How hard could it be to flip a bit?

KVM PV feature enablement up the virtualization stack.

KVM Forum 2021

Vitaly Kuznetsov <vkuznets@redhat.com>

About myself

- KVM contributor and reviewer
  Areas of interest include:
  - PV features
  - Hyper-V emulation, Windows guests
  - Nesting including Hyper-V-on-KVM and KVM-on-Hyper-V

- “Occasional” QEMU developer
  - Hyper-V and KVM PV feature enablement.

Paravirtualized features in KVM

"Extra" features, not present in the emulated hardware:

- "Native" KVM PV features:
  - `kvmclock, kvm-nopiodelay, kvm-asyncpf, kvm-steal-time, kvm-pv-eoi, kvm-pv-unhalt, kvm-pv-tlb-flush, kvm-async-pf-vmexit, kvm-pv-ipi, kvm-poll-control, kvm-pv-sched-yield, kvm-asyncpf-int, kvm-msi-ext-dest-id, kvm-hc-map-gpa-range, kvm-migration-control`
- Emulating other hypervisors:
  - Hyper-V emulation
    - `hv-relaxed, hv-vapic, hv-spinlocks, hv-vpindex, hv-runtime, hv-crash, hv-time, hv-synic, hv-stimer, hv-tlbflush, hv-ipi, hv-reset, hv-frequencies, hv-reenlightenment, hv-evmcs, hv-stimer-direct, hv-no-nonarch-coresharing`
  - Xen emulation
    - Hypercalls, shared_info, vcpu_info, vcpu_runstate info, …
  - Vmware hypervisor emulation
    - 'Vmware backdoor' (vmport)

Paravirtualized features in KVM

- PV features are:
  - Performance related.
  - Introducing some ‘unique’ capabilities unavailable/unneeded in bare hardware.
- It may be hard to notice the absence of a performance related feature
  - *Your guest could’ve run faster!*
- Normally, guest decides whether to use the feature or not
  - There is (almost) no reason to not give all performance related features to all guests.

Paravirtualized features in KVM

- Performance related PV features are usually implemented in KVM itself.

- PV features need to be ‘presented’ to guests:
  - The ‘usual’ interface for feature discovery is CPUID.
  - Userspace VMM (e.g. QEMU) has to:
    - Query KVM for the supported feature set.
    - Expose a subset to the guest by populating guest visible CPUIDs.

POP QUIZ!

- Can a KVM guest VM use a PV feature **not** exposed to it in CPUID but supported by KVM?

POP QUIZ!

- Can a KVM guest VM use a PV feature **not** exposed to it in CPUID but supported by KVM?
  - **Yes!**
  - Two recently added options to ‘harden’ the behavior:
    - KVM_CAP_ENFORCE_PV_FEATURE_CPUID - for ‘native’ KVM PV features
    - KVM_CAP_HYPERV_ENFORCE_CPUID - for Hyper-V PV features.
    - None of them are supported by QEMU atm.

Paravirtualized features in KVM

- PV features need to be ‘presented’ to guests
  - The ‘usual’ interface for feature discovery is CPUID
  - VMM (e.g. QEMU) has to:
    - Query KVM for the supported feature set
    - Expose a subset to the guest by populating guest visible CPUIDs

So flipping a bit is all it takes VMM to enable a new PV feature! Sounds really easy!

“Native” KVM PV feature enablement

“Example” PV feature

Interrupt based asynchronous page fault mechanism (`kvm-asyncpf-int`):

- Significantly improves throughput in memory-overcommitted environments.
- Merged into Linux-5.10, supported by QEMU-5.2.0+.
- Replaces legacy asynchronous page fault mechanism (`kvm-asyncpf`) which is now deprecated/disabled in KVM.

KVM PV feature enablement with QEMU

Example QEMU command line:

```
qemu-system-x86_64 -machine q35,accel=kvm -cpu Skylake-Server ....
```

- **Does this expose any ‘native’ KVM PV features to the guest?**

KVM PV feature enablement with QEMU

Example QEMU command line:

```
qemu-system-x86_64 -machine q35,accel=kvm -cpu Skylake-Server ....
```

- Does this expose any ‘native’ KVM PV features to the guest?
    - **Yes!**

KVM PV feature enablement with QEMU

target/i386/kvm/kvm-cpu.c (excerpt, shortened):

```
/*
 * KVM-specific features that are automatically added/removed
 * from cpudef models when KVM is enabled.
 *
 * NOTE: features can be enabled by default only if they were
 *       already available in the oldest kernel version supported
 *       by the KVM accelerator (see "OS requirements" section at
 *       docs/system/target-i386.rst)
 */
static PropValue kvm_default_props[] = {
    { "kvmclock", "on" },
    { "kvm-nopiodelay", "on" },
    { "kvm-asyncpf", "on" },
    { "kvm-steal-time", "on" },
    { "kvm-pv-eoi", "on" },
    { "kvmclock-stable-bit", "on" },
};
```

KVM PV feature enablement with QEMU

docs/system/target-i386.rst:

```
On x86_64 hosts, the default set of CPU features enabled by the KVM
accelerator require the host to be running Linux v4.5 or newer. Red Hat
Enterprise Linux 7 is also supported, since the required
functionality was backported.
```

```
$ git show v4.5
...
commit b562e44f507e863c6792946e4e1b1449fbbac85d (tag: v4.5)
Author: Linus Torvalds <torvalds@linux-foundation.org>
Date:   Sun Mar 13 21:28:54 2016 -0700

    Linux 4.5
```

