KVM-UNIT-TESTS

PAST, PRESENT, AND FUTURE

Drew Jones

KVM Forum 2015

What's a KVM unit test?

- Guest kernel with lots of shortcuts
  - Minimal system initialization (may vary test to test)
  - Hard-coded page tables, hard-coded I/O addresses, …
- Test is written in C and assembler
  - libc API (only a tiny bit)
  - kvm-unit-tests specific API

kvm-unit-tests: Outline

Past: Once upon a time…

Present:

- Recent additions to the framework
- How to write and run tests

Future:

- Work in progress

Closing: The end

History: a test guest

- kvm-unit-tests is as old as KVM
  - Early KVM development was rapid prototyping
    - The unit test was a guest
- As KVM evolved its test guest evolved
  - Started with just a handful of instructions
  - Instructions added along with KVM support

```
        mov     $0, %al
        mov     $10000, %ebx
1:      mov     %rbx, %rcx
2:      loop    2b
        out     %al, $0x80
        inc     %al
        add     $10000, %rbx
        jmp     1b
```

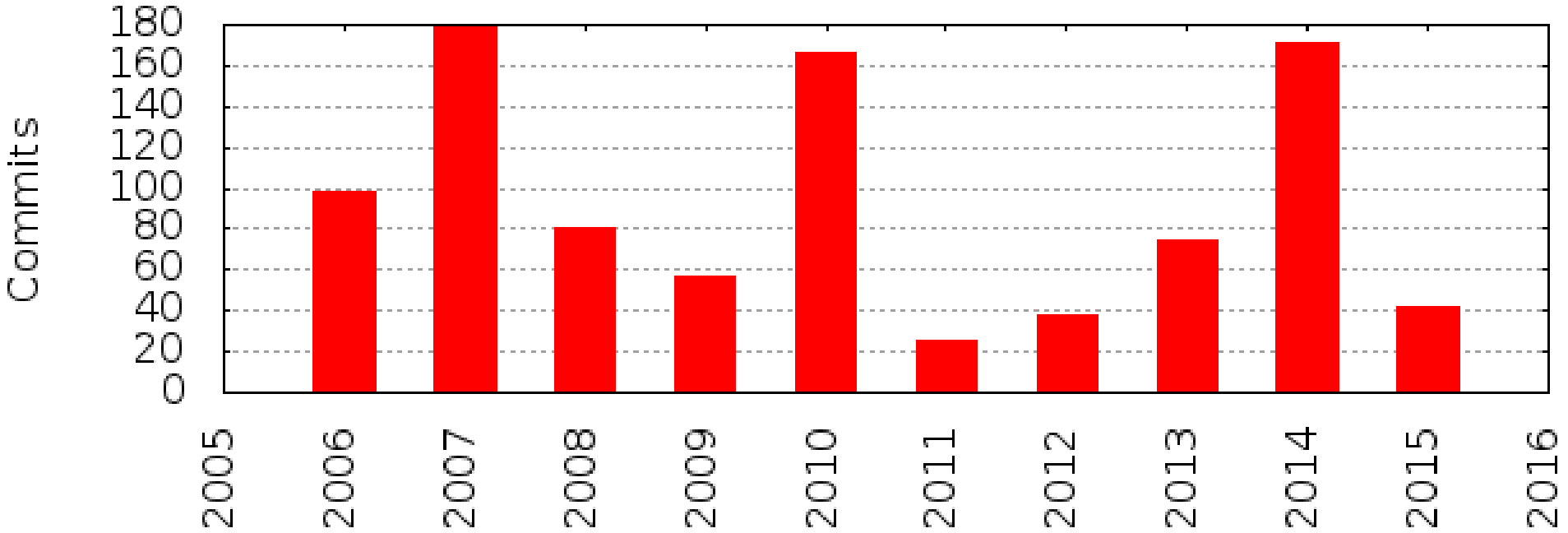
History: a test framework

- Share common code among unit tests
- Share code location among architectures
  - The location changed a few times...
    - `qemu-kvm.git`

      `kvm/user → kvm/user/test → kvm/test`
    - `kvm-unit-tests.git`

```
Directory structure:
.:              configure script, top-level Makefile, and run_tests.sh
./config:       collection of architecture dependent makefiles
./lib:          general architecture neutral services for the tests
./lib/<ARCH>:   architecture dependent services for the tests
./<ARCH>:       the sources of the tests and the created objects/images
```

Project activity



ARM support

- Started with the typical kvm-unit-tests style (taking shortcuts)
  - e.g. converted DT to simple address table during build
  - Why code it, when you can ***hard***-code it…
- Eventually cleaner, Linux-like approaches
  - libfdt and lib/devicetree
  - Linux-style asm-offsets generation
  - Even a tiny bit of virtio support

ARM support cont.

- Both ARMv7 and ARMv8 supported
  - Share as much code as possible
  - Linux's asm symlink for #include <asm/foo.h>
- Documentation?
  - Also Linux-like, i.e. grep
  - Use existing tests, e.g. arm/selftest.c, as templates

Example ARM unit test

```
int main(int ac, char **av)
{
    int cpu;

    for_each_present_cpu(cpu) {
        if (cpu == 0)
            continue;
        smp_boot_secondary(cpu, read_mpidr);
    }
    read_mpidr();

    while (!cpumask_full(&ready))
        cpu_relax();

    report("check_mpidrs(nr_cpus == %d)\n",
                check_mpidrs(ac, av), nr_cpus);

    return report_summary();
}
```

