kvm-unit-tests: When "KVM" doesn't mean KVM

Building unit tests as EFI apps

Andrew Jones
drjones@redhat.com

Red Hat

Outline:

Quick kvm-unit-tests introduction

Current status of non-KVM targets

Motivation for building tests as EFI apps

Current status of EFI app targets

EFI app target implementation

Wrap-up

Quick kvm-unit-tests introduction

Beware of bugs in the above code; I have only proved it correct, not tried it.

— Donald Knuth

What is kvm-unit-tests?

A test framework and collection of unit tests for KVM

How does it test KVM?

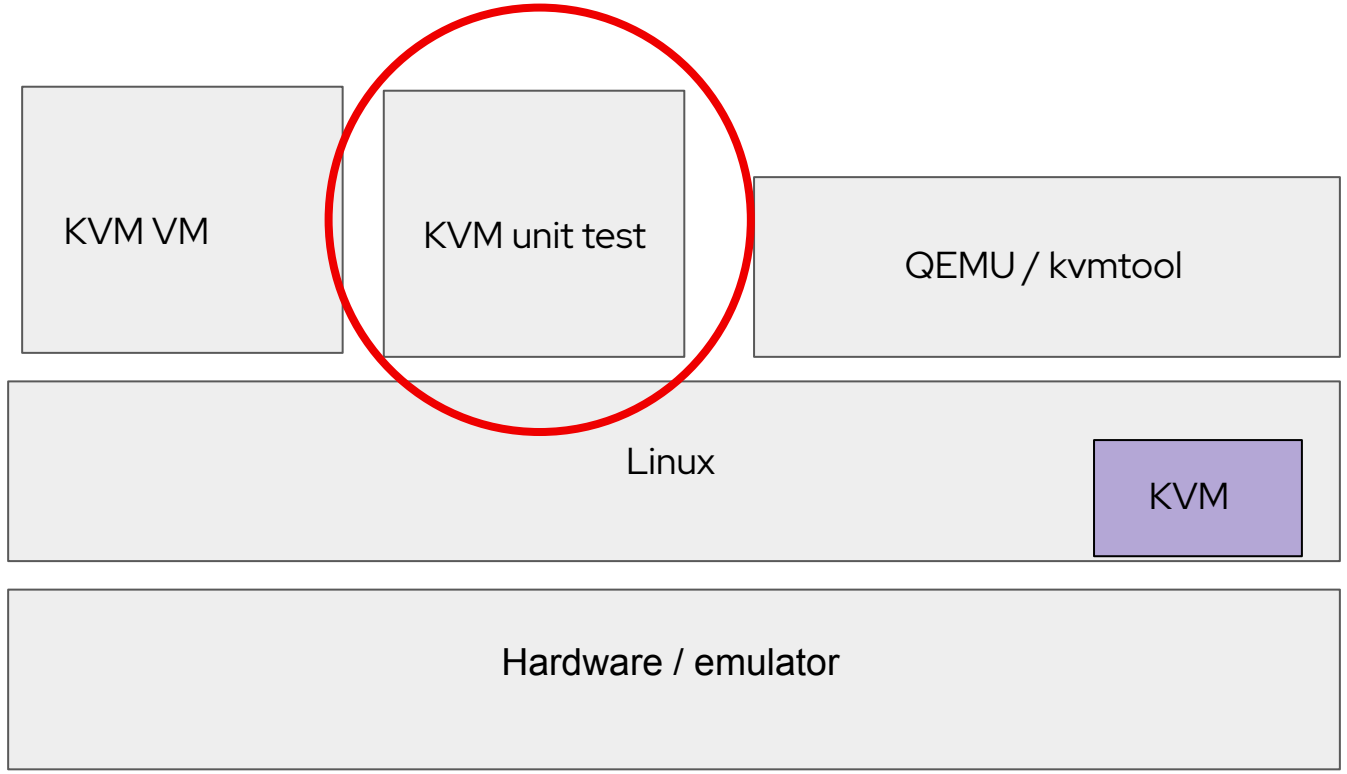
Runs tiny guests which execute instructions generating traps to KVM and exits to QEMU

What's a tiny guest look like to the test developer?

main() with a mixed API of kernel (irq_enable, virt_to_phys, ...) and libc (printf, malloc, ...) functions

Source:
https://www.linux-kvm.org/page/KVM-unit-tests
https://gitlab.com/kvm-unit-tests/kvm-unit-tests.git

Quick kvm-unit-tests introduction (cont.)



