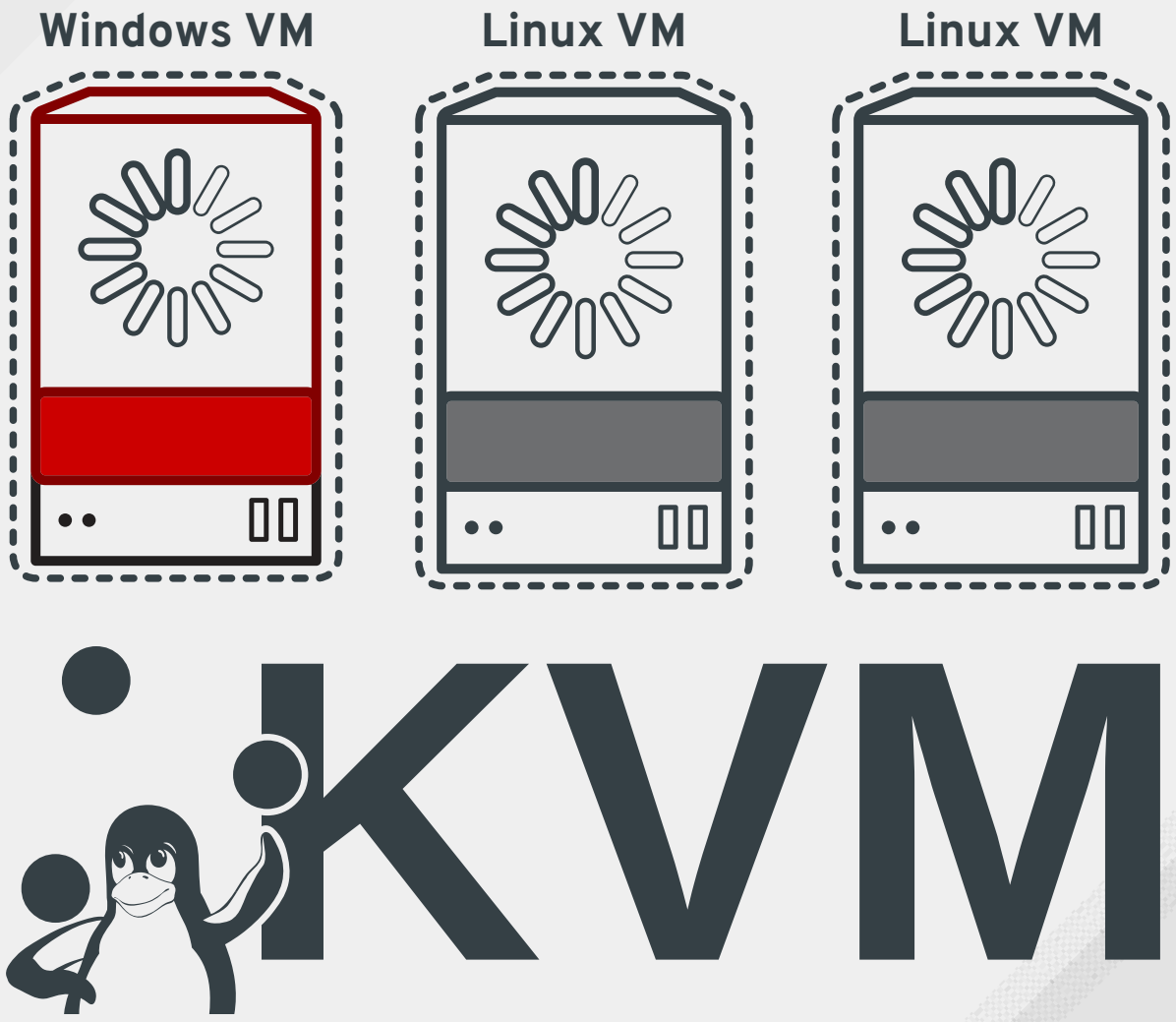
"ENLIGHTENING" KVM

HYPER-V EMULATION

VITALY KUZNETSOV

<vkuznets@redhat.com>

FOSDEM 2019


Windows VM
Linux VM
Linux VM
KVM

DOES GUEST OS MAKE A DIFFERENCE?

