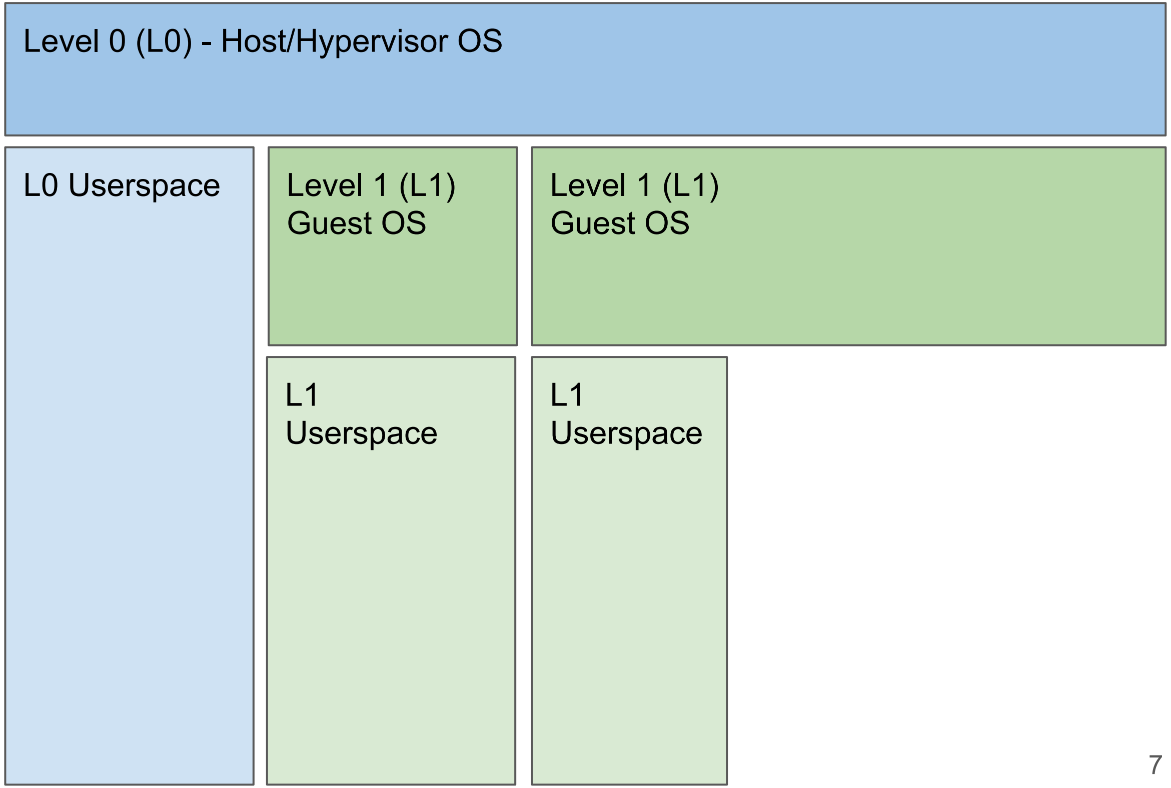
Some Terminology

- What is KVM?
- What is Nested KVM?
    - L0 Hypervisor



Some Terminology

- What is KVM?
- What is Nested KVM?
    - L0 Hypervisor
    - L1 Guest (Hypervisor)



Level 0 (L0) - Host/Hypervisor OS

L0 Userspace

Level 1 (L1)
Guest OS

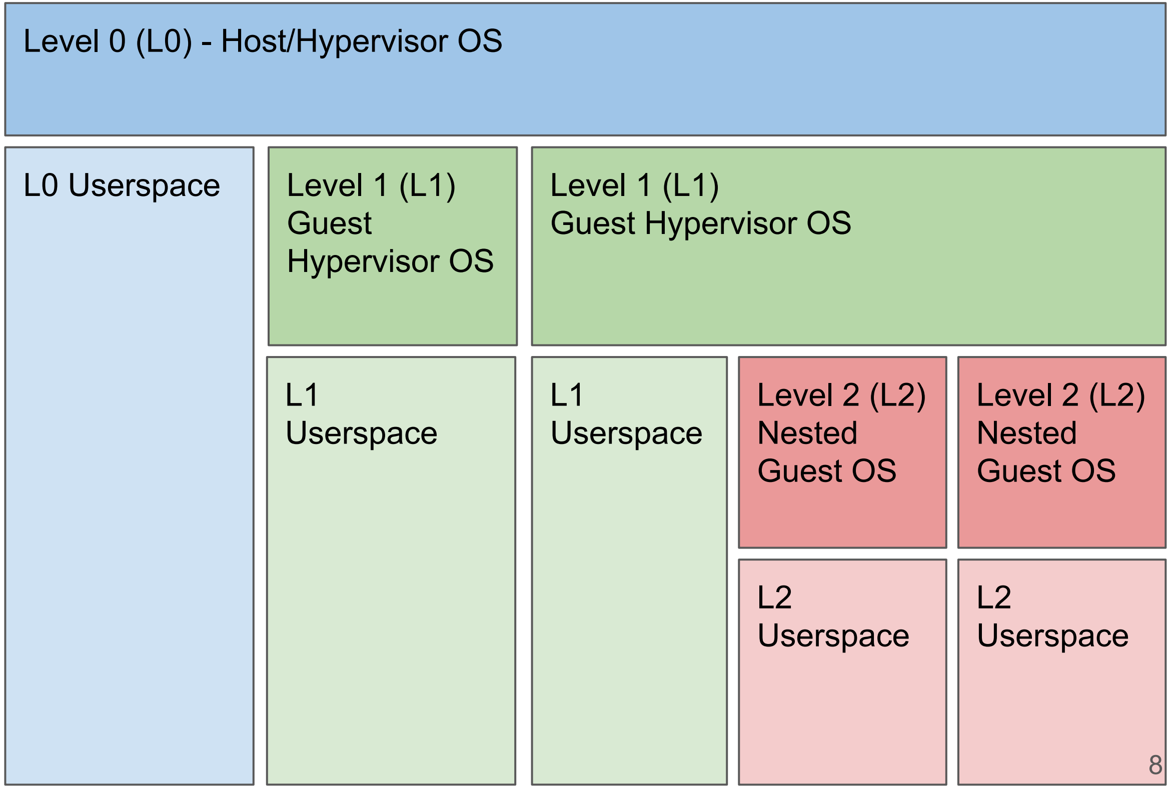
Level 1 (L1)
Guest OS

L1
Userspace

L1
Userspace

Some Terminology

- What is KVM?
- What is Nested KVM?
  - L0 Hypervisor
  - L1 Guest (Hypervisor)
  - L2 (Nested) Guest


Level 0 (L0) - Host/Hypervisor OS
L0 Userspace
Level 1 (L1) Guest Hypervisor OS
Level 1 (L1) Guest Hypervisor OS
L1 Userspace
L1 Userspace
Level 2 (L2) Nested Guest OS
Level 2 (L2) Nested Guest OS
L2 Userspace
L2 Userspace

Hasn’t this been done before?

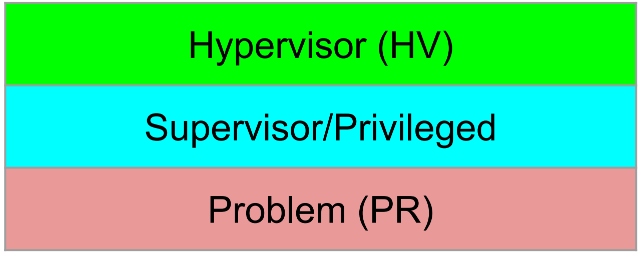
- Feature already present in:
    - x86
    - ARM
    - s390
    - PowerPC
        - KVM-PR

Hasn’t this been done before?

- Feature already present in:
  - x86
  - ARM
  - s390
  - PowerPC
    - KVM-PR
- KVM-HV vs KVM-PR

Hasn’t this been done before?

- Feature already present in:
    - x86
    - ARM
    - s390
    - PowerPC
        - KVM-PR
- 3 Privilege Levels - HV/SV/PR



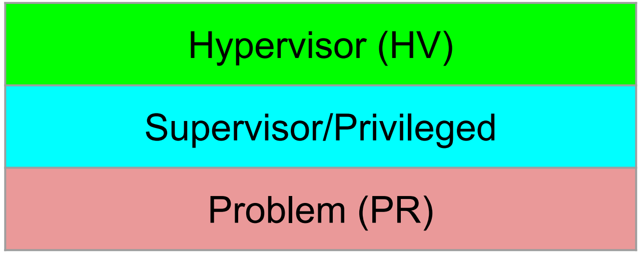
Hypervisor (HV)

Supervisor/Privileged

Problem (PR)

Hasn’t this been done before?

- Feature already present in:
  - x86
  - ARM
  - s390
  - PowerPC
    - KVM-PR
- 3 Privilege Levels - HV/SV/PR
- KVM-HV vs KVM-PR



Hypervisor (HV)

Supervisor/Privileged

Problem (PR)

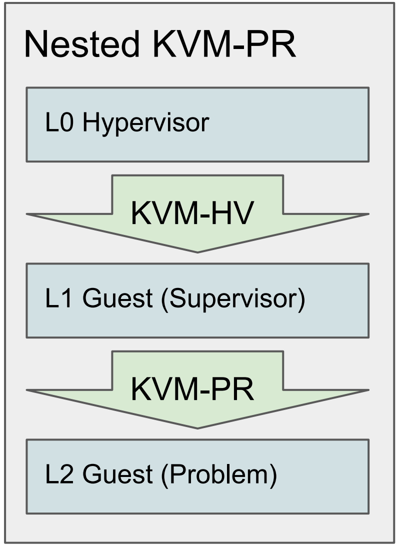
Hasn’t this been done before?

Hypervisor (HV)

Supervisor/Privileged

Problem (PR)

- Feature already present in:
    - x86
    - ARM
    - s390
    - PowerPC
        - KVM-PR
- Nested KVM-PR
    - L1 guest runs in supervisor mode
    - L2 guest runs in userspace
    - L1 emulates supervisor instructions for L2



Nested KVM-PR

L0 Hypervisor

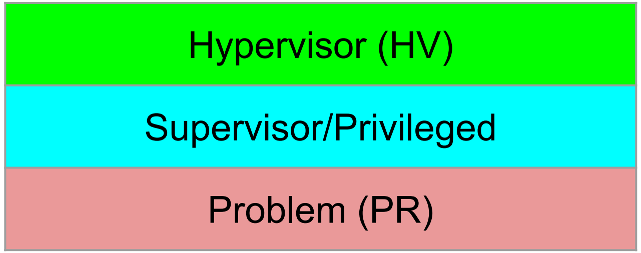
KVM-HV

L1 Guest (Supervisor)

KVM-PR

L2 Guest (Problem)

Hasn’t this been done before?

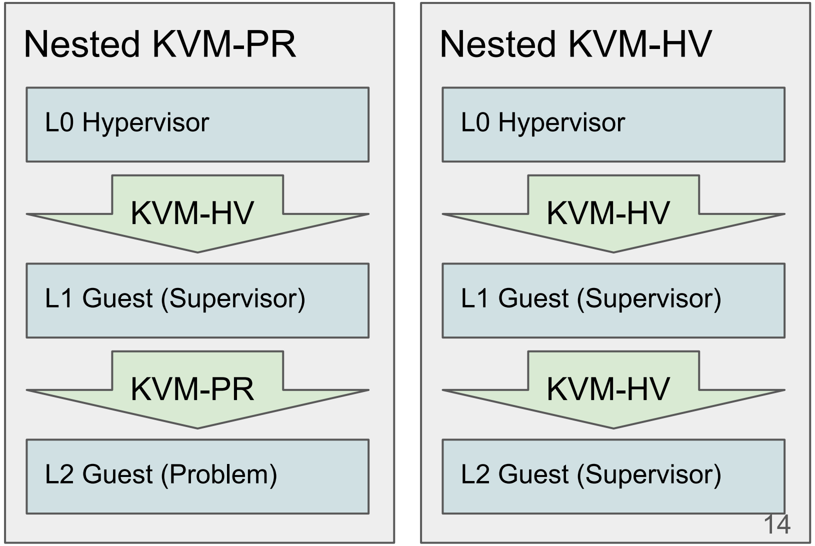


Hypervisor (HV)

Supervisor/Privileged

Problem (PR)

- Feature already present in:
  - x86
  - ARM
  - s390
  - PowerPC
    - KVM-PR
- Nested KVM-PR
  - L1 guest runs in supervisor mode
  - L2 guest runs in userspace
  - L1 emulates supervisor instructions for L2
- Nested KVM-HV
  - L1 guest runs in supervisor mode
  - L2 guest runs in supervisor mode
  - No need to emulate supervisor instructions
  - L0 emulates hypervisor instructions for L1



But Why?

- Testing
  - Openstack requires large number of hardware configurations
  - Able to test hypervisor changes in a virtualised environment
  - Able to test hypervisor management software
  - Able to test migration of hypervisors
- Ability to run guests even if already virtualised (e.g. the cloud)
- Faster development process
- Because we could!!!

¯_(ツ)_/¯

Breath

So how do we make this happen?

- Nested KVM-HV
- Want to run a KVM-HV guest inside another KVM-HV guest

1.

L1 → L2

So how do we make this happen?

- Nested KVM-HV
- Want to run a KVM-HV guest inside another KVM-HV guest
- Getting from the L1 guest into the L2 guest

1.	2.
L1 → L2	EA-GRA-HRA

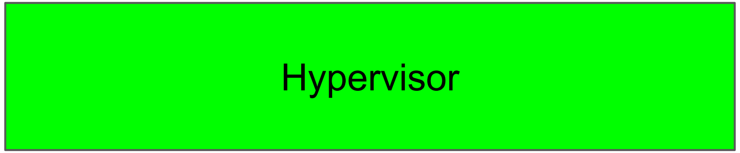
So how do we make this happen?

- Nested KVM-HV
- Want to run a KVM-HV guest inside another KVM-HV guest
- Getting from the L1 guest into the L2 guest
- L2 guest address translation
  - Instruction Address
  - Data Address

1.	2.
L1 → L2	EA-GRA-HRA

What happens when you run a guest?

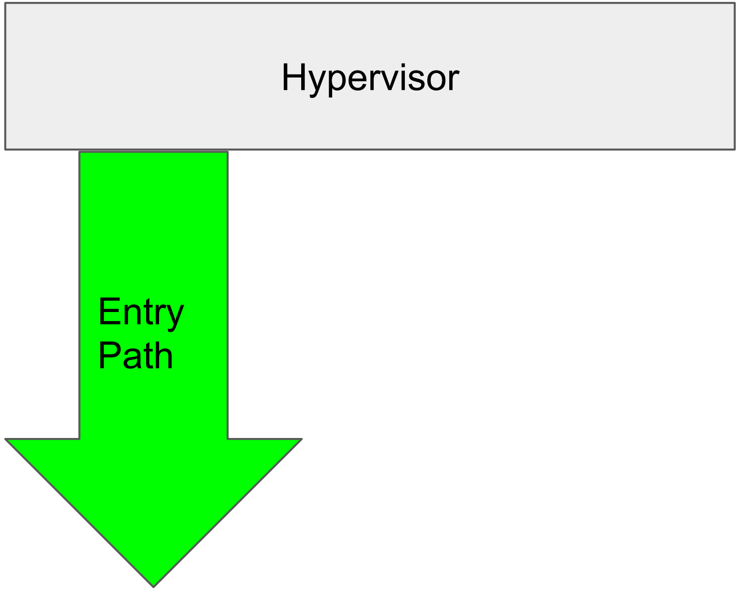
- L0 has the state of the L1 guest saved in memory



1.	2.
L1 → L2	EA-GRA-HRA

What happens when you run a guest?

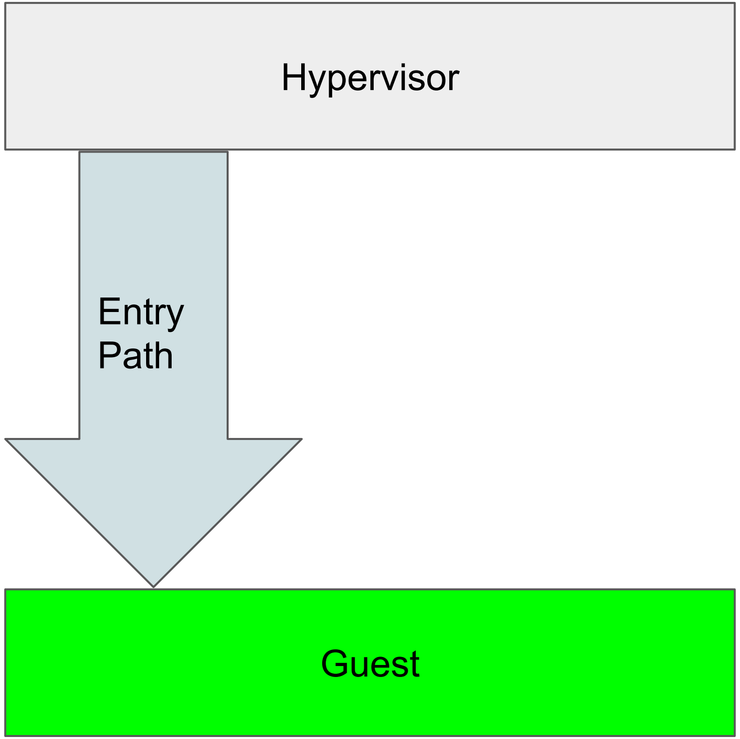
- L0 has the state of the L1 guest saved in memory
- Entry Path:
  - L0 decides to schedule L1 guest
  - Load L1 state onto the cpu
  - HRFID to guest



1.	2.
L1 → L2	EA-GRA-HRA

What happens when you run a guest?

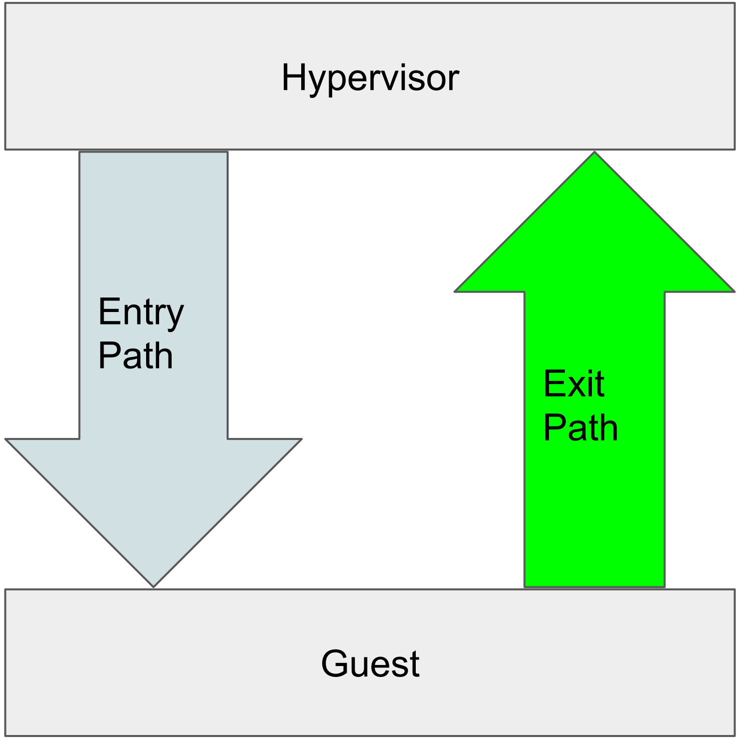
- L0 has the state of the L1 guest saved in memory
- Entry Path:
  - L0 decides to schedule L1 guest
  - Load L1 state onto the cpu
  - HRFID to guest
  - Guest is now executing



1.	2.
L1 → L2	EA-GRA-HRA

What happens when you run a guest?

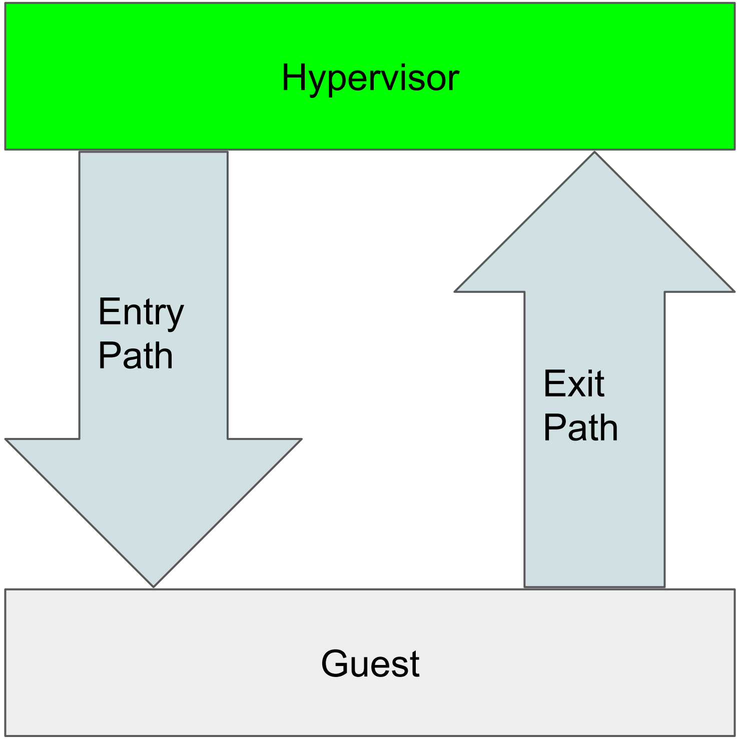
- L0 has the state of the L1 guest saved in memory
- Entry Path:
  - L0 decides to schedule L1 guest
  - Load L1 state onto the cpu
  - HRFID to guest
  - Guest is now executing
- Exit Path:
  - Interrupt returns control to L0 hypervisor
  - Save L1 state off the cpu into memory



1. L1 → L2	2. EA-GRA-HRA

What happens when you run a guest?

