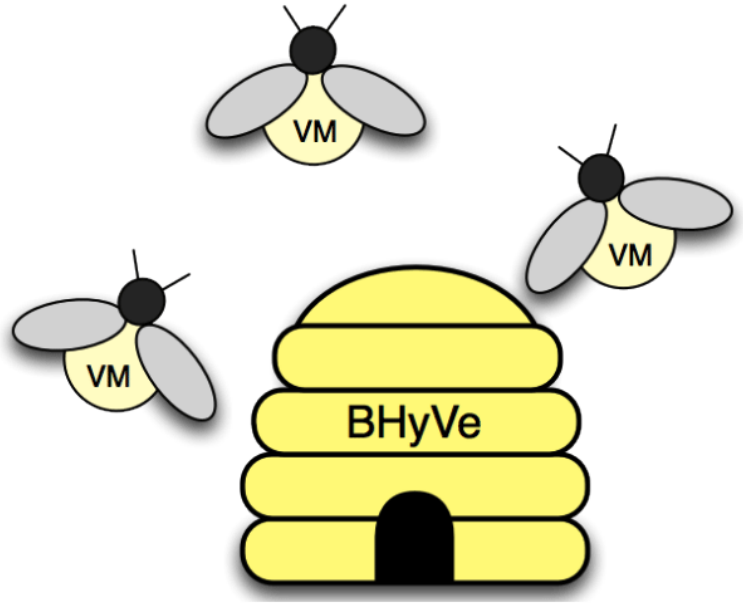

BHyVe

The BSD HyperVisor In Depth

Michael Dexter

editor@callfortesting.org @michaeldexter
www.bhyve.org

MeetBSD 2012
Santa Something, California

History

MeetBSD California

UnConference 2010

History

UnConference Format:

1. Congregate
2. Propose and select sessions
3. Break out into sessions
4. Repeat the next day

History

Huh?

History

Huh?

100% Virtualization session attendance

Follow-up session the next day

Conclusion: We need a hypervisor

History

Result:

Neel Natu and Peter Grehan

Announce BHyVe at the BSDCan 2011 DevSummit

Slides and audio are online

http://wiki.freebsd.org/201105DevSummit?action=AttachFile&do=view&target=BHyVe.pdf
http://www.bsdcan.org/2011/audio/BHyVeNativeBSDHypervisor.mp3

History

March 2012: CallForTesting.org/bhyve

April 2012: bhyve.org

Incomplete Google Summer of Code BIOS Project for foreign OS's

tl:dr

Requires Intel Extended Page Tables (EPT)

Guests are booted from disk images

jhb@'s as(1) fixes allow for a GPLv2 assembler with EPT

CURRENT import Real Soon Now, MFC-able to 9.0

Easiest to test on 9.0 and 10-CURRENT (Not 8.x or 9.1)

7.2 and 8.0 guests feasible, 9.0 and newer supported

Minor fixes planned to allow for building with CLANG

Works with VMware Fusion VT-x pass-through

Context

Proprietary and dominant: VMware (GPL Violations?)

Linux: KVM/QEMU Hypervisor / LXC Containers

SmartOS: KVM Hypervisor / Zones

FreeBSD: Xen EC2 / jail(8)

Honorable mention: NetBSD Xen

Context

Xen KVM BhyVe

(No Cup Holders – Yet)

Context

Robert P. Goldberg, 1973:
“Architectural Principles for Virtual Computer Systems”

Type 1 Native/Bare Metal Hypervisor: VMware

Type 2 Hosted Hypervisor: BHyVe, NetBSD/Xen, Linux/KVM

Gerald J. Popek and Robert P. Goldberg, 1974:
“Formal Requirements for Virtualizable Third Generation Architectures”

Properties of a Hypervisor

http://en.wikipedia.org/wiki/Hypervisor
http://en.wikipedia.org/wiki/Popek_and_Goldberg_virtualization_requirements

Context

Equivalence / Fidelity

A program running under the VMM should exhibit a behavior essentially identical to that demonstrated when running on an equivalent machine directly

Resource control / Safety

The VMM must be in complete control of the virtualized resources

Efficiency / Performance

A statistically dominant fraction of machine instructions must be executed without VMM intervention

http://en.wikipedia.org/wiki/Popek_and_Goldberg_virtualization_requirements

Hardware-Assisted Virtulalization

“It's all built on VT-x exits. I/O exits are used to build the PCI emulation since I/O instructions are used for PCI config space. VT-x sets up state to enter/exit "non-root" mode, aka VM mode.

