Nesting & Testing

KVM Forum 2019

Vitaly Kuznetsov <ykuznets@redhat.com>

Red Hat

About myself

- Focusing (mostly) on Linux kernel
- My areas of interest include:
  - Linux as guest on Hyper-V and Azure
  - Hyper-V Enlightenments in KVM
  - Running nested KVM on Hyper-V
  - Running nested Hyper-V on KVM

History of x86 nesting in KVM

commit cd232ad02f00286c3f8c9df30948da17212ef905
Author: Nadav Har'El <nyh@il.ibm.com>
Date: Wed May 25 23:10:33 2011 +0300

KVM: nVMX: Implement VMLAUNCH and VMRESUME

commit 3d6368ef580a4dff012960834bba4e28d3c1430c
Author: Alexander Graf <agraf@suse.de>
Date: Tue Nov 25 20:17:07 2008 +0100

KVM: SVM: Add VMRUN handler

Nesting in production

- Google Cloud Platform
- Oracle Cloud
- Microsoft Azure (KVM on Hyper-V)
- OpenStack testing at Red Hat
- ...

How do we test the feature?

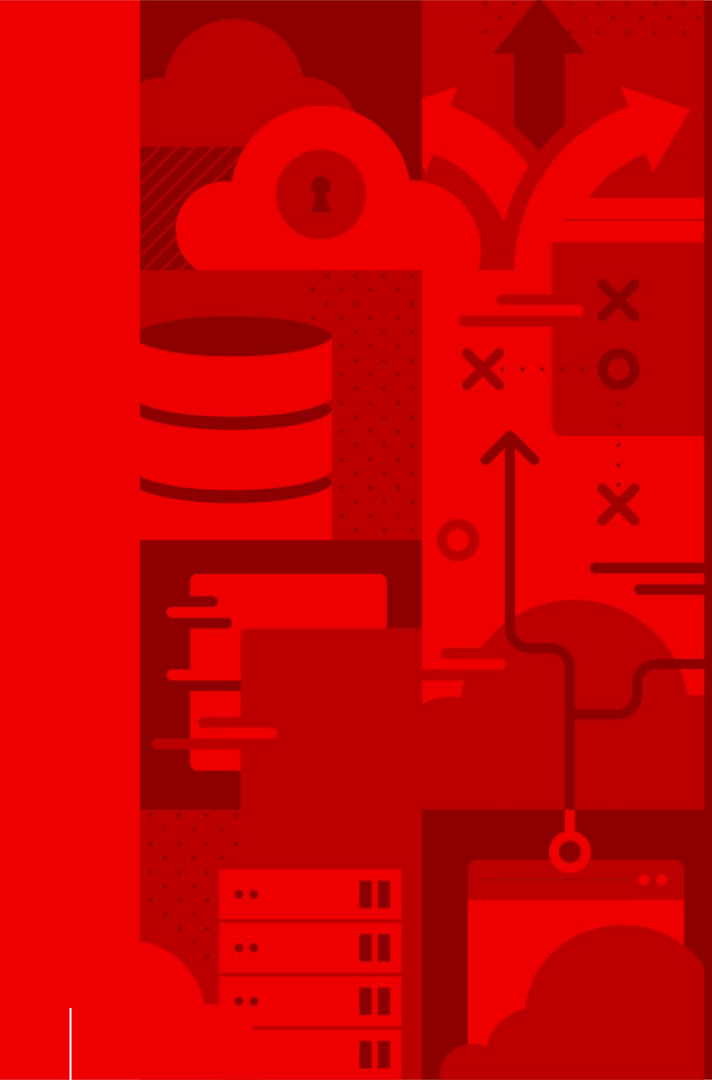
- By running dedicated nesting test suites:
  - VMX/SVM tests in kvm-unit-tests
  - Nested related tests in KVM selftests
- By running L1s and running hypervisor test suites there.
  - All tests in kvm-unit-tests
  - All tests in KVM selftests
- By running L1s+L2s and checking that everything works as expected

kvm-unit-tests

- git://git.kernel.org/pub/scm/virt/kvm/kvm-unit-tests.git
- Utilizes QEMU to run guests
- Pros:
  - We can use QEMU devices and features
  - SMP support
  - Mature codebase, rich library,...
- Cons:
  - We can't do what QEMU's not capable of (e.g. issue specially crafted or not yet supported ioctl)

