

Emulating Hyper-V in 2022

KVM Forum 2022

Vitaly Kuznetsov, Red Hat

About myself

- ▶ Long time KVM contributor, focusing mainly on Hyper-V emulation, running KVM on Hyper-V, KVM PV features and nested virt.

- ▶ Newly appointed KVM X86 HYPER-V co-maintainer.

Why emulating some other hypervisor?

- ▶ Make existing [proprietary] operating systems run faster!

- ▶ ... much faster in some cases.

How do we know what proprietary OS need?

- ▶ Publicly available Hyper-V Top Level Functional Specification (TLFS):
 - <https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/tlfs/tlfs>
 - <https://github.com/MicrosoftDocs/Virtualization-Documentation>

“Legacy” enlightenments

(which I’m not talking about today)

- ▶ hv-relaxed
- ▶ hv-vapic
- ▶ hv-spinlocks
- ▶ hv-vpindex
- ▶ hv-runtime
- ▶ hv-synic
- ▶ hv-stimer
- ▶ hv-reset
- ▶ hv-frequencies
- ▶ hv-reenlightenment
- ▶ hv-tlbflush
- ▶ hv-ipi
- ▶ hv-evmcs
- ▶ hv-stimer-direct


Previously given talks:

- ▶ DevConf2019/FOSDEM2019: "Enlightening" KVM: Hyper-V emulation
 - <https://www.youtube.com/watch?v=kKmsLAIctQ8>
 - https://archive.fosdem.org/2019/schedule/event/vai_enlightening_kvm/
- ▶ KVM Forum 2018: "Hybrid" Nesting: KVM on Hyper-V by Vitaly Kuznetsov & Tianyu Lan
<https://www.youtube.com/watch?v=Fn7mQYObkvs>

(Some) documentation

- ▶ QEMU:

<https://gitlab.com/qemu-project/qemu/-/blob/master/docs/system/i386/hyperv.rst>



Recently implemented enlightenments

New feature: hv-avic

- ▶ The enlightenment allows to use Hyper-V SynIC (hv-synic) with hardware APICv/AVIC enabled
 - Supported since v5.15
 - Windows has to obey to the 'deprecate AutoEOI' recommendation
 - "Ancient" Windows versions may ignore the recommendation and keep using AutoEOI effectively disabling the feature.

New feature: hv-emsr-bitmap

- ▶ Enlightened MSR-Bitmap
 - Nested specific, allows L0 (KVM) and L1 (Hyper-V) hypervisors to collaborate to avoid unnecessary updates to L2 MSR-Bitmap upon vmexits.
 - Supported for both nVMX (v5.17) and nSVM (v5.18)
 - -700 CPU cycles per Vmexit
 - Also supported for KVM-on-Hyper-V on VMX (since v4.18) and SVM (v5.14)

New feature: hv-xmm-input

- ▶ Use XMM registers for hypercall input values
 - Allows to pass parameters for certain hypercalls (HvFlushVirtualAddressSpace{,Ex}, HvFlushVirtualAddressList{,Ex}, HvCallSendSyntheticClusterIpiEx) using XMM registers
 - Supported since v5.14

New feature: hv-synDBG

- ▶ Hyper-V Synthetic debugger support
 - Enables synthetic debugger interface.
 - Used by Windows Kernel debugger rather than sending packets via serial/network, adding significant performance boost.
 - Supported since v5.10

New feature: hv-enforce-cpuid

- ▶ Limit available Hyper-V features to what was presented to the guest in CPUID:
 - By default, KVM allows guests to use ALL currently supported Hyper-V enlightenments, regardless of what is exposed in guest visible CPUIDs
 - **'hv-enforce-cpuid'** can be used to limit the guest to use only exposed Hyper-V enlightenments.
 - Dependencies between enlightenments have to be tracked by VMM.
 - Supported since v5.14

WIP feature: improved hv-tlbflush and hv-tlbflush-ext

- ▶ Previously, Hyper-V TLB flush hypercalls were handled in 'all or nothing' mode (per-vCPU) by KVM but the interface is actually fine grained, allowing to specify the particular GVAs to flush:
 - + up to 4096 consequent GFNs per one entry ("Extended GVA ranges")
- ▶ Current status: v9 on the mailing list
 - <https://lore.kernel.org/kvm/20220803134110.397885-1-vkuznets@redhat.com/>
- ▶ Prerequisite for 'L2 TLB flush'

WIP feature: hv-tlbflush-direct


- ▶ Allows L0 (KVM) to directly handle TLB flush hypercalls from L2 guest without the need to exit to L1 (Hyper-V) hypervisor.
- ▶ Current status: v9 on the mailing list
 - <https://lore.kernel.org/kvm/20220803134110.397885-1-vkuznets@redhat.com/>
- ▶ Supported for KVM on Hyper-V on VMX (v5.10) and SVM (v5.14) for Windows/Hyper-V guests.

