kvmtool - a QEMU alternative?

Andre Przywara <andre.przywara@arm.com>

ARM Ltd., Cambridge, UK
October 13, 2016

© ARM 2016
kvmtool - A QEMU alternative?

- Overview
- Brief History
- Comparison to QEMU
- Usage
- Numbers....
- Outlook
Native Linux KVM tool - Overview

"kvmtool is a lightweight tool for hosting KVM guests."

What kvmtool does:

- Run Linux guests under KVM
- Provide network and block devices (using virtio)
- Emulate some platform devices (8250 UART, PS/2, RTC, VGA)
- Is portable (i386, x86-64, arm, arm64, mips64, ppc)

What kvmtool does not do:

- Instruction emulation
- Emulate different architectures
- Emulate existing platforms (boards)
- Run Windows
- Run on non-KVM-capable machines
- Live in some kernel tree anymore
Brief History

- 2010ish: first posts into tools/kvm of the Linux tree
- 2011: longish discussion about being merged or not
- end of 2012: gains ARM support
- beginning of 2013: gains AArch64 support
- 2014: gains MIPS support
- 2015: separated from the kernel tree
Disadvantages

- Missing a lot of features
- Missing platform emulation: ACPI (no poweroff on x86!), UEFI
- Non-Linux guest OSes not really supported
- Much less tested
- Some convenience features missing
Advantages

- Small: around 1.6 MB code size, 35 KLOC
- Builds in seconds
- Easily(TM) cross-compilable
- Static binary possible (roughly 1MB in size)
- Very small dependency chain
- Easily hackable (Yeah!)
- License (GPL-v2)
Usage

$ lkvm run -k bzImage

- Loads the kernel, passes through the host filesystem (9pfs/virtio)
- Connects the guest serial console to your terminal
- Provides (user level) networking
- Starts a shell prompt ...

- But disk images, initrds and ptys are supported as well!

$ lkvm run -k bzImage -i initrd.img -d disk.img -p "root=/dev/vda" -c 2 -m 512 --tty 0
"If kvmtool is so much lighter than QEMU, can one load more guests on a host?"

- Memory usage is dominated by guest's RAM demand
- Guest RAM consumption is driven by the guest and fulfilled by the kernel
- Multiple instances of the same userland process share resources

- What are those guests supposed to do?
- Is there another limit that we run into (CPU, storage, network)?
- Shall I use virtualization in the first place?
## Memory usage: `/proc/<pid>/status`

<table>
<thead>
<tr>
<th></th>
<th>kvmtool</th>
<th>QEMU</th>
</tr>
</thead>
<tbody>
<tr>
<td>VmPeak</td>
<td>943392 kB</td>
<td>599124 kB</td>
</tr>
<tr>
<td>VmSize</td>
<td>877856 kB</td>
<td>598352 kB</td>
</tr>
<tr>
<td>VmHWM</td>
<td>53472 kB</td>
<td>193412 kB</td>
</tr>
<tr>
<td>VmRSS</td>
<td>53472 kB</td>
<td>193412 kB</td>
</tr>
<tr>
<td>VmData</td>
<td>840164 kB</td>
<td>476876 kB</td>
</tr>
<tr>
<td>VmStk</td>
<td>136 kB</td>
<td>136 kB</td>
</tr>
<tr>
<td>VmExe</td>
<td>180 kB</td>
<td>7532 kB</td>
</tr>
<tr>
<td>VmLib</td>
<td>4504 kB</td>
<td>14708 kB</td>
</tr>
<tr>
<td>VmPTE</td>
<td>292 kB</td>
<td>668 kB</td>
</tr>
<tr>
<td>VmPMD</td>
<td>16 kB</td>
<td>12 kB</td>
</tr>
<tr>
<td>Threads</td>
<td>17</td>
<td>4</td>
</tr>
</tbody>
</table>
Kernel same page merging

- Scans memory to find identical pages
- One (physical) page is freed, both VAs now point to the same location
- Marked as read-only, copy-on-write takes care of faults
- `CONFIG_KSM and echo 1 > /sys/kernel/mm/ksm/run`
- Works great with many guests, but:
  - Takes CPU cycles to scan the memory
  - Heavy memory over-commitment may invoke OOM killer upon usage
KSM memory usage

QEMU guests with 2 cores and 512MB RAM with 9pfs sandboxed shell

![Graph showing KSM memory usage](chart.png)

The graph illustrates the free memory in KB and the number of guests running as a function of the number of guests started. The graph shows the following:

- **Free memory in KB**: The graph plots the free memory in KB on the left y-axis.
- **Guests running**: The graph plots the number of guests running on the right y-axis.
- **Guests KSM**: The graph shows the number of guests using KSM for memory sharing.

The x-axis represents the number of guests started, ranging from 0 to 600. The graph demonstrates how the free memory decreases as more guests are started, while the number of guests running increases. The use of KSM for memory sharing is also illustrated, showing how it affects the free memory.
Run as many guests as possible with:

- small RAM size (40 MB)
- two cores
- 9pfs shared filesystem
  - no block device, as disk images consume buffer cache
  - no initrd to allow smaller guest memory
- running with kvmtool (HEAD) and QEMU (2.7.0) on:
  - Core i7 3770 "Ivy Bridge", 16 GB RAM + 8GB swap
  - Calxeda Midway (4 * ARM® Cortex®-A15), 8 GB RAM + 8GB swap
  - (ARM Juno r0 (2 * ARM Cortex®-A57 + 4 * ARM Cortex®-A53), 8 GB RAM + 8GB swap)
- Launch guests until number no longer increases (OOM killer)
- Don't forget to enable KSM!
Results: Number of guests

guests with 2 cores and 52MB RAM with 9pfs sandboxed shell

© ARM 2016
Results: Memory usage

QEMU guests with 2 cores and 512MB RAM with 9pfs sandboxed shell

available swap
available RAM
running guests

free memory in KB
guests running
number of guests started

© ARM 2016
Known issues

- **virtio bug on SMP: guest waits forever for virtio transaction**
  - Some race condition in the (heavily multithreaded) virtio implementation
  - Does not manifest with UP guests pinned on a host core
  - Can be observed while booting with NFS root
  - On some host quickly shows even with virtio-blk
  - Debugging and fixes welcome!

- Security issues around file path parsing and string functions
- Some messages have double line endings ;-)
- CPU load of idle guests (observed on x86 only)
- x86 CPUID vendor value is "LKVMLKVMLKVM", prevents some features?
kvmtool works (tm) right now

Some bugs get fixed, some new (ARM) features added

Invest in sandboxing feature?

Adding new features:
- New emulated devices? No.
- ACPI emulation? Mmmh.
- New KVM kernel features? Yes.

Please send bug fixes and reports! Tell about your usecase!
Conclusions

- kvmtool is a different implementation of the KVM ABI
- kvmtool is an easier to handle application (cross compilation, static binary)
  - Does that matter?
- kvmtool is no real competition to QEMU
- kvmtool can show advantages in some cases
  - Wimpy hosts (low memory)
  - Rapid prototyping

Please contribute to QEMU (and help fixing it)!