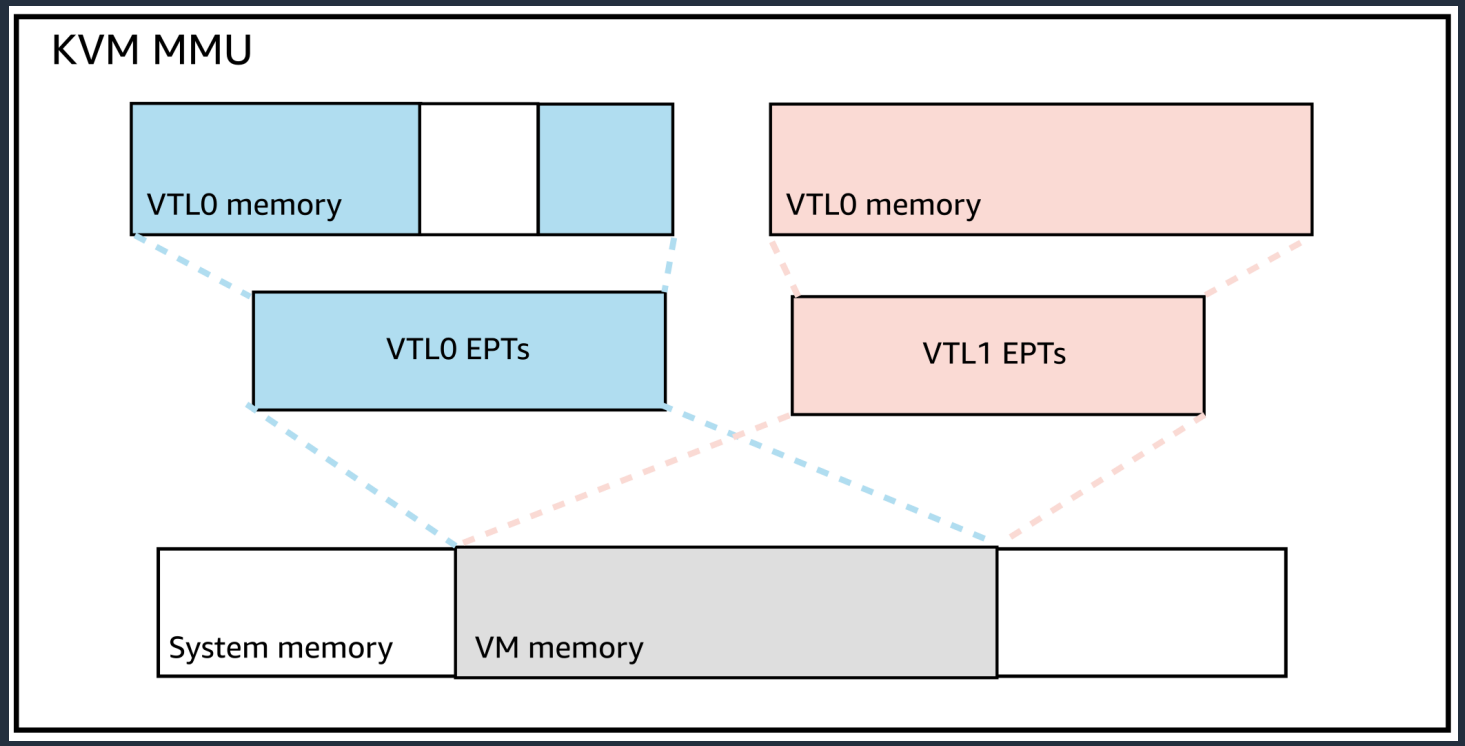
Hyper-V's Virtual Secure Mode in KVM

NICOLAS SAENZ JULIENNE



Intro

Kernel and hypervisor engineer at AWS, working alongside Anel Orazgaliyeva.

Introduced VSM and our plans for upstreaming at the KVM forum 2023 (*https://kvm-forum.qemu.org/2023/talk/TK7YGD/*)

Sent first VSM enablement RFC on Nov 8th (*https://lore.kernel.org/lkml/20231108111806.92604-1-nsaenz@amazon.com/*)

The aim of this session is to reach an agreement on what are the right abstractions for emulating VSM in KVM.

Finding the right abstraction: one kvm_vcpu per VTL

One memory slot address space, MMU role, and memory attributes array per VTL.

