Native Linux KVM Tool

Asias He
Beihang University, Beijing, China
Aug 15, 2011
Agenda

- What is it?
- A brief history
- Who is developing it?
- Features
- Features in the future
- How to use it?
- Demos
- Q&A
What is it? (1/2)

* Native Linux KVM Tool is a clean, from-scratch, lightweight KVM host tool implementation
  * Source Code
    * 15K lines of clean C code
    * From scratch and lightweight
    * Great learning tool
    * Integrate more tightly with the kernel source tree
  * Care about both Desktop and Server users
    * Usability
      * As little configuration as possible
    * Performance
      * Multi-threaded and para-virtualized device model
What is it? (2/2)

- Young, only 1 year and 5 months old
- Still under heavy development
- Already have some cool features
  - SMP
    - Up to 254 VCPUs per VM
  - Devices
    - Minimal legacy devices emulation
    - Rely heavily on virtio devices
    - Disk, Network, Serial, Mouse and Keyboard, RTC, VESA, SDL and VNC support
- More features and improve usability & performance
A brief history

- [RFC] Unify KVM kernel-space and user-space code into a single project
- Initial Commit 3 files 93 LOC
- Announcement V1 atLKML
- Announcement V2 atLKML
- Pull Request Mainline kernel
- Pull Request Targetting 3.2

- 2010.3.17
- 2010.3.22
- 2011.4.1
- 2011.6.15
- 2011.7.25
- Current

- 301 Commits
- 301 Commits
- 129 Commits
- 62 Commits

- 793 Commits
Who is developing it? (1/2)

- Developers (17 people)
  - Pekka Enberg (326)
  - Sasha Levin (153)
  - Asias He (120)
  - Cyrill Gorcunov (110)
  - Prasad Joshi (29)
  - Aneesh Kumar K.V (18)
  - Ingo Molnar (11)
  - Liming Wang (7)
  - John Floren (6)
  - Amos Kong (4)
  - Amerigo Wang (2)
  - Giuseppe Calderaro (2)
  - Anton Vorontsov (1)
  - David Ahern (1)
  - Emil Renner Berthing (1)
  - Konstantin Khlebnikov (1)
  - Paul Bolle (1)

- Special thanks to
  - Avi Kivity
    - KVM internal
  - Ingo Molnar
    - All around support
    - Encouragement
Who is developing it? (2/2)

- Mail List
  - kvm@vger.kernel.org
- IRC
  - #pvm @ freenode
- Git Repo
  - git://github.com/penberg/linux-kvm.git

- We need you!
  - Patches and ideas are more than welcome ;

Features (1/12)

- User Interface support
  - Command line user interface
    - Very similar CLI interface like git and perf.
  - Text Console
    - Serial console
    - Virtio console
  - GUI Framebuffer
    - SDL
    - VNC
Features (2/12)

- SMP support
  - Up to 254 VCPUS per VM
    - KVM_CAP_NR_VCPUS 64
    - KVM_CAP_MAX_VCPUS 254
      - [PATCH] x86: Raise the hard VCPU count limit by Sasha Levin
  - Implement MPtable specification
    - Easier than ACPI specification
    - Implement the minimum needed for smp
Features (3/12)

- Disk support
  - Disk image support
    - Raw disk images
    - QCOW/QCOW2 disk images (experimental)
    - Raw block devices (e.g. /dev/sdb7)
  - Boot a directory as a root filesystem.
    - Plain directory which contains root filesystem
Features (4/12)

- Network support
  - TAP Mode
    - NAT
    - Bridge
    - Special privilege (CAP_NET_ADMIN)
    - Setup
  - UIP Mode (User mode TCP/IP)
    - No special privilege
    - From scratch and no ancient slirp code
      - `qemu.git$ cat slirp/*.c,h net/slirp/*.c,h | wc -l` -> 11790 LOC -> 11.7 KLOC
      - `tools/kvm$ cat net/uip/*.c,h include/kvm/uip.h | wc -l` -> 1588 LOC -> 1.5 KLOC
    - Protocols
      - ARP, ICMP, IP, TCP, UDP, DHCP
      - Up layer: HTTP, FTP, SSH, DNS
    - Zero configuration network
      - Built-in DHCP server
      - No setup in host side
  - Multi-threaded
    - UDP thread
    - Per Connect TCP thread
  - Performance
    - Almost achieves the the same TCP and UDP performance as in host
Features (5/12)

- Device emulation
  - Two type of devices
    - Virtio devices
    - Legacy devices
  - Device emulation infrastructures
    - PIO and MMIO
      - KVM_EXIT
      - KVM_IOEVENTFD
    - Interrupt
      - KVM_IRQ_LINE
Features (6/12)