EPT-violation exits are used for memory-mapped I/O. Guest physical memory is set up in EPT tables. Guest access to anything else causes an EPT-violation exit. Then instruction emulation is used to determine what is written and where reads should go.”

– Peter Grehan

Hardware-Assisted Virtulalization

VT-x: Virtualization Extensions:
Interception of privileged instructions

VT-d: Virtualization for Directed I/O:
IOMMU Virtualization / PCI Pass-Through

EPT: Extended Page Tables: MMU Virtualization
Previously handled in software

AMD-V, AMD-Vi and RVI/Nested Page Tables Planned

http://en.wikipedia.org/wiki/X86_virtualization
http://en.wikipedia.org/wiki/Extended_Page_Table

Hardware-Assisted Virtulalization

dmesg(8):

Copyright (c) 1992-2012 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
The Regents of the University of California. All rights reserved.
FreeBSD is a registered trademark of The FreeBSD Foundation.
FreeBSD 9.0-RELEASE-p3 #0: Tue Jun 12 02:52:29 UTC 2012
root@amd64-builder.daemonology.net:/usr/obj/usr/src/sys/GENERIC amd64
CPU: Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz (2491.97-MHz K8-class CPU)
Origin = "GenuineIntel" Id = 0x206a7 Family = 6 Model = 2a Stepping = 7

Features=0xbfebfbff<FPU,VME,DE,PSE,TSC,MSR,PAE,MCE,CX8,APIC,SEP,MTRR,PGE,MCA,CMOV,PAT,PSE36,CLFLUSH,DTS,ACPI,MMX,FXSR,SSE,SSE2,SS,HTT,TM,PBE>

Features2=0x17bae3ff<SSE3,PCLMULQDQ,DTES64,MON,DS_CPL,**VMX**,SMX,EST,TM2,SSSE3,CX16,xTPR,PDCM,PCID,SSE4.1,SSE4.2,x2APIC,**POPCNT**,TSCDLT,AESNI,XSAVE,AVX>

POPCNT Accompanies EPT

Hardware-Assisted Virtulalization

Most, if not all “Nehalem”, “Sandy Bridge” and “Ivy Bridge” Core, Xeon, Pentium and Celeron Processors:

EPT: Intel Core i3, i5, i7 Processors

EPT: Most same-generation Xeon Processors

EPT: Some Pentium Mobile and Celeron Processors (!)

VT-d: Many Core i5, i7 and Xeon Processors

Caveat: It may be disabled in BIOS or blocked entirely

Hardware-Assisted Virtulalization

ark.intel.com is your friend

Intel® Pentium® 2020M Processor (2M Cache, 2.40 GHz)

Advanced Technologies	
Intel® Turbo Boost Technology	No
Intel® vPro Technology	No
Intel® Hyper-Threading Technology	No
Intel® Virtualization Technology (VT-x)	Yes
Intel® Virtualization Technology for Directed I/O (VT-d)	No
Intel® VT-x with Extended Page Tables (EPT)	Yes

as(1) Implications

The GNU binutils assembler with the EPT instructions is GPLv3-licensed and will not be included in BASE

9.0 Short-term solution: Install devel/binutils (v22.2_3)

HEAD solution: jhb@ implemented the missing instructions and they are manually MFC-able

See FreeBSD svn revisions 238123 and 238167

as(1) Implications

Revision 238123

Add support for the 'xsave', 'xrstor', 'xsaveopt', 'xgetbv', and 'xsetbv' instructions. I reimplemented this from scratch based on the Intel manuals and the existing support for handling the fxsave and fxrstor instructions. This will let us use these instructions natively with GCC rather than hardcoding the opcodes in hex.

