redhat.

A KVM-unit-tests and KVM selftests update for aarch64

Eric Auger
KVM Forum 2020

Overview

- Introduction
- Test Frameworks overview
- Test Code Base on aarch64
- Advertise the frameworks
  - Examples
  - Highlights
  - Lessons learnt on some kvm-unit-tests developments
  - Develop tests on models
  - Test migration with kvm-unit-tests
- Conclusion

Introduction

- KVM/arm64
  - now used in production systems
  - Some areas have stabilized (VGIC, ... \o/)
  - A significant kernel code base
  - Lots of traffic on the ML
    - Code reworks (page table code, mitigations ...)
    - Many new ARM v8.++ feature kernel developments without HW
- KVM unit test frameworks
  - few unitary tests are contributed
  - New features generally do not come with unitary tests
  - Unitary tests generally come too late, do not have significant coverage
  - Very few bug reproducers
- Time for introspection?
  - Why? How to improve?

In a nutshell (1/2)

	KVM selftests	kvm-unit-tests
When	Since 2018, ARM support since 2018	Before 2010 [1], ARM support since 2014
Where	in the linux tree [2]	In a separate repo [3]
Tester writes	KVM user API function calls + guest code (C/asm @ EL1)	Guest code only (C/asm @EL1/EL2 [4])
Dependency	none	qemu (kvm/tcg), kvmtool, ...
Framework brings	- KVM API wrappers & helpers - gva/gpa allocation/mapping and gva/gpa/hva translation - host/guest basic sync	- Guest code: - basic OS services (vectors, SMP, UART, ...) - few libc functions - test specific utilities (error reporting) - Set of bash scripts (config, grouping, migration)

[1] existed before but not in its own repo
[2] tools/testing/selftests/kvm
[3] https://gitlab.com/kvm-unit-tests/kvm-unit-tests
[4] [RFC PATCH v3 0/7] arm64: Run at EL2

In a nutshell (2/2)

	KVM selftests	**kvm-unit-tests**
Very adapted to	- Tests with simple guest code - Existing & new KVM user API testing - Init sequence Testing - Nested testing	- tests with more complex guest code (interrupts, timers, dt ...) - qemu/kvmtool (KVM/TCG) testing - in-kernel emulated devices testing - microbenches - migration testing (with QEMU) - nested testing

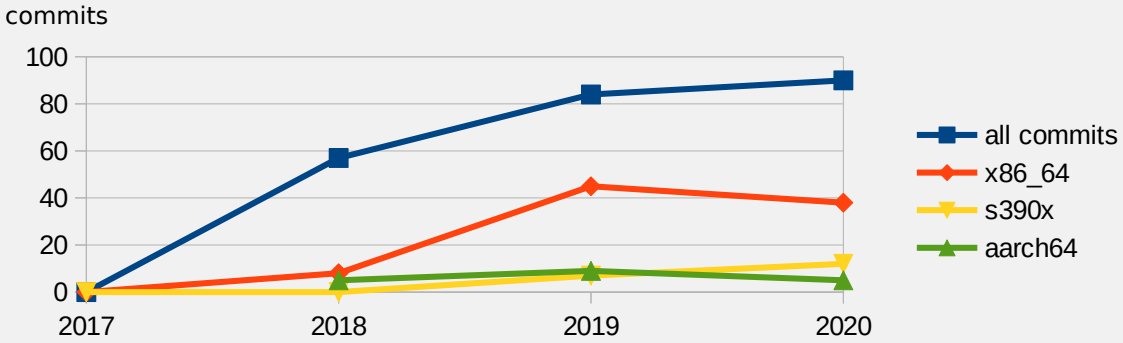
Facts about aarch64 tests

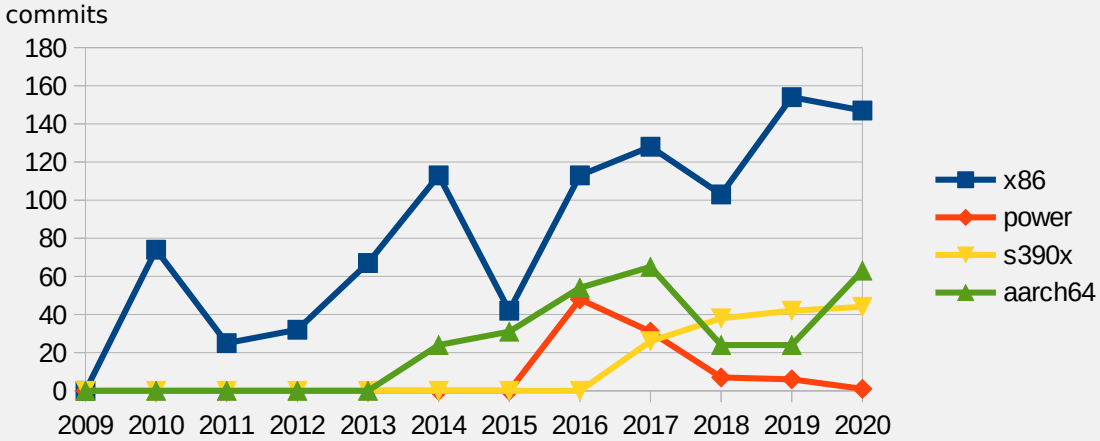
kvm selftests

- No aarch64 specific tests!
- Only framework and few tests shared with other archs (stolen time, max vcpu, max memslots, dirty log test, demand paging)

kvm-unit-tests

SMP tests
GIC MMIO/IPI
ITS MSI controller
PMU v3
PL031
cache tests
vTIMER/pTIMER
microbenchs
PSCI





git log --pretty=oneline --after='YYYY-01-01' --before='YYYY-12-31' -- lib/x86_64 -- lib/x86_64 -- arm | wc -l