KVM selftests

- ‘tools/testing/selftests/kvm’ in linux.git
- Every test is a ‘KVM userspace of its own’
- Pros:
  - Everything is possible (any ioctl, any guest code, ...)
  - Same git repository with KVM, patches can go in simultaneously
- Cons:
  - Requires low-level implementation for everything
  - Single concurrently running vCPU at this moment
  - Relatively young, limited library



Dedicated nesting testsuites

Red Hat

Running kvm-unit-tests on Intel hardware

```
FAIL vmx (18328 tests, 3 unexpected failures, 2 expected failures, 2 skipped)
PASS ept (7788 tests)
PASS vmx_eoi_bitmap_ioapic_scan (7 tests)
PASS vmx_hlt_with_rvi_test (7 tests)
PASS vmx_apicv_test (9239 tests)
PASS vmx_apic_passthrough_thread (8 tests)
PASS vmx_init_signal_test (8 tests)
PASS vmx_vmcs_shadow_test (142218 tests)
```

Running kvm-unit-tests on AMD hardware

```
PASS svm (24 tests)
```

VMX

- “Correctness”
  - EPT: all bits/all levels, access, misconfig/violation
    - 7788 assertions
  - INVVPID: validity, exceptions, no functional testing
    - 1562 assertions
  - VMX controls (vmlaunch success/failure):
    - Control MSRs: 329 assertions
    - I/O, MSR bitmaps: 817 assertions
    - APIC/vAPIC, Posted interrupts, vTPR, NMI/vNMI: 2856 assertions
    - PML: 317 assertions
    - EPT: 160 assertions
    - MSR-store/MSR-load: 380 assertions
    - Invalid event injection: 246 assertions

VMX

- “Correctness” (continued)
  - Host state area (vmlaunch success/failure)
    - 1006 assertions
  - Guest state area (vmlaunch success/failure)
    - 994 assertions
  - APIC tests (xAPIC/x2APIC, TPR shadow, all registers)
    - 9239 assertions
  - Shadow VMCS (all VMCS fields)
    - 142218 assertions

VMX

- “Functional”
  - ‘Basic’ VMX (launch/resume, capabilities MSRs, PAT/EFER control fields): 55 assertions
  - CR shadowing: 12 assertions
  - Preemption timer: 5 assertions
  - I/O bitmap: 15 assertions
  - Instruction intercept: 38 assertions
  - EPT: 36 assertions
  - PML: 2 assertions
  - VM-Entry in MOVSS shadow: 5 assertions
  - INIT signal: 8 assertions
  - Store TSC: 2 assertions
  - Pending event: 2 assertions

VMX

- “Regression”
  - #NM reflection: 2 assertions
  - #DB tests: 35 assertions
  - CR load: 3 assertions
  - EOI-exit-bitmap IOAPIC scan: 1 assertion
  - IOAPIC & LAPIC passthrough: 8 assertions
  - HLT with interrupt in RVI: 5 assertions