Revision 238167

Add support for the 'invept' and 'invvpid' instructions. Beyond simply adding appropriate table entries, the assembler had to be adjusted as these are the first non-SSE instructions to use a 3-byte opcode (and a mandatory prefix to boot).

as(1) Implications

My cheat for FreeBSD 9.0's version 2.17.50:

```
sudo svn export --force
http://svn.freebsd.org/base/head/contrib/binutils/gas/config/tc-
i386.c@238167 /usr/src/contrib/binutils/gas/config/

sudo svn export --force
http://svn.freebsd.org/base/head/contrib/binutils/opcodes/i386-
dis.c@238167 /usr/src/contrib/binutils/opcodes/

sudo svn export --force
http://svn.freebsd.org/base/head/contrib/binutils/opcodes/i386-
opc.h@238123 /usr/src/contrib/binutils/opcodes/

sudo svn export --force
http://svn.freebsd.org/base/head/contrib/binutils/opcodes/i386-
opc.tbl@238167 /usr/src/contrib/binutils/opcodes/

sudo svn export --force
http://svn.freebsd.org/base/head/contrib/binutils/opcodes/i386-
tbl.h@238167 /usr/src/contrib/binutils/opcodes/
```

as(1) Implications

```
cd /usr/src/gnu/usr.bin/binutils
make depend
make
cp /usr/src/gnu/usr.bin/binutils/as/as /usr/bin
```

The Versioning Moving Target

BHyVe began life in FreeBSD 8.1

Is manageable in FreeBSD 9.0

FreeBSD 9.1 suffers from AVX floating point changes

FreeBSD 10 projects/bhyve is the official home

My solution: Start with FreeBSD 9.0R and cherry pick BHyVe and VirtIO sources and build them individually

Build Environment

Union mount the sources and patch /usr/src-bhyve/

```
mkdir /usr/src-bhyve/
mount -t unionfs -o noatime -o below /usr/src /usr/src-bhyve
```

Build to a memory disk for speed and SSD preservation

```
mdmfs -M -s 2000m md /usr/obj
```

See only what is modified!

```
umount /usr/src-bhyve
```

BHyVe Host Components

"/usr/sbin/bhyve", the user-space sequencer and I/O emulation
/usr/src/usr.sbin/bhyve/*

"/usr/sbin/bhyveload", the user-space FreeBSD loader that can load the kernel and metadata inside a BHyVe-based virtual machine
/usr/src/usr.sbin/bhyveload/*

"/usr/sbin/vmmctl", a utility to dump hypervisor register state
/usr/src/usr.sbin/vmmctl/* ← Renamed bhyvectl in 10

"/usr/lib/libvmmapi.a, /usr/lib/libvmmapi.so,
/usr/lib/libvmmapi.so.5, /usr/lib/libvmmapi_p.a"
The front-end to the vmm.ko chardev interface
/usr/src/lib/libvmmapi/*

"/boot/kernel/vmm.ko" Kernel module for VT-x, VT-d and hypervisor control
/usr/src/sys/modules/vmm/*
/usr/src/sys/amd64/vmm/* User Utilities
/usr/src/sys/amd64/include/vmm* User Visible

BHyVe Guest Kernel Components

The BIOS MPTable in-memory structures used to get APIC ID's etc.
`/usr/src/sys/x86/x86/mptable.c`
`/usr/src/sys/x86/x86/mptable_pci.c`

BVM, the "BSD Virtual Machine" Console, or you can connect by serial UART
`/usr/src/sys/dev/bvm/*`

The modified `local_apic.c` to enable Intel x2apic support for performance
`/usr/src/sys/x86/x86/local_apic.c`

The modified `mp_machdep.c` to allow CPUs without 'unrestricted guest' support to bypass the real-mode bootstrap when running under BHyVe
`/usr/src/sys/amd64/amd64/mp_machdep.c`

The BHYVE kernel configuration file
`/usr/src/sys/amd64/conf/BHYVE`

`device`	`bvmconsole`	Or you can use the serial UART
`device`	`mptable`	Obsolted come ACPI support

BHyVe Guest Helpers and Build

Bryan Venteicher's VirtIO modules to allow for guest networking (imported to FreeBSD 10.0-CURRENT)
`/usr/src/sys/dev/virtio/*`
`/usr/src/sys/modules/virtio/*`

(bryanv@'s VirtIO, not kuriyama@'s or Takeshi Hasegawa's)

The traditional FreeBSD tunnel software network interface
`/usr/src/sys/modules/if_tap/*`

Guest Kernel Build:

```
make -DNO_MODULES KERNCONF=BHYVE buildkernel
```

Goal: BHyVe File Layout

```
~/guest/
          boot/kernel/      Containing loader, kernel and modules
          diskdev           Disk image for guest's / partition
          userboot.so       Required by the bhyve* utilities

~/guest/boot/
beastie.4th          kernel              menu.rc
brand.4th            loader.4th          screen.4th
check-password.4th   loader.conf         shortcuts.4th
color.4th            loader.help         support.4th
defaults             loader.rc           userboot.so
delay.4th            menu-commands.4th   version.4th
frames.4th           menu.4th

~/guest/boot/kernel
if_vtnet.ko          virtio.ko           virtio_pci.ko
kernel               virtio_balloon.ko
mdroot (optional)    virtio_blk.ko           The Heavy Lifting
```

