Nested EPT to Make Nested VMX Faster

Red Hat

Author Gleb Natapov

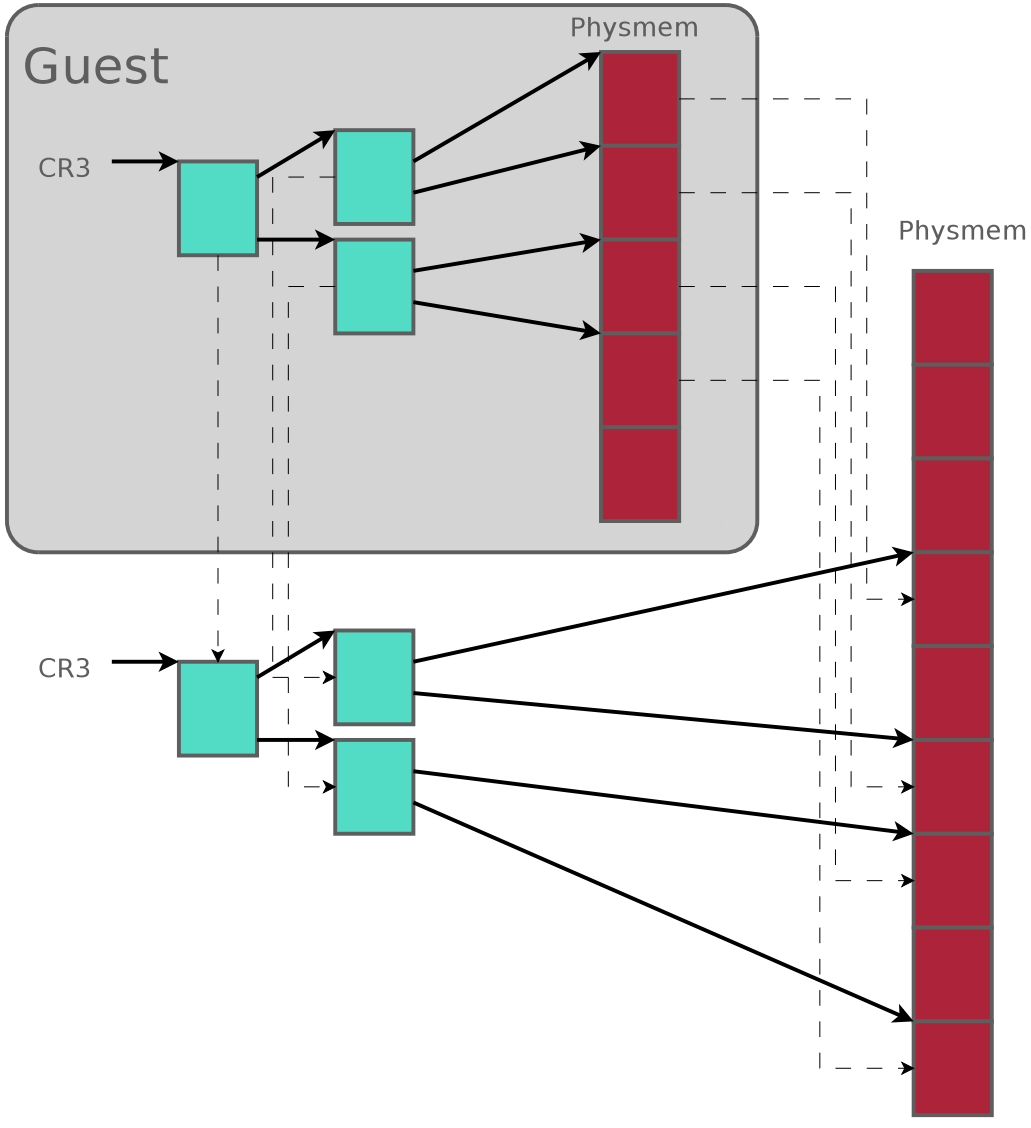
October 21, 2013

Section 1

Background

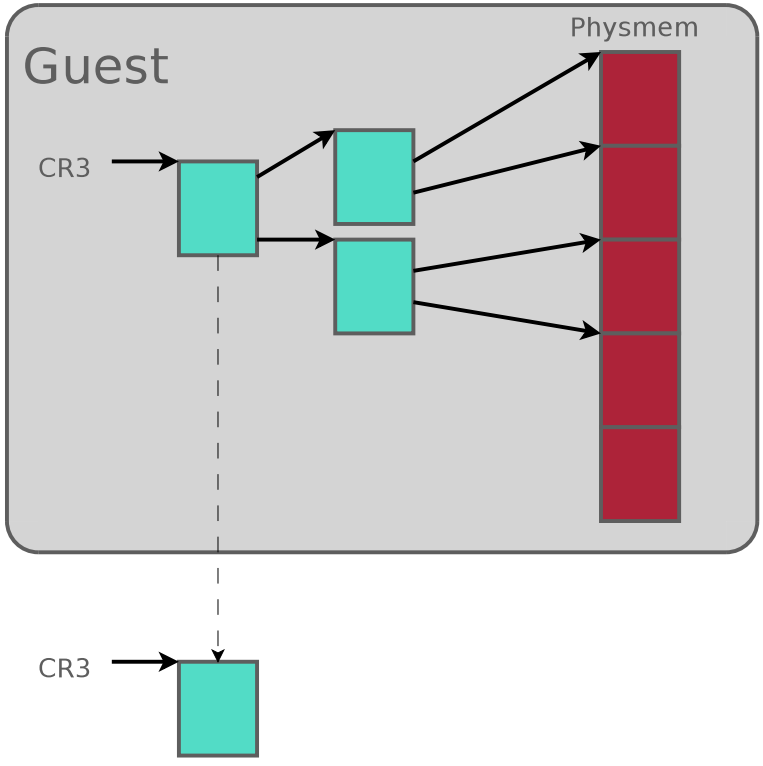
redhat.