- virtio pci
  - Simple PCI controller
  - PCI configuration space
    - PCI_CONFIG_ADDRESS 0xcf8
    - PCI_CONFIG_DATA 0xcfc
  - PCI discovery/configuration
    - VENDOR_ID
      - PCI_VENDOR_ID_REDHAT_QUMRANET 0x1af4
    - DEVICE_ID
      - PCI_DEVICE_ID_VIRTIO_NET 0x1000
      - PCI DEVICE_ID_VIRTIO_BLK 0x1001
      - PCI DEVICE_ID_VIRTIO_CONSOLE 0x1003
      - PCI DEVICE_ID_VIRTIO_RNG 0x1004
      - PCI DEVICE_ID_VIRTIO_BLN 0x1005
      - PCI DEVICE_ID_VIRTIO_9P 0x1009
    - SUBSYSTEM_ID
      - VIRTIO_ID_NET 1
      - VIRTIO_ID_BLOCK 2
      - VIRTIO_ID_CONSOLE 3
      - VIRTIO_ID_RNG 4
      - VIRTIO_ID_BALLOON 5
      - VIRTIO_ID_9P 9
  - BAR[0]
    - IO space
    - Virtio configuration
Features (7/12)

- **virtio blk**
  - Process multiple virtio-blk *requests* in parallel
  - Process multiple virtio-blk *devices* in parallel
- **Backends**
  - Raw block device
  - Raw disk image
  - QCOW image
  - QCOW2 image
Features (8/12)

- virtio net
  - Multi-thread
    - TX thread
    - RX thread
  - Backends
    - TAP Mode
    - UIP Mode

```
Guest OS
QUEUE_NOTIFY

KVM_EXIT      KVM_IOEVENTFD

Virtio Net Device
virtio_net_handle_callback()

VIRTIO_NET_TX_QUEUE       VIRTIO_NET_RX_QUEUE

virtio_net_tx_thread()    virtio_net_tx_thread()

Handle Request

TAP_MODE       UIP_MODE

TX

tap_ops_tx     RX

uip_ops_tx     tap_ops_rx

Interrupt

Guest

TX

Interrupt

Guest

RX

uip_ops_rx
```
Features (9/12)

- **virtio 9p**
  - 9p: Plan 9 Filesystem Protocol
  - Transport: Named pipe, TCP connection, File descriptor, RDMA channel, virtio
  - No network setup is needed
  - Share files between host and guest
    - `kvm run -k ./bzImage -d ./disk.img -9p ./dir_to_share`
    - `mount -t 9p -otrans=virtio -oversion=9p2000.u kvm_9p /mnt`
  - Boot a directory as a guest root filesystem using 9p
    - `kvm run -k ./bzImage -d ./guest_rootfs`
Features (10/12)

- virtio console
  - /dev/hvc0
- virtio rng
  - /dev/urandom
  - /dev/hwrng
- virtio balloon
  - kvm balloon inflate/deflate size instance
Features (11/12)

- Legacy device emulation
  - Serial device 16550
    - Guest console
  - PS/2 Keyboard and Mouse i8042
    - SDL and VNC
  - VESA
    - SDL and VNC
  - RTC
    - Real time clock
Features (12/12)

- BIOS emulation
  - Very tiny and lightweight BIOS layer
  - No external BIOS dependency
  - Functions
    - e820 memory map
    - real-mode interrupt vector table
    - mptable
Features in the future (1/2)

- Vhost net/blk
- Macvtap Mode
- Virtio-scsi virtio-based SCSI HBA
- IO bandwidth limits
- More disk image format support (e.g. vmdk, vdi, etc.)
- 9p + overlayfs for COW filesystem layer for guest
- Boot disk images without external linux kernel image.
- Grub support
- External BIOS support (e.g. Seabios)
Features in the future (2/2)

- Non-Linux OS support
- QXL paravirtual graphic card
- Integrate with *perf* for profiling and tracing
- Integrate with *gdb* for debugging
- Libvirt support
- Live migration
How to use it (1/6)

- Command line interface
  - `kvm run/stop`
  - `kvm pause/resume`
  - `kvm list`
  - `kvm balloon`
  - `kvm debug`
  - `kvm help`
  - `kvm version`
How to use it (2/6)

Details for ‘kvm run’