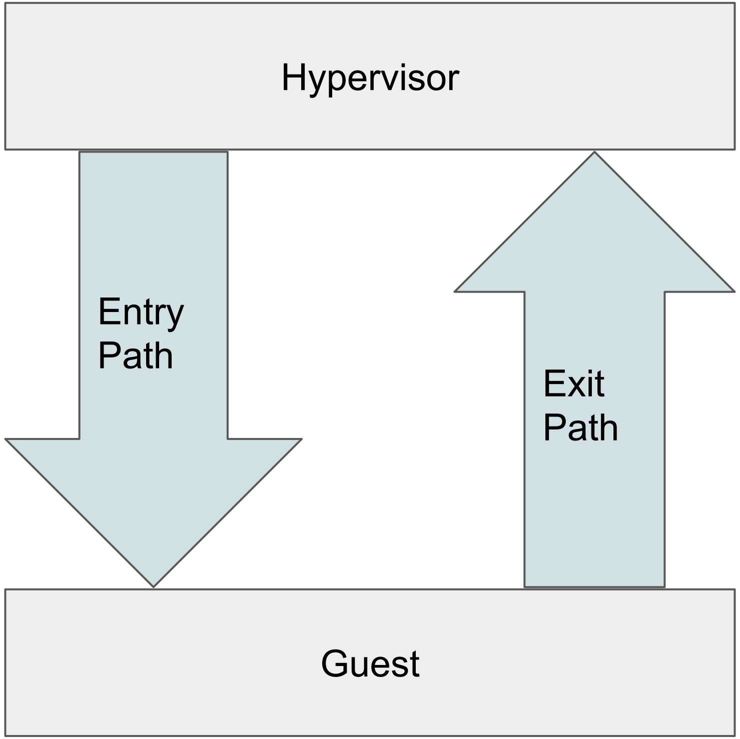
- L0 has the state of the L1 guest saved in memory
- Entry Path:
  - L0 decides to schedule L1 guest
  - Load L1 state onto the cpu
  - HRFID to guest
  - Guest is now executing
- Exit Path:
  - Interrupt returns control to L0 hypervisor
  - Save L1 state off the cpu into memory
  - Resume execution in the host



1.	2.
L1 → L2	EA-GRA-HRA

What happens when you run a guest?

- L0 has the state of the L1 guest saved in memory
- Entry Path:
  - L0 decides to schedule L1 guest
  - Load L1 state onto the cpu
  - HRFID to guest
  - Guest is now executing
- Exit Path:
  - Interrupt returns control to L0 hypervisor
  - Save L1 state off the cpu into memory
  - Resume execution in the host
- L0 also maintains page tables to manage the partitioning of memory for the guest real address space

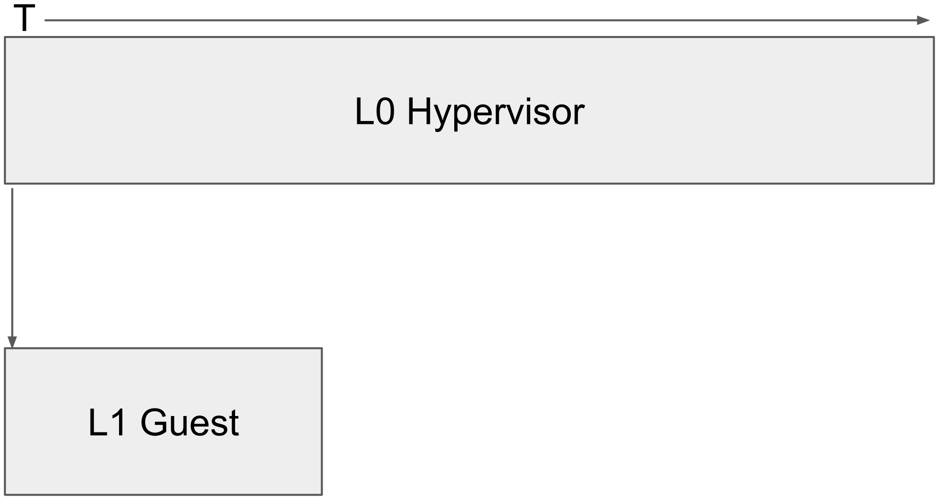


1. L1 → L2

2. EA-GRA-HRA

Nested Guest Entry - Trap and Emulate

- L0 runs L1



T

L0 Hypervisor

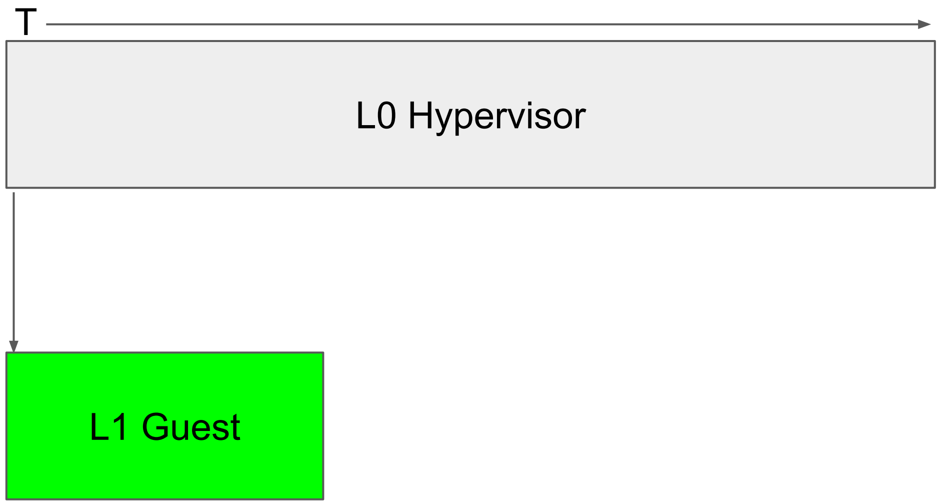
L1 Guest

1. L1 → L2

2. EA-GRA-HRA

Nested Guest Entry - Trap and Emulate

- L0 runs L1
- L1 tries to run L2
    - L1 Supervisor mode



1.

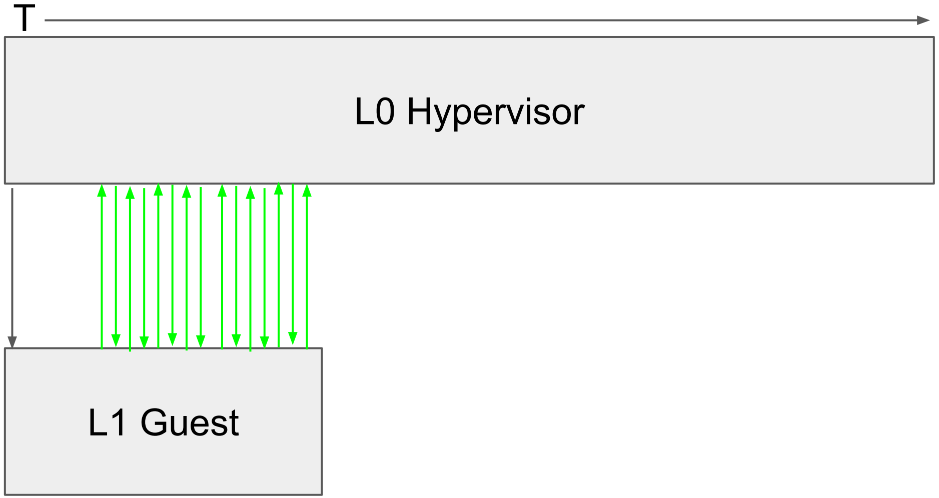
L1 → L2

2.

EA-GRA-HRA

Nested Guest Entry - Trap and Emulate

- L0 runs L1
- L1 tries to run L2
  - L1 Supervisor mode
  - L1 uses KVM-HV entry path to load up L2 state
    - HV instructions
    - HV SPRs
  - Trap to L0 and emulate

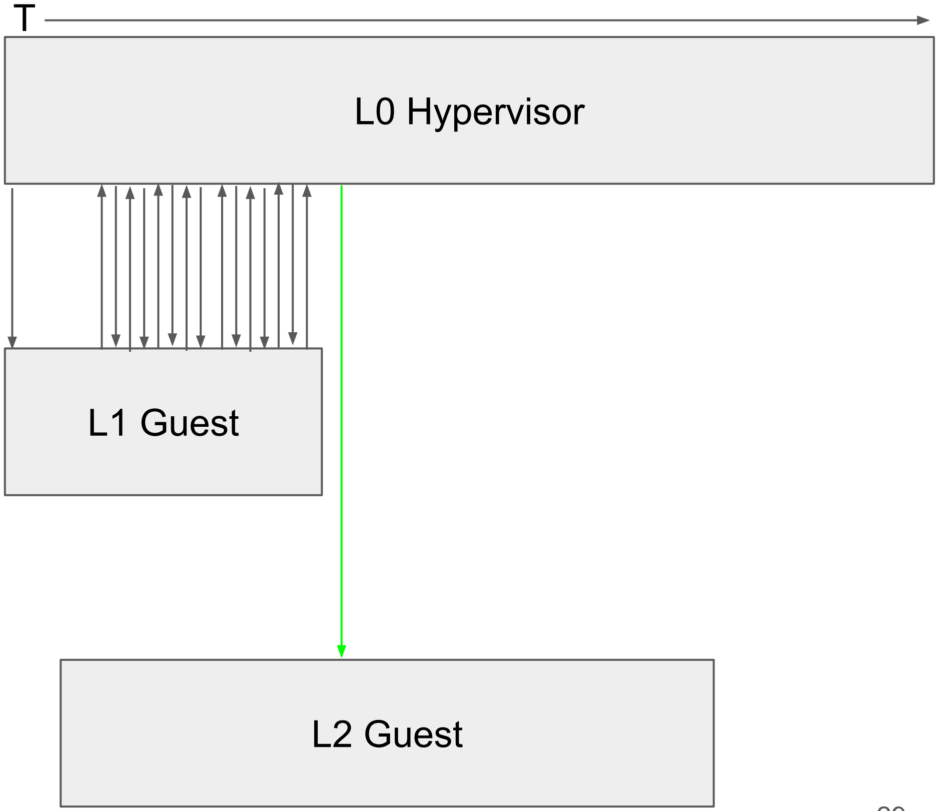


1. L1 → L2

2. EA-GRA-HRA

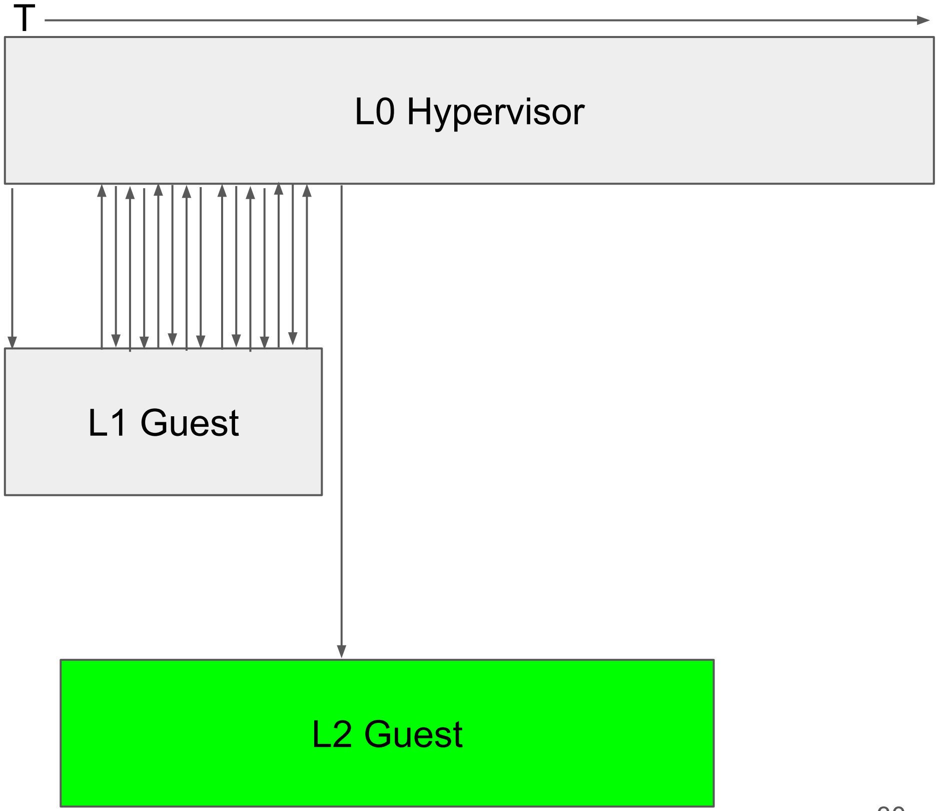
Nested Guest Entry - Trap and Emulate

- L0 runs L1
- L1 tries to run L2
  - L1 Supervisor mode
  - L1 uses KVM-HV entry path to load up L2 state
    - HV instructions
    - HV SPRs
  - Trap to L0 and emulate
  - L1 executes HRFID
  - L0 knows L1 wants to enter its guest
  - L0 loads L2 state onto the cpu and HRFIDs



Nested Guest Entry - Trap and Emulate

- L0 runs L1
- L1 tries to run L2
  - L1 Supervisor mode
  - L1 uses KVM-HV entry path to load up L2 state
    - HV instructions
    - HV SPRs
  - Trap to L0 and emulate
  - L1 executes HRFID
  - L0 knows L1 wants to enter its guest
  - L0 loads L2 state onto the cpu and HRFIDs
  - L2 guest is now executing in supervisor state just as L1 was

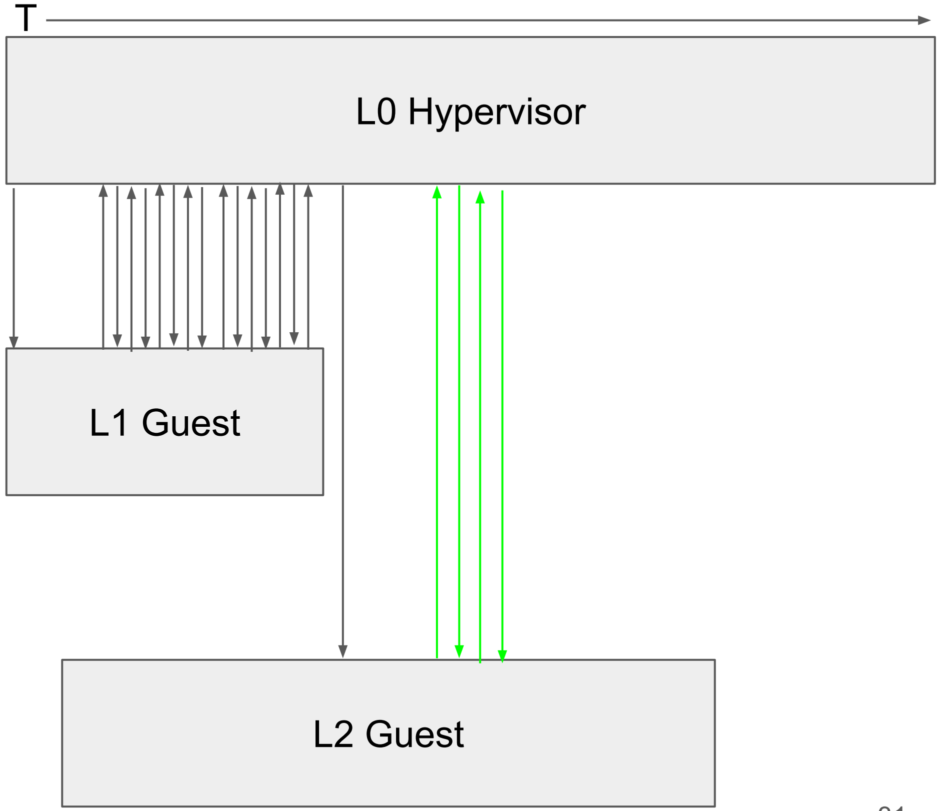


1. L1 → L2

2. EA-GRA-HRA

Nested Guest Entry - Trap and Emulate

- Trap returns execution to L0
    - Trap handled by L0 and immediately returns to L2

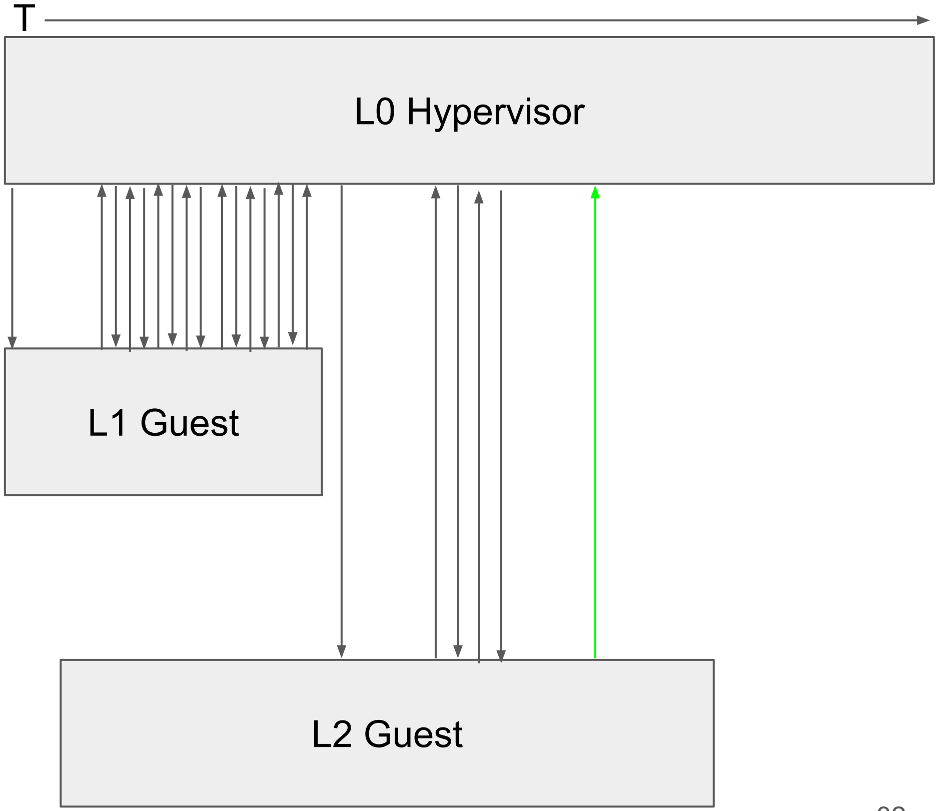


1. L1 → L2

2. EA-GRA-HRA

Nested Guest Entry - Trap and Emulate

- Trap returns execution to L0
  - Trap handled by L0 and immediately returns to L2
- Trap which requires handling in L1
  - L0 forwards the trap down to L1

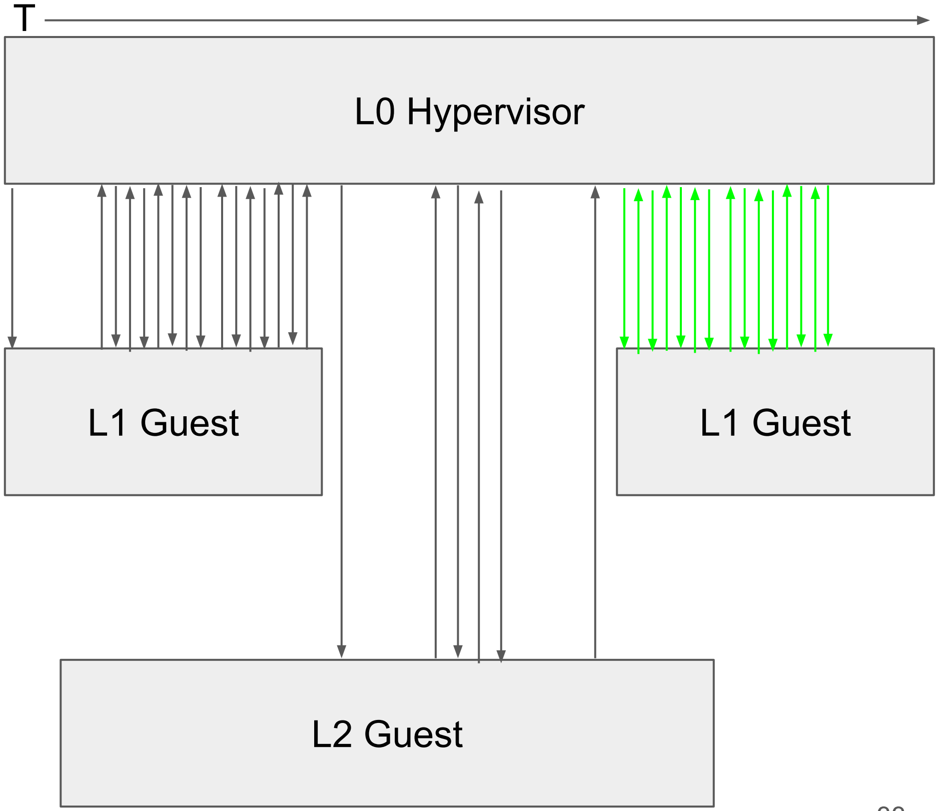


1. L1 → L2

2. EA-GRA-HRA

Nested Guest Entry - Trap and Emulate

- Trap returns execution to L0
  - Trap handled by L0 and immediately returns to L2
- Trap which requires handling in L1
  - L0 forwards the trap down to L1
  - L1 uses the KVM exit path to save L2 state
    - HV Instructions
    - HV SPRs
  - Trap to L0 and emulate