Shadow Paging



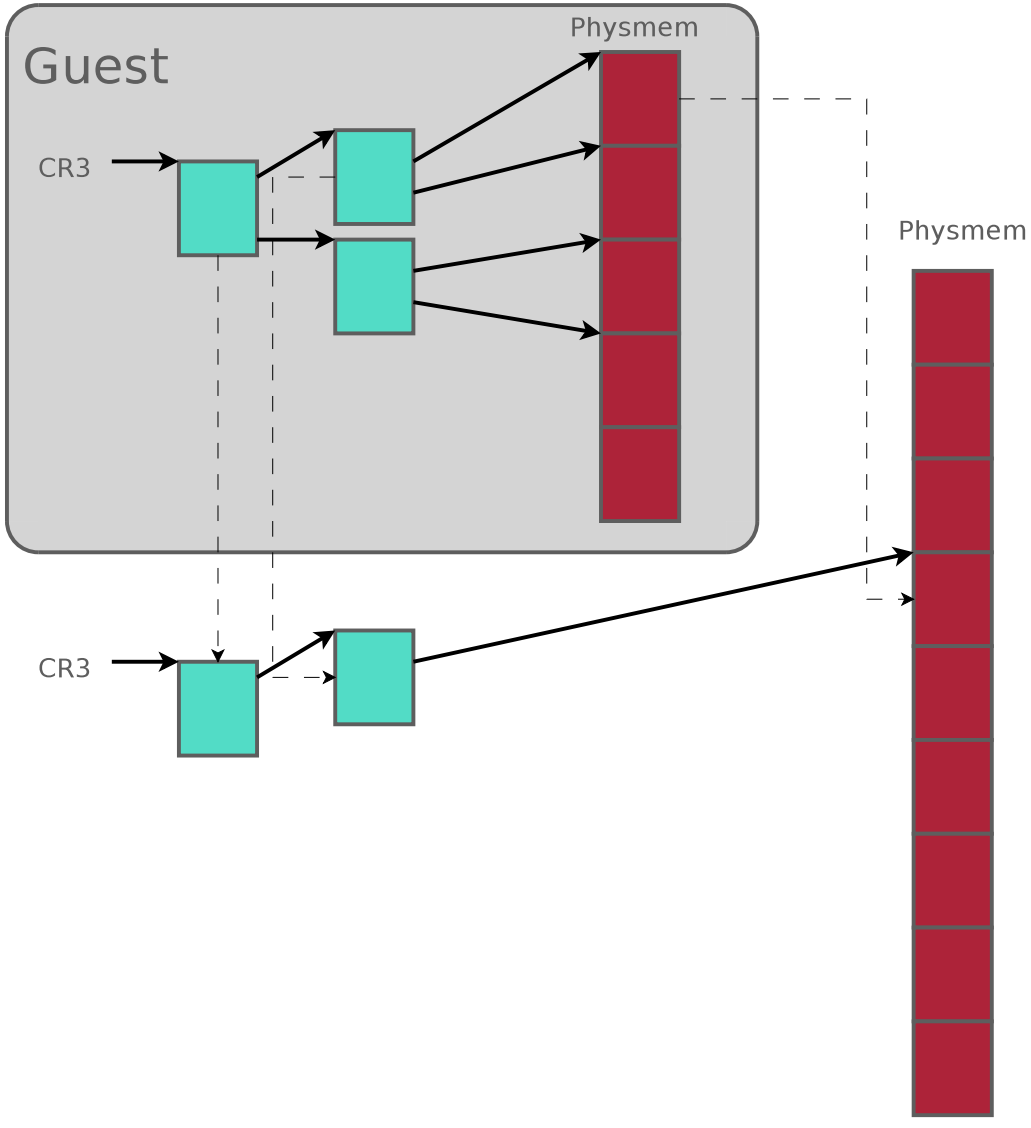
redhat.