APIC groups for IPI routing and APIC ID filtering.

Cross VTL communication and VTL semantics enforcing becomes simpler.

Flexible in case moving behavior into KVM is necessary.

QEMU implementation straightforward.

User-space/kernel responsibility split less clear.

Needs a vCPU *poll()* interface.

Finding the right abstraction: one struct kvm per VTL

Memory slots, MMU roles, memory attributes and LAPIC no longer need to be VTL aware.

All VTL awareness moves to user-space, complicates cross VTL interactions

Kernel looses the ability to influence VSM behavior, may affect our ability to implement optimizations.

Clear responsibility split.

High complexity of supporting two “struct kvm” VMs in user-space.

Needs a vCPU *poll()* interface.

Feedback

Are we simplifying the memory management aspects of VSM in detriment of introducing a lot of complexity in user-space?

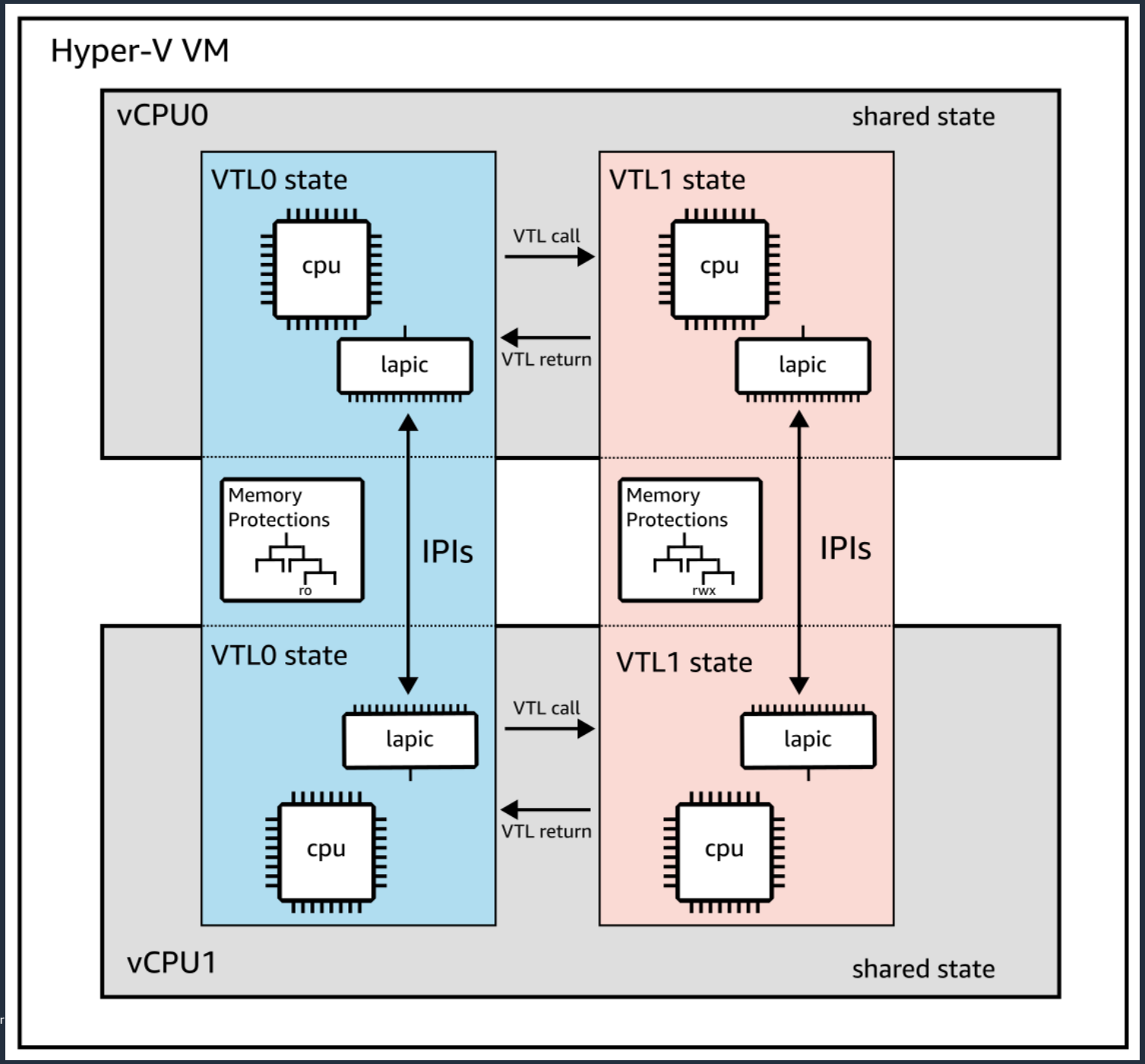
Not having VTL awareness in-kernel could be detrimental if we ever need to implement optimizations.