Current status of non-KVM targets

QEMU accelerators

TCG, Hypervisor.framework (macOS HVF), Windows Hypervisor Platform (WHPX)

s390x

z/VM, LPAR hypervisors

x86 bare-metal and VMware

The unit tests are launched from grub and use environment variables in place of some hardware discovery

Source:
https://gitlab.com/kvm-unit-tests/kvm-unit-tests/-/blob/master/README.macOS.md
KVM Forum 2019 "Moving s390 kvm-unit-tests up the stack"
x86: Support environments without test-devices

Motivation for building tests as EFI apps

The good thing about standards is that there are so many to choose from.

— Andrew S. Tanenbaum

One portage, many targets (theoretically)

Environment variables can manage configuration differences

EFI is a relatively easy target

No need to determine the memory map nor implement reset

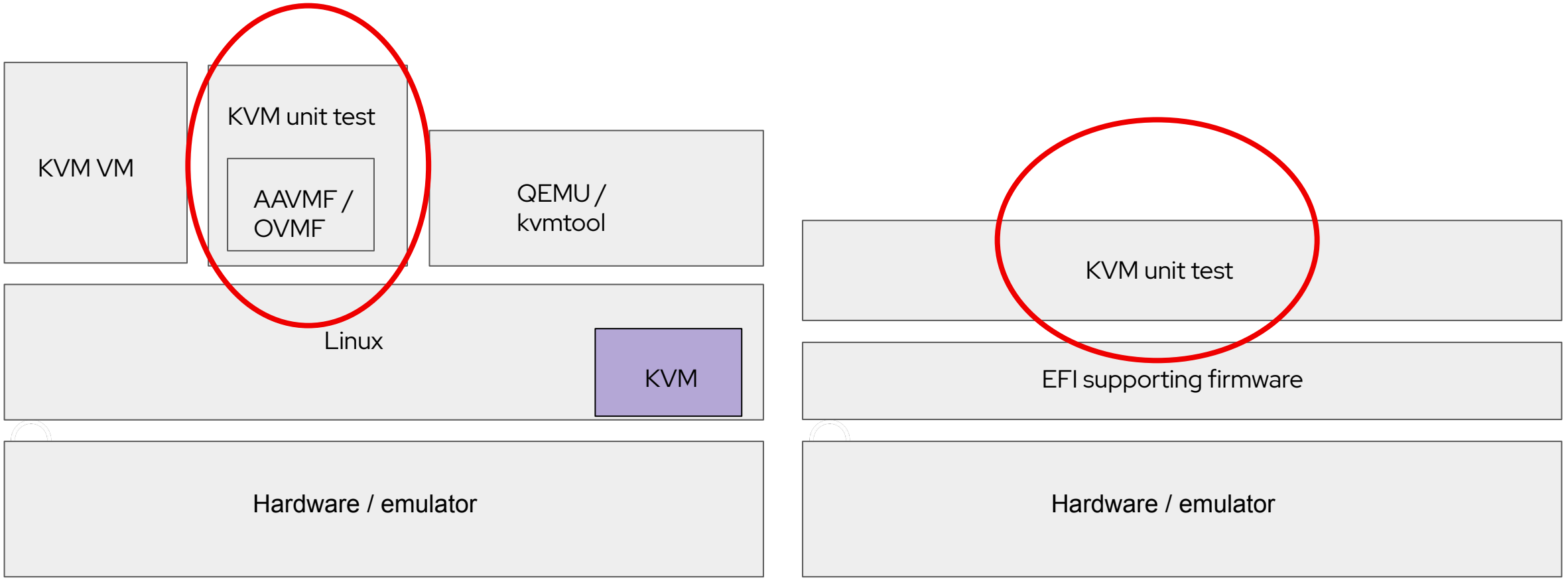
May enable faster KVM unit test development

Emulators are used when hardware isn't available. It's faster to only boot firmware.