SVM

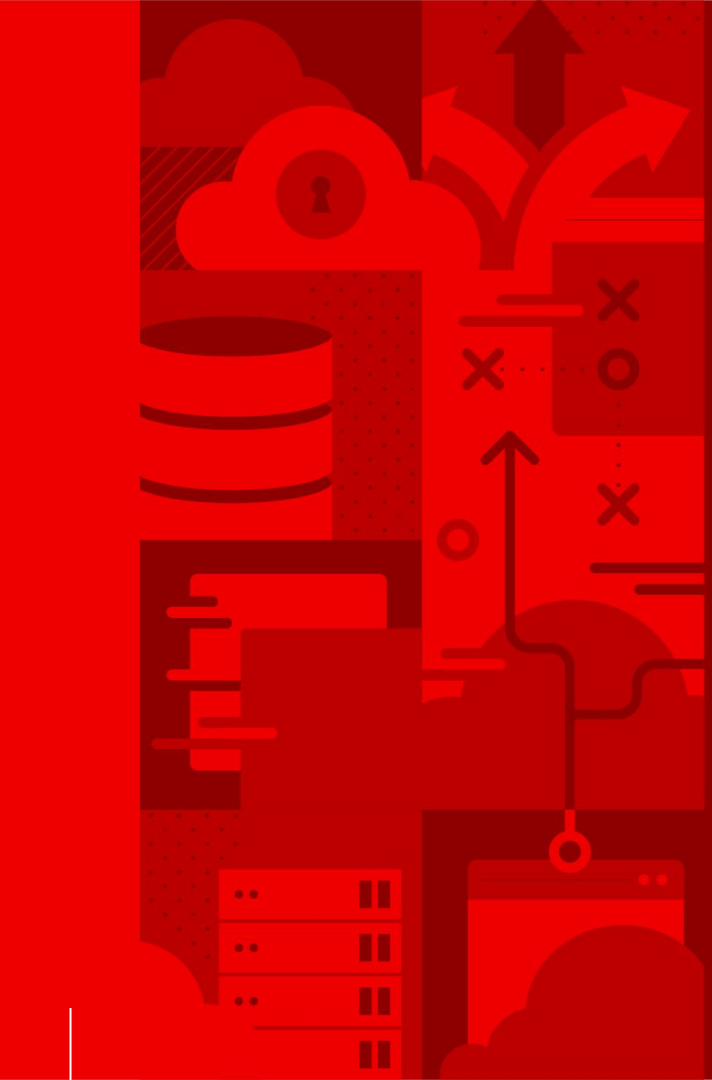
- Basic VMRUN
- IOIO
- Intercepts
  - VMRUN
  - CR3
  - DR
  - MSRs
  - Selective CR0
- Next RIP (rdtsc)
- Mode switch
- ASID == 0
- Latency

SVM

- NPT
  - NX bit
  - USER bit
  - WRITABLE bit (PT walk/page access)
  - RESERVED bit (PT walk/page access)

KVM selftests

- 16 tests total, 5 VMX-only tests, 0 SVM-only tests (no SVM library)
- VMX-only tests:
  - Enlightened VMCS
  - Close while nested
  - Dirty log
  - Set nested state
  - TSC adjust
  - SMM (with VMX enabled)



Running KVM testsuites in L1

Red Hat

Running KVM testsuites in L1

- Pros:
  - Much richer L2s
  - Test code reuse (what was running in L1 now runs in L2)
  - Allow us to test 3 level nesting!
- Cons:
  - We don't test corner cases as L1 is a 'sane VM'
  - One extra step during development

- *… Can be the only possible option (e.g. for KVM on Hyper-V) …*

Using nested testing as a tool

- Tests usually run with a fixed set of CPU features tied to the host (like '-cpu host')
  - No options for KVM selftests
- Making sure things work on with different CPUs require testing on different hosts
- We can emulate different CPUs with QEMU and run tests in L1!
  - This will test both L0's KVM nesting capabilities and L1's KVM acting correctly on the specified 'hardware'
  - QEMU recently added options for fine-grained VMX capabilities setting ('vmx-*' features)

Typical development workflow:

1. Write a patch for KVM/QEMU, write a test
2. Compile, install
3. Run kvm-unit-tests, selftests
   - This involves dedicated VMX/SVM testsuites
4. No regressions -> Submit!

Typical development workflow (improved):

1. Write a patch for KVM/QEMU, write a test
2. Compile, install
3. Run kvm-unit-tests, selftests
   - This involves dedicated VMX/SVM testsuites
4. **Deploy artifacts on the testing VM**
   - **If tests were altered deploy them too**
5. **Run kvm-unit-tests, selftests in the VM**
   - **May make sense to try different L1 configs (CPU features, hugepages, …)**
6. No regressions -> Submit!