Shadow Paging



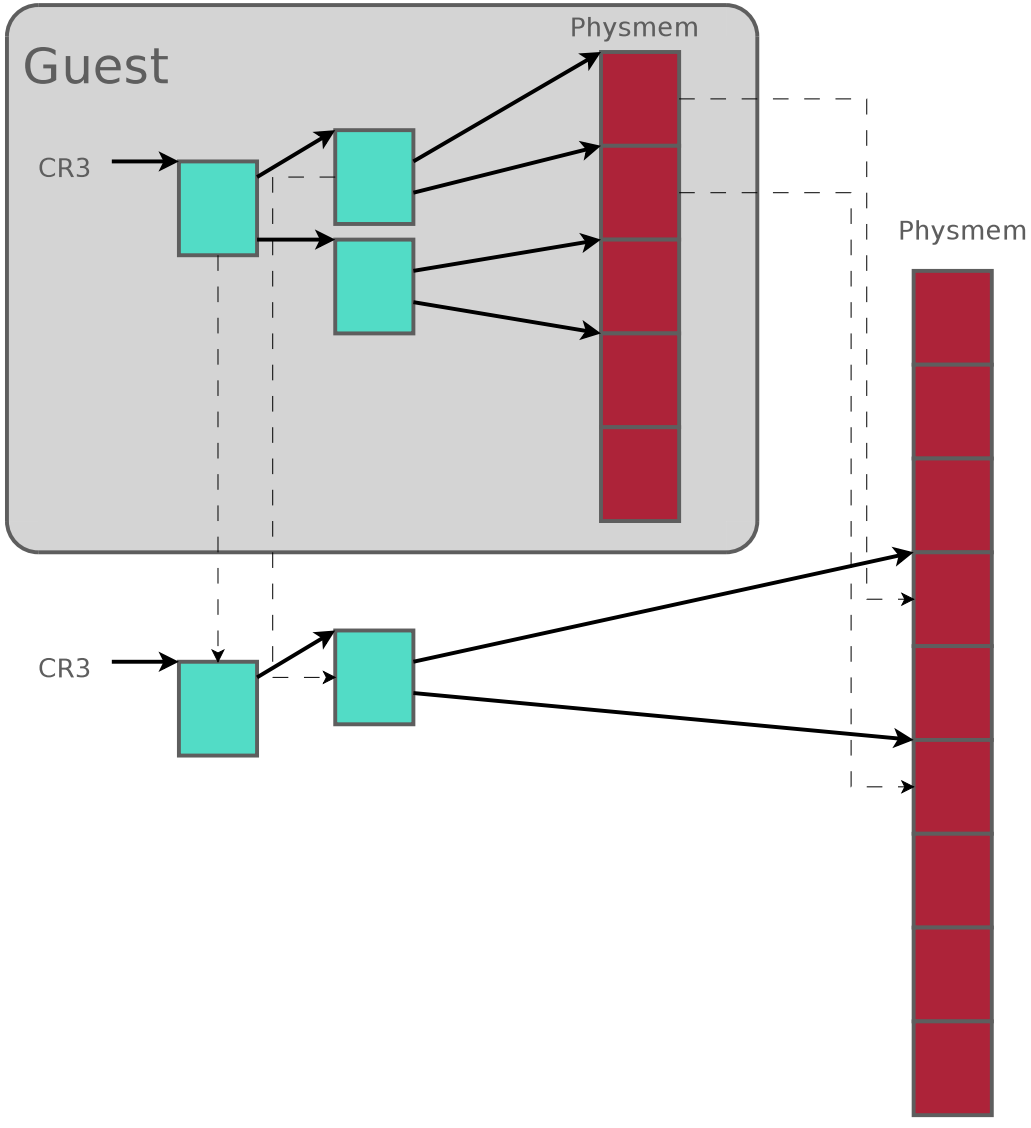
redhat.