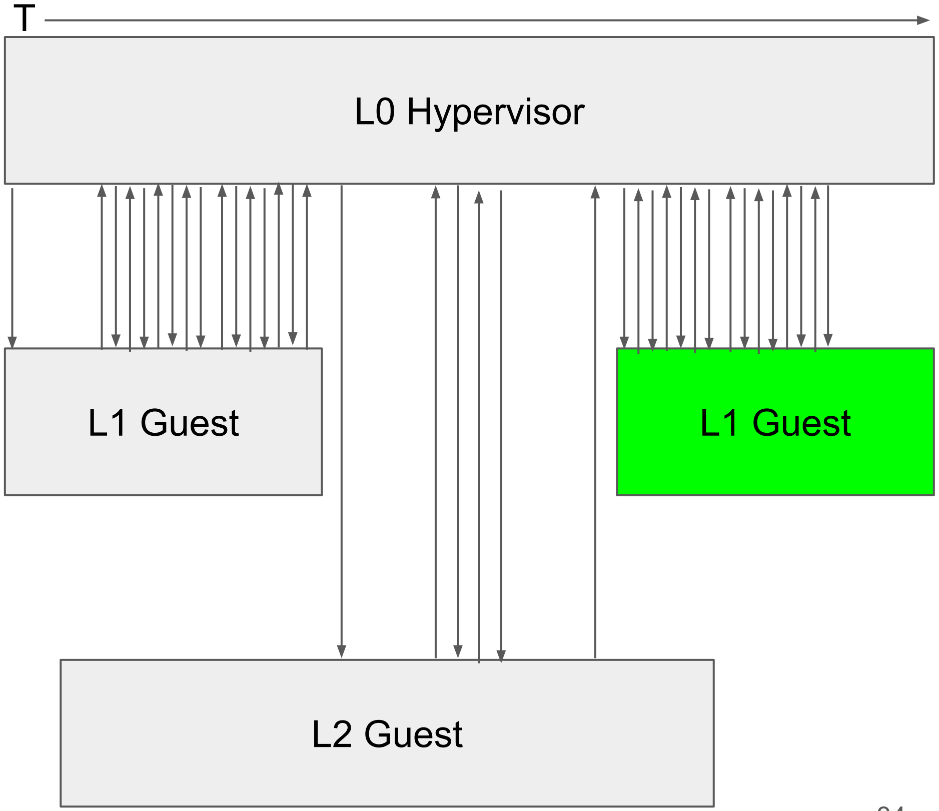


1. L1 → L2

2. EA-GRA-HRA

Nested Guest Entry - Trap and Emulate

- Trap returns execution to L0
  - Trap handled by L0 and immediately returns to L2
- Trap which requires handling in L1
  - L0 forwards the trap down to L1
  - L1 uses the KVM exit path to save L2 state
    - HV Instructions
    - HV SPRs
  - Trap to L0 and emulate
  - L1 guest continues to execute as normal

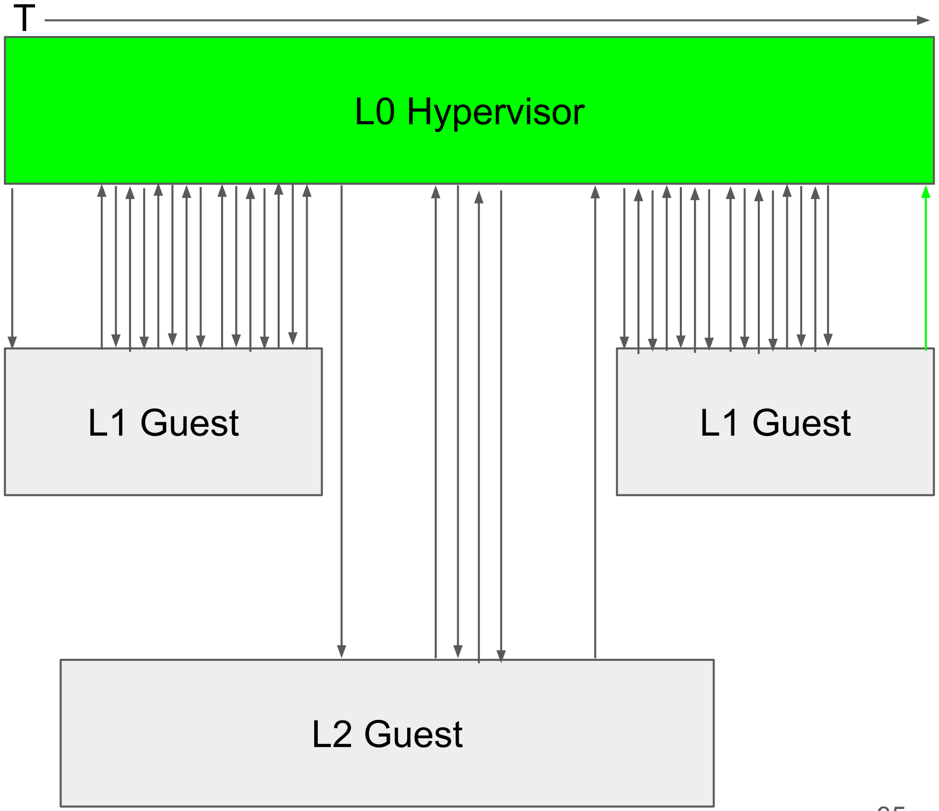


1. L1 → L2

2. EA-GRA-HRA

Nested Guest Entry - Trap and Emulate

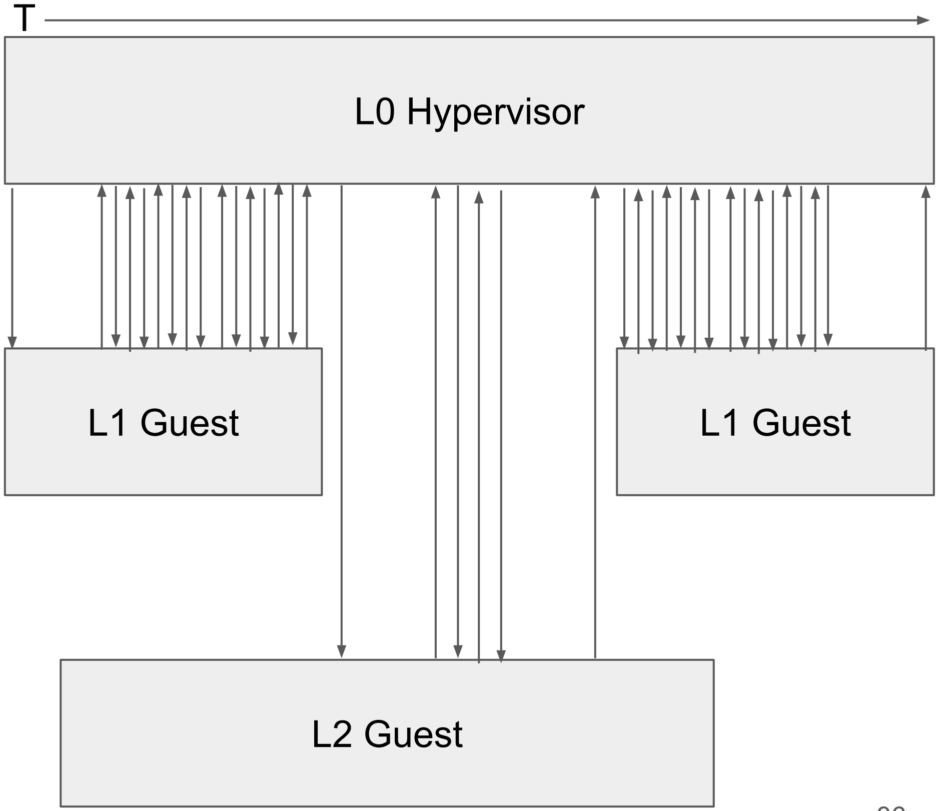
- Trap returns execution to L0
  - Trap handled by L0 and immediately returns to L2
- Trap which requires handling in L1
  - L0 forwards the trap down to L1
  - L1 uses the KVM exit path to save L2 state
    - HV Instructions
    - HV SPRs
  - Trap to L0 and emulate
  - L1 guest continues to execute as normal
- Trap returns execution to L0
  - L1 waits to be scheduled again



1. L1 → L2	2. EA-GRA-HRA

Nested Guest Entry - Trap and Emulate

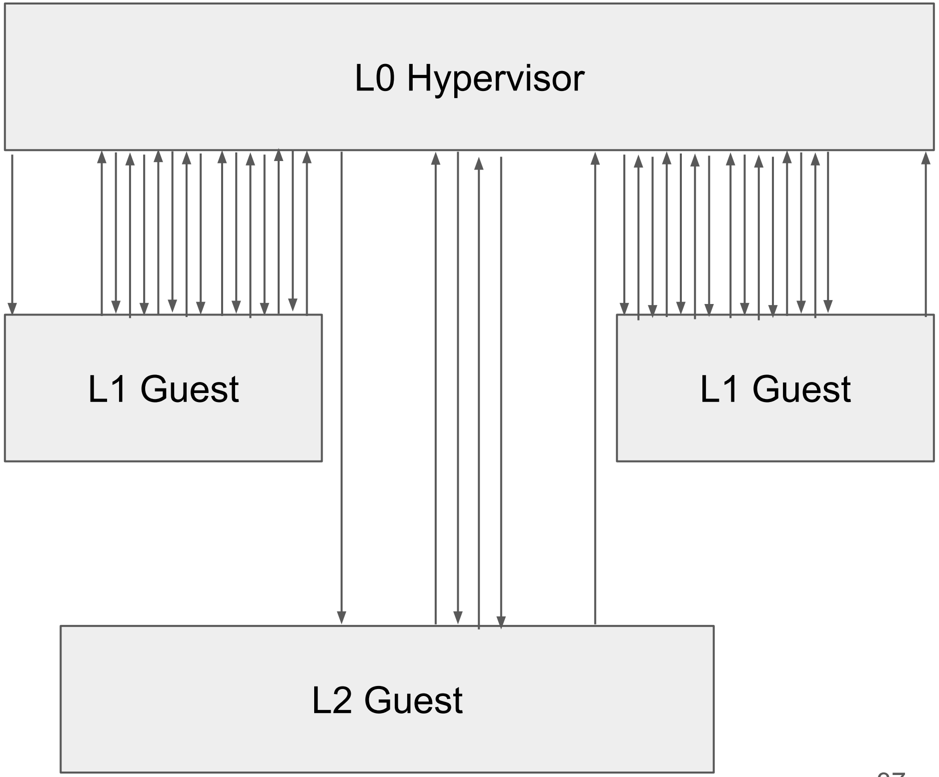
- Trap and emulate approach is slow
  - Many context switches from L0 <-> L1 to enter L2
  - Gets worse the deeper you nest



1. L1 → L2	2. EA-GRA-HRA

Is there a better way?

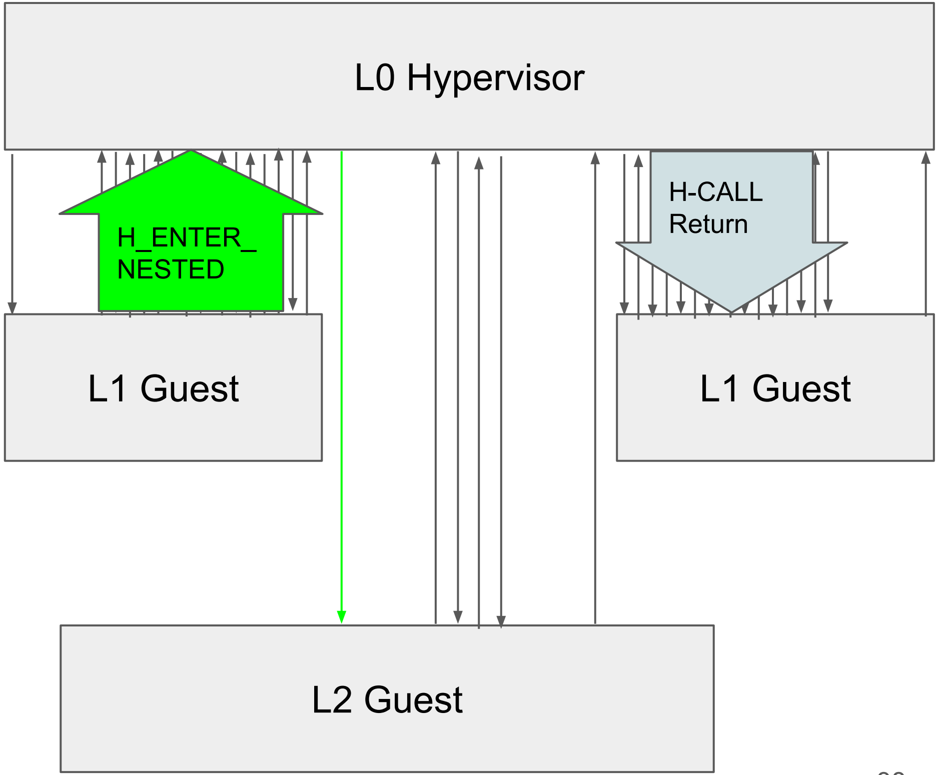
- Paravirtualise with an H-CALL



1. L1 → L2	2. EA-GRA-HRA

Is there a better way?

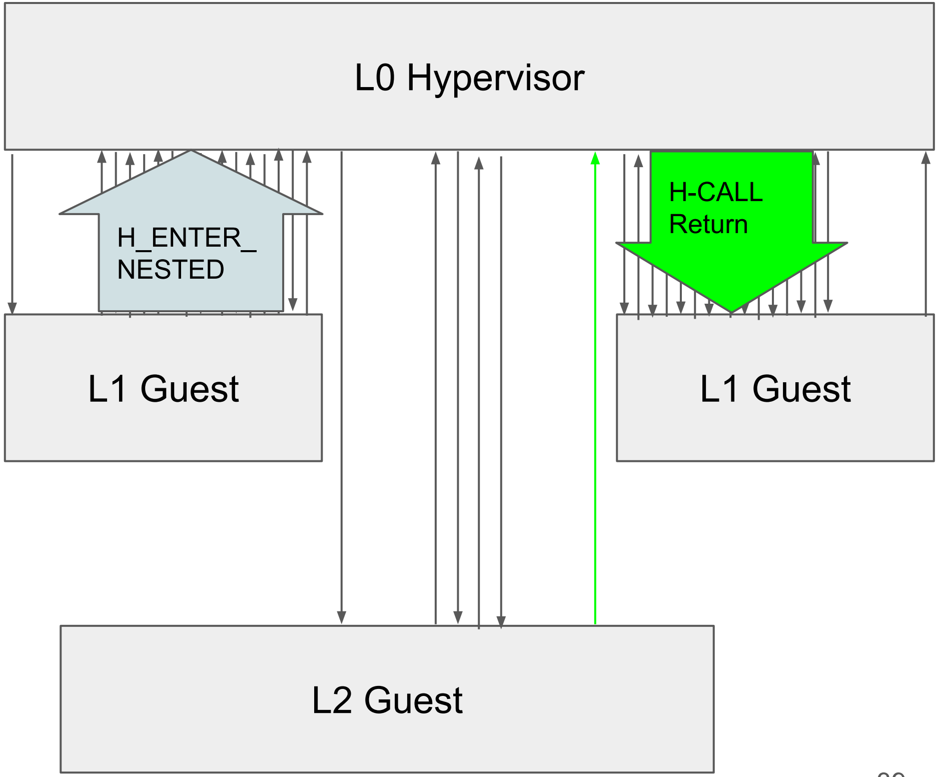
- Paravirtualise with an H-CALL
- H_ENTER_NESTED
    - L1 makes H-CALL to L0
        - Location in L1 memory of L2 state to use
        - L0 loads L2 state onto the cpu



1. L1 → L2	2. EA-GRA-HRA

Is there a better way?

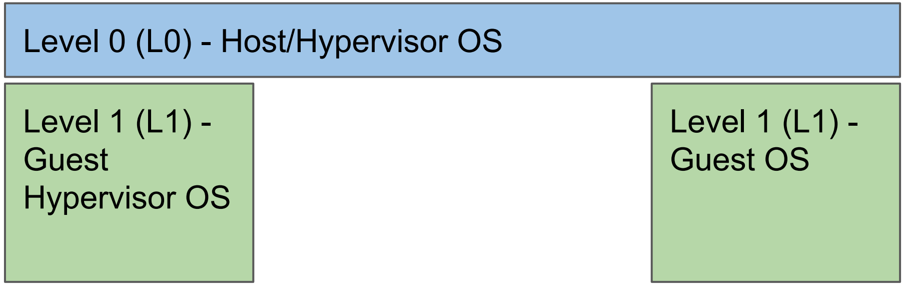
- Paravirtualise with an H-CALL
- H_ENTER_NESTED
  - L1 makes H-CALL to L0
    - Location in L1 memory of L2 state to use
    - L0 loads L2 state onto the cpu
  - Interrupt which needs handling in L1
    - Write L2 state back in to L1 memory
    - L0 returns to L1 from H-CALL



1. L1 → L2	2. EA-GRA-HRA

What L0 Sees

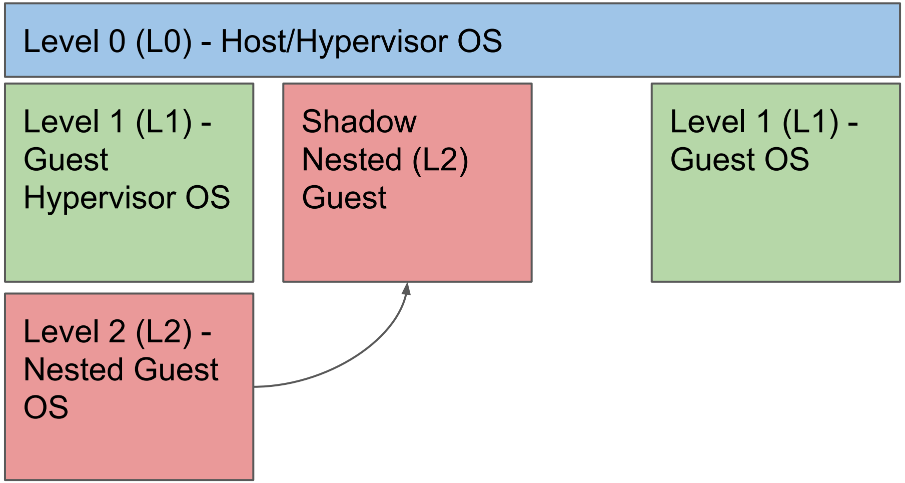
- How much state does L0 have to track for L2
    - L2 state mainly stored in L1 memory


Level 0 (L0) - Host/Hypervisor OS
Level 1 (L1) - Guest Hypervisor OS
Level 1 (L1) - Guest OS

1. L1 → L2	2. EA-GRA-HRA

What L0 Sees

- How much state does L0 have to track for L2
  - L2 state mainly stored in L1 memory
- Each nested guest essentially a “shadow” guest of L0

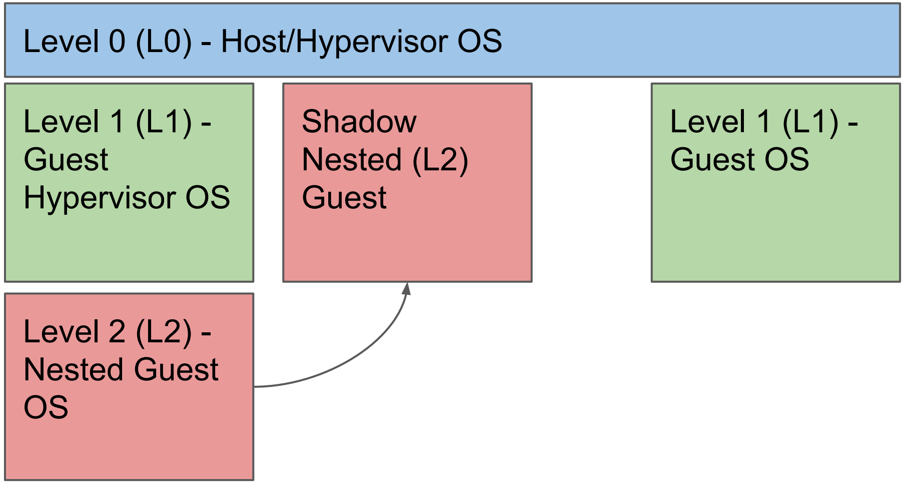


Level 0 (L0) - Host/Hypervisor OS
Level 1 (L1) - Guest Hypervisor OS
Shadow Nested (L2) Guest
Level 1 (L1) - Guest OS
Level 2 (L2) - Nested Guest OS

1. L1 → L2	2. EA-GRA-HRA

What L0 Sees

- How much state does L0 have to track for L2
  - L2 state mainly stored in L1 memory
- Each nested guest essentially a “shadow” guest of L0
- L0 must maintain some state for each nested guest
  - L1 LPID of this guest
  - Shadow L0 LPID for this guest
  - Shadow Page Tables
  - L2 Process Table



Level 0 (L0) - Host/Hypervisor OS
Level 1 (L1) - Guest Hypervisor OS
Shadow Nested (L2) Guest
Level 1 (L1) - Guest OS
Level 2 (L2) - Nested Guest OS

```
/*
 * Structure for a nested guest, that is, for a guest that is managed by
 * one of our guests.
 */
struct kvm_nested_guest {
        struct kvm *l1_host;            /* L1 VM that owns this nested guest */
        int l1_lpid;                    /* lpid L1 guest thinks this guest is */
        int shadow_lpid;                /* real lpid of this nested guest */
        pgd_t *shadow_pgtable;          /* our page table for this guest */
        u64 l1_gr_to_hr;                /* L1's addr of part'n-scoped table */
        u64 process_table;              /* process table entry for this guest */
        long refcnt;                    /* number of pointers to this struct */
        struct mutex tlb_lock;          /* serialize page faults and tlbies */
        struct kvm_nested_guest *next;
        cpumask_t need_tlb_flush;
        cpumask_t cpu_in_guest;
        short prev_cpu[NR_CPUS];
};
```

1. L1 → L2	2. EA-GRA-HRA

What Now?

- Enter Nested Guest
  - We can load up a nested guest context and start executing

1. L1 → L2

2. EA-GRA-HRA

What Now?

- Enter Nested Guest
  - We can load up a nested guest context and start executing
- Nested Guest Address Translation
  - We will take a page fault on the first L2 instruction
  - How do we translate L2 addresses?