DOES GUEST OS MAKE A DIFFERENCE?

IN THEORY, IT DOESN'T

DOES GUEST OS MAKE A DIFFERENCE?

IN PRACTICE, IT DOES

```
# dmesg | grep -i kvm
[    0.000000] DMI: Red Hat KVM, BIOS rel-1.11.1-0-g0551a4be2c-prebuilt.qemu-project.org 0
[    0.000000] Hypervisor detected: KVM
[    0.000000] kvm-clock: Using msrs 4b564d01 and 4b564d00
[    0.000000] kvm-clock: cpu 0, msr 2768001, primary cpu clock
[    0.000000] kvm-clock: using sched offset of 9962523967 cycles
[    0.000003] clocksource: kvm-clock: mask: 0xffffffffffffffff max_cycles: 0x1cd42e4dffb,
[    0.038540] Booting paravirtualized kernel on KVM
[    0.147439] KVM setup async PF for cpu 0
[    0.147444] kvm-stealtime: cpu 0, msr 13ba16140
[    0.480396] KVM setup pv remote TLB flush
[    0.584919] clocksource: Switched to clocksource kvm-clock
```

Emulating hardware interfaces can be slow

Emulating hardware interfaces can be slow

Invent virtualization-friendly (paravirtualized) interfaces!

Emulating hardware interfaces can be slow

Invent virtualization-friendly (paravirtualized) interfaces!

Add support to guest OSes

Emulating hardware Interfaces can be slow

Invent virtualization-friendly (paravirtualized) interfaces!

Add support to guest OSes

... but what about proprietary OSes?

We can try writing device drivers for such OSes

We can try writing device drivers for such OSes

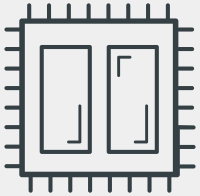
... but some core features (interrupt handling, timekeeping,...) are not devices

We can try writing device drivers for such OSes

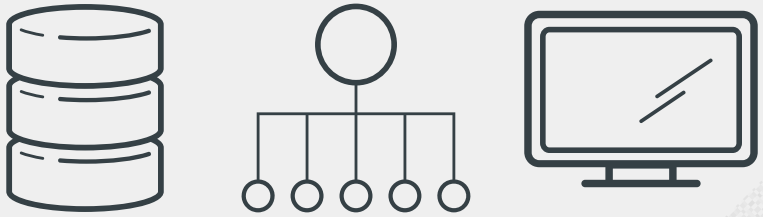
... but some core features (interrupt handling, timekeeping,...) are not devices

Emulate an already supported (proprietary) hypervisor interfaces solving the exact same issues!