Shadow Paging



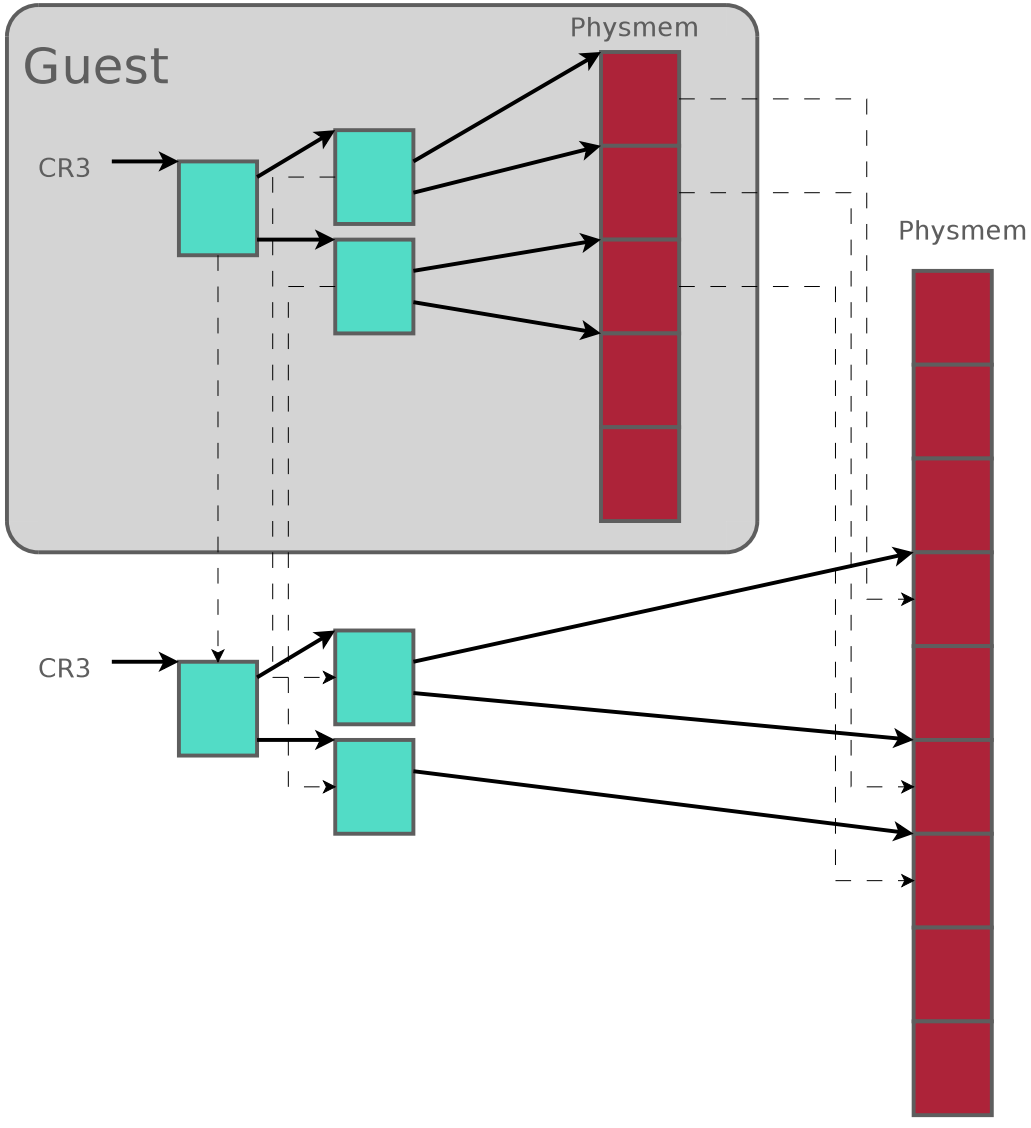
redhat.

Shadow Paging



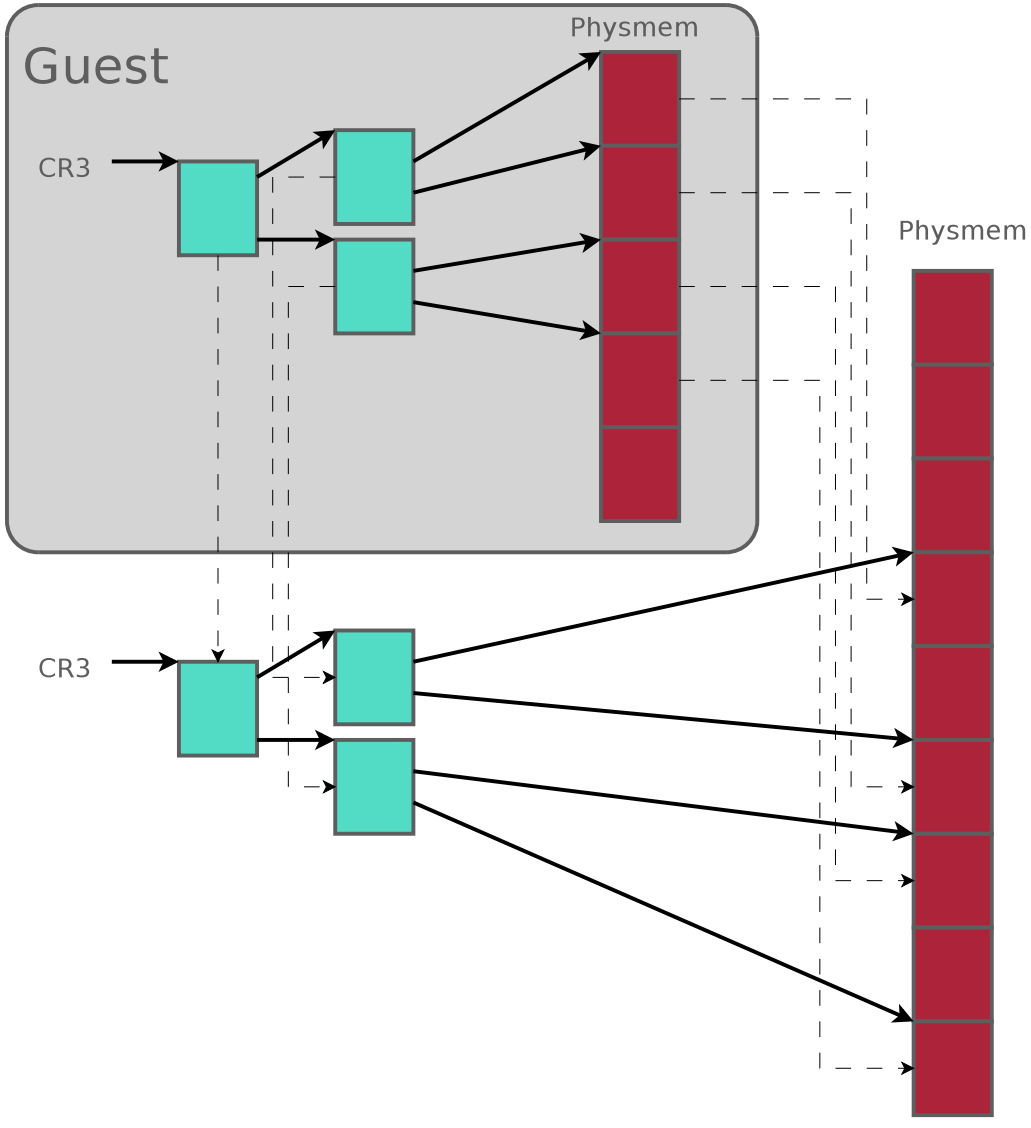
redhat.