Example ARM unit test cont.

```
static void read_mpidr(void)
{
    int cpu = smp_processor_id();
    unsigned long mpidr;

#ifdef __arm__
    asm volatile("mrc p15, 0, %0, c0, c0, 5" : "=r" (mpidr));
#else
    asm volatile("mrs %0, mpidr_el1" : "=r" (mpidr));
#endif
    printf("cpu%d: %lx\n", cpu, mpidr);

    mpidrs[cpu] = mpidr;

    cpumask_set_cpu(cpu, &ready);
    if (cpu != 0)
        halt();
}
```

ARM: API

I/O

```
ioremap,
readl,writel,
…
```

MMU

```
mmu_enable_idmap,
mmu_set_range_ptes,
…
```

SMP

```
smp_boot_secondary,
for_each_cpu,
smp_processor_id,
…
```

ARM: API

User Mode

```
start_usr,
is_user,
…
```

Primitives

```
spin_lock,
smp_mb,
test_and_set_bit,
…
```

Vectors

```
install_exception_handler,
show_regs,
…
```

ARM: API

Device Tree

```
dt_device_find_compatible,
dt_get_default_console_node,
dt_get_bootargs,
…
```

libc

```
printf,
strcpy,
malloc,
exit,
…
```

exit

- x86 doesn't use ACPI
  - ACPI support wasn't available at first
  - Instead exit code is written to a debug exit I/O port
- ARM: mach-virt
  - Need for exit also led power management (PSCI)
  - And, command line device configuration had to be virtio
  - Enter special char dev backend, exposed through virtio-serial (chr-testdev)

ARM: API

- Virtio support
  - Minimal
  - So far only virtio-mmio with DT device discovery

virtio

```
virtqueue_add_outbuf,
virtqueue_kick,
virtqueue_get_buf,
...
```

testdev

- chr-testdev vs. chr-exit?
  - Could extend it for tests that need QEMU's cooperation
- x86 has pc-testdev
  - Different I/O ports invoke different tests
- There's also pci-testdev
  - Could use this in ARM too after adding some PCIe host bridge support

Running the tests

```
$ git clone git://git.kernel.org/pub/scm/virt/kvm/kvm-unit-tests.git
$ cd kvm-unit-tests/
$ ./configure
$ make
$ ./run_tests.sh
```

Or just one test

```
$ export QEMU=/some/path/qemu-system-aarch64
$ ./arm-run arm/mpidr-test.flat -smp 4 -append clusters=2
cpu1: 80000001
cpu2: 80000002
cpu3: 80000003
cpu0: 80000000
PASS: check_mpidrs(nr_cpus == 4)

SUMMARY: 1 tests, 0 unexpected failures
```

Everything about running...

```
$ cat arm/unittests.cfg
…
[selftest-setup]
file = selftest.flat
smp = 2
extra_params = -m 256 -append 'setup smp=2
mem=256'
groups = selftest
…
```

Relies on QEMU's -kernel command line option

```
$ ./run_tests.sh
PASS selftest-setup
PASS selftest-vectors-kernel
PASS selftest-vectors-user
PASS selftest-smp
```

```
$ cat test.log
…
PASS: selftest: setup: smp: nr_cpus = 2
PASS: selftest: setup: mem: size = 256 MB

SUMMARY: 2 tests, 0 unexpected failures
…
```

ARM vs. x86

- Different setup designs
  - x86 is less Linux-like and has less device discovery
- Unit test support
  - ARM still needs interrupt controller framework
- API
  - Only lib code (and thus API) shared, not a bunch
- What could be merged?
  - Maybe x86 would like to use asm-offsets generation
  - ??

ARM vs. x86 cont.

- What else is different?
  - Oh yeah, x86 actually has tests!
- But ARM interest is picking up!
  - MTTCG
  - PSCI, PMU, VFP

"If you build it, they will come"

Types of tests (x86)

- Spec conformance
- Latency measurement
- Bug reproducers

instruction emulation	vmexits	nested virt
kvmclock	page access	msr access
apic	ioapic	event injection
debug register access	realmode	tsc

The future...

x86	More tests
ARM	Way more tests
And...	

PowerPC

- First started in 2008
  - Lost momentum, source removed in 2014
- PowerPC also boots with DT
  - Can build on ARM framework
- Only first stepping stones posted so far

PowerPC first stepping stones

- QEMU sPAPR machine type
  - Expects bootloader → built one (`b 0x400000`)
  - Manipulates load address → unit tests must be relocatable
- Printing
  - Just put-term-char hypervisor call
- Exiting
  - Cheated… Prints exit code and issues RTAS call
- That's it so far...

Closing

- Unit tests are an important part of development
  - Functional testing, latency measurement, regression testing
- kvm-unit-tests
  - Makes it possible for a test to be just a few tens of lines
  - Makes it easy to build and run tests
  - Supports i386, x86_64, ARMv7, ARMv8
  - Support for PowerPC coming soon

"Sometimes you eat the bear …
and sometimes the bear eats you"

kvm-unit-tests helps keep the bug from eating you!

Thank you

Happy Coding! *and Testing*

drjones@redhat.com

BACKUP SLIDES

Is exit too complex?

- Should we switch to ACPI and PSCI?
  - Con: QEMU would always exit with zero
  - But, unit tests print most status anyway
  - And, the output stream could be monitored for an exit code easily enough
  - Con2: exit would need to be “syscall-ified” for user mode tests to use it

selftests

- kvmarm rapid prototyping also had unit tests
- Long lost branch in long lost kernel repo
  - tools/testing/selftests/kvm/arm/
- Interesting approach with own userspace (no QEMU)
  - MMIO addresses used like pc-testdev
- We can steal inspiration for tests to write from it for kvm-unit-tests
  - e.g. the vmexit latency test

standalone

```
$ make standalone
$ make install
$ /usr/local/share/kvm-unit-tests/eventinj
```