Get yet another test target

Can now test firmware's EFI implementations too

Motivation for building tests as EFI apps (cont.)



Red Hat

Current status of EFI app targets

- Simply configure with a new switch, *--target-efi*, and run make
- So far AArch64 only, works with QEMU and AAVMF, but that's not overly exciting...
- It's a work-in-progress for bare-metal (tested on AMD Seattle)
- Plan to try x86 tests with QEMU and OVMF next
- Second stop for x86 tests will be VirtualBox, as VirtualBox also supports booting VMs with OVMF

Links::
https://github.com/rhdrjones/kvm-unit-tests/commits/target-efi

EFI app target implementation

What needs to be added?

Arch-neutral EFI support code, linked with gnu-efi

What needs to be removed or bypassed?

As little as possible, but target-specific code must go

Other changes

Driver and paging setup improvements

EFI app target implementation (cont.)

What needs to be added?

- gnu-efi is an EFI development environment for the GNU toolchain
- Requires an odd build process; compile and link as a shared library and then objcopy select sections
- All gnu-efi apps start in efi_main(), which is implemented by the app
- A goal is to share one efi_main implementation among all architectures and tests
- kvm-unit-tests efi_main uses the gnu-efi API and direct UEFI calls to prepare the unit test for launch and then exit boot services
- exit() for EFI app calls the UEFI reset runtime service

EFI app target implementation (cont.)

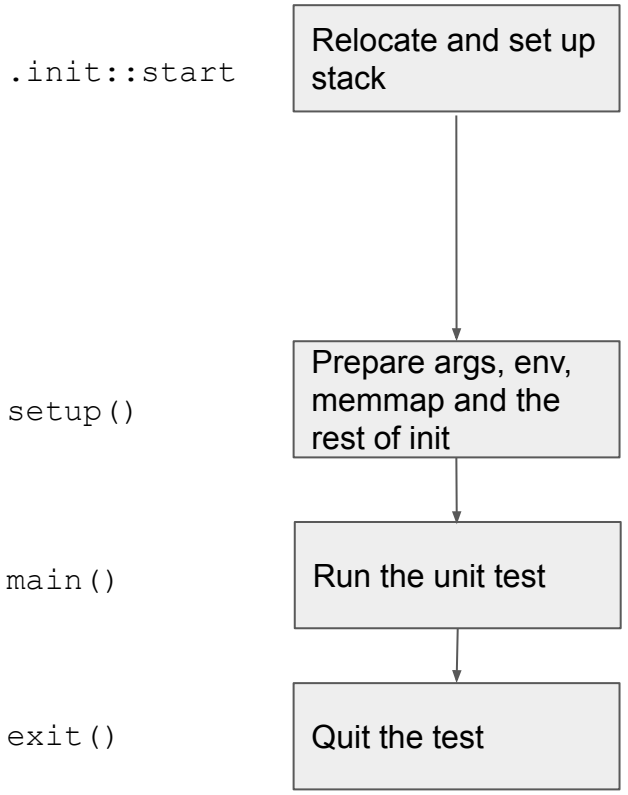
What needs to be removed or bypassed?