Shadow Paging



redhat.

Shadow Paging



redhat.

Shadow Paging (Cont.)

Slow!

- CR3 change traps to hypervisor
- Page table modification by a guest traps to hypervisor
- New address space creation (fork) requires new shadow page table to be created

redhat.

Shadow Paging (Cont.)

What actually happens

$$\left.\begin{array}{c} GVA \\ \Downarrow \\ GPA \\ \Downarrow \\ HPA \end{array}\right\} \xrightarrow{shadow} \left\{\begin{array}{c} GVA \\ \Downarrow \\ HPA \end{array}\right.$$

redhat.

EPT Saves the Day

Two level paging in HW so shadow is not needed!

$$
\begin{array}{ll}
GVA & \\
\Downarrow & \textit{Guest Page Table} \\
GPA & \\
\Downarrow & \textit{Extended Page Table} \\
HPA &
\end{array}
$$

redhat.

EPT Saves the Day (Cont.)

Guest manages its address space by itself

Section 2

What About Nested

redhat.

Nested Guest is Running

Three levels of address translation!

$$nGVA$$
$$\Downarrow$$
$$nGPA$$
$$\Downarrow$$
$$GPA$$
$$\Downarrow$$
$$HPA$$

redhat.

Nested Guest is Running (Cont.)

But HW has only two levels!

redhat.

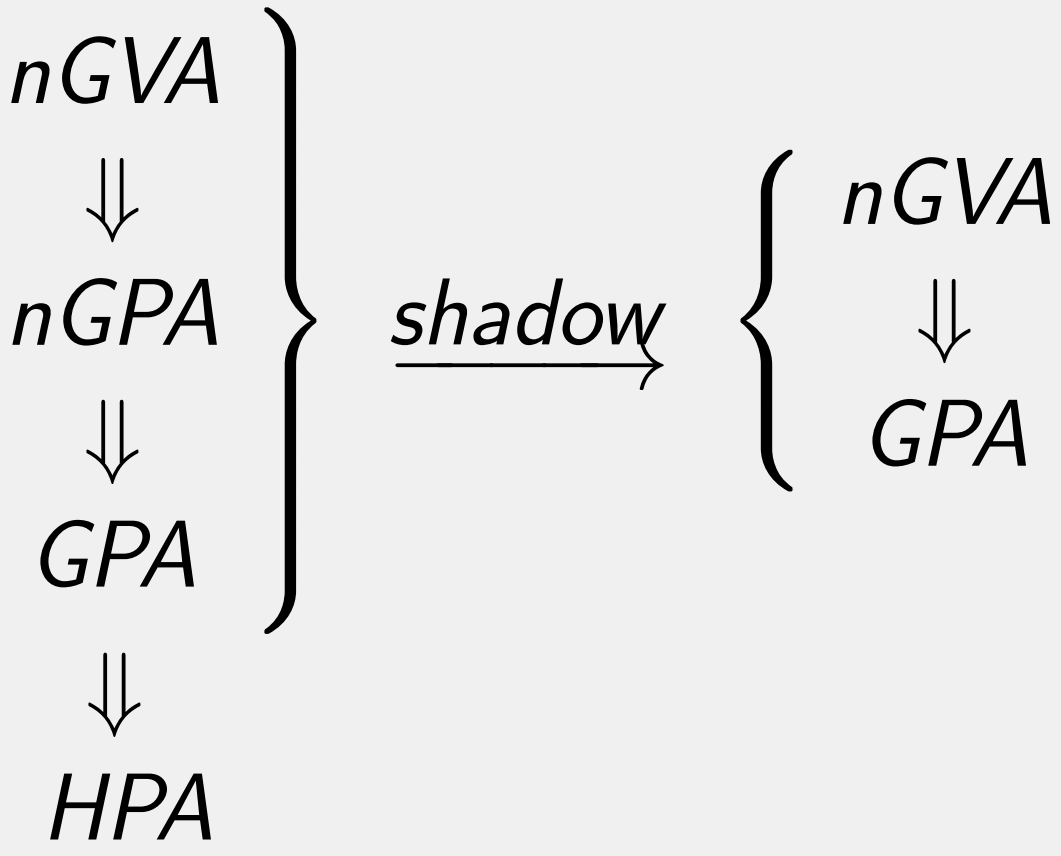
Nested Guest is Running (Cont.)