Is there common ground with other use-cases that would justify introducing new memory slot modification primitives? For ex. first class memory overlay support.

We have to agree on what constitutes a vCPU event that will wake up *poll()*.

Are memory slot address spaces really *that* bad? ☺

VSM



Finding the right abstraction: VTL aware kvm_vcpu

Resulting code is highly intrusive. VTL awareness is necessary in a lot of x86 generic code.

vCPU has multiple local APICs and requires a rework of the KVM LAPIC.

Async event injection becomes overly complicated.

Needs memory slot address spaces, VTL aware MMU roles, VTL aware memory attributes.

Serialization/Deserialization of vCPUs needs complete rework.

VTL switch simpler/faster.

Very little needs to be done in VMM.

vcpu poll()

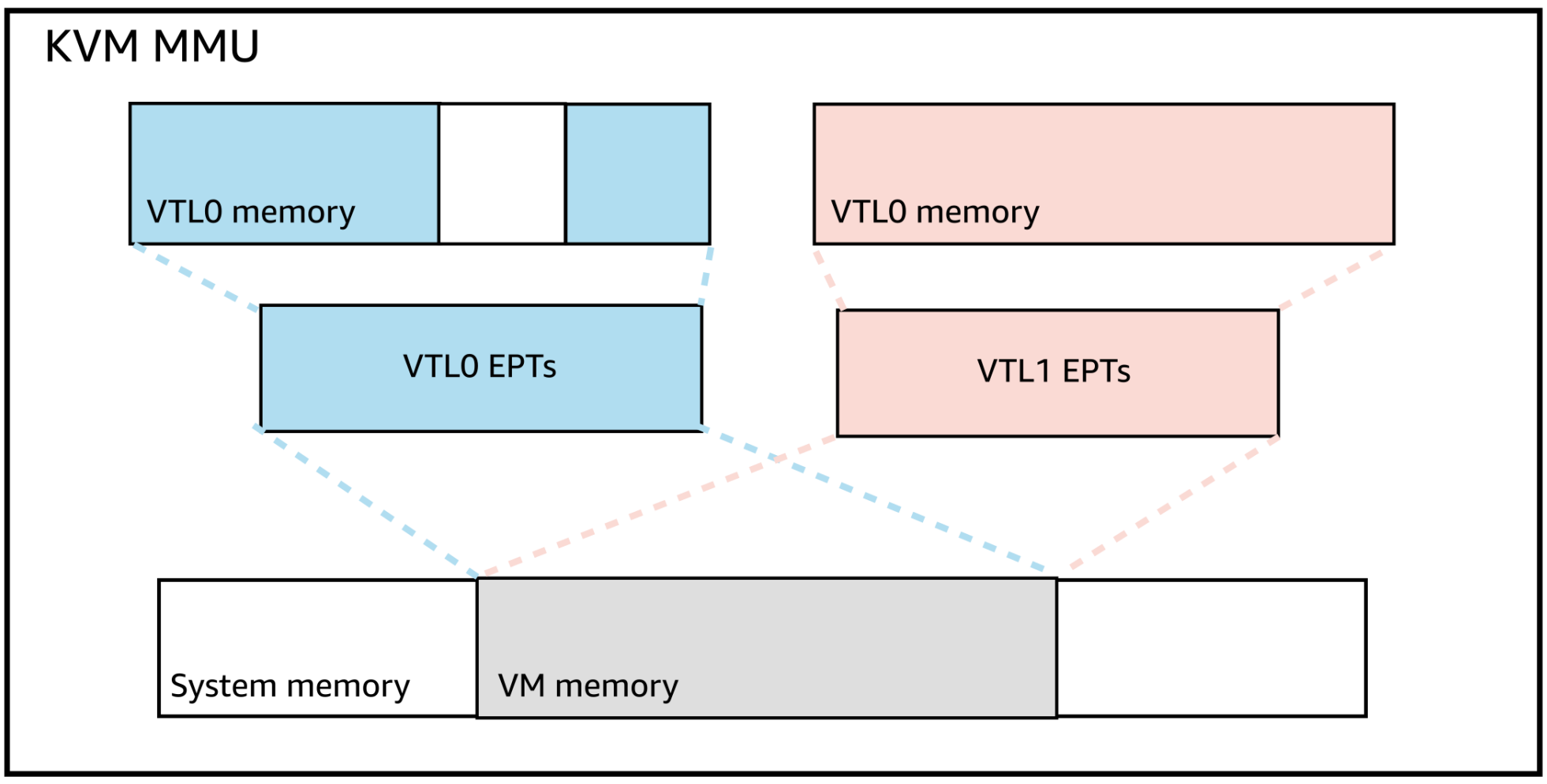
Inactive privileged VTL vCPUs need to be monitored for pending events (IPIs, timers).