Share host's filesystem with L1 to avoid the hassle

- I use virtme (https://github.com/amluto/virtme) as a QEMU wrapper

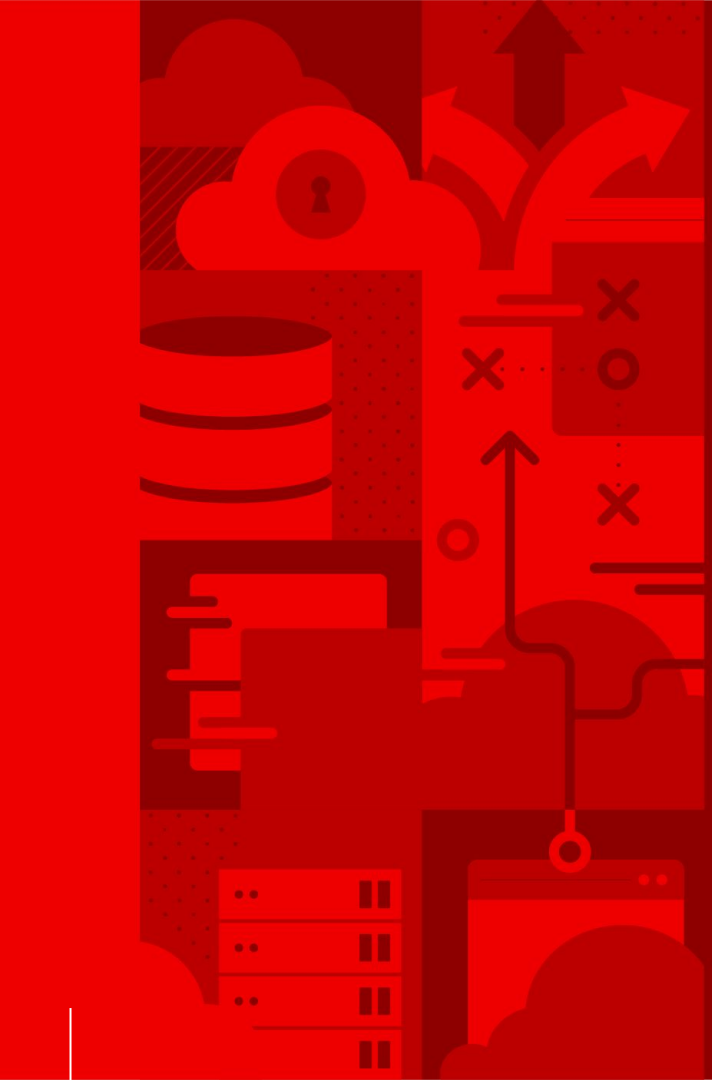
Example: run kvm-unit-tests with L1 backed by huge pages:

```
# ~/virtme/virtme-run --memory 4096 --installed-kernel --rwdir `pwd` --script-sh
"cd `pwd` && ./run_tests.sh" --qemu-opts -smp 4 -mem-path /dev/hugepages/

PASS apic-split (53 tests)
PASS ioapic-split (19 tests)
PASS apic (53 tests)
PASS ioapic (19 tests)
...
```

How can we embed something like this into standard development workflow?

- Promote usage of existing tools
  - Like “virtme is awesome! :-)”
- Pick a tool and add a dependency to kvm-unit-tests
  - ./run_tests.sh && ./run_tests_nested.sh
- Add a [QEMU] wrapper to kvm-unit-tests
- … do something else?
- … and what about selftests?