Something has to be shadowed

redhat.

Shadow on EPT

What actually happens



redhat.

Shadow on EPT (Cont.)

Slow for all the same reasons as regular shadowing

Plus each L2's #PF and CR3 access traps to L0 and forwarded to L1

redhat.

Shadow on EPT (Cont.)

Slow for all the same reasons as regular shadowing
Plus each L2's #PF and CR3 access traps to L0 and forwarded to L1

redhat.

Nested EPT

Key observation

Guests are created/destroyed much less frequently than processes

redhat.

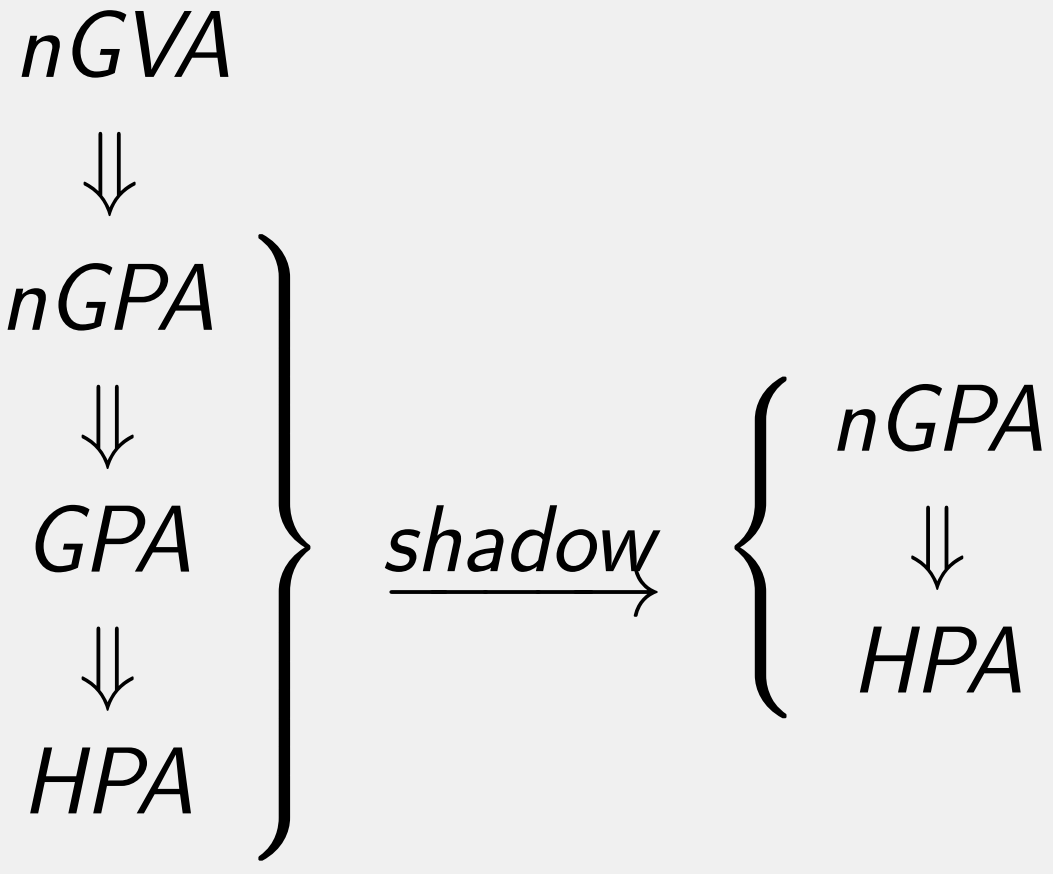
Nested EPT (Cont.)

Why not shadow nGPA to HPA translation instead?

redhat.

Nested EPT (Cont.)

What actually happens



redhat.

Nested EPT (Cont.)

Nested guest manages its address space by itself

Section 3

Implementation

redhat.

Good

KVM already has shadow paging code

redhat.

Good (Cont.)

KVM shadow code understands all guest's paging modes

- 32-bit Paging
- PAE Paging
- IA-32e Paging

redhat.

32-bit Paging

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Address of page directory[1]																				Ignored							PCD	PWT	Ignored			CR3
Bits 31:22 of address of 4MB page frame										Reserved (must be 0)						Bits 39:32 of address[2]			PAT	Ignored			G	1	D	A	PCD	PWT	U/S	R/W	1	PDE: 4MB page
Address of page table																				Ignored				0	Ign	A	PCD	PWT	U/S	R/W	1	PDE: page table
Ignored																															0	PDE: not present
Address of 4KB page frame																				Ignored			G	PAT	D	A	PCD	PWT	U/S	R/W	1	PTE: 4KB page
Ignored																															0	PTE: not present

redhat.