Hyper-V Emulation in KVM

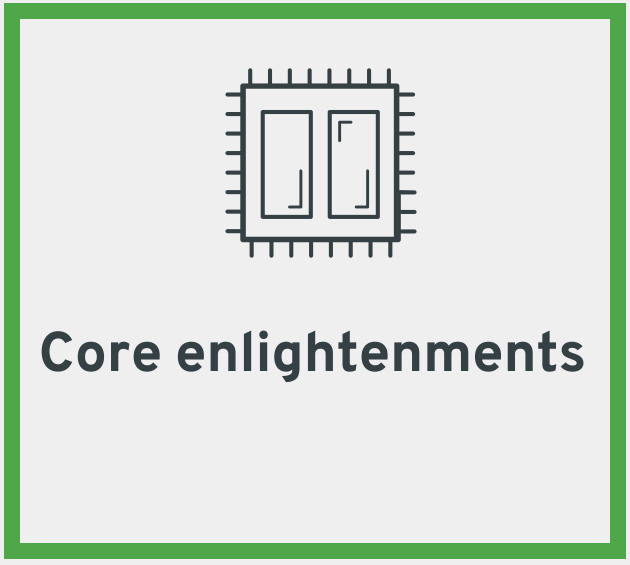


Core enlightenments

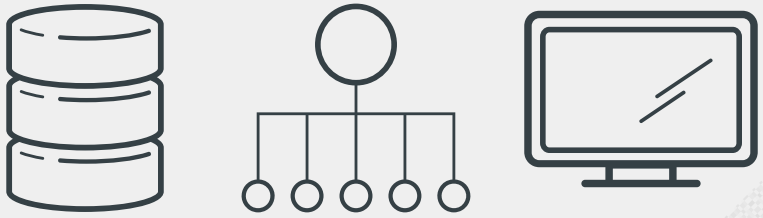


Device drivers (VMBus)

Hyper-V Emulation in KVM



Core enlightenments



Device drivers (VMBus)

Existing documentation

- https://libvirt.org/formatdomain.html

hyperv

Enable various features improving behavior of guests running Microsoft Windows.

Feature	Description	Value	Since
relaxed	Relax constraints on timers	on, off	*1.0.0 (QEMU 2.0)*
vapic	Enable virtual APIC	on, off	*1.1.0 (QEMU 2.0)*
spinlocks	Enable spinlock support	on, off; retries - at least 4095	*1.1.0 (QEMU 2.0)*
vpindex	Virtual processor index	on, off	*1.3.3 (QEMU 2.5)*
runtime	Processor time spent on running guest code and on behalf of guest code	on, off	*1.3.3 (QEMU 2.5)*
synic	Enable Synthetic Interrupt Controller (SyNIC)	on, off	*1.3.3 (QEMU 2.6)*
stimer	Enable SyNIC timers	on, off	*1.3.3 (QEMU 2.6)*
reset	Enable hypervisor reset	on, off	*1.3.3 (QEMU 2.5)*
vendor_id	Set hypervisor vendor id	on, off; value - string, up to 12 characters	*1.3.3 (QEMU 2.5)*
frequencies	Expose frequency MSRs	on, off	*4.7.0 (QEMU 2.12)*
reenlightenment	Enable re-enlightenment notification on migration	on, off	*4.7.0 (QEMU 3.0)*
tlbflush	Enable PV TLB flush support	on, off	*4.7.0 (QEMU 3.0)*
ipi	Enable PV IPI support	on, off	*4.10.0 (QEMU 3.1)*
evmcs	Enable Enlightened VMCS	on, off	*4.10.0 (QEMU 3.1)*

Existing documentation

- https://libvirt.org/formatdomain.html

hyperv

Enable various features improving behavior of guests running Microsoft Windows.

Feature	Description	Value	Since
relaxed	Relax constraints on timers	on, off	*1.0.0 (QEMU 2.0)*
vapic	Enable virtual APIC	on, off	*1.1.0 (QEMU 2.0)*
spinlocks	Enable spinlock support	on, off; retries - at least 4095	*1.1.0 (QEMU 2.0)*
vpindex	Virtual processor index	on, off	*1.3.3 (QEMU 2.5)*
runtime	Processor time spent on running guest code and on behalf of guest code	on, off	*1.3.3 (QEMU 2.5)*
synic	Enable Synthetic Interrupt Controller (SyNIC)	on, off	*1.3.3 (QEMU 2.6)*
stimer	Enable SyNIC timers	on, off	*1.3.3 (QEMU 2.6)*
reset	Enable hypervisor reset	on, off	*1.3.3 (QEMU 2.5)*
vendor_id	Set hypervisor vendor id	on, off; value - string, up to 12 characters	*1.3.3 (QEMU 2.5)*
frequencies	Expose frequency MSRs	on, off	*4.7.0 (QEMU 2.12)*
reenlightenment	Enable re-enlightenment notification on migration	on, off	*4.7.0 (QEMU 3.0)*
tlbflush	Enable PV TLB flush support	on, off	*4.7.0 (QEMU 3.0)*
ipi	Enable PV IPI support	on, off	*4.10.0 (QEMU 3.1)*
evmcs	Enable Enlightened VMCS	on, off	*4.10.0 (QEMU 3.1)*