Breath

Guest Address Translation

- Two level radix tree translation to get to a hardware address

Hardware Address

1. L1 → L2

2. EA-GRA-HRA

Guest Address Translation

- Two level radix tree translation
- Guest Effective Address
  - Analogous to a “Virtual Address”

Guest Effective Address (EA)

(Virtual Address)

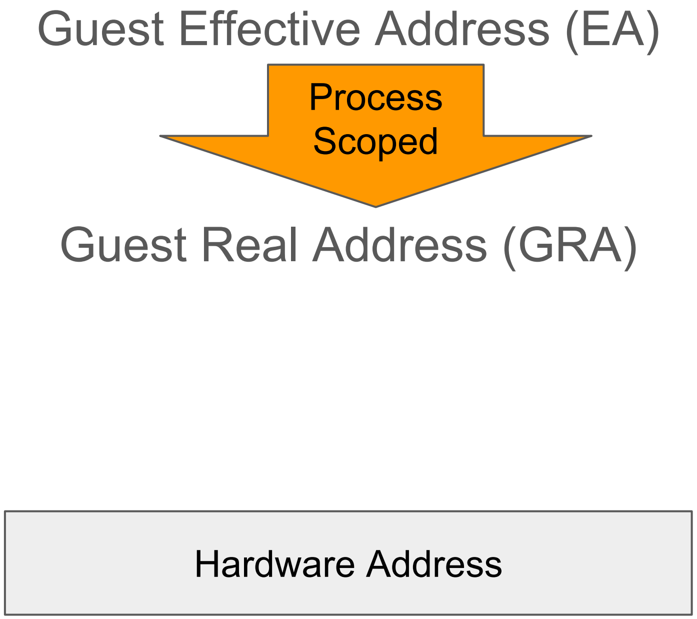
Hardware Address

1. L1 → L2

2. EA-GRA-HRA

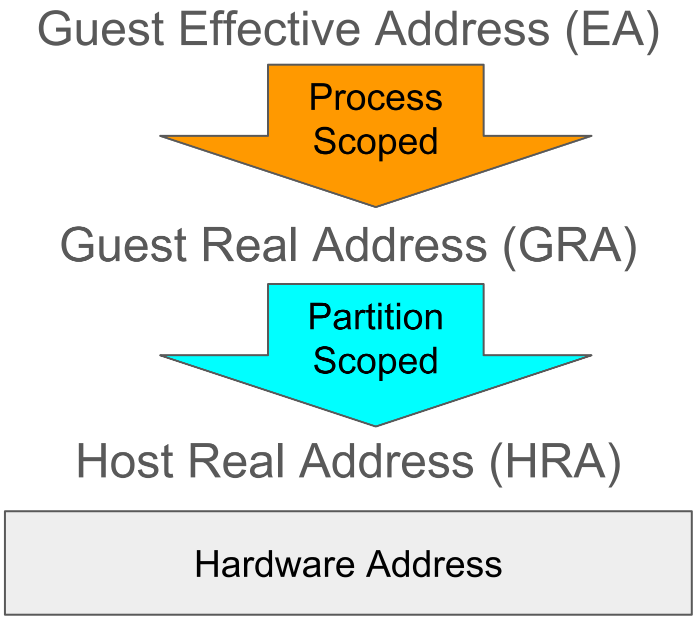
Guest Address Translation

- Two level radix tree translation
- Guest Effective Address
  - Analogous to a “Virtual Address”
- Process Scoped Translation
  - Radix trees in L1 memory
  - Managed by L1 to divide its memory
  - Associated with PID
  - Results in a Guest Real Address



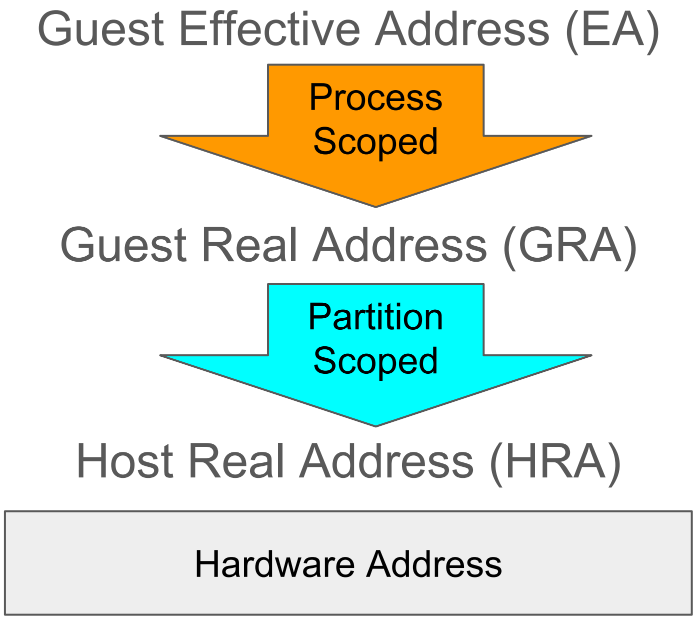
Guest Address Translation

- Two level radix tree translation
- Guest Effective Address
  - Analogous to a “Virtual Address”
- Process Scoped Translation
  - Radix trees in L1 memory
  - Managed by L1 to divide its memory
  - Associated with PID
  - Results in a Guest Real Address
- Partition Scoped Translation
  - Radix trees in L0 memory
  - Managed by L0 to divide its memory
  - Associated with LPID
  - Results in a Host Real Address
    - Hardware Address



Guest Address Translation

- Guest EA
    - Virtual Address
- PID
    - Per Process ID
    - Used to tag cache entries
    - Used for Process Scoped Translation
- LPID
    - Per Logical Partition ID
    - Used to tag cache entries
    - Host has one
        - Normally 0
    - One allocated for each Guest
        - 1, 5, 127
        - Unique to that Guest
    - Used for Partition Scoped Translation



Guest Effective Address (EA)

Process Scoped

Guest Real Address (GRA)

Partition Scoped

Host Real Address (HRA)

Hardware Address

Guest Address Translation

- All a bit hand wavy
- Let's walk through an example
  - EA -> HRA
  - LPID = 7
  - PID = 0
- Remember this is what the hardware is doing

1.	2.
L1 → L2	EA-GRA-HRA

Guest Address Translation

- Partition Table
  - In L0 memory
  - Entry per LPID
  - Pointer to partition scoped radix tree
  - Pointer to process table
    - In L1 memory

Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2	2. EA-GRA-HRA

Guest Address Translation

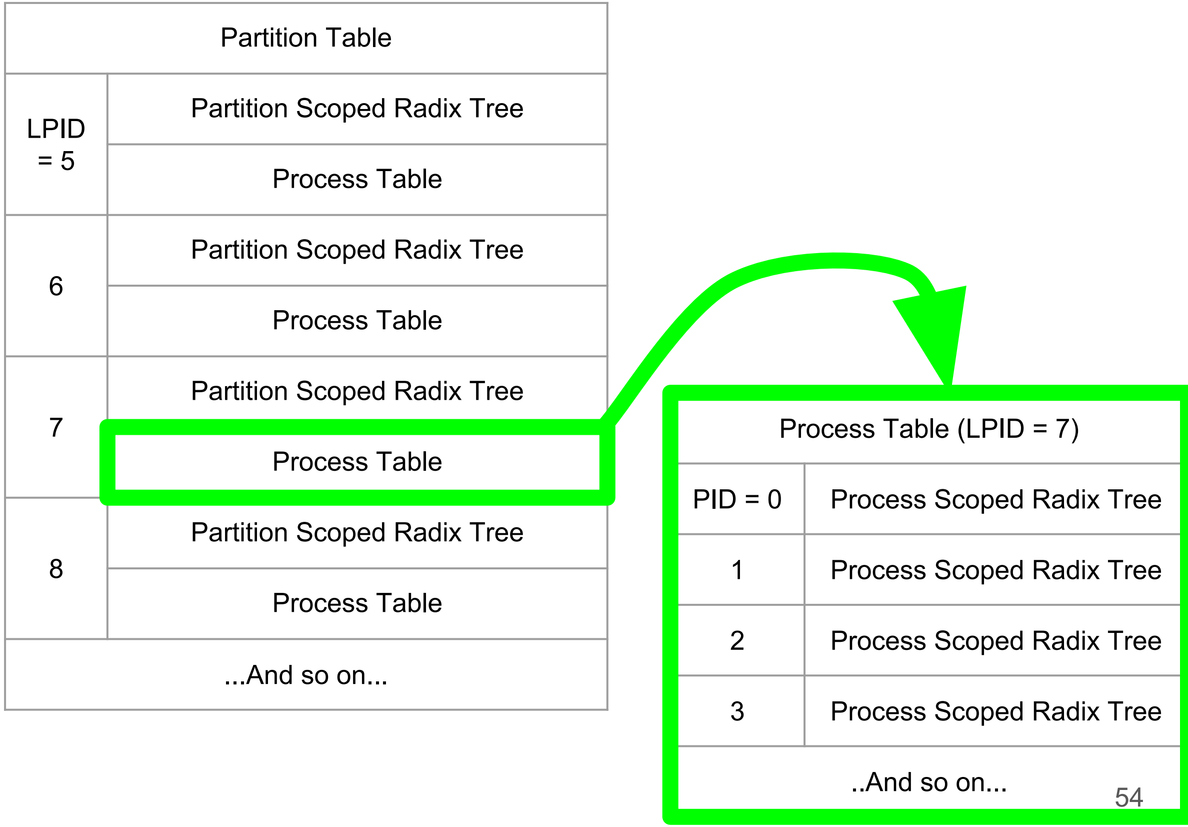
- Index by LPID = 7
- Select Partition Table Entry

Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2	2. EA-GRA-HRA

Process Scoped Address Translation

- Find the Process Table



Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

Process Table (LPID = 7)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

1. L1 → L2	2. EA-GRA-HRA

Process Scoped Address Translation

- Index by PID = 0
- Select the Process Table Entry
  - Pointer to Process Scoped Radix Tree

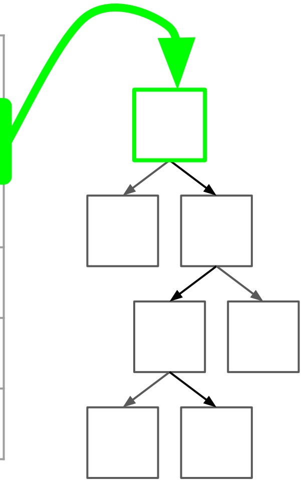
Process Table (LPID = 7)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

1. L1 → L2	2. EA-GRA-HRA

Process Scoped Address Translation

- Found the Process Scoped Radix Tree
- Translate Guest Effective Address (EA) to Guest Real Address (GRA)
  - By walking the radix tree

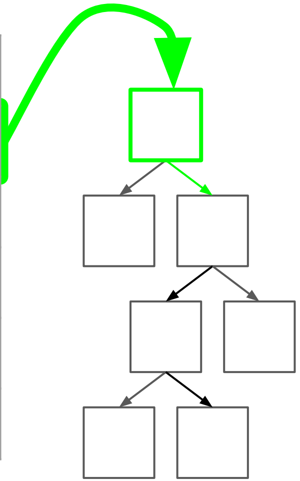
Process Table (LPID = 7)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	



Process Scoped Address Translation

- Found the Process Scoped Radix Tree
- Translate Guest Effective Address (EA) to Guest Real Address (GRA)
  - By walking the radix tree

Process Table (LPID = 7)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

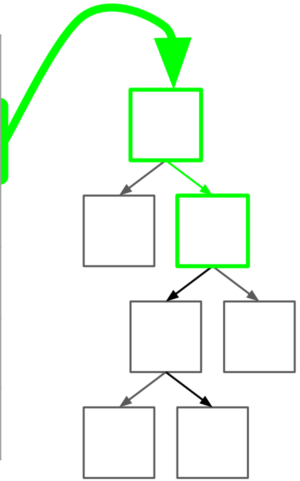


1. L1 → L2	2. EA-GRA-HRA

Process Scoped Address Translation

- Found the Process Scoped Radix Tree
- Translate Guest Effective Address (EA) to Guest Real Address (GRA)
    - By walking the radix tree

Process Table (LPID = 7)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

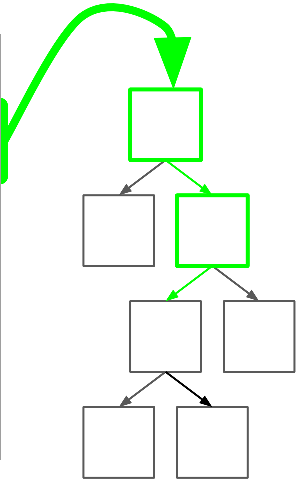


1.	2.
L1 → L2	EA-GRA-HRA

Process Scoped Address Translation

- Found the Process Scoped Radix Tree
- Translate Guest Effective Address (EA) to Guest Real Address (GRA)
  - By walking the radix tree

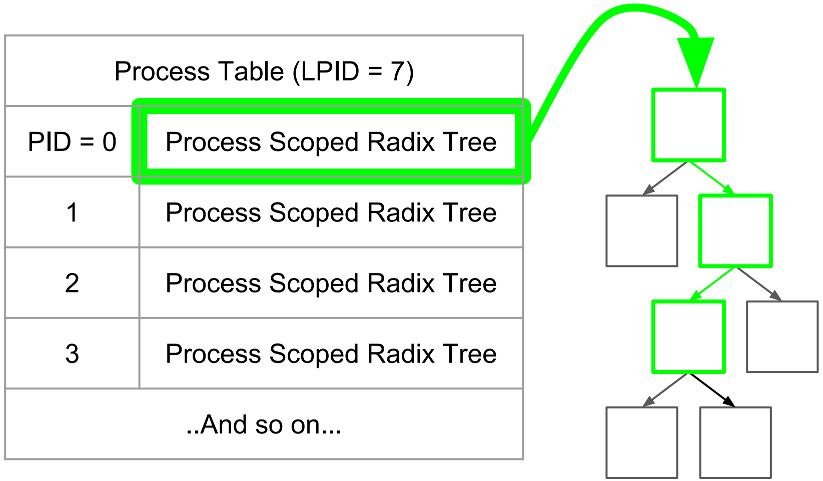
Process Table (LPID = 7)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	



1.	2.
L1 → L2	EA-GRA-HRA

Process Scoped Address Translation

- Found the Process Scoped Radix Tree
- Translate Guest Effective Address (EA) to Guest Real Address (GRA)
    - By walking the radix tree



Process Table (LPID = 7)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

1.

L1 → L2

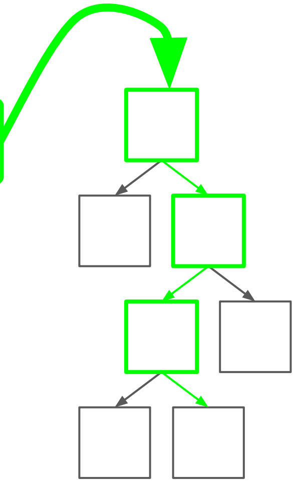
2.

EA-GRA-HRA

Process Scoped Address Translation

- Found the Process Scoped Radix Tree
- Translate Guest Effective Address (EA) to Guest Real Address (GRA)
    - By walking the radix tree

Process Table (LPID = 7)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

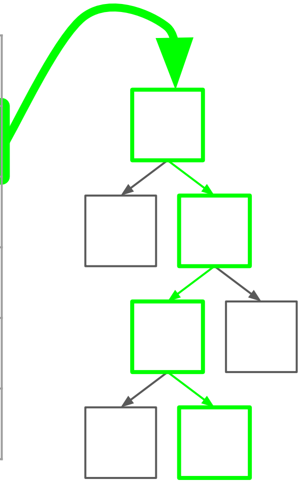


1. L1 → L2	2. EA-GRA-HRA

Process Scoped Address Translation

- Found the Process Scoped Radix Tree
- Translate Guest Effective Address (EA) to Guest Real Address (GRA)
  - By walking the radix tree

Process Table (LPID = 7)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

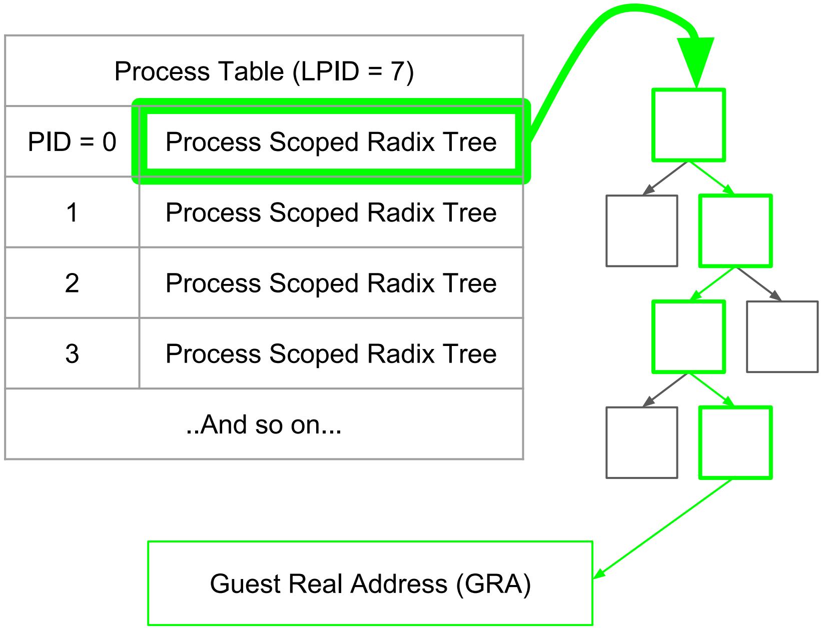


1. L1 → L2

2. EA-GRA-HRA

Process Scoped Address Translation

- We now have our Guest Real Address (GRA)



Process Table (LPID = 7)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

1. L1 → L2	2. EA-GRA-HRA

Partition Scoped Address Translation

- Now need to do partition scoped translation
- Index by LPID = 7

Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2	2. EA-GRA-HRA

Partition Scoped Address Translation

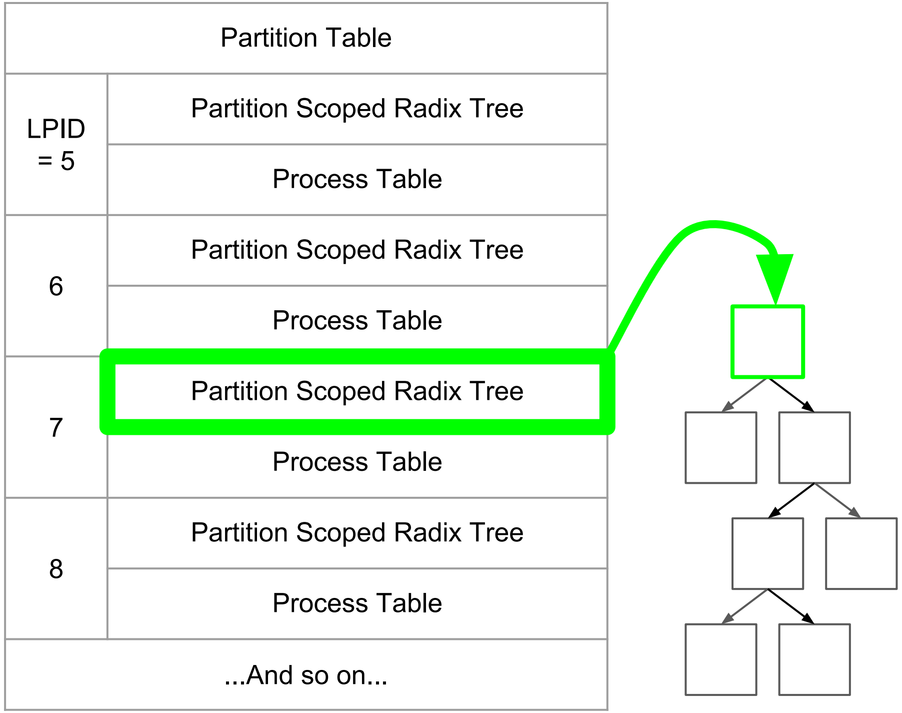
- Now need to do partition scoped translation
- Index by LPID = 7
- Select the Partition Scoped Radix Tree

Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2	2. EA-GRA-HRA

Partition Scoped Address Translation

- Found the Partition Scoped Radix Tree
- Translate Guest Real Address (GRA) to a Host Real Address (HRA)
  - By walking the radix tree



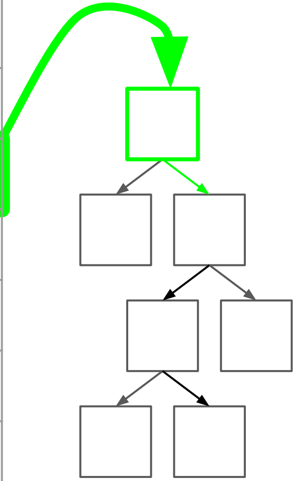
Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1.	2.
L1 → L2	EA-GRA-HRA

Partition Scoped Address Translation

- Found the Partition Scoped Radix Tree
- Translate Guest Real Address (GRA) to a Host Real Address (HRA)
    - By walking the radix tree

Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

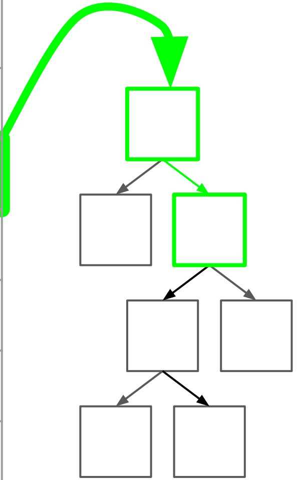


1.	2.
L1 → L2	EA-GRA-HRA

Partition Scoped Address Translation

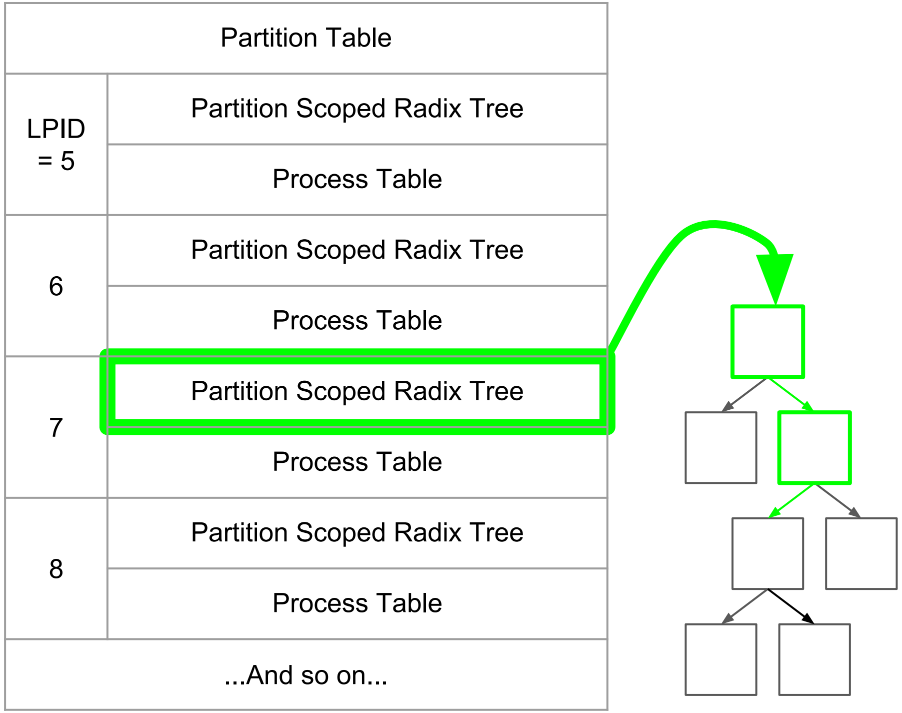
- Found the Partition Scoped Radix Tree
- Translate Guest Real Address (GRA) to a Host Real Address (HRA)
    - ○ By walking the radix tree

Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	



Partition Scoped Address Translation

- Found the Partition Scoped Radix Tree
- Translate Guest Real Address (GRA) to a Host Real Address (HRA)
  - By walking the radix tree

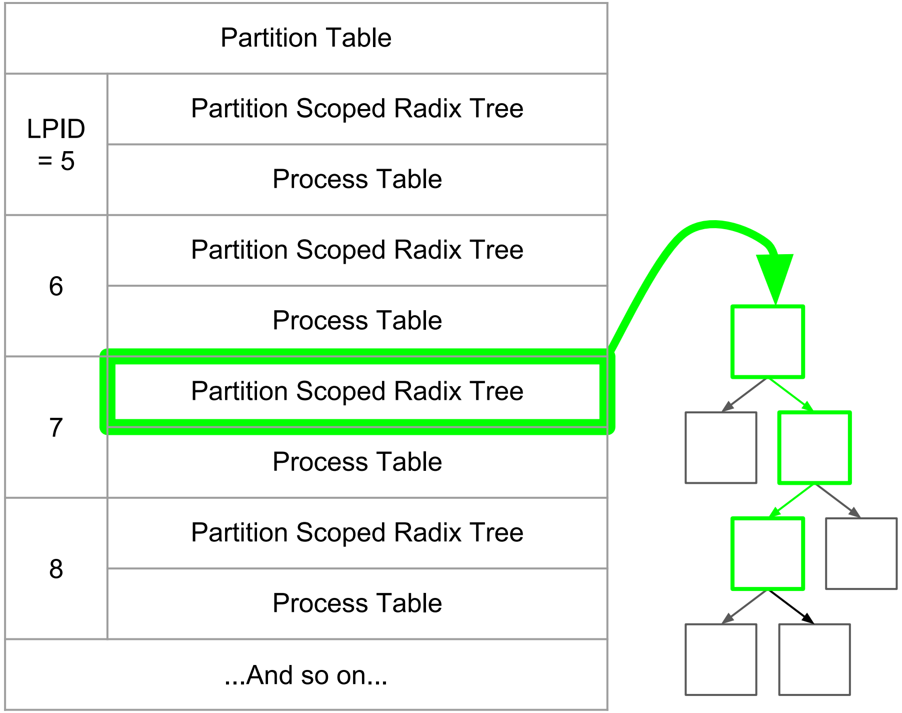


Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2	2. EA-GRA-HRA

Partition Scoped Address Translation

- Found the Partition Scoped Radix Tree
- Translate Guest Real Address (GRA) to a Host Real Address (HRA)
    - By walking the radix tree

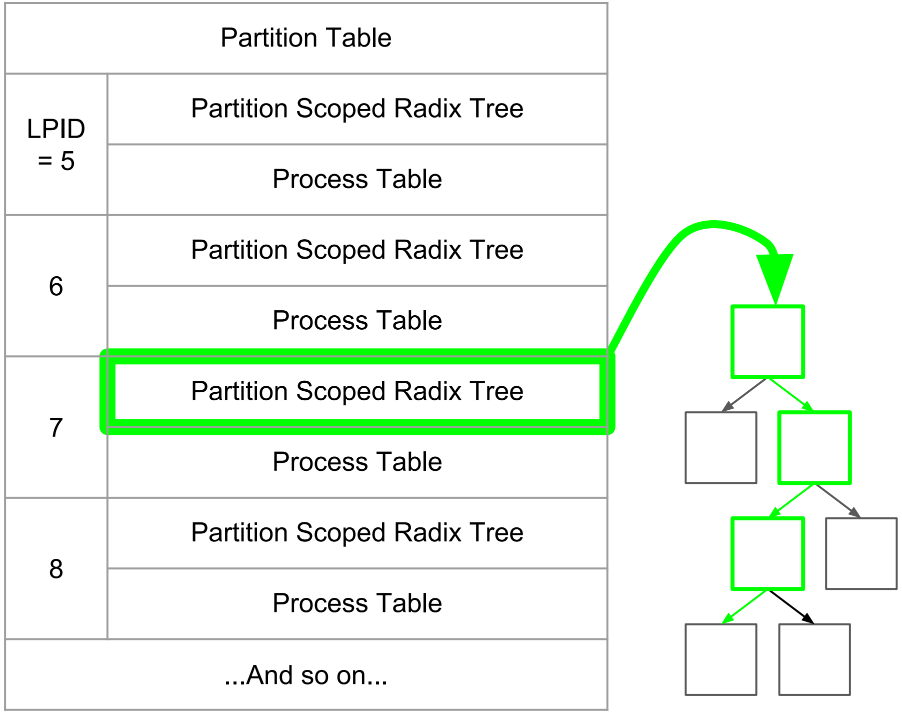


Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1.	2.
L1 → L2	EA-GRA-HRA

Partition Scoped Address Translation

- Found the Partition Scoped Radix Tree
- Translate Guest Real Address (GRA) to a Host Real Address (HRA)
  - By walking the radix tree



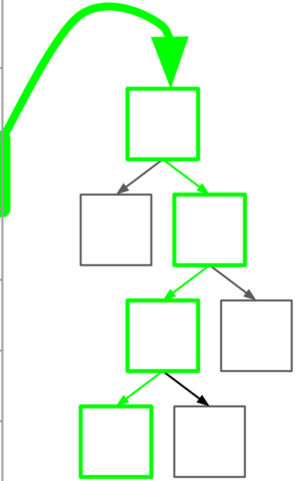
Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1.	2.
L1 → L2	EA-GRA-HRA

Partition Scoped Address Translation

- Found the Partition Scoped Radix Tree
- Translate Guest Real Address (GRA) to a Host Real Address (HRA)
  - By walking the radix tree

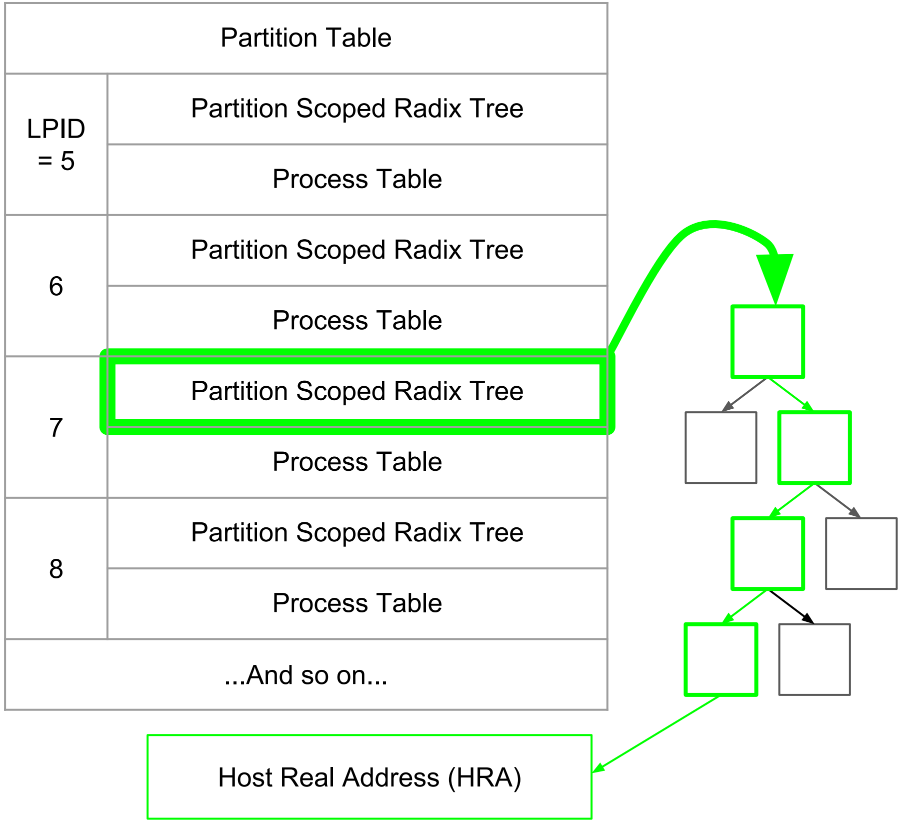
Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	



1.	2.
L1 → L2	EA-GRA-HRA

Partition Scoped Address Translation

- We now have our Host Real Address (HRA)
  - Can do the hardware access

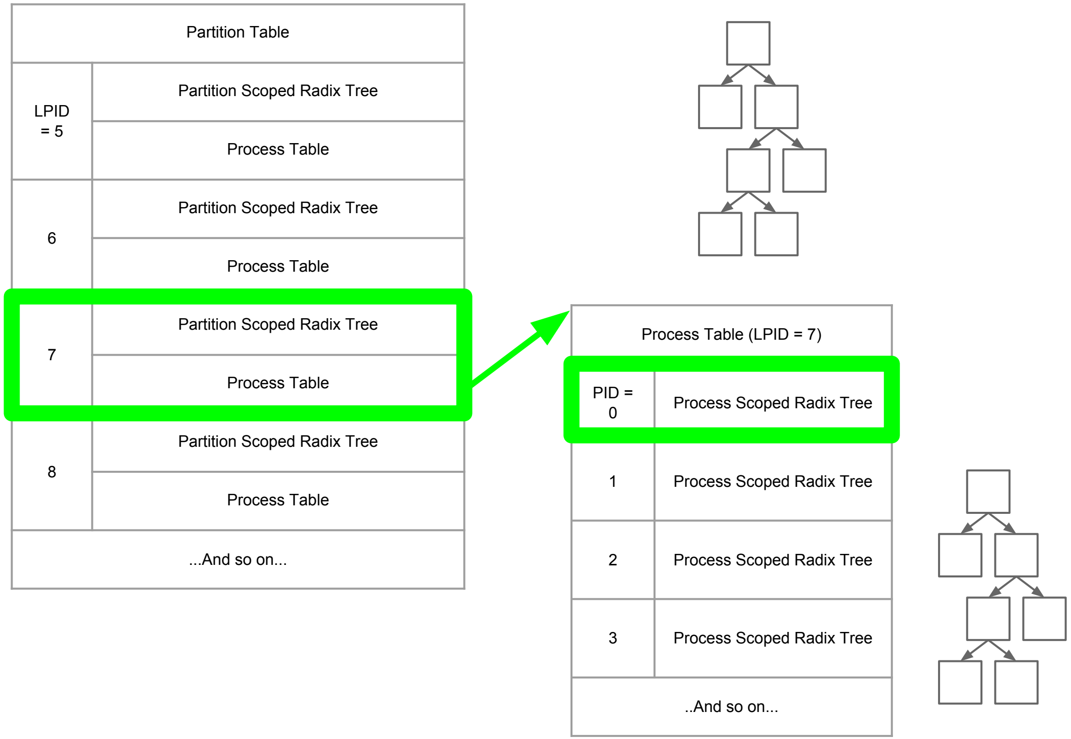


Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1.	2.
L1 → L2	EA-GRA-HRA

Guest Address Translation

- Quick Recap



Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

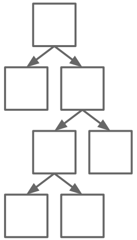
Process Table (LPID = 7)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

1.	2.
L1 → L2	EA-GRA-HRA

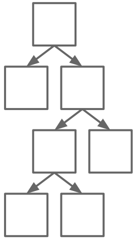
Guest Address Translation

Guest Effective Address (EA)

Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

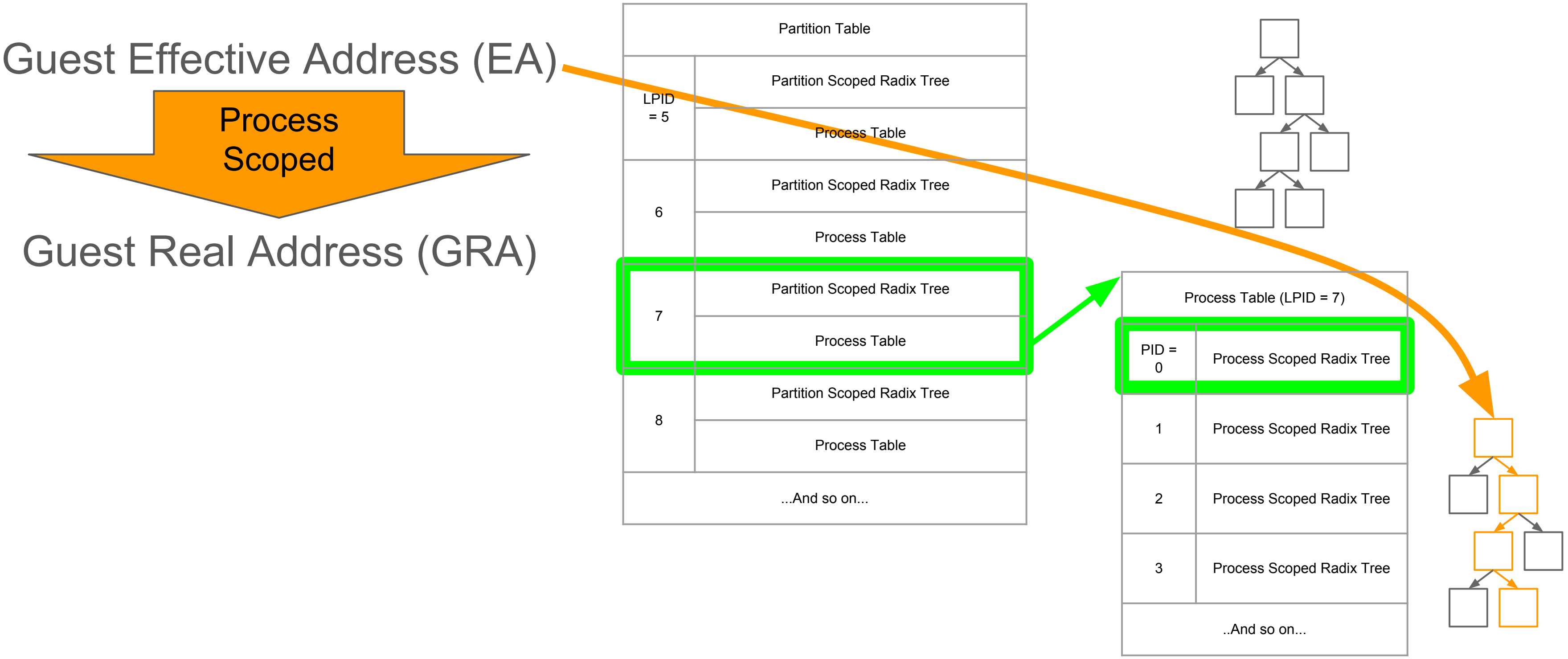


Process Table (LPID = 7)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	



1. L1 → L2	2. EA-GRA-HRA

Guest Address Translation



Guest Effective Address (EA)

Process Scoped

Guest Real Address (GRA)

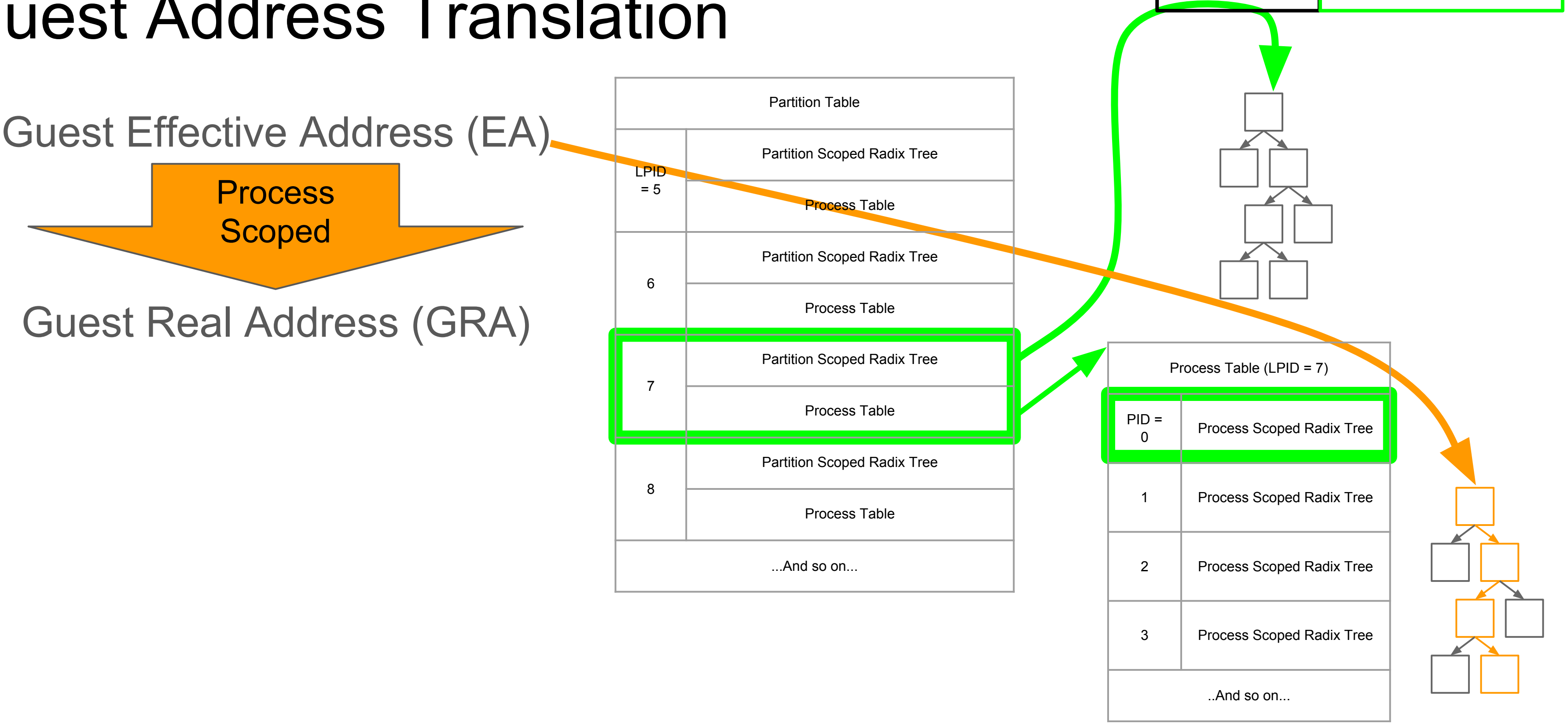
Partition Table		
LPID = 5	Partition Scoped Radix Tree	
	Process Table	
6	Partition Scoped Radix Tree	
	Process Table	
7	Partition Scoped Radix Tree	
	Process Table	
8	Partition Scoped Radix Tree	
	Process Table	
...And so on...		

Process Table (LPID = 7)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

Guest Address Translation

1. L1 → L2

2. EA-GRA-HRA



Guest Effective Address (EA)

Process Scoped

Guest Real Address (GRA)

Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

Process Table (LPID = 7)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

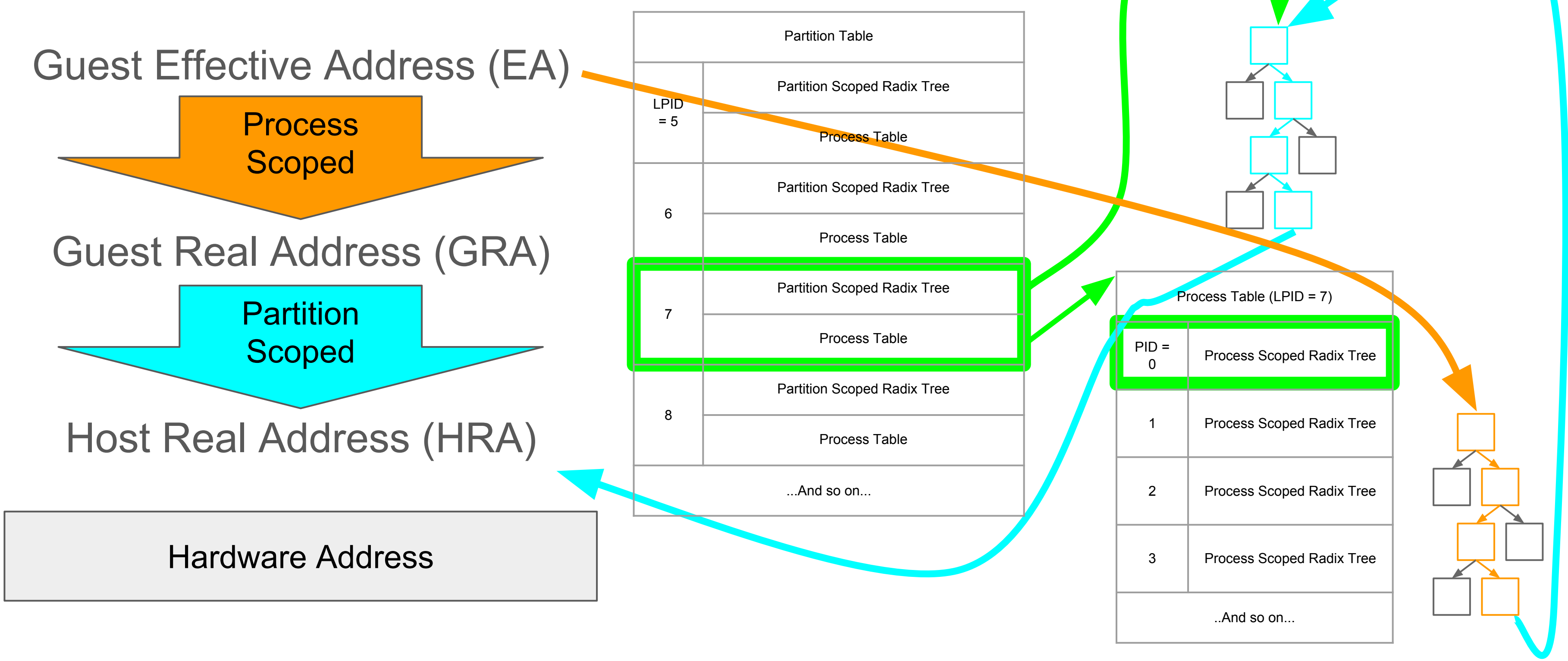
Guest Address Translation

1.