- gnu-efi provides its own linker script
- Need to remove or replace code referencing symbols defined in the original / default linker script (or rename them, e.g. etext →_etext)
- Goal is to push all original / default target assumptions into its linker script and the initial start assembly code (all code in the .init section)
- Then, when building as an EFI app, we only need to #ifdef out the .init section
- Any initialization common to the original / default target and to the EFI app target should be done in setup()

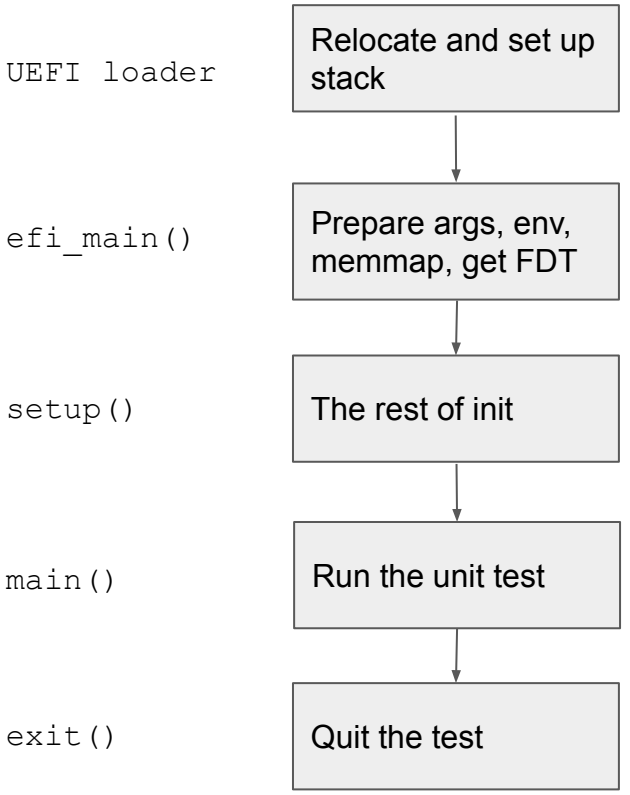
EFI app target implementation (cont.)

Comparison of the target startups

Original / default target



EFI app target



Red Hat

EFI app target implementation (cont.)

Comparison of the target startups

Original / default target

- .init::start relocates and sets up the stack
- DT and multiboot info come from QEMU
- Command line args extracted from DT or multiboot info
- Env provided by an initrd
- Memory map comes from DT or multiboot info, or is hardcoded
- setup() called with MMU off and no devices initialized

EFI app target

- EFI app is relocated and given a stack by the UEFI loader
- DT comes from a DTB file on the EFI FS (we may need more ACPI support for x86)
- Command line args extracted gnu-efi call
- Env extracted with a UEFI runtime service call
- Memory map comes from gnu-efi call
- setup() called with MMU on and some devices initialized

EFI app target implementation (cont.)

Other changes

- Drivers should do device reset before init
- Improvements to device drivers, e.g. proper UART FIFO usage
- Generalize paging setup to work for the EFI memory map as well as the original / default memory map
- On bare-metal a '\r' → '\r\n' hack may be necessary. It shouldn't hurt to always do it, but that could be a test suite config option
- Support choice as to what to do when starting with the MMU enabled and when exiting the unit test (environment variables and auxinfo)

EFI app target implementation (cont.)

Current and future challenges to finish the PoC's

- AArch64 and maybe x86: Implement device reset before init and possibly other driver improvements
- AArch64: Support 4K pages (currently only supports 64K)
- x86: Some hardware descriptions may require parsing ACPI tables

Wrap-up

- kvm-unit-tests is already targeting more than just KVM
- Adding an EFI app build to kvm-unit-tests will help to further expand the set of test targets
- An EFI app build may also be useful for faster KVM unit test development when emulators are used, since only firmware boots
- A PoC for AArch64 is pretty far along, but more platform assumptions must be removed to run all tests on bare-metal
- A PoC for x86 is planned. It's expected to have a different set of challenges to those of AArch64.

Thank you

https://www.linux-kvm.org/page/KVM-unit-tests
https://gitlab.com/kvm-unit-tests/kvm-unit-tests.git
https://github.com/rhdrjones/kvm-unit-tests/commits/target-efi

Andrew Jones <drjones@redhat.com>

Red Hat