- **OR** https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/reference/tlfs

EXISTING HYPER-V ENLIGHTENMENTS

RELAXED TIMING

```
QEMU syntax:
 - cpu ....,hv-relaxed

libvirt syntax:
<features>
     <hyperv>
          ...
          <relaxed state='on' />
     </hyperv>
</features>
```

- Tells guest OS to disable watchdog timeouts
- Some Windows versions do this regardless of the setting when running on Hyper-V

PARAVIRTUALIZED APIC

```
QEMU syntax:
 - cpu ....,hv-vapic

libvirt syntax:
<features>
     <hyperv>
          ...
          <vapic state='on' />
     </hyperv>
</features>
```

- Provides "VP assist page" MSR for Paravirtualized EOI signalling (exit-less).
- Required for Enlightened VMCS (**hv-evmcs**) feature
- Some features are not yet implemented in KVM.

PARAVIRTUALIZED SPINLOCKS

```
QEMU syntax:
 - cpu ....,hv-spinlocks=4096

libvirt syntax:
<features>
     <hyperv>
          ...
          <spinlocks state='on' retries='4096'/>
     </hyperv>
</features>
```

- Spinlock retry attempts [0xfff .. 0xffffffff]
  - 0xffffffff means 'never retry' (default)
- Allows **other** guests to run when vCPU is blocked on a spinlock

VP INDEX

```
QEMU syntax:
 - cpu ....,hv-vpindex

libvirt syntax:
<features>
     <hyperv>
          <vpindex state='on'/>
     </hyperv>
</features>
```

- "The partition has access to the synthetic MSR that returns the virtual processor index"
- Required for **hv-tlblush**, **hv-ipi** enlightenments

RUN TIME INFORMATION

```
QEMU syntax:
 - cpu ....,hv-runtime

libvirt syntax:
<features>
     <hyperv>
          ...
          <runtime state='on' />
     </hyperv>
</features>
```

- Provides virtual MSR with time spent in the guest/hypervisor information.
- Windows may use the info for better scheduling.

CRASH INFORMATION

```
QEMU syntax:
 - cpu ....,hv-crash

libvirt syntax:
<devices>
  ...
  <panic model='hyperv'/>
</devices>
```

- Provides additional crash information when Windows crashes
  - available in libvirt domain log
  - useful for analyzing crashes at scale

HYPER-V CLOCKSOURCE

```
QEMU syntax:
 - cpu ....,hv-time

libvirt syntax:
<clock offset='localtime'>
    ...
    <timer name='hypervclock' present='yes'/>
</clock>
```

- Significantly speeds up time related operations
- Libvirt's syntax is quite different from other Hyper-V enlightenments
- Requires stable TSC on the host! **(check that you have 'tsc' in /sys/devices/system/clocksource/clocksource0/current_clocksource!)**

SYNTHETIC INTERRUPT CONTROLLER

```
QEMU syntax:
 - cpu ....,hv-synic

libvirt syntax:
<features>
     <hyperv>
          <synic state='on'/>
     </hyperv>
</features>
```

- Enables synthetic interrupt controller implementation
  - Post messages, Signal events
- Required for VMBus emulation (not yet in qemu)
- Required for **hv-stimer** enlightenment

SYNTHETIC TIMERS

```
QEMU syntax:
 - cpu ....,hv-time,hv-synic,hv-stimer

libvirt syntax:
<features>
     <hyperv>
          <synic state='on'/>
          <stimer state='on'/>
     </hyperv>
</features>
<clock offset='localtime'>
    ...
    <timer name='hypervclock' present='yes'/>
</clock>
```

- Requires **hv-synic** and **hv-time** enlightenments
- Provide 4 synthetic timers per vCPU
- Significantly reduces CPU load for Win10+

PARAVIRTUALIZED TLB SHOOTDOWN

```
QEMU syntax:
 - cpu ....,hv-vpindex,hv-tlbflush

libvirt syntax:
<features>
     <hyperv>
          <vpindex state='on'/>
          <tlbflush state='on'/>
     </hyperv>
</features>
```