WIP feature: updated Enlightened VMCS

- ▶ Enlightened VMCS v.1 gained support for the following features with 2022 'update':
 - PerfGlobalCtrl
 - EnclsExitingBitmap
 - Tsc Scaling
 - GuestLbrCtl
 - CET
 - SSP
- ▶ Initial implementation makes it hard to update :-)
- ▶ Current status: v5 on the mailing list:
 - <https://lore.kernel.org/kvm/20220802160756.339464-1-vkuznets@redhat.com/>

WIP feature: hv-invtsc

- ▶ Normally, architectural 'invariant TSC' bit (CPUID.80000007H:EDX[8]) is masked in CPUID until the guest flips bit 0 in HV_X64_MSR_TSC_INVARIANT_CONTROL MSR.
- ▶ With 'invariant TSC' bit set, 'Reenlightenment' becomes unneeded.
- ▶ Existing Windows versions are known to work well even when architectural 'invariant TSC' is set from boot.
- ▶ Status: v1 on the mailing list
 - <https://lore.kernel.org/kvm/20220713150532.1012466-1-vkuznets@redhat.com/>



Enlightenments
described in TLFS
but no work has
started (yet)

TODO feature: hv-stimer-unhalted

- ▶ Synthetic Time-Unhalted Timer MSR:
 - “Unlike regular synthetic timers that accumulate time when the guest has halted (ie: gone idle), the Synthetic Time-Unhalted Timer accumulates time only while the guest is not halted”.
 - Can only send interrupts (no Vmbus messages).

TODO feature: hv-npiep

- ▶ “Non-Privileged Instruction Execution Prevention”:
 - Block the execution of the SIDT, SGDT, SLDT, and STR instructions by user mode.
 - Similar to the already present UMIP emulation in KVM.

TODO feature: hv-tlbflush and hv-tlbflush-direct further improvements

- ▶ “KVM: x86: hyper-v: Fine-grained TLB flush + L2 TLB flush features” series (on the mailing list at this moment) introduces KFIFOs for L1 and L2 TLB flushes (queued work for vCPUs):
 - Makes it possible to analyze ‘AddressSpace’ (CR3) argument (ignored now)
 - Should eliminate some unnecessary TLB flushes.

TODO feature: hv-tdp-flush


- ▶ The HvFlushGuestPhysicalAddressSpace hypercall invalidates cached L2 GPA to GPA mappings within a second level address space.
- ▶ The HvFlushGuestPhysicalAddressSpace hypercall invalidates cached GVA / L2 GPA to GPA mappings within a portion of a second level address space.
- ▶ Operate on all vCPUs
- ▶ "Enlightened TLB" on SVM: ASID invalidations "just flush TLB entries derived from first level address translation".
- ▶ Already implemented for KVM on Hyper-V on VMX (v4.19) and SVM (v5.14)

TODO feature: free page reporting

- ▶ HvExtCallMemoryHeatHint hypercall, allows the guest to report 'cold' memory to the host so it can be utilized differently.
- ▶ Already supported for Linux on Hyper-V guests (v5.13)

Elephant in the room: VSM

- ▶ “Creation and management of new security boundaries within operating system software”.
- ▶ Isolates:
 - Memory Access Protections
 - Virtual Processor State
 - Interrupts
 - Overlay Pages
- ▶ Used by Device Guard, Credential Guard, virtual TPMs and shielded VMs features in Windows.



Using Hyper-V emulation for something else?

KVM-on-KVM: use Hyper-V features?

- ▶ KVM doesn't have [m]any PV features to speed up KVM on KVM workloads but KVM on Hyper-V and Hyper-V on KVM code already implements quite a few:
 - Enlightened VMCS
 - Enlightened MSR-Bitmap
 - "Direct" (L2) TLB flush
 - EPT/NPT flush
- ▶ It is already possible to make L0 KVM pretend it's "genuine Hyper-V" to make L1 KVM use PV features but maybe we should "glorify" the hack and introduce a KVM PV bit?

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.



[linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)



[youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)



[facebook.com/redhatinc](https://www.facebook.com/redhatinc)



twitter.com/RedHat