PAE Paging

63	62	61	60	59	58	57	56	55	54	53	52	51		M^1	M-1			32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Ignored[2]																		Address of page-directory-pointer table																														Ignored				CR3
Reserved[3]														Address of page directory																											Ign.		Rsvd.				PCD	PWT	Rsvd	1	PDPTE: present
Ignored																																																	0	PDTPE: not present
XD[4]	Reserved													Address of 2MB page frame																					Reserved						PAT	Ign.		G	1	D	A	PCD	PWT	U/S	R/W	1	PDE: 2MB page
XD	Reserved																																											Address of page table						Ign.		0	Ign	A	PCD	PWT	U/S	R/W	1	PDE: page table
Ignored																																																			0	PDE: not present
XD	Reserved																																												Address of 4KB page frame						Ign.		G	PAT	D	A	PCD	PWT	U/S	R/W	1	PTE: 4KB page
Ignored																																																					0	PTE: not present

redhat.

IA-32e Paging

63	62	61	60	59	58	57	56	55	54	53	52	51		M^1	M-1			32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved[2]															Address of PML4 table																								Ignored								PCD	PWT	Ign.			CR3
XD[3]	Ignored													Rsvd.	Address of page-directory-pointer table																								Ign.			Rsvd	Ign		A	PCD	PWT	U/S	R/W	1	PML4E: present
Ignored																																																	0	PML4E: not present
XD	Ignored													Rsvd.	Address of 1GB page frame																								Reserved								PAT	Ign.		G	1	D	A	PCD	PWT	U/S	R/W	1	PDPTE: 1GB page
XD	Ignored													Rsvd.	Address of page directory																								Ign.											0	Ign		A	PCD	PWT	U/S	R/W	1	PDPTE: page directory
Ignored																																																	0	PDTPE: not present
XD	Ignored													Rsvd.	Address of 2MB page frame																								Reserved								PAT	Ign.		G	1	D	A	PCD	PWT	U/S	R/W	1	PDE: 2MB page
XD	Ignored													Rsvd.	Address of page table																								Ign.											0	Ign		A	PCD	PWT	U/S	R/W	1	PDE: page table
Ignored																																																	0	PDE: not present
XD	Ignored													Rsvd.	Address of 4KB page frame																								Ign.								PAT		G	D	A	PCD	PWT	U/S	R/W	1	PTE: 4KB page
Ignored																																																	0	PTE: not present

redhat.

What is Common?

- `bit 0` - Present
- `bit 1` - R/W
- `bit 2` - User
- `bit 5` - Accessed
- `bit 6` - Dirty
- `bit 7` - Large Page
- `bit 63` - Execute Disabled (PAE & IA-32e)

redhat.

What is Different?

- PTE size (32bit vs 64bit)
- Number of page table levels

redhat.

How Differences are Handled

- Shadow paging code is a template
- All differences are template parameters
- Template code is compiled for each paging mode
- `vcpu->mmu` is initialized according to current guest mode

redhat.

Bad

EPT page table format is very different

redhat.

EPT Page Table Format

63 62 61 60 59 58 57 56 55 54 53 52 51	M[1]	M-1	32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6	5 4 3	2 1 0	
Reserved			Address of EPT PML4 table	Rsvd.	A/D	EPT PWL–1	EPT PS MT	EPTP[2]
Ignored	Rsvd.		Address of EPT page-directory-pointer table	Ign.	A	Reserved	X W R	PML4E: present
SVE[3]	Ignored						0 0 0	PML4E: not present
SVE	Ignored	Rsvd.	Physical address of 1GB page	Reserved	Ign. D A 1	IPAT	EPT MT	X W R	PDPTE: 1GB page
Ignored	Rsvd.		Address of EPT page directory		Ign. A 0	Rsvd.		X W R	PDPTE: page directory
SVE	Ignored						0 0 0	PDTPE: not present
SVE	Ignored	Rsvd.	Physical address of 2MB page	Reserved	Ign. D A 1	IPAT	EPT MT	X W R	PDE: 2MB page
Ignored	Rsvd.		Address of EPT page table		Ign. A 0	Rsvd.		X W R	PDE: page table
SVE	Ignored						0 0 0	PDE: not present
SVE	Ignored	Rsvd.	Physical address of 4KB page		Ign. D A Ign	IPAT	EPT MT	X W R	PTE: 4KB page
SVE	Ignored						0 0 0	PTE: not present

redhat.

Find the Differences

Bit	Regular Paging	EPT
0	present	readable
1	writable	writable
2	user	executable
5	accessed	memory type
6	dirty	ignore pat
7	large page	large page
8	ignored	accessed
9	ignored	dirty
63	XD	Suppress #VE

redhat.

Step One: Make PTE handling parameterizable

- Reserved bits
- Present
- Dirty
- Accessed
- Permission

redhat.

Step Two: Teaching Shadow About EPT

```
arch/x86/kvm/mmu.c         |    5 +++++
arch/x86/kvm/paging_tmpl.h |   37 ++++++++++++++++++++++++++++++++++++-
2 files changed, 41 insertions(+), 1 deletion(-)
```

redhat.