- Requires **hv-vpindex**
- Significantly improves performance in overcommited environments

PARAVIRTUALIZED IPI

```
QEMU syntax:
 - cpu ....,hv-vpindex,hv-ipi

libvirt syntax:
<features>
     <hyperv>
          <vpindex state='on'/>
          <ipi state='on'/>
     </hyperv>
</features>
```

- Requires **hv-vpindex**
- Similar to PV tlb flush, significantly improves performance of overcommited environments

VENDOR ID

```
QEMU syntax:
 - cpu ....,hv-vendor-id='KVM Hv'

libvirt syntax:
<features>
     <hyperv>
          ...
          <vendor_id state='on' value='KVM Hv'/>
     </hyperv>
</features>
```

- Defaults to "Microsoft Hv"
  - Windows doesn't care about the value
- Does NOT enable Hyper-V identification in QEMU
  - Some other hv_* feature needs to be enabled

RESET

```
QEMU syntax:
 - cpu ....,hv-reset

libvirt syntax:
<features>
     <hyperv>
          ...
          <reset state='on' />
     </hyperv>
</features>
```

- Just another fancy way to reset your guest
- Even genuine Hyper-V doesn't suggest using it

NESTED RELATED ENLIGHTENMENTS

STABLE CLOCKSOURCE FOR L2

```
QEMU syntax:
 - cpu ....,hv-frequencies,hv-reenlightenment

libvirt syntax:
<features>
     <hyperv>
          <frequencies state='on'/>
          <reenlightenment state='on'/>
     </hyperv>
</features>
```

- Enables synthertic MSRs with APIC/TSC frequencies and notifications on TSC frequency change (migration)
- Essential for Hyper-V to pass stable clocksource to L2
- Not yet fully supported by KVM

ENLIGHTENED VMCS

```
QEMU syntax:
 - cpu ....,hv-vapic,hv-evmcs

libvirt syntax:
<features>
     <hyperv>
          <vapic state='on'/>
          <evmcs state='on'/>
     </hyperv>
</features>
```

- Requires **hv-vapic**
- Speeds up L2 vmexits (10%)
- But disables certain virtualization features (posted interrupts)

DIRECT MODE STIMERS (WIP)

```
QEMU syntax (proposed):
 - cpu ....,hv-stimer-direct

libvirt syntax (proposed):
<features>
     <hyperv>
          <stimer_direct state='on'/>
     </hyperv>
</features>
```

- Same as **hv-stimer** but uses real interrupts instead of VMBus messages
- Used by Hyper-V when running nested

SOME BENCHMARKS

HYPER-V CLOCKSOURCE

```
before = rdtsc();

for (i = 0; i < COUNT; i++)
    clock_gettime(CLOCK_REALTIME, &tp);

after = rdtsc();

printf("%d\n", (after - before)/COUNT);
```

Without hv-time	With hv-time
17600	430

ENLIGHTENED VMCS (NESTED GUEST)

```
before = rdtsc();

for (i = 0; i < COUNT; i++)
    cpuid(0x1);

after = rdtsc();

printf("%d\n", (after - before)/COUNT);
```

Without hv-evmcs	With hv-evmcs
20850	19400

PARAVIRTUALIZED TLB SHOOTDOWN

Phisical host: 12 CPUs

Test: 64 pthreads doing (simplified)

```
for (j = 0; j < nrounds; j++) {
    for (i = 0; i < nchunks; i++)
        addr[i] = mmap(NULL, PAGE_SIZE * pagecount, PROT_READ, MAP_SHARED,
                       fd, i * PAGE_SIZE);
    for (i = 0; i < nchunks; i++)
        v += *addr[i];
    for (i = 0; i < nchunks; i++)
        munmap(addr[i], PAGE_SIZE * pagecount);
}
```

No of vCPUs	Without hv-tlbflush (sec)	With hv-tlbflush (sec)
12	22.08	22.43
24	24.79	22.90
36	26.74	22.99

redhat®

THANK YOU!