KVM selftests (1/2)
Steal-time Example (aarch64/x86)

Test that stolen time value reported to the guest by KVM matches procfs schedstat

- host:
  - KVM USER API calls (host):
    - Create a VM
    - Create a memslot
    - Create a VM GVA/GPA mapping
    - Creates VPUS
    - Check the stolen time capability is supported (KVM_HAS_DEVICE_ATTR)
    - Configure the PV_TIME IPA (KVM_SET_DEVICE_ATTR)
    - kvm_run's
  - Set vcpu affininity, Spawn a thread on the same pCPU that steals time to the VM
  - Read guest stolen time and compare it against procfs value
- Guest:
  - smccc call to read the stolen time and write it at some place readable by the host (GVA/GPA/HVA)

KVM selftests (2/2)

- Highlights:
  - No need to wait for userspace integration (QEMU/kvmtool)
  - Testing efforts easily visible from the kernel community
  - Good way to learn the KVM user API
  - Very easy setup & fast iterations

- Needs
  - Missing aarch64 version of few common tests (memslot related tests)
  - No aarch64 specific tests yet!
    - New KVM APIs could be unitary tested here, before userspace integration
    - Need fuzzing of the KVM API (Marc's input)
      - Ill-behaved userspace
      - Ill-behaved guest, trying to use features the host does not support
      - looming nested virt
  - Write some doc on the framework?

kvm-unit-tests (1/2)
Examples and lessons learnt

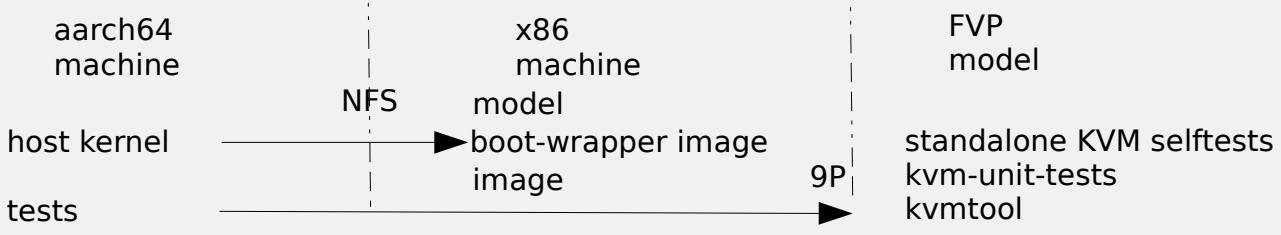
- ITS MSI controller
  - programming required a lot of logic for translation table setup ➡ ~ rewrote a driver (?!)
  - HW device ignores most of the wrong programming, not tester friendly
  - Came too late in the development process. Regression tests now
  - Opportunity to enable migration testing though (bug reproducer)
- PMU
  - low level (register writing) ➡ much more efficient
  - Incremental efforts based on cycle counter existing test
  - Very interested since you get a fine grain control as opposed to the perf layer
  - In sync with chain counter new support (found bugs!)
  - Those tests pave the way to ARMv8.5-PMU 64b counter support
  - Discovered some tests were not passing on some HW
  - Was also used for QEMU TCG PMU event counter support

kvm-unit-tests (2/2)

- Highlights
  - Also test a userspace
  - Focus on guest code
  - Automation (config, grouping, migration)
  - Errata framework (adapt the test if the host has a specific commit) which helps CI integration
- Needs
  - Improve the coverage of existing tests
  - fuzzing: ill-behaved guest
  - test vcpus features
  - nested
  - Better Advertise the framework from linux?
    - “Reported-by: kvm-unit-tests” on top of usual R-b's [Alexandru's input]

Develop Tests on FVP Model (1/3)

- How to write tests without HW and keep up the pace with KVM developments?
- Free-of-charge models: foundation model and FVP base model
- A good blog to start with
  - https://www.thegoodpenguin.co.uk/blog/booting-linux-with-fvp (Andrew Murray)
- Most difficult is
  - To find a good image (light but rich enough to compile the userspace)
  - Find/hack the device tree (ARM Trusted Firmware) ??!!
  - Get familiar with the model options (virtio_net, 9p, has_*, ...)