L1 → L2

2.

EA-GRA-HRA



Guest Effective Address (EA)

Process Scoped

Guest Real Address (GRA)

Partition Scoped

Host Real Address (HRA)

Hardware Address

Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

Process Table (LPID = 7)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

Breath

1.	2.
L1 → L2	EA-GRA-HRA

Nested Address Translation

- That seems pretty easy
- What about nested address translation?

1.	2.
L1 → L2	EA-GRA-HRA

Nested Address Translation

- L0 has a Partition Table for its guests
  - In L0 memory
  - Used to setup mappings for L1 GRA

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1.	2.
L1 → L2	EA-GRA-HRA

Nested Address Translation

- L0 has a Partition Table for its guests
- L1 has a Partition Table for its guests
  - In L1 memory
  - Used to setup mappings for L2 GRA

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2	2. EA-GRA-HRA

Nested Address Translation

- L0 has a Partition Table for its guests
- L1 has a Partition Table for its guests
- Hardware can only know about one partition table
  - Could switch it
    - Flush caches

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2	2. EA-GRA-HRA

Nested Address Translation

- L0 has a Partition Table for its guests
- L1 has a Partition Table for its guests
- Hardware only knows about one partition table
  - Could switch it
    - Flush caches
  - Each partition table only does a single level of translation
    - L2 GRA -> L1 GRA

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2	2. EA-GRA-HRA

Nested Address Translation

- L0 has a Partition Table for its guests
- L1 has a Partition Table for its guests
- Hardware only knows about one partition table
  - Could switch it
    - Flush caches
  - Each partition table only does a single level of translation
    - L2 GRA -> L1 GRA
    - L1 GRA -> L0 HRA
    - Hardware needs L2 GRA -> L0 HRA

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1.

L1 → L2

2.

EA-GRA-HRA

Nested Address Translation

- L0 allocates a “shadow LPID” for the nested guest
  - e.g. LPID = 8
- Create an entry in the L0 partition table
  - Will contain mappings for the Nested (L2) Guest

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1.	2.
L1 → L2	EA-GRA-HRA

Process Scoped Nested Translation

- L2 process table is in L2 memory
    - Managed by L2

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

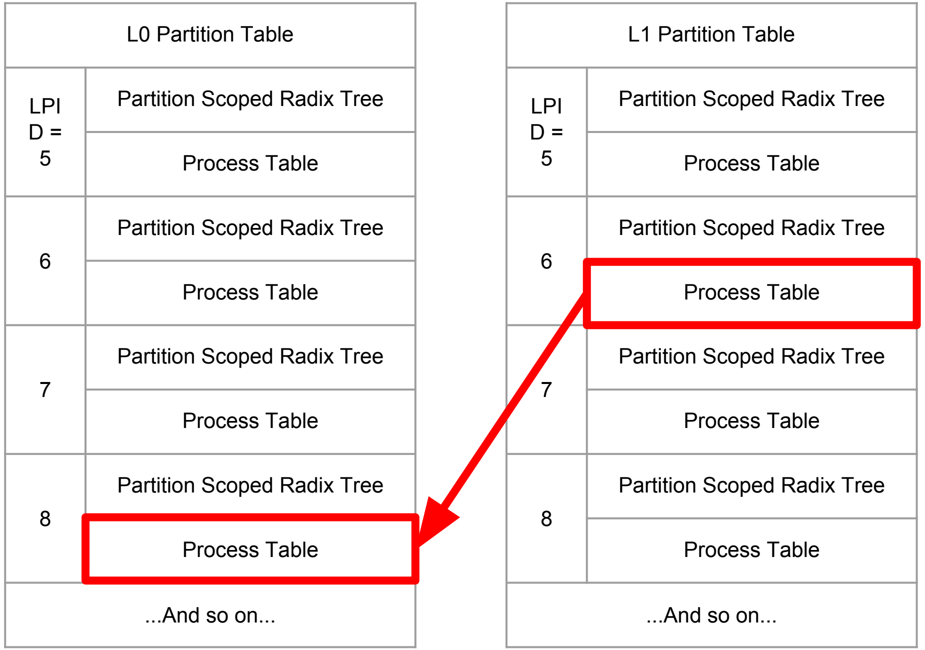
L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2

2. EA-GRA-HRA

Process Scoped Nested Translation

- L2 process table is in L2 memory
    - Managed by L2
- L0 can copy the process table from the L1 partition table into its entry for the "shadow LPID" allocated for the L2 guest
- Hardware can find the process table
    - L2 EA -> L2 GRA translation



L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2

2. EA-GRA-HRA

Partition Scoped Nested Translation

- What about Partition Scoped Translation?
  - Have a L2 GRA from process scoped
  - Need a hardware accessible mapping for L2 GRA -> L0 HRA translation
  - Hardware needs a single radix tree
    - Can’t just walk the two in the two partition tables
    - But software can
    - So let’s see what happens when we handle a page fault

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

Partition Scoped Nested Translation

L2 Guest Real Address

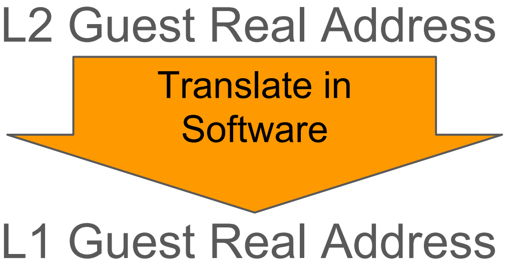
- L2 GRA -> L1 GRA
- Mapping in L1 Partition Table

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2	2. EA-GRA-HRA

Partition Scoped Nested Translation


L2 Guest Real Address

Translate in Software

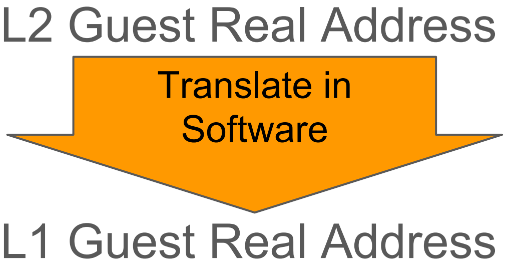
L1 Guest Real Address

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2	2. EA-GRA-HRA

Partition Scoped Nested Translation



L2 Guest Real Address

Translate in Software

L1 Guest Real Address

- No PTE?
  - Synthesise interrupt to the L1 OS
  - L1 OS will fault in an entry
  - Can retry next time

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2	2. EA-GRA-HRA

Partition Scoped Nested Translation

L2 Guest Real Address

Translate in Software

L1 Guest Real Address

- L1 GRA -> L0 HRA
- Mapping in L0 Partition Table

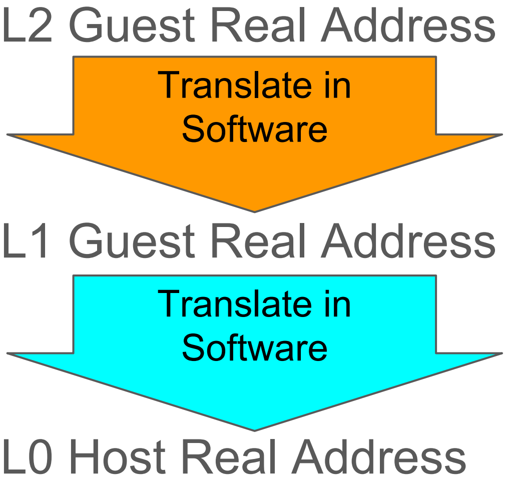
L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2

2. EA-GRA-HRA

Partition Scoped Nested Translation



L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

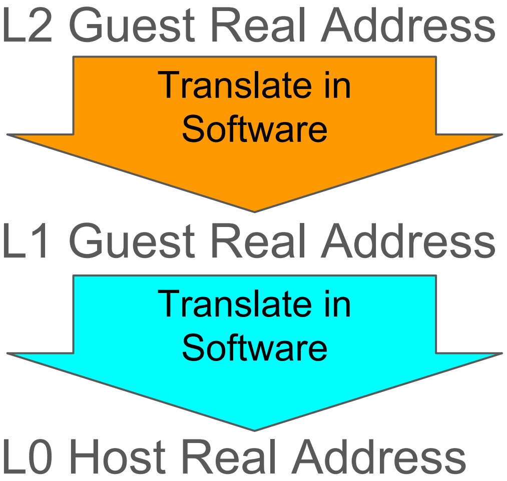
1.

L1 → L2

2.

EA-GRA-HRA

Partition Scoped Nested Translation



- No PTE?
  - Fault in an entry

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

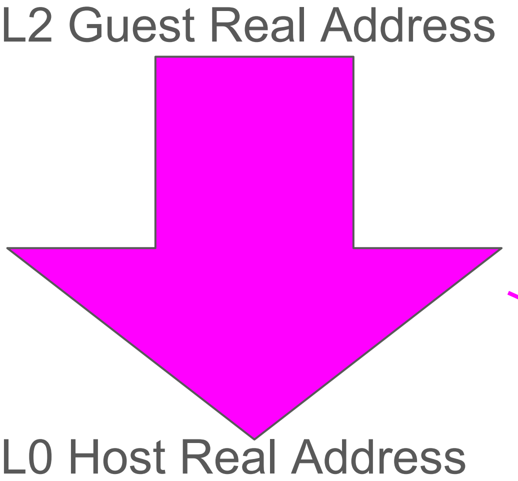
1.

L1 → L2

2.

EA-GRA-HRA

Partition Scoped Nested Translation



L2 Guest Real Address

L0 Host Real Address

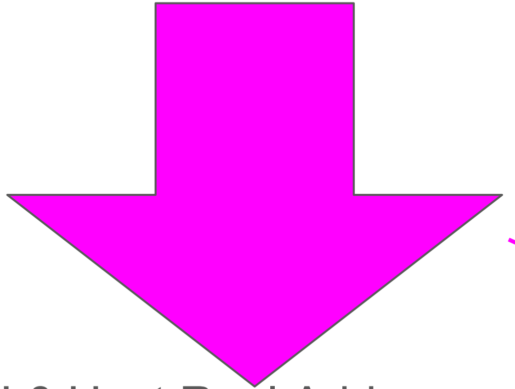
L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1.	2.
L1 → L2	EA-GRA-HRA

Partition Scoped Nested Translation

L2 Guest Real Address



L0 Host Real Address

- Shadow Page Table for Nested Guest
    - Combination of the two levels of partition scoped translation
    - Hardware can access this mapping

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1.

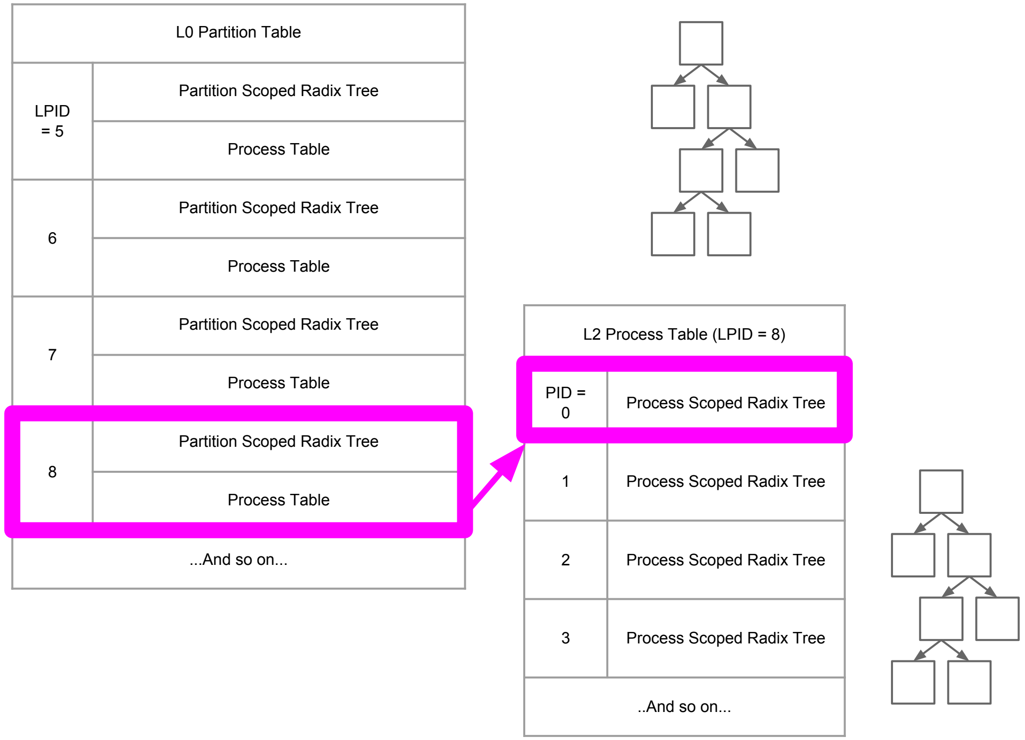
L1 → L2

2.

EA-GRA-HRA

Nested Address Translation

- What does the hardware end up doing



L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L2 Process Table (LPID = 8)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

1.

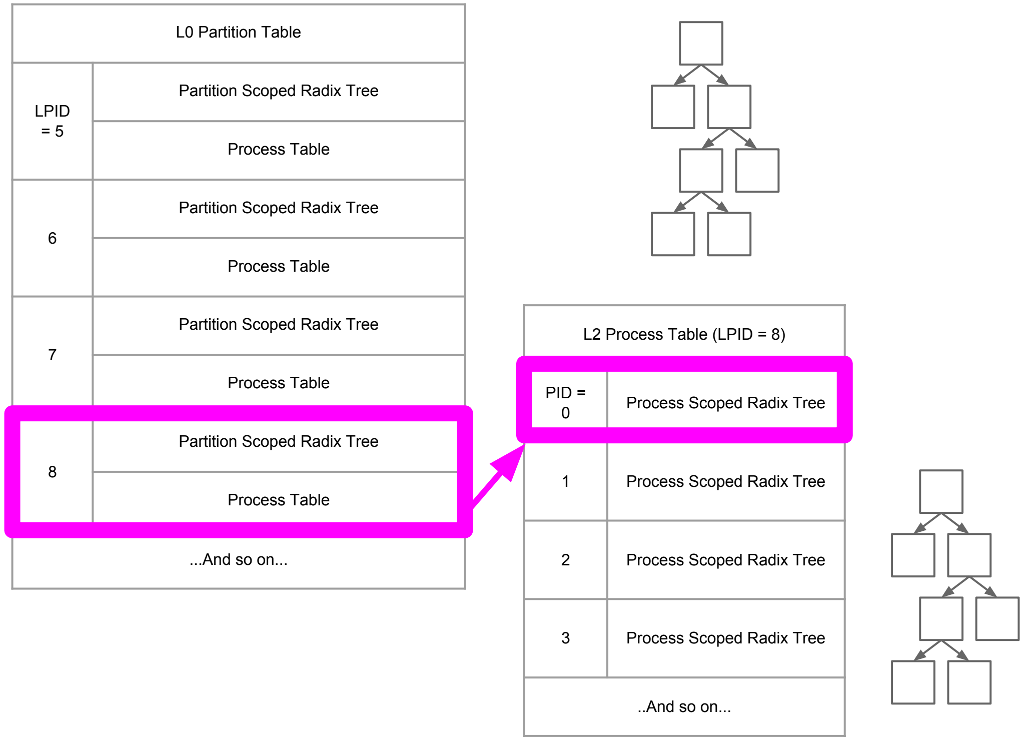
L1 → L2

2.

EA-GRA-HRA

Nested Address Translation

L2 Guest Effective Address (EA)

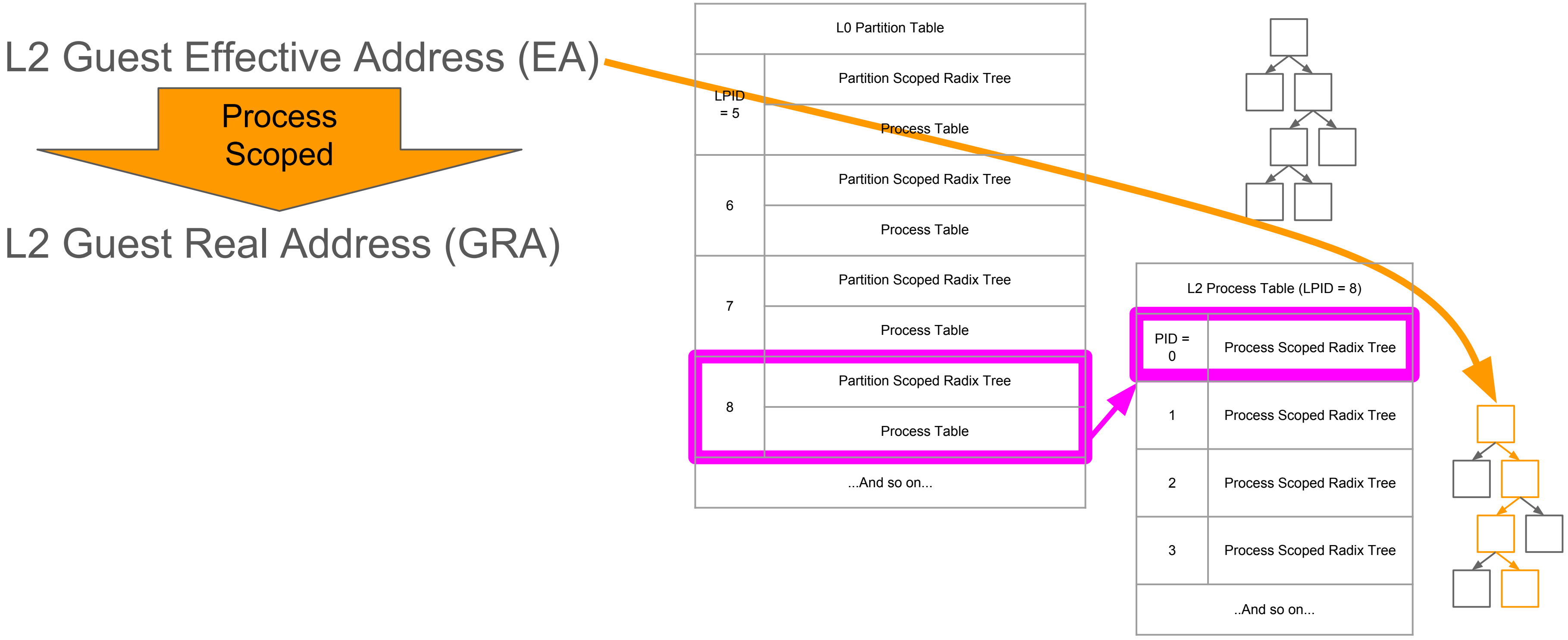


L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L2 Process Table (LPID = 8)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

1. L1 → L2	2. EA-GRA-HRA

Nested Address Translation



L2 Guest Effective Address (EA)

L2 Guest Real Address (GRA)

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

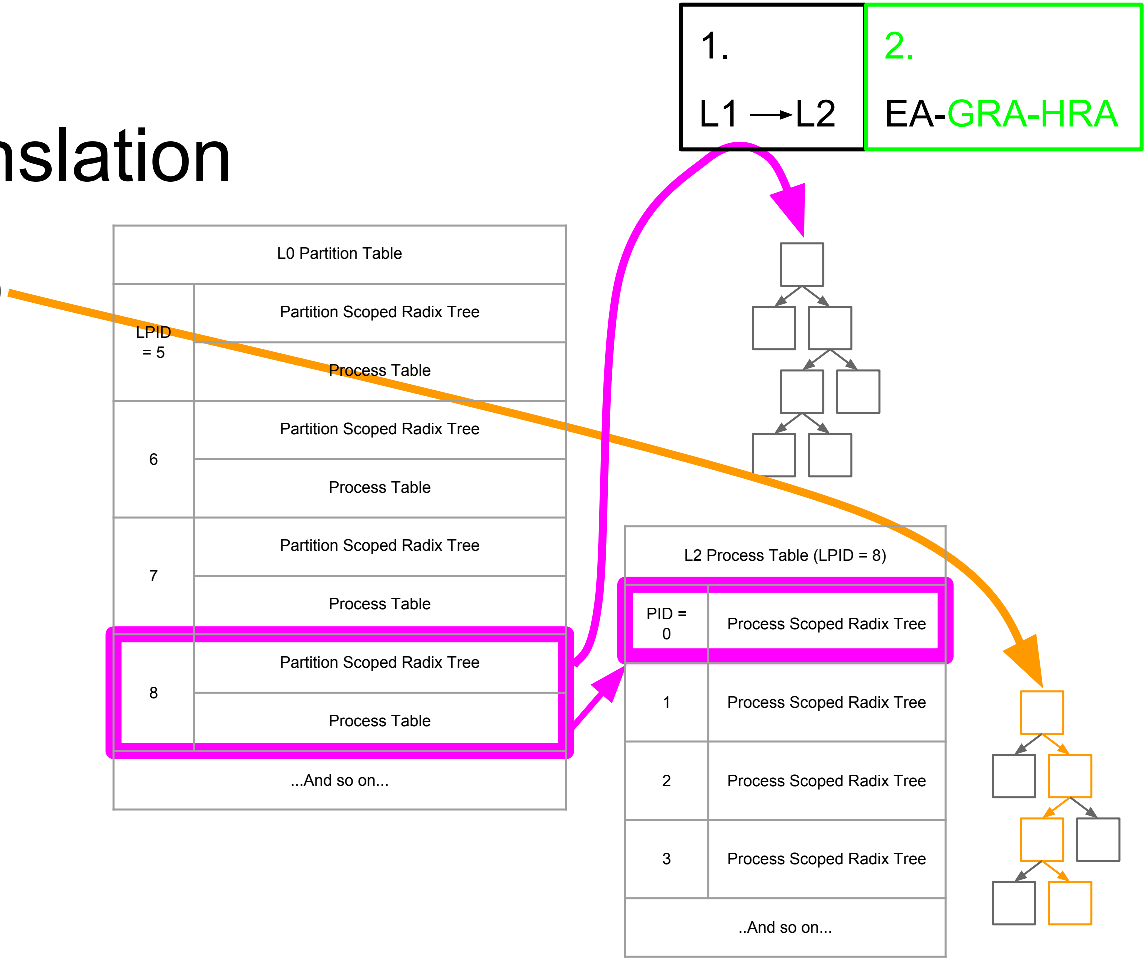
L2 Process Table (LPID = 8)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

Nested Address Translation

L2 Guest Effective Address (EA)

Process Scoped

L2 Guest Real Address (GRA)



1\.

L1 → L2

2\.

EA-GRA-HRA

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L2 Process Table (LPID = 8)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

Nested Address Translation

L2 Guest Effective Address (EA)

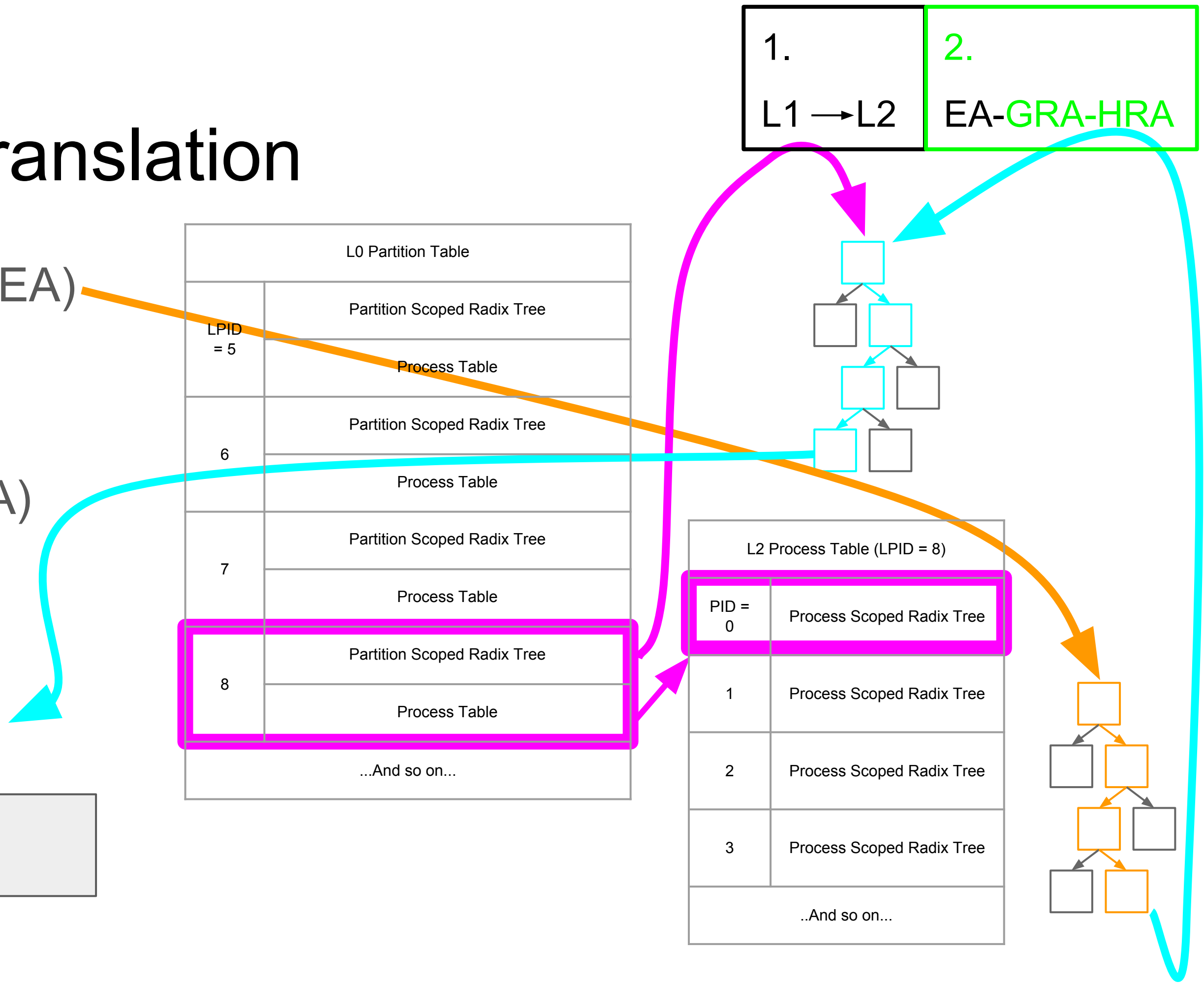
Process Scoped

L2 Guest Real Address (GRA)

Partition Scoped

L0 Host Real Address (HRA)

Hardware Address



L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L2 Process Table (LPID = 8)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	

1.

L1 → L2

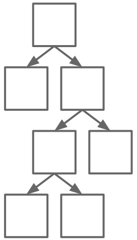
2.

EA-GRA-HRA

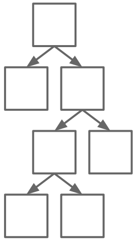
Nested Address Translation

- To the hardware all guests are the same
  - Process Table in guest memory
    - Associated with PID
    - EA -> GRA Mapping
  - Partition Scoped Page Table in L0 Host Memory
    - Associated with LPID
    - GRA -> HRA Mapping
- L0 Shadow Page Table just the collapse of all Partition Scoped Page Tables below it
  - Each level manages its own mappings

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	



L2 Process Table (LPID = 8)	
PID = 0	Process Scoped Radix Tree
1	Process Scoped Radix Tree
2	Process Scoped Radix Tree
3	Process Scoped Radix Tree
..And so on...	



Breath

1. L1 → L2	2. EA-GRA-HRA

Nested Address Translation Invalidation

- We can insert nested address translations
- But how do we invalidate them?
  - L1 invalidates a page it mapped through to L2
  - L0 invalidates a page it mapped through to L1

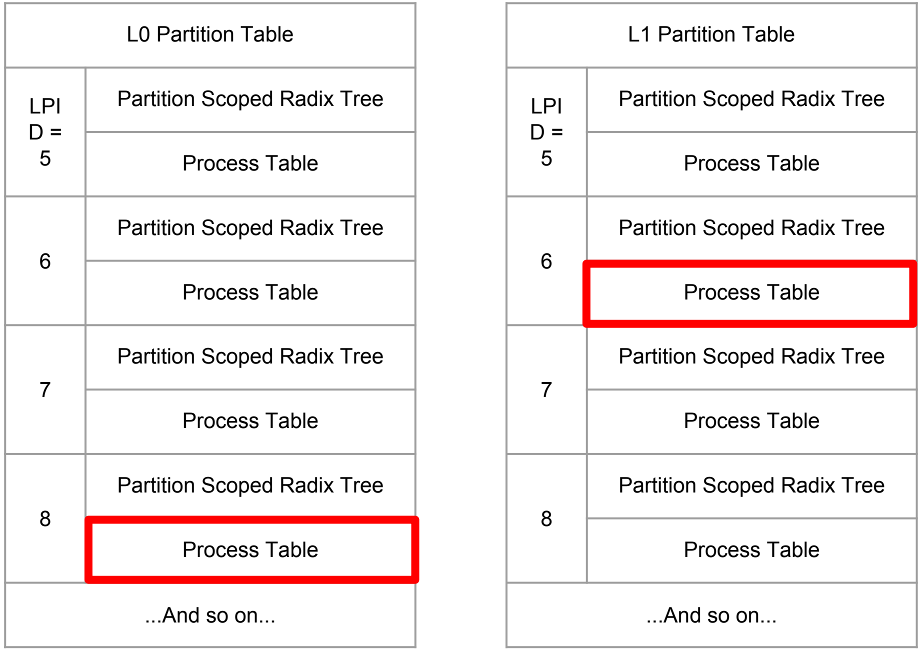
L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2	2. EA-GRA-HRA

Process Scoped Invalidation

- L2 invalidating the L2 EA -> L2 GRA process scoped translation



L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1.

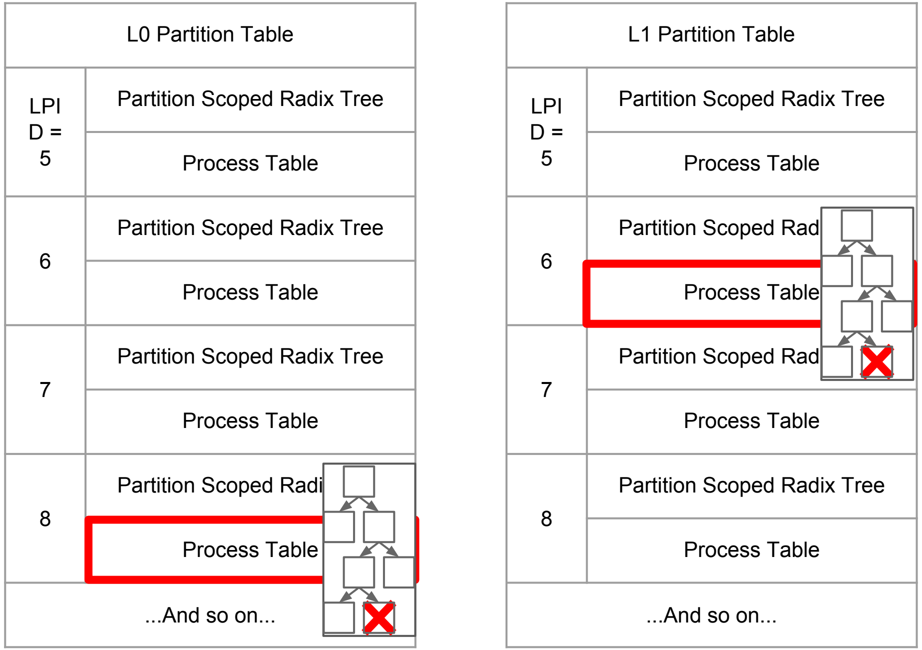
L1 → L2

2.

EA-GRA-HRA

Process Scoped Invalidation

- L2 invalidating the L2 EA -> L2 GRA process scoped translation
  - Process table is in L2 memory
    - L2 can invalidate ptes
  - L2 runs in supervisor mode
    - Able to use supervisor instructions to invalidate the caching of these
- No hypervisor assistance required



L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radi
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Rad
	Process Table
7	Partition Scoped Rad
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1.

L1 → L2

2.

EA-GRA-HRA

Partition Scoped Invalidation

- Invalidating entries in the Shadow Page Table for the Nested Guest

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

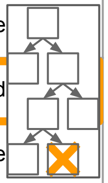
1.	2.
L1 → L2	EA-GRA-HRA

Partition Scoped Invalidation

- L1 invalidates a page it mapped through to L2
    - Invalidation of partition scoped mappings requires HV privileged instructions
    - Guest hypervisor uses an H-CALL
        - Provides L2 GRA

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Rad
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	



1.

L1 → L2

2.

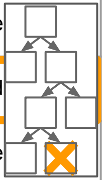
EA-GRA-HRA

Partition Scoped Invalidation

- L1 invalidates a page it mapped through to L2
  - Invalidation of partition scoped mappings requires HV privileged instructions
  - Guest hypervisor uses an H-CALL
    - Provides L2 GRA
- Can walk our shadow page table for the nested guest - keyed on L2 GRA

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

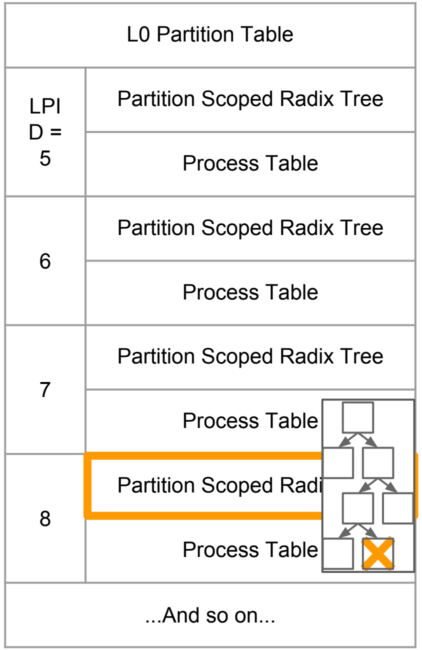
L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Rad
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	



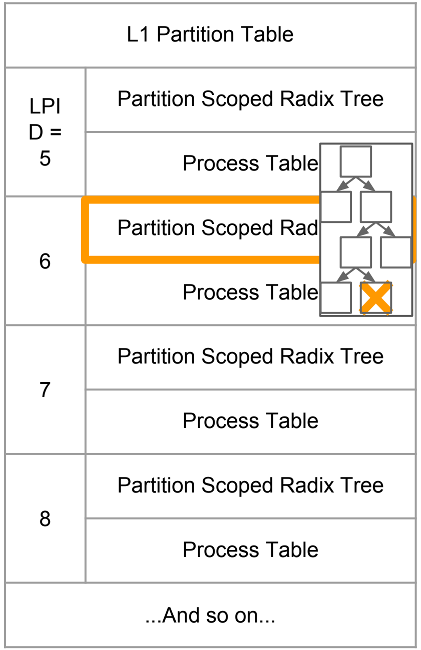
1.	2.
L1 → L2	EA-GRA-HRA

Partition Scoped Invalidation

- L1 invalidates a page it mapped through to L2
  - Invalidation of partition scoped mappings requires HV privileged instructions
  - Guest hypervisor uses an H-CALL
    - Provides L2 GRA
- Can walk our shadow page table for the nested guest - keyed on L2 GRA
  - Invalidate PTE if any



L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radi
	Process Table
...And so on...	



L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Rad
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1.

L1 → L2

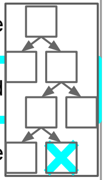
2.

EA-GRA-HRA

Partition Scoped Invalidation

- L0 invalidates a page it mapped through to L1
  - The page might also have been mapped through to L2

L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Rad
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

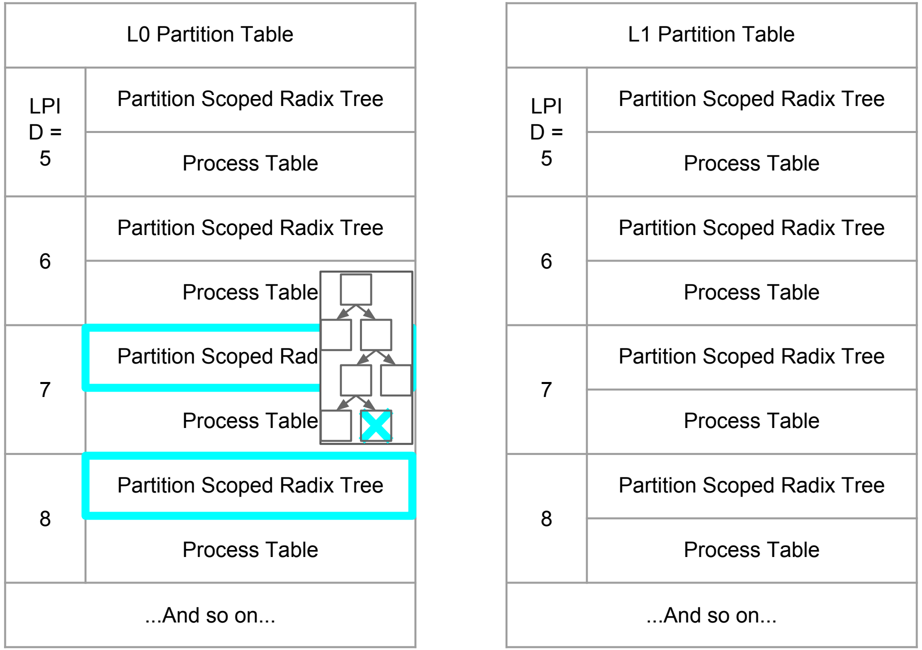


L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1.	2.
L1 → L2	EA-GRA-HRA

Partition Scoped Invalidation

- L0 invalidates a page it mapped through to L1
  - The page might also have been mapped through to L2
  - KVM code provides L1 GRA here
- How do we find the corresponding entry in the shadow page table for the nested guest
  - This translation in the shadow page table is keyed on L2 GRA
  - Only have L1 GRA



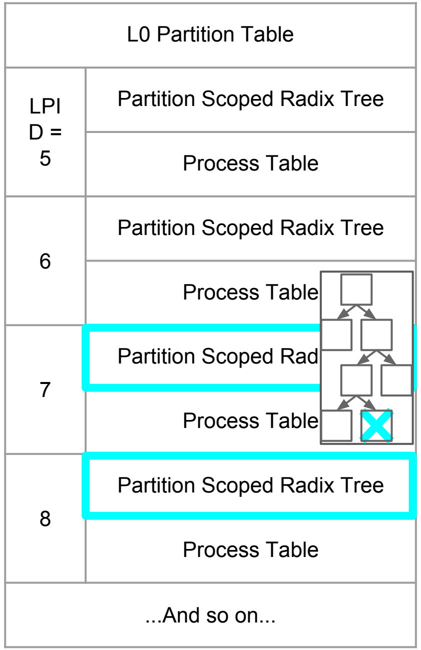
L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Rad
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1.	2.
L1 → L2	EA-GRA-HRA

Partition Scoped Invalidation

- L0 invalidates a page it mapped through to L1
  - The page might also have been mapped through to L2
  - KVM code provides L1 GRA here
- How do we find the corresponding entry in the shadow page table for the nested guest
  - Keep an rmap (reverse mapping) which stores the L1 GRA -> L2 GRA mapping whenever an entry in the shadow page table is created



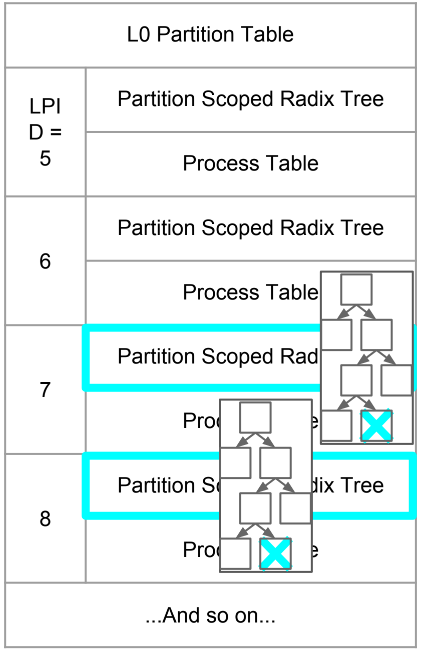
L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Rad
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2	2. EA-GRA-HRA

Partition Scoped Invalidation

- L0 invalidates a page it mapped through to L1
  - The page might also have been mapped through to L2
  - KVM code provides L1 GRA here
- How do we find the corresponding entry in the shadow page table for the nested guest
  - Keep an rmap (reverse mapping) which stores the L1 GRA -> L2 GRA mapping whenever an entry in the shadow page table is created
  - Use the L2 GRA to find and invalidate any valid ptes



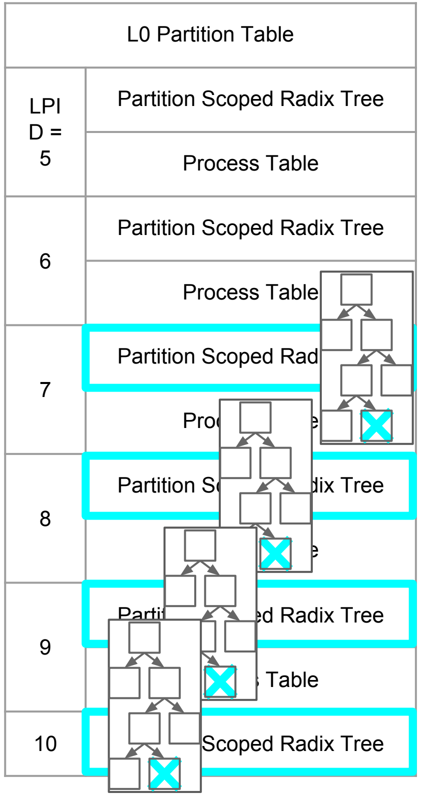
L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Rad
	Proc e
8	Partition S dix Tree
	Proc e
...And so on...	

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1.	2.
L1 → L2	EA-GRA-HRA

Partition Scoped Invalidation

- L0 invalidates a page it mapped through to L1
  - A single L1 page may have been mapped to multiple L2 guests
    - To accommodate this the rmap is a list
    - Traverse the list and invalidate all ptes in shadow pages tables for all nested guests of the same L1 with a matching pte



L0 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Rad
	Proc e
8	Partition Sc dix Tree
	e
9	Parti ed Radix Tree
	Table
10	Scoped Radix Tree

L1 Partition Table	
LPID = 5	Partition Scoped Radix Tree
	Process Table
6	Partition Scoped Radix Tree
	Process Table
7	Partition Scoped Radix Tree
	Process Table
8	Partition Scoped Radix Tree
	Process Table
...And so on...	

1. L1 → L2	2. EA-GRA-HRA

So how do we make this happen?

- Two things needed to run a nested KVM-HV guest

1.	2.
L1 → L2	EA-GRA-HRA

So how do we make this happen?

- Two things needed to run a nested KVM-HV guest
- L1 -> L2 Guest Entry

1. L1 → L2 ✓	2. EA-GRA-HRA

So how do we make this happen?

- Two things needed to run a nested KVM-HV guest
- L1 -> L2 Guest Entry
    - H-CALL H_ENTER_NESTED

1. L1 → L2 ✓	2. EA-GRA-HRA

So how do we make this happen?

- Two things needed to run a nested KVM-HV guest
- L1 -> L2 Guest Entry
  - H-CALL H_ENTER_NESTED
- L2 Guest Address Translation

So how do we make this happen?

