Extending bhyve beyond FreeBD guests

Peter Grehan
grehan@freebsd.org
EuroBSDcon 2013

What is bhyve ?

- A “minimally viable x86 hypervisor”
- serial console, PCI virtio block/net, 64 bit host/guests (covers most)
- Requires VT-x/EPT CPU support (core i*)
- In base-system FreeBSD as of 10.0

Pieces #1: vmm.ko

- Kernel module implementing:
  - VT-x state setup, enter/exit context switching
  - Local APIC emulation
  - VT-d IOMMU for PCI pass-thru
  - Guest physical memory mgmt
  - user-space cdev interface
- Small: 64KB text

Pieces #2: bhyveload

- user-space bootloader
- FreeBSD “userboot” library + bhyve API
- Creates VM; lays out kernel + metadata; sets up initial VM register state
- ... and then exits.

Pieces #3: bhyve

- User-space run loop
- Implements PCI bus/device emulation
  - PCI devs - uart, virtio block/net
- Device backends
  - stdin/out, tap device, file/block device
- Threads for vCPUs, i/o devs, kqueue loop.
- Small: 130KB text (less than ifconfig)

Pieces #4: bhyvectl

- Simple utility
- Dump/modify VM state (registers, VMCS)
- Dump VM stats (e.g. # HLT exits)
- Delete VMs

Why non-FreeBSD ?

- Simple: what users expect
- Intent of bhyve was always to run unmodified guests
- Restricting to modified guest o/s dooms to slow decay, or as a limited audience developer-only tool
    - UML Linux

Ordering Development

- Development started with modified guest o/s - reduces unknowns and complexity
- Then moved to an unmodified guest
- ... older versions of guest
- ... guest o/s's with source access/buildability
- binary-only guest o/s's last.

First “other”: UEFI

- BSD-licensed BIOS replacement
- “OVMF” build target for virtual machines
- Extracts parameters from hypervisor
  - Memory map, # CPUs etc

UEFI issues

- Simple linear loader, but with 16-bit entry
  - set up memory area for params
- Wanted 8259/8254 for timer support
  - Replaced with HPET
- PCI BARs reprogrammed
  - Blew away PCI serial port: changed to use LPC bridge and ISA-style addresses

grub

- "grub-emu" build target for user-space
- Mainly for filesystem debug
- Hacked^h^h^h^hCo-opted for a Linux/*BSD userspace loader

grub-1.98

- grub-emu builds OOTB on FreeBSD
- However, boot-loader code assumes 32-bit. Lots of 64-bit issues
- Assumes it is running in target address space with linear mapping
  - Direct pointer derefs
  - Trampolines to launch o/s
- Too much conditional code. Abandoned.

grub-2.00

- grub-emu nightmare build on FreeBSD
  - gnulib, aaaargh !!!
- Once done, excellent clean codebase
- Indirection for target mem/reg sets
  - Easy interface to bhyve API
- No conditional code in Linux/*BSD loaders

Linux

- 32-bit “flat” entry point
  - Intended for non-BIOS environments
  - Well documented !
- Requires VT-x ‘unrestricted guest’ support
- Lots of trial and error to fix initial TR/TSS state to avoid “invalid guest state” exit on vmenter

Linux #2

- CR0 unconditional write exposed bug
- Lots of CPUID probing
    - Required hiding more and more fields
- Lots of MSRs that required emulation/ ignorance
- “Interesting” use of 8254 timer

Mo' Linux

- Requires PCI hostbridge, or no PCI probing
- ISO initrd doesn't have virtio block drivers
  - Fixed with GSoC AHCI emulation
- IOAPIC changed to ignore level-trigger
- SMP, virtio devices worked fine
- No custom kernels. Lots of code reading

OpenBSD

- Using 5.3. Not quite working.
- Well supported by grub-2.00
- No MSI in virtio drivers
  - Well... non-standard and bhyve-specific
  - 1-line change, committed (thanks!)

OpenBSD #2

- APIC code exposed instruction emul bugs
  - RIP-relative addressing, SIB + displacement
- Required RTC NVRAM implementation
- PCI MSI capability writes to r/o regs
  - Fine on real h/w; no point in trapping
- Extensive kernel rebuilds for debug

Future o/s's

- NetBSD - no MSI in drivers (fixed?)
- D'flyBSD - BIOS emul for loader
- Illumos
  - Requires BIOS emul
  - Useful for ZFS development
- Windows - daunting, no source

Thanks to...

- neel@freebsd.org, who did (and does) all the hard stuff
- Tycho Nightingale from Pluribus Networks who contributed the UEFI work

Questions ?