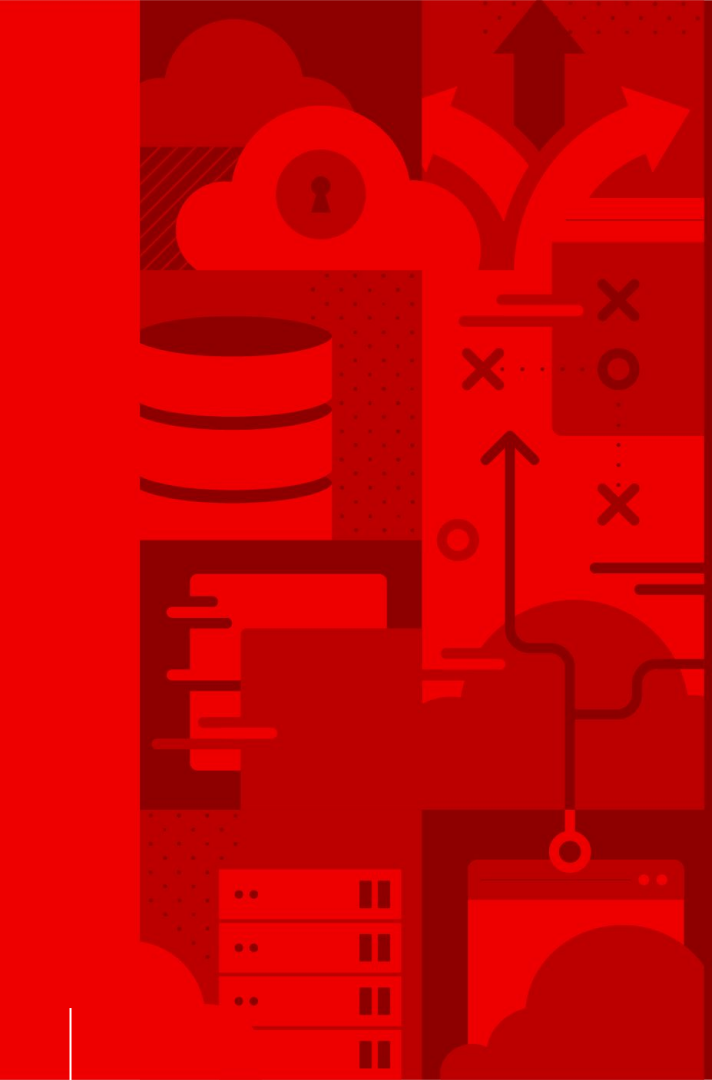


My personal testing wishlist

Red Hat

Would appreciate some love...

- SVM testing in kvm-unit-tests
  - NPT, VMCB controls, AVIC, ...
- SVM library for KVM selftests
- More event injections (both SVM and VMX)
- Enlightened VMCS support in kvm-unit-tests
- SMM with nesting tests (selftest, kvm-unit-tests?)
- Functional tests for translation buffers invalidation
- Hyper-V enlightenments tests (PV TLB flush, PV IPI, ...)



Credits

Red Hat

kvm-unit-tests

$ git log --no-merges --since 2018-10-27 --pretty=short | git shortlog -s -n

38 Nadav Amit
30 Janosch Frank
29 Krish Sadhukhan
29 Sean Christopherson
20 Paolo Bonzini
9 Thomas Huth
8 Liran Alon
7 Marc Orr
6 Alexandru Elisei
6 Bill Wendling
6 Vitaly Kuznetsov
5 Jim Mattson
4 Andrew Jones
3 Tambe, William
2 David Gibson
2 David Hildenbrand
2 Oliver Upton
2 Stefan Raspl
2 Suraj Jitindar Singh
1 Andre Przywara
1 Cathy Avery
1 Christian Borntraeger
1 Christoffer Dall
1 Evgeny Yakovlev
1 Haozhong Zhang
1 Peter Xu
1 Sergey Bronnikov
1 Wanpeng Li

KVM selftests

$ git log --no-merges --since 2018-10-27 --pretty=short tools/testing/selftests/kvm/ | git shortlog -s -n

```
16  Paolo Bonzini
16  Thomas Huth
15  Vitaly Kuznetsov
14  Andrew Jones
 5  Aaron Lewis
 5  Peter Xu
 4  Sean Christopherson
 2  Liran Alon
 2  Shuah Khan
 2  Thomas Gleixner
 1  Ben Gardon
 1  Christian Borntraeger
 1  Dan Carpenter
 1  Naresh Kamboju
```

Thank you!

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat