Aneesh Kumar K.V <aneesh.kumar@linux.vnet.ibm.com> twitter @kvaneesh

09 Jul 2016

Kernel debugging using Qemu

Agenda

- Kernel development
- Introduction to Qemu
- Virtual Machines
- GDB server
- GDB remote debugging

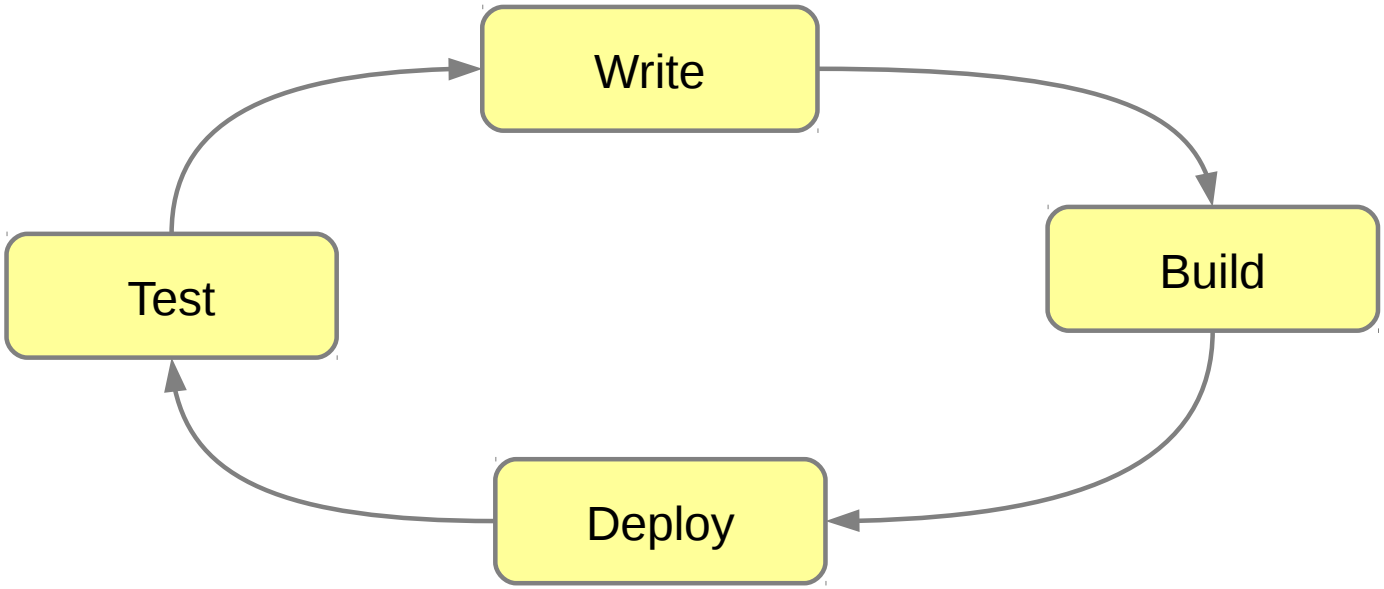
Some of the slides are based on http://vmsplice.net/~stefan/stefanha-kernel-recipes-2015.pdf

About Me

- Works in IBM’s Linux Technology Center, Bangalore
- Ext4, NFS, KVM, MM
- Maintainer for VirtFS (9p file system)
- Maintainer for PPC64 MMU subsystem

Kernel development

- Ends up writing lots of code, before you could start testing.
- Testing is complex, due to machine/hardware dependency



Kernel development

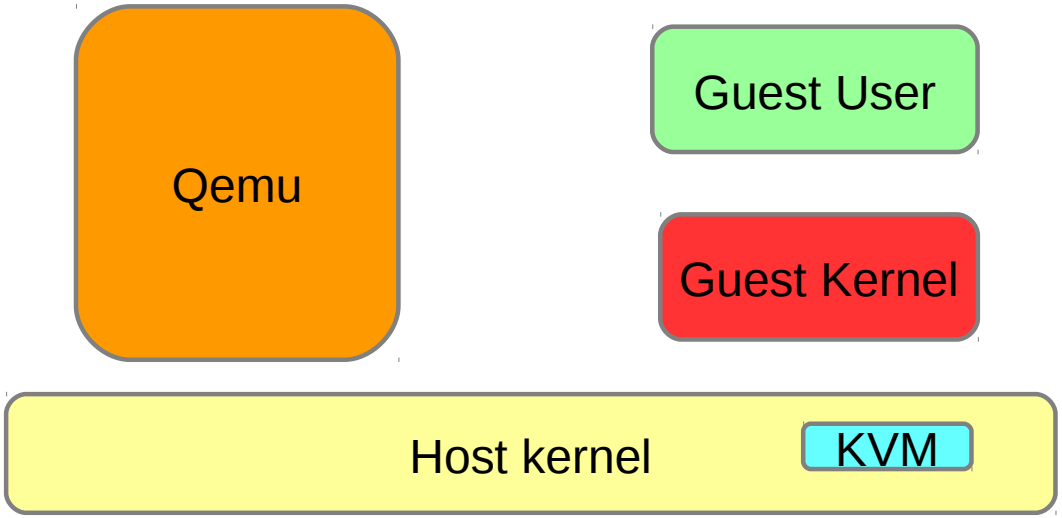
- Kernel bugs results in machine crashes
  - Can’t use your development machines for testing
- Remote Machine based development is cumbersome
  - Requires kernel/source to be copied to the machine
    - tftp setup/scp
    - Log collection requires console monitoring
    - Longer deploy time due to longer boot cycle
    - File system corruption causing reinstall

Virtual Machines

- Easy to start/stop
- Use the same machine for development and testing
- Full access to memory

QEMU

- http://www.qemu.org
- Runs on Linux, Mac, BSD and windows
- Emulates large number of cpu architectures (x86, ppc, arm..)
- Opensource GPL V2 license

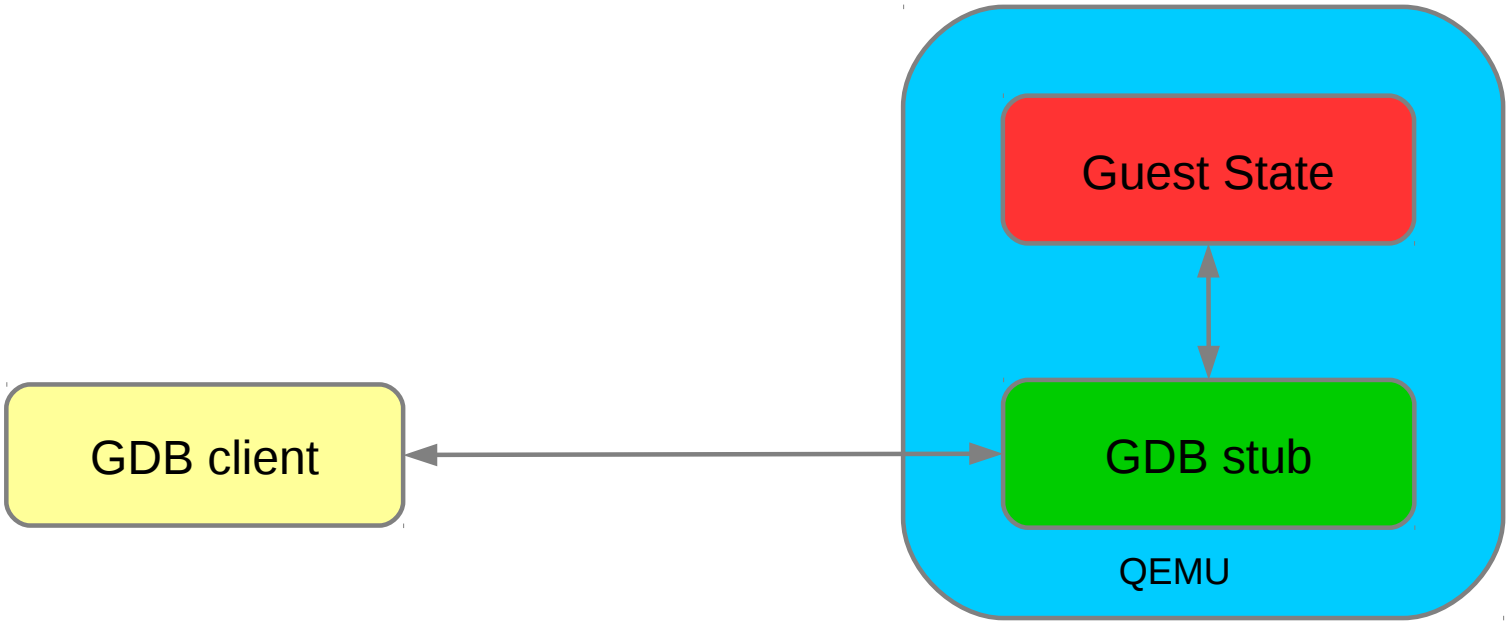


QEMU

- Guest code runs in virtual machine
- Hardware devices are emulated
- QEMU performs I/O on behalf of guest
- QEMU appears as as a normal userspace process on the host

Debugging a virtual machine

- QEMU supports GDB remote debugging to attach to a virtual machine
- Remote debugging != Qemu debugging



Debugging

- How to start

#gdb ./vmlinux

GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.04) 7.11.1

…….

Type "apropos word" to search for commands related to "word"...

Reading symbols from ./vmlinux...target done.

(gdb) target remote localhost:1237

Remote debugging using localhost:1237

default_idle () at arch/x86/kernel/process.c:307

307 trace_cpu_idle_rcuidle(PWR_EVENT_EXIT, smp_processor_id());

(gdb)

Debugging

- Breakpoint, backtrace, register dump and variable print

(gdb) info reg

```
rax            0xffffffff81c19500       -2118019840
rbx            0x0      0
rcx            0xffffffff81c5a440       -2117753792
rdx            0xffff88007fc0eb90       -131939251983472
rsi            0x1      1
rdi            0xffffffff81c19500       -2118019840
....
```

(gdb) b _do_fork

```
Breakpoint 1 at 0xffffffff8109001e: file kernel/fork.c, line 1739.
(gdb) c
Continuing.

Breakpoint 1, _do_fork (clone_flags=8390417, stack_start=18446744071579545266, stack_size=18446612134355350080, parent_tidptr=0x0
<irq_stack_union>, child_tidptr=0x0 <irq_stack_union>,
    tls=0) at kernel/fork.c:1739
1739    {
```

(gdb) bt

```
#0  _do_fork (clone_flags=8390417, stack_start=18446744071579545266, stack_size=18446612134355350080, parent_tidptr=0x0 <irq_stack_union>,
child_tidptr=0x0 <irq_stack_union>, tls=0)
    at kernel/fork.c:1739
#1  0xffffffff8109037e in kernel_thread (fn=<optimized out>, arg=<optimized out>, flags=<optimized out>) at kernel/fork.c:1823
#2  0xffffffff810ab77e in create_kthread (create=<optimized out>) at kernel/kthread.c:233
#3  kthreadd (unused=<optimized out>) at kernel/kthread.c:529
#4  0xffffffff8167cebf in ret_from_fork () at arch/x86/entry/entry_64.S:389
```

Debugging

- Single stepping and variable print

```
(gdb) n
1774            pid = get_task_pid(p, PIDTYPE_PID);
(gdb) n
1775            nr = pid_vnr(pid);
(gdb) n
1774            pid = get_task_pid(p, PIDTYPE_PID);
(gdb) p pid
$4 = (struct pid *) 0xffff88007159f000
(gdb) p *pid
$5 = {count = {counter = 2}, level = 0, tasks = {{first = 0xffff88006e4096e0}, {first = 0x0 <irq_stack_union>}, {first = 0x0 <irq_stack_union>}}, rcu = {next =
0xffff88007158dc98,     func = 0xffffffff810a96c9 <delayed_put_pid>}, numbers = {{nr = 2450, ns = 0xffffffff81c536e0 <init_pid_ns>, pid_chain = {next =
0x0 <irq_stack_union>, pprev = 0xffff88007ffcd3a0}}}}
```

DEMO

Legal Statement

- This work represents the view of the author, and does not necessarily represent the view the IBM.
- IBM and IBM (logo) are trademark of International Business Machines in the United States and/or other.
- Linux is the registered trademark of Linus Torvalds.
- Other company, product and service names may be trademarks of others