Step Three: Switch to Shadow EPT

On nested guest entry switch `vcpu->mmu` to EPT

redhat.

But...

KVM uses `vcpu->mmu` for two purposes:

1. Virtualize guests memory
2. Translate GVA to GPA during instruction emulation

redhat.

But... (Cont.)

What if L0 wants to emulate L2's instruction?

It needs to translate an address from nGVA to GPA

EPT `vcpu->mmu` translates from nGPA to GPA

redhat.

But... (Cont.)

What if L0 wants to emulate L2's instruction?

It needs to translate an address from nGVA to GPA

EPT `vcpu->mmu` translates from nGPA to GPA

redhat.

But... (Cont.)

What if L0 wants to emulate L2's instruction?
It needs to translate an address from nGVA to GPA
EPT `vcpu->mmu` translates from nGPA to GPA

redhat.

Solution

Nested MMU

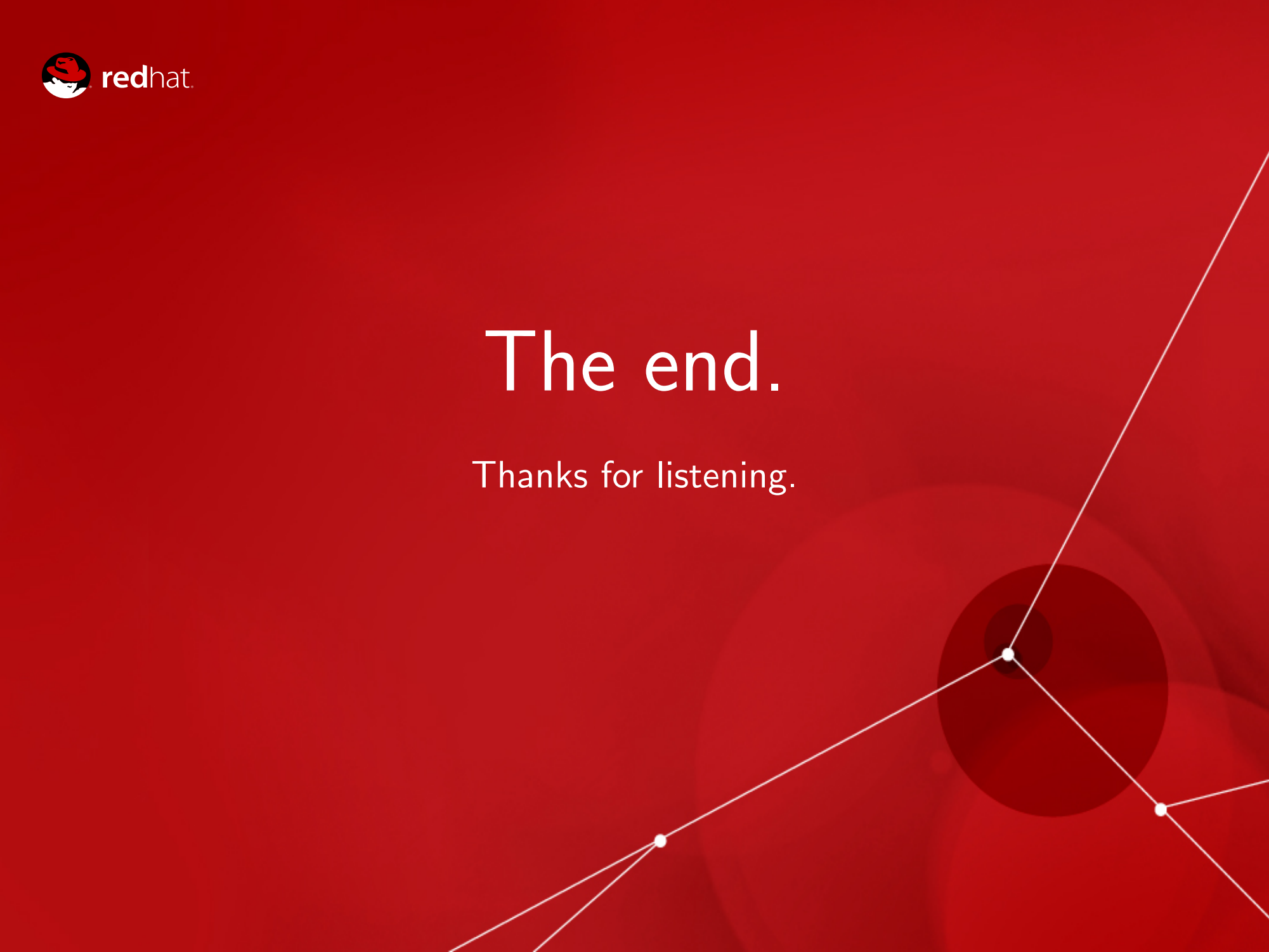
- Pointed to by `vcpu->nested_mmu`
- Translates nested guest's address twice:
  1. nGVA→nGPA
  2. nGPA→GPA

redhat.

Numbers

Kernel compile

Shadow-on-EPT:	33m22s
Nested EPT:	9m46s


redhat.
The end.
Thanks for listening.