"ENLIGHTENING" KVM
HYPER-V EMULATION
VITALY KUZNETSOV
<vkuznets@redhat.com>
FOSDEM 2019
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: 0xffffffffffffff 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](https://libvirt.org/formatdomain.html)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Value</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>relaxed</td>
<td>Relax constraints on timers</td>
<td>on, off</td>
<td>1.0.0 (QEMU 2.0)</td>
</tr>
<tr>
<td>vapi</td>
<td>Enable virtual APIC</td>
<td>on, off</td>
<td>1.1.0 (QEMU 2.0)</td>
</tr>
<tr>
<td>spinlocks</td>
<td>Enable spinlock support</td>
<td>on, off; retries - at least 4095</td>
<td>1.1.0 (QEMU 2.0)</td>
</tr>
<tr>
<td>vpin</td>
<td>Virtual processor index</td>
<td>on, off</td>
<td>1.3.3 (QEMU 2.5)</td>
</tr>
<tr>
<td>runtime</td>
<td>Processor time spent on running guest code and on behalf of guest code</td>
<td>on, off</td>
<td>1.3.3 (QEMU 2.5)</td>
</tr>
<tr>
<td>sync</td>
<td>Enable Synthetic Interrupt Controller (SyNIC)</td>
<td>on, off</td>
<td>1.3.3 (QEMU 2.6)</td>
</tr>
<tr>
<td>timer</td>
<td>Enable SyNIC timers</td>
<td>on, off</td>
<td>1.3.3 (QEMU 2.6)</td>
</tr>
<tr>
<td>reset</td>
<td>Enable hypervisor reset</td>
<td>on, off</td>
<td>1.3.3 (QEMU 2.5)</td>
</tr>
<tr>
<td>vendor_id</td>
<td>Set hypervisor vendor id</td>
<td>on, off; value - string, up to 12 characters</td>
<td>1.3.3 (QEMU 2.5)</td>
</tr>
<tr>
<td>frequencies</td>
<td>Expose frequency MSRs</td>
<td>on, off</td>
<td>4.7.0 (QEMU 2.12)</td>
</tr>
<tr>
<td>reenlightenment</td>
<td>Enable re- enlightenment notification on migration</td>
<td>on, off</td>
<td>4.7.0 (QEMU 3.0)</td>
</tr>
<tr>
<td>tlbflush</td>
<td>Enable PV TLB flush support</td>
<td>on, off</td>
<td>4.7.0 (QEMU 3.0)</td>
</tr>
<tr>
<td>ipl</td>
<td>Enable PV IPI support</td>
<td>on, off</td>
<td>4.10.0 (QEMU 3.1)</td>
</tr>
<tr>
<td>evmcs</td>
<td>Enable Enlightened VMCS</td>
<td>on, off</td>
<td>4.10.0 (QEMU 3.1)</td>
</tr>
</tbody>
</table>
Existing documentation

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

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

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-mpi 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/clocksourc/clocksourc0/curren_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>
<cloc...
PARAVIRTUALIZED TLB SHUTDOWN

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 overcommitted 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 overcommitted 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-vmcs

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
```c
before = rdtsc();

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

after = rdtsc();

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

<table>
<thead>
<tr>
<th>Without hv-time</th>
<th>With hv-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>17600</td>
<td>430</td>
</tr>
</tbody>
</table>
ENLIGHTENED VMCS (NESTED GUEST)

```c
before = rdtsc();

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

after = rdtsc();

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

<table>
<thead>
<tr>
<th>Without hv-evmcs</th>
<th>With hv-evmcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>20850</td>
<td>19400</td>
</tr>
</tbody>
</table>
**PARAVIRTUALIZED TLB SHOOTDOWN**

Physical host: 12 CPUs

Test: 64 pthreads doing (simplified)

```c
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);
}
```

<table>
<thead>
<tr>
<th>No of vCPUs</th>
<th>Without hv-tlbflush (sec)</th>
<th>With hv-tlbflush (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>22.08</td>
<td>22.43</td>
</tr>
<tr>
<td>24</td>
<td>24.79</td>
<td>22.90</td>
</tr>
<tr>
<td>36</td>
<td>26.74</td>
<td>22.99</td>
</tr>
</tbody>
</table>
THANK YOU!