Basic options:

```
--name <guest name>  
A name for the guest
-c, --cpus <n>       Number of CPUs
-m, --mem <n>        Virtual machine memory size in MiB.
-d, --disk <image or rootfs_dir>  
Disk image or rootfs directory
--balloon           Enable virtio balloon
--vnc               Enable VNC framebuffer
--sdl               Enable SDL framebuffer
--rng               Enable virtio Random Number Generator
--9p <dir_to_share,tag_name>  
Enable virtio 9p to share files between host and guest
--console <serial or virtio>  
Console to use
--dev <device_file>  
KVM device file
```
How to use it (3/6)

- Details for ‘kvm run’

  Kernel options:
  
  -k, --kernel <kernel>
  Kernel to boot in virtual machine

  -i, --initrd <initrd>
  Initial RAM disk image

  -p, --params <params>
  Kernel command line arguments
How to use it (4/6)

● Details for ‘kvm run’

Networking options:
- `--network <user, tap, none>`
  Network to use
- `--host-ip <a.b.c.d>`
  Assign this address to the host side networking
- `--guest-ip <a.b.c.d>`
  Assign this address to the guest side networking
- `--host-mac <aa:bb:cc:dd:ee:ff>`
  Assign this address to the host side NIC
- `--guest-mac <aa:bb:cc:dd:ee:ff>`
  Assign this address to the guest side NIC
- `--tapscript <Script path>`
  Assign a script to process created tap device
How to use it (5/6)

● Details for ‘kvm run’

BIOS options:
  --vidmode <n>     Video mode

Debug options:
  --debug           Enable debug messages
  --debug-single-step Enable single stepping
  --debug-iomport   Enable ioport debugging
  --debug-iodelay <n> Delay IO by millisecond
## How to use it (6/6)

### Details for ‘kvm debug’

#### Registers:

- **rip**: 00000000c1035061
- **rsp**: 00000000c199ffb8
- **flags**: 0000000000000246
- **rax**: 0000000000000000
- **rbx**: 00000000c19fa1e4
- **rcx**: 00000000d78027d0
- **rdx**: 0000000000000003
- **rsi**: 0000000000000000
- **rdi**: 00000000c19a0000
- **rbp**: 0000000000000003
- **r8**: 0000000000000000
- **r9**: 0000000000000000
- **r10**: 0000000000000000
- **r11**: 0000000000000000
- **r12**: 0000000000000000
- **r13**: 0000000000000000
- **r14**: 0000000000000000
- **r15**: 0000000000000000
- **cr0**: 000000008005003b
- **cr2**: 0000000000000000
- **cr3**: 0000000000000000
- **cr4**: 0000000000000000
- **cr8**: 0000000000000000

#### Segment registers:

<table>
<thead>
<tr>
<th>register</th>
<th>selector</th>
<th>base</th>
<th>limit</th>
<th>type</th>
<th>p</th>
<th>dpl</th>
<th>db</th>
<th>s</th>
<th>l</th>
<th>g</th>
<th>avl</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs</td>
<td>0060</td>
<td>0000000000000000</td>
<td>ffffffff</td>
<td>0b</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ss</td>
<td>0068</td>
<td>0000000000000000</td>
<td>ffffffff</td>
<td>03</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ds</td>
<td>007b</td>
<td>0000000000000000</td>
<td>ffffffff</td>
<td>03</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>es</td>
<td>007b</td>
<td>0000000000000000</td>
<td>ffffffff</td>
<td>03</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>fs</td>
<td>00d8</td>
<td>0000000000000000</td>
<td>ffffffff</td>
<td>03</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gs</td>
<td>0000</td>
<td>0000000000000000</td>
<td>ffffffff</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>tr</td>
<td>0080</td>
<td>0000000000000000</td>
<td>ffffffff</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ldt</td>
<td>0000</td>
<td>0000000000000000</td>
<td>ffffffff</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gdt</td>
<td>0000</td>
<td>0000000000000000</td>
<td>ffffffff</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>idt</td>
<td>000000000000c19a0000</td>
<td>0000007ff</td>
<td>11111111</td>
<td>01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

#### APIC:

- **efer**: 0000000000000000
- **apic base**: 00000000fee00900
- **nmi**: enabled

#### Interrupt bitmap:

```
0000000000000000 0000000000000000 0000000000000000 0000000000000000
```

#### Code:

```
---
rip: [00000000c1035061] <unknown>
```
Demos

- 1.demo.sdl.sh
- 2.demo.vnc.sh
- 3.demo.serial.console.sh
- 4.demo.virtio.console.sh
- 5.demo.dir.as.rootfs.sh
- 6.demo.dir.to.share.sh
- 7.demo.64vcpus.sh
Q&A

Questions?