These events have priority over the execution of less privileged VTLs.

Need to narrow down what constitutes an “event”:

- Consider a *vcpu_kick()* as the event source.
- Make *mp_state* aware of vCPU halted in user-space. Trigger poll event if *mp_state* changes.

VSM



Thanks!

References

Per VTL State

Private

SYSENTER_CS, SYSENTER_ESP, SYSENTER_EIP, STAR,
LSTAR, CSTAR, SFMASK, EFER, PAT,
KERNEL_GSBASE, FS.BASE, GS.BASE, TSC_AUX
HV_X64_MSR_HYPERCALL, HV_X64_MSR_GUEST_OS_ID
HV_X64_MSR_REFERENCE_TSC, HV_X64_MSR_APIC_FREQUENCY
HV_X64_MSR_EOI, HV_X64_MSR_ICR
HV_X64_MSR_TPR, HV_X64_MSR_APIC_ASSIST_PAGE
HV_X64_MSR_NPIEP_CONFIG
HV_X64_MSR_SIRBP, HV_X64_MSR_SCONTROL
HV_X64_MSR_SVERSION, HV_X64_MSR_SIEFP
HV_X64_MSR_SIMP, HV_X64_MSR_EOM
HV_X64_MSR_SINT0 - HV_X64_MSR_SINT15
HV_X64_MSR_STIMER0_CONFIG -
HV_X64_MSR_STIMER3_CONFIG
HV_X64_MSR_STIMER0_COUNT - HV_X64_MSR_STIMER3_COUNT

Local APIC registers (including CR8/TPR)
RIP, RSP
RFLAGS
CR0, CR3, CR4
DR7
IDTR, GDTR
CS, DS, ES, FS, GS, SS, TR, LDTR
TSC

Shared

HV_X64_MSR_TSC_FREQUENCY
HV_X64_MSR_VP_INDEX
HV_X64_MSR_VP_RUNTIME
HV_X64_MSR_RESET
HV_X64_MSR_TIME_REF_COUNT
HV_X64_MSR_GUEST_IDLE
HV_X64_MSR_DEBUG_DEVICE_OPTIONS
HV_X64_MSR_BELOW_1MB_PAGE
HV_X64_MSR_STATS_PARTITION_RETAIL_PAGE
HV_X64_MSR_STATS_VP_RETAIL_PAGE
MTRRs
MCG_CAP
MCG_STATUS

Rax, Rbx, Rcx, Rdx, Rsi, Rdi, Rbp
CR2
R8 - R15
DR0 - DR5
X87 floating point state
XMM state
AVX state
XCR0 (XFE)

References

Hypervisor Top Level Functional Specification, v6.0b. Microsoft Feb, 2020 (GitHub)

Battle of the SKM and IUM: How Windows 10 Rewrites OS Architecture. Alex Ionescu, Chief Architect, CrowdStrike (Black Hat USA 2015)

AWS official Credential Guard documentation (docs.aws.amazon.com)

VSM enablement RFC (https://lore.kernel.org/lkml/ZUuzFshjO7NO5k3b@google.com)

Introduced VSM and our plans for upstreaming at the KVM forum 2023 (https://kvm-forum.qemu.org/2023/talk/TK7YGD/)