KVM PV feature enablement with QEMU

- For a new PV feature in KVM
  - It’ll take roughly **5 years** before it can be enabled ‘by default’.
  - Manual enablement is possible:

```
qemu-system-x86_64 -machine q35,accel=kvm -cpu Skylake-Server,+kvm-asyncpf-int ...
```

  - “-cpu host” also enables everything (but generally it is not migratable):

```
qemu-system-x86_64 -machine q35,accel=kvm -cpu host ...
```

  - Enablement should also happen all the way up the stack (QEMU -> libvirt -> …).
  - Not all users are aware of the new feature and updating VM configs is not an easy task.
  - The result is low adoption of new PV features. Users don’t benefit from new PV features in KVM.
  - **Can we do better?**

KVM PV feature enablement with QEMU

- Why can’t we enable new PV features by default?
  - QEMU will not start on anything but the latest KVM

- Can we enable the feature conditionally, only if it is supported by the host?
  - The same QEMU command line should create the exact same configuration, this is crucial for live migration.
  - We could support migrating VMs to destination which supports a superset of PV features but not the other way around.

KVM PV feature enablement with QEMU

- Can we enable new features by default for new machine types?

```
qemu-system-x86_64 -machine q35,accel=kvm -cpu Skylake-Server …
```

Equals to (QEMU-6.1):

```
qemu-system-x86_64 -machine pc-q35-6.1,accel=kvm -cpu Skylake-Server …
```

  - It is expected that the latest machine type can be created even when the host has the oldest supported kernel (4.5 atm).
  - Changing this will force users to hardcode older machine types in their configurations.
  - This may reduce the adoption of **all** new features in QEMU, not only KVM PV.

KVM PV feature enablement with QEMU

- Can we have another “configuration dimension” (made up syntax)?

```
qemu-system-x86_64 -machine q35,accel=kvm -host-platform 5.14 -cpu …
```

  - Pros:
    - Clearly separates the required host version from machine type.
    - Can be used for all kernel-dependent features in QEMU (e.g. vhost, vfio, …).
  - Cons:
    - Users will still have to manually update their configurations.
    - Test matrix is going to explode ([‘machine type’ x ‘cpu type’] vs [‘machine type’ x ‘cpu type’ **x ‘host version’**]).
    - Unclear what to do with downstream kernels which may have features backported (`-host-platform rhel8.3` maybe?)

Additional ideas

- Maybe we need to solve the problem on a higher level?
  - Moving the issue up the stack doesn’t magically solve the problem.
  - There are multiple (even open source) higher level applications using QEMU/KVM stack.
  - All lower levels (e.g. QEMU, libvirt, …) should enable the feature before it is considered for a high level tool.
  - It is still a hard task to know all possible migration target hosts (and their kernel versions at the time of migration) in advance.

Additional ideas

- **Raise the minimum required KVM version when a new machine type is introduced, e.g.**
  - `pc-q35-6.1 requires Linux >= 5.9`
  - `pc-q35-6.2 requires Linux >= 5.10`
  - `....`

- **Add an option to limit migrations to the same or newer hosts, this will allow to enable all KVM PV features supported by the source host by default.**

```
qemu-system-x86_64 -machine q35,accel=kvm -migration same-or-newer-host ...
```

- **A new PV interface to revoke features from guests upon migration?**

- **Better document new KVM PV features when they are introduced**
  - **There’s no documentation for KVM PV features in QEMU currently. This is about to change.**

Hyper-V PV feature enablement

Hyper-V PV feature enablement with QEMU

- Unlike “native” KVM features nothing is enabled by default.
- Generally, users are advised to enable all currently supported Hyper-V enlightenments.
- Some features (‘hv-time’, ‘hv-stimer’,...) are not really optional as Windows’ performance without them is really poor.
  - Users google for them and hardcode years old suggestion to their configuration.
  - Real world adoption of new features stays low.
- Non-migratable ‘`hv-passthrough`’ CPU flag to enable everything supported by the host already exists.

Enabling all Hyper-V enlightenments by default

- An effort to introduce migratable ***'hv-default'*** CPU flag to enable all currently supported Hyper-V enlightenments was made. Problems were:
    - It is unclear what should get in the set. "Everything" would require a very recent kernel. Following QEMU's "Linux >= 4.5" support promise will leave too many features out.
    - There are Intel- and AMD- specific enlightenments in Hyper-V (e.g. already existing 'hv-evmcs'), it is unclear if these should be included in the *'hv-default'* set.

- Can be combined with the idea for elevated minimum required Linux version for newer machine types.

Summary and future work plans

Summary

- There is a problem with PV features enablement up the virtualization stack causing low adoption of the newly introduced KVM features.

- The problem is fundamentally caused by the architecture of the stack which consists of loosely coupled components.

- Live migration plays an important role in making the issue hard to resolve.

My future work plans:

- Finish this talk and hopefully get some feedback :-)
- Introducing “***-host-platform***” may be worth a try.
- Raising the required kernel version for new machine types in QEMU is an alternative approach.
- Hardening: enable KVM_CAP_ENFORCE_PV_FEATURE_CPUID and KVM_CAP_HYPERV_ENFORCE_CPUID in QEMU.
- Resume ***‘hv-default’*** work.

Thank you!

in linkedin.com/company/red-hat

▶ youtube.com/user/RedHatVideos

f facebook.com/redhatinc

twitter.com/RedHat

Red Hat