Develop tests on FVP model (2/3) kvm-unit-tests

- Running QEMU on ARM model is terrible!
  - Need to continue efforts shrinking the executable [1]

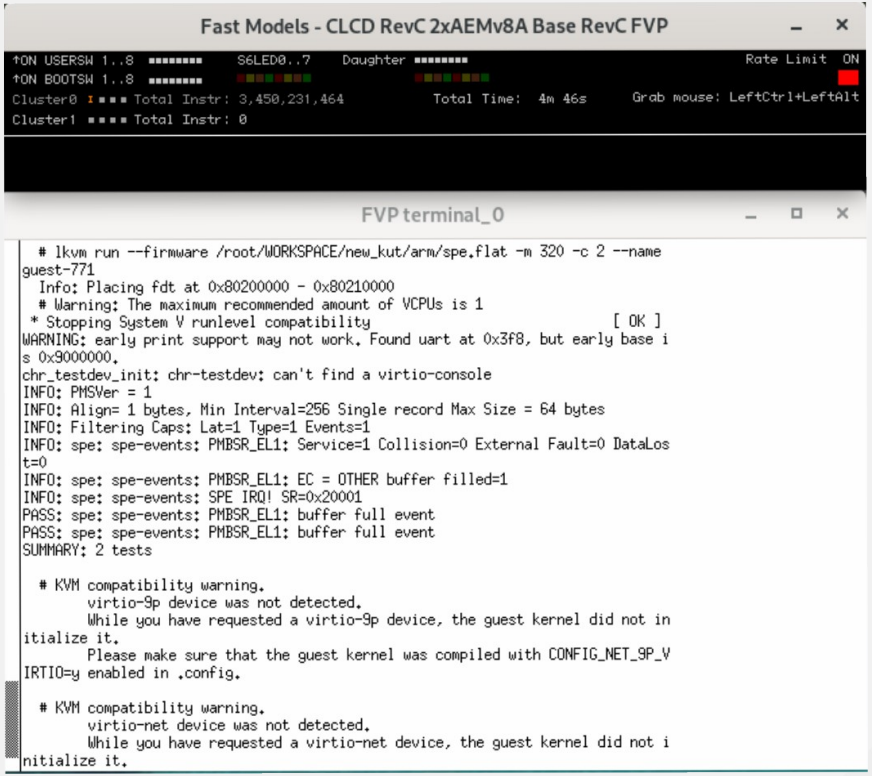
- Develop unit tests using kvmtool [2]
  - Statistical Profiling Extension Test RFC [3] was developed on model

- Not possible to test QEMU integration or migration though!
- kvmtool integration lacks automation (integration with arm/unittests.cfg)

[1] among others, [PATCH v4 00/12] Support disabling TCG on ARM (part 2)
[2] kvm-unit-tests PATCH v3 0/5] arm/arm64: Add support for running under kvmtool, Feb 2019
[3] [kvm-unit-tests RFC 0/4] KVM: arm64: Statistical Profiling Extension Tests, Sept 2020

Develop Tests on FVP Model (3/3)

```
FVP_Base_RevC-2xAEMv8A -C cache_state_modelled=0 \
-C bp.refcounter.non_arch_start_at_default=1 \
-C bp.secure_memory=false \
-C bp.virtio_net.enabled=1 \
-C bp.virtio_net.hostbridge.userNetworking=1 \
-C bp.virtiop9device.root_path=/home/augere/GIGA_VM/9P \
-C bp.virtiop9device.mount_tag=FW \
-C bp.virtioblockdevice.image_path=image.raw \
-C cluster0.has_arm_v8-1=1 -C cluster0.has_arm_v8-2=1 \
-C cluster0.has_statistical_profiling=1 \
-C cluster0.pmu-num_counters=8 \
-C cluster0.pmu_has_chain_event=1 \
-C cluster0.has_amu=1 \
-C cluster0.NUM_CORES=4 \
-a "cluster0.*=base-image/linux-system.axf" \
--disable-analytics
```



mount -t 9p -o trans=virtio,version=9p2000.L FW WORKSPACE

~/kvmtool/lkvm run --cpus 2 --spe --pmu --console serial
--params "spe-events" --irqchip gicv3 --firmware ~/WORKSPACE/new_kut/arm/spe.flat

Testing QEMU Migration with kvm-unit-tests

arm: unittests.cfg excerpt

```
[its-migration]
file = gic.flat
smp = $MAX_SMP
accel = kvm
extra_params = -machine gic-version=3 -append 'its-migration'
groups = its migration
arch = arm64
```

- The test must belong to the **migration** group in unittests.cfg
- The framework launches both source and destination qemu
- Guest code initiates the migration by outputting the "migrate" keyword: puts("**migrate**\n");
- Guest code then waits for the migration completion by calling blocking **getchar**()
- Once the migration is over, the run script provides the stdin input which unblocks the guest
- Following guest code is executed on the destination and can check the state is consistent

Conclusions

- Two really nice test frameworks completely underused on ARM
- Test Development on ARM model/TCG is feasible (not with QEMU though): develop tests on time!
- Fast iterations
- CI integrated
- Fast & nice way to learn and contribute!
- Crying needs: bug reproducers, fuzzing, ...
- Make QEMU lighter to be runnable on ARM FVP model

Feel free to contact me at eric.auger@redhat.com!

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHat

youtube.com/user/RedHatVideos