1. L1 → L2 ✓	2. EA-GRA-HRA ✓

- Two things needed to run a nested KVM-HV guest
- L1 -> L2 Guest Entry
  - H-CALL H_ENTER_NESTED
- L2 Guest Address Translation
  - Shadow Page Table
  - rmap for invalidations

1. ✓ L1 → L2	2. ✓ EA GRA-HRA

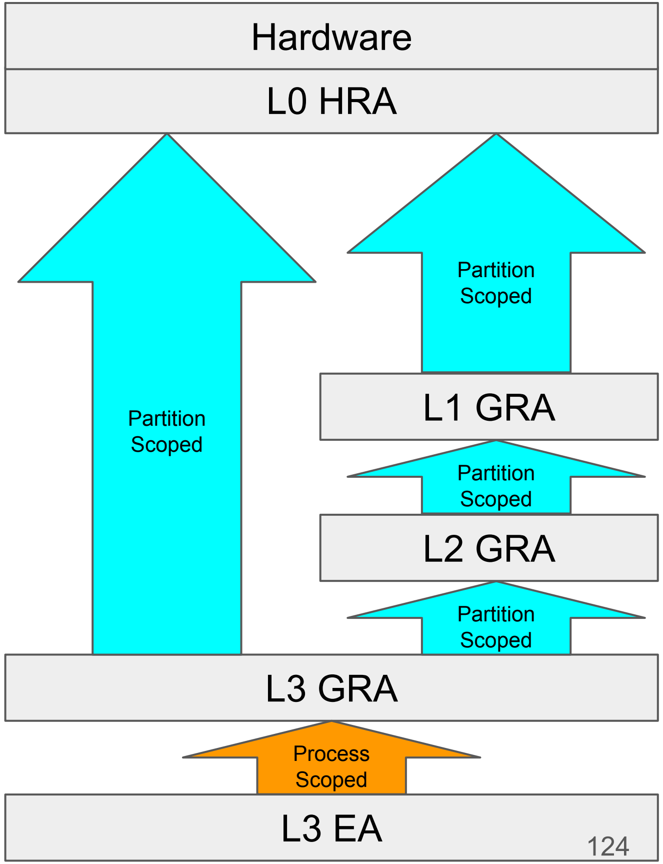
So how do we make this happen?

- Two things needed to run a nested KVM-HV guest
- L1 -> L2 Guest Entry
  - H-CALL H_ENTER_NESTED
- L2 Guest Address Translation
  - Shadow Page Table
  - rmap for invalidations

Breath

Interesting Features

- Nested Nested
  - There is no reason L2 can't run it's own L3 nested guest
  - L1 manages a shadow page table for L3
    - Just as L0 did for L2
  - L0 sees L3 as just another guest of L1
  - L0 manages its own shadow page table for L3
    - Just as it did for L2
  - L0 doesn't know whether L3 is a guest of L2 or just another guest of L1

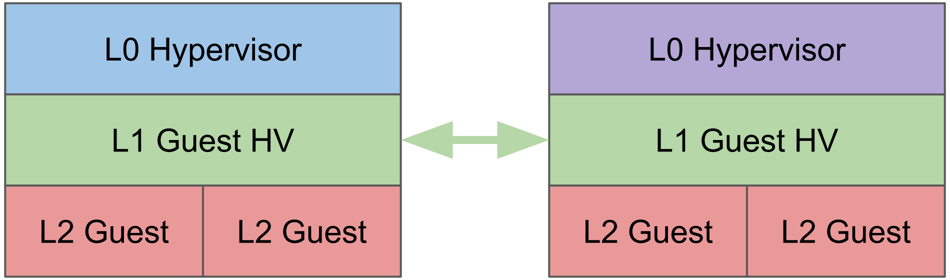


Interesting Features

- Theoretically possible to nest indefinitely
  - Given enough memory
  - …and time
  - ...and with some caveats

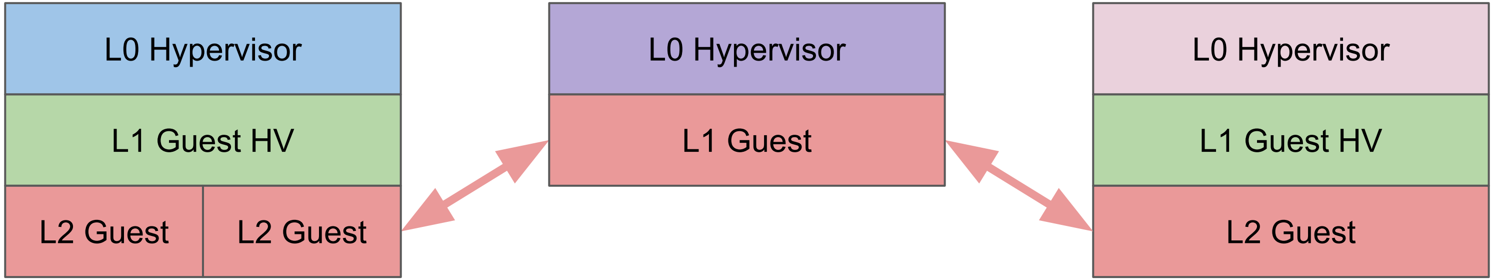
Interesting Features

- Migration of Nested Guests
  - Possible to migrate a L1 guest and all the nested guests below it
  - The state and memory of all the nested guests is stored in L1 memory
    - Already migrated as part of the migration stream
  - All of the state stored in L0 can be generated/allocated again on the receiving side
    - Except the location of the L1 partition table in L1 memory



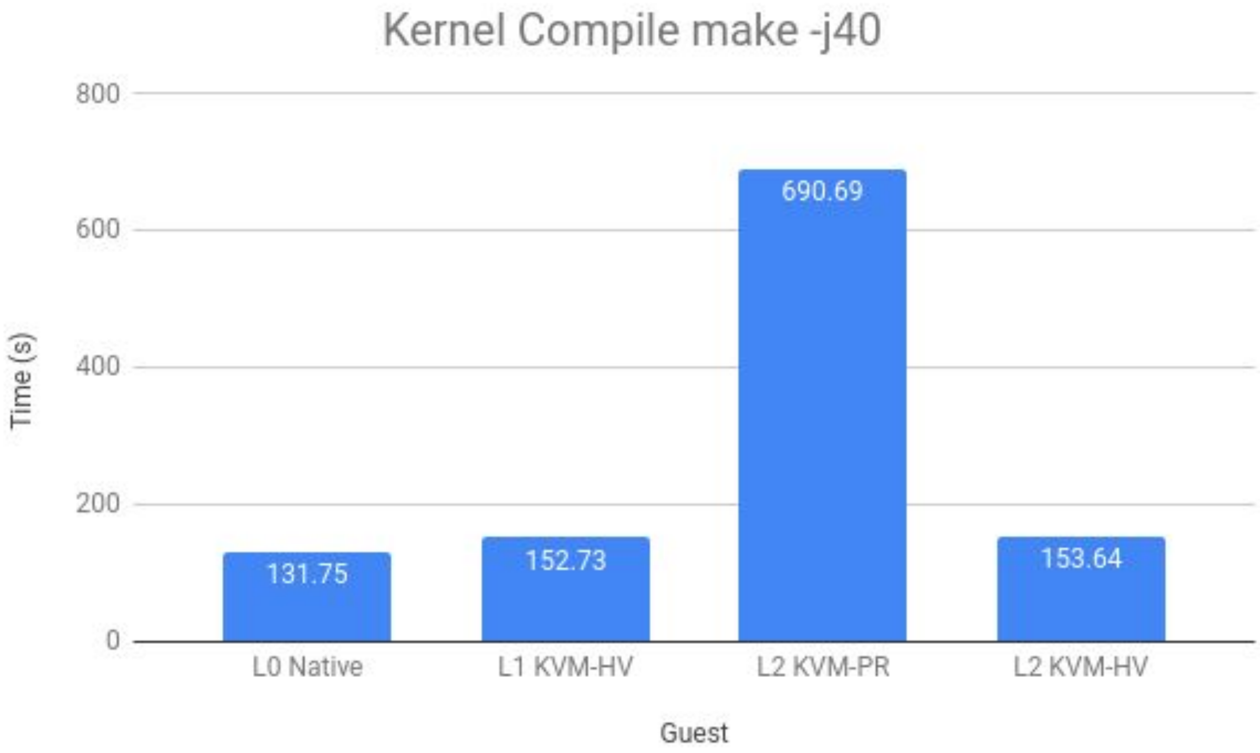
Interesting Features

- Migration Between Levels
  - All pseries guests are technically the same
  - Possible to migrate a L2 guest to become a L1 guest
  - Possible to migrate a L1 guest to become a L2 guest
  - Assuming a transport between L0 and L1



Performance

- Kernel Compile
  - 40 Threads
  - 20G Memory
  - pseries_le_defconfig
  - make -j40 -s
  - Hot run to ensure page tables populated
- Total Time Elapsed



How Many Levels Can You Nest?

- Ran a level 11 guest last week
- Significant slow down booting level 12
  - Due to the bouncing around of H-Calls

State of the Code

- KVM/Kernel
    - Patches in the kvm-next tree
    - Hopefully in 4.20
- QEMU
    - Patches posted to the list
    - Hopefully in 3.1 once the cap number in upstream

How to Use It?

- KVM/Kernel L0
  - echo Y > /sys/modules/kvm_hv/parameters/nested
- QEMU L0
  - qemu-system-ppc64 -machine pseries,cap-nested-hv=true
- KVM/Kernel L1
  - Requires the patch series to implement nested kvm
  - No other specific steps
- QEMU L1
  - Nothing special required
- Kernel L2
  - Nothing special required

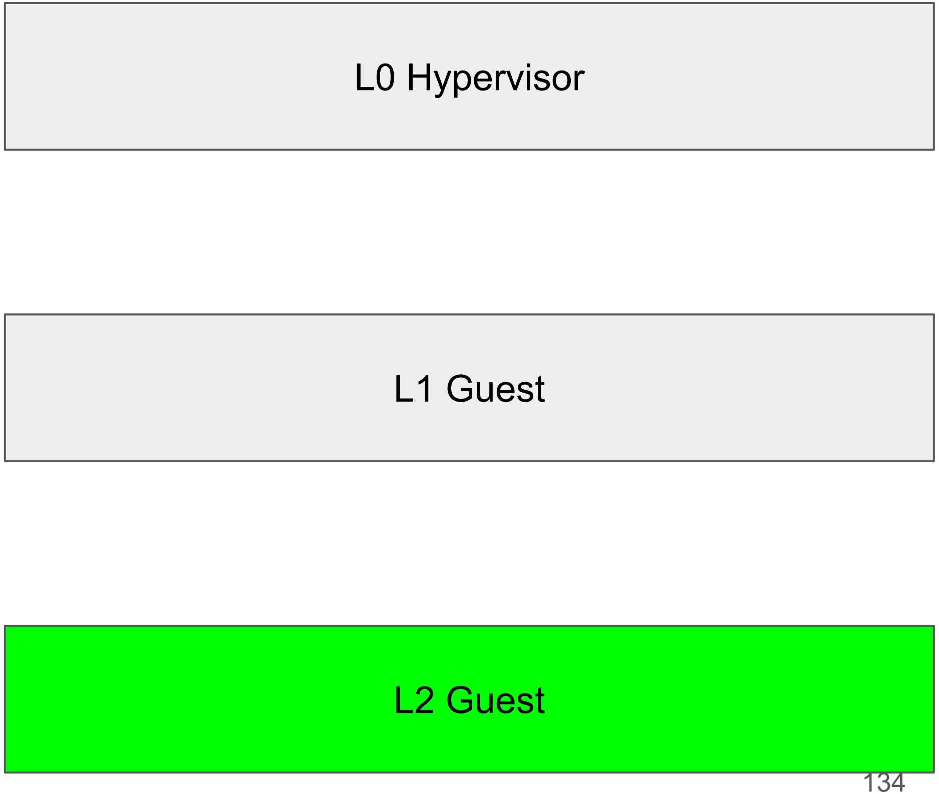
Now you can run your own nested KVM-HV guests

- Thank you for listening

Questions?

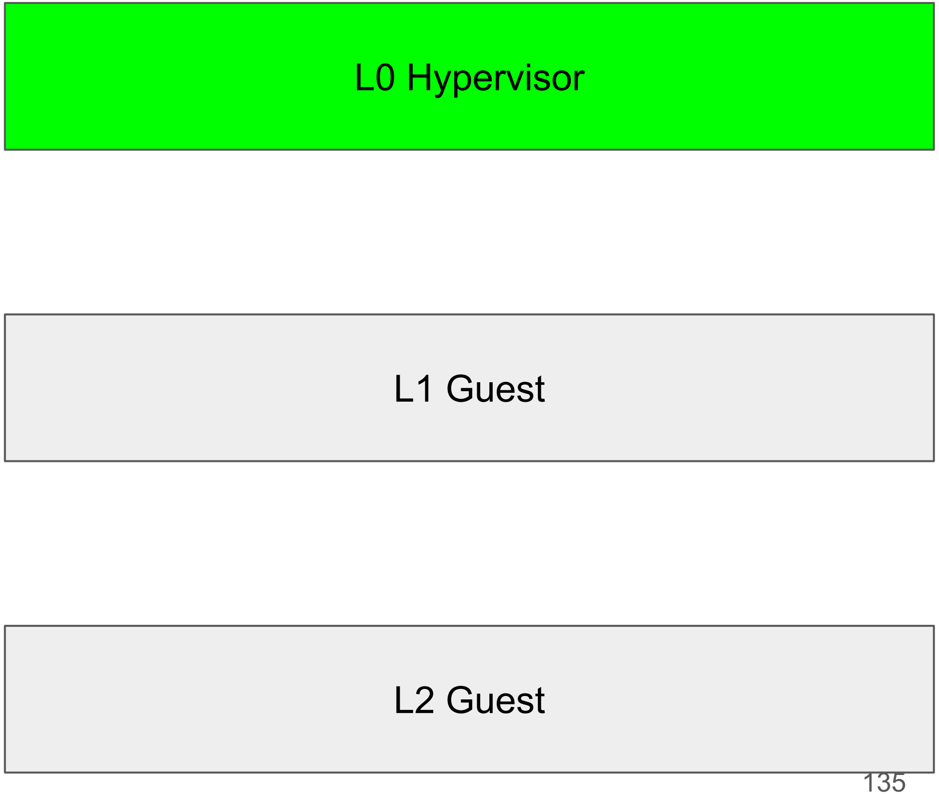
A Quick Word on Interrupts

- L2 Runs in Supervisor Mode
  - OS Interrupts delivered directly to the L2 OS
    - OS Level Page Faults
    - Decrementer
    - System Call
    - etc.



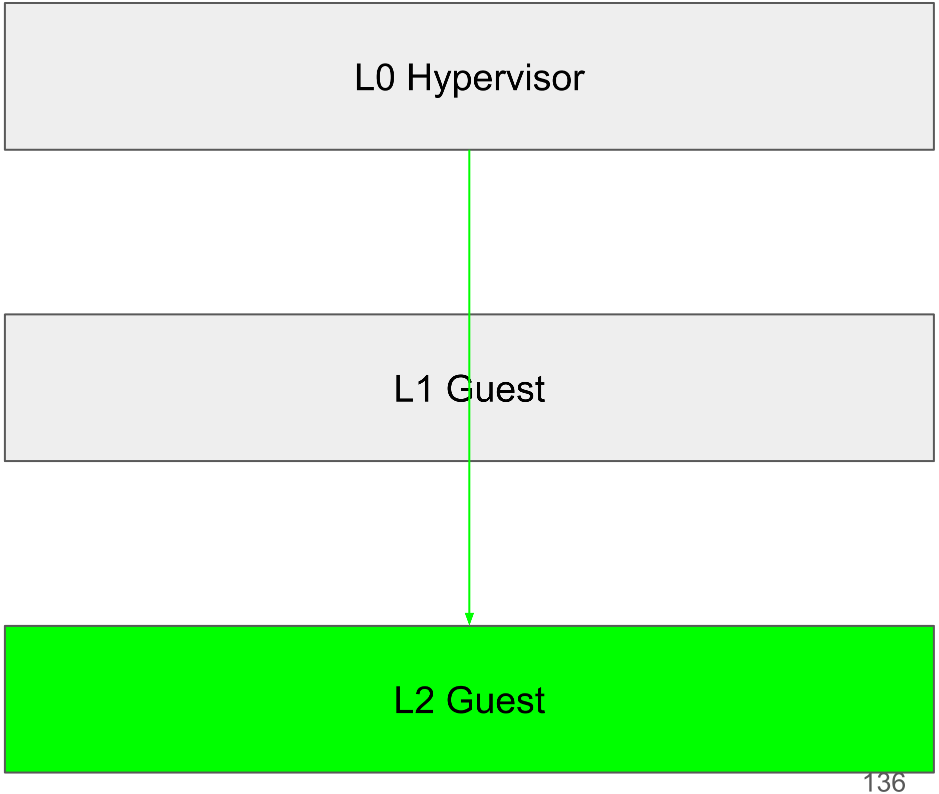
A Quick Word on Interrupts

- L2 Runs in Supervisor Mode
  - OS Interrupts delivered directly to the L2 OS
- HV Interrupts delivered to L0
  - Hypervisor Page Fault
  - Hypervisor Decrementer
  - Hypervisor Doorbell
  - H-CALL (Hypervisor System Call)
  - etc.



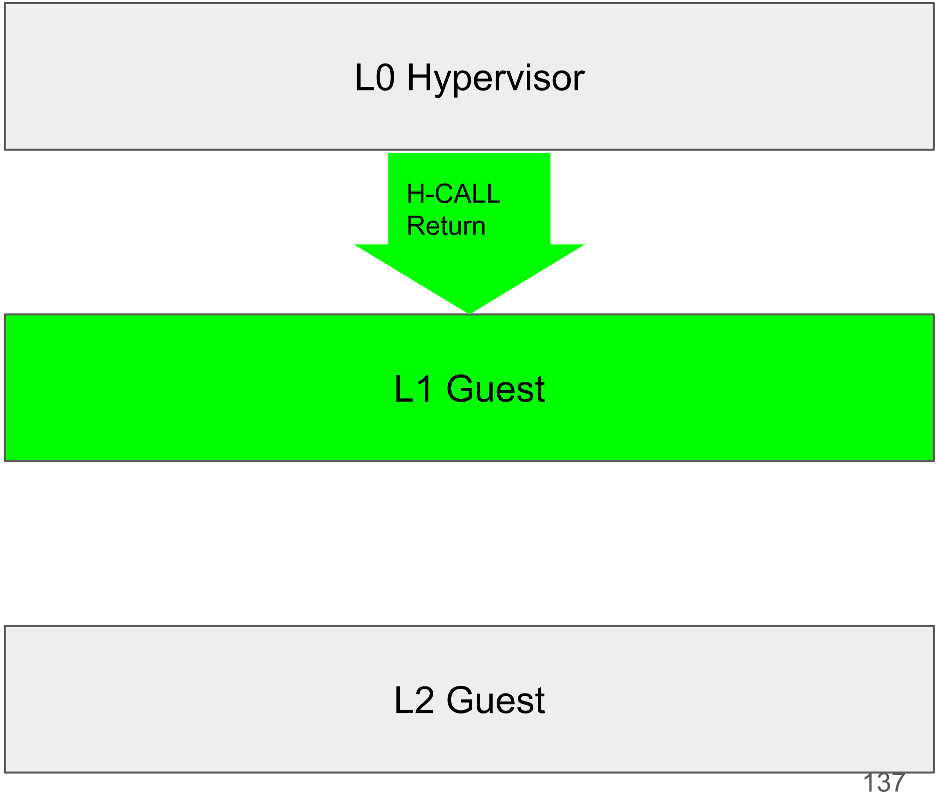
A Quick Word on Interrupts

- L2 Runs in Supervisor Mode
  - OS Interrupts delivered directly to the L2 OS
- HV Interrupts delivered to L0
  - Hypervisor Page Fault
  - Hypervisor Decrementer
  - Hypervisor Doorbell
  - H-CALL (Hypervisor System Call)
  - etc.
- If handled return directly to L2



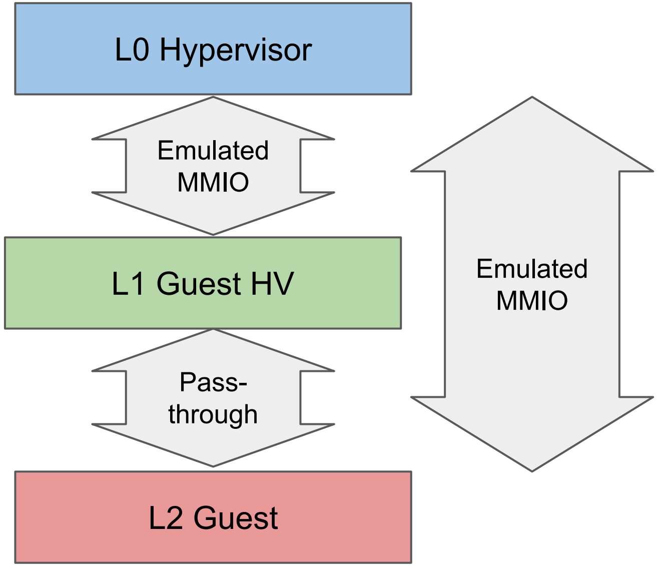
A Quick Word on Interrupts

- L2 Runs in Supervisor Mode
  - OS Interrupts delivered directly to the L2 OS
- HV Interrupts delivered to L0
  - Hypervisor Page Fault
  - Hypervisor Decrementer
  - Hypervisor Doorbell
  - H-CALL (Hypervisor System Call)
  - etc.
- When required HV interrupts delivered to L1
  - As part of return from H-CALL



Interesting Features

- Emulated MMIO Passthrough
  - L0 emulates a device for L1
  - L1 sees it as a real device and passes it through to L2
  - L0 emulates L2 accesses



Limitations

- The L0 hypervisor, all nested hypervisors and all nested guests must use radix translation
- If the host is scheduling on a per core level then only one nested vcpu can run at a time on a core, the secondary threads will be idle
- A nested hypervisor can’t use a smaller page size than that of the hypervisors in the levels above it
- There can only be 1023 guests on a system as a whole, irrespective of at which level they run
    - Since the L0 hypervisor must allocate a real LPID for each