These can be read from the guest image with the `bhyve -d option`

Host Preparation

Deduct memory from the host for the guests and load modules:

`/boot/loader.conf`

```
# 4GB for the host:
hw.physmem="0x100000000"
# 8GB for the host:
#hw.physmem="0x200000000"

# Suppress noise:
debug.witness.watch="0"

vmm_load="YES"
if_tap_load="YES"
bridgestp_load="YES"
if_bridge_load="YES"
```

← Reboot to take effect

This is eliminated in 10!

← Loadable with `kldload`

Neel's vm1.tar.gz Notes

Neel has long offered a downloadable guest image:

http://people.freebsd.org/~neel/bhyve/vm1.tar.gz

Uses a memory-backed file system:
`~/vm1/boot/kernel/mdroot`

Changes are not persistent, over 100% full and a large mdroot requires a kernel option for stability:

```
options                         NKPT=256
```

Neel's vm1.tar.gz Notes

NEWS FLASH

Neel's latest guest image allows for guest booting to an installation ISO

http://people.freebsd.org/~neel/meetbsd2012/vm.tar.gz

BHyVe Guest Boot

To explicity boot the guest in `/usr/vm1/`

```
/usr/sbin/vmmctl –vm=myguest --destroy;     \
/usr/sbin/bhyveload -m 768 -M 2048 -h       \
/usr/vm1/ myguest && /usr/sbin/bhyve        \
-c 2 -m 768 -M 2048 -g 0 -P -H -s           \
1,virtio-net,tap0 -s 2,virtio-blk,diskdev \
myguest && sleep 20 && ifconfig tap0 up
```

```
FreeBSD/amd64 User boot, Revision 1.1
(root@bhyve, Fri Mar 30 01:41:20 PDT 2012)
Loading /boot/defaults/loader.conf
/boot//kernel/kernel text=0x41f64f data=0x57810+0x273590 syms=[0x8+0x73788+0x8+0x6af0b]
/boot//kernel/virtio.ko size 0x4bc0 at 0xbca000
/boot//kernel/if_vtnet.ko size 0xae10 at 0xbcf000
/boot//kernel/virtio_pci.ko size 0x57d8 at 0xbda000
/boot//kernel/virtio_blk.ko size 0x4f68 at 0xbe0000
```


```
GDB: debug ports: bvm
GDB: current port: bvm
KDB: debugger backends: ddb gdb
KDB: current backend: ddb
Copyright (c) 1992-2012 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
```

BHyVe Guest Shutdown

Either call `shutdown(8)` or reboot and press `ESC` at the loader and type "`quit`"

How We Got There: Build Notes

Build `~/src/lib/libvmmapi` First or the other components will not build

Tricks to build the components discreetly, rather than with buildworld:

```
svn co
http://svn.freebsd.org/base/projects/bhyve/usr.sbin/bhyve@225757
/usr/src-bhyve/usr.sbin/bhyve/

echo 'CFLAGS+= -I. -I../../lib/libvmmapi' >> /usr/src-
bhyve/usr.sbin/bhyve/Makefile

echo 'LDFLAGS+= -L../../lib/libvmmapi' >> /usr/src-
bhyve/usr.sbin/bhyve/Makefile

ln -s ../../sys/amd64/include /usr/src-
bhyve/usr.sbin/bhyve/machine
```

How I Got There: Guest Image

Similar to building a jail in a 400mb disk image:

```
mdconfig -d -u md0
dd if=/dev/zero of=/usr/vm1/diskdev bs=1k count=400k
mdconfig -a -f /usr/vm1/diskdev -u 0
bsdlabel -w md0 auto
newfs md0
mount /dev/md0 /mnt

fetch -o - ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/amd64/9.0-
RELEASE/base.txz | tar --unlink -xpJf - -C /mnt/
```

Manually configure:

```
mnt/etc/fstab
       /rc.conf
       /resolv.conf
       /ttys

console "/usr/libexec/getty std.9600"  vt100  on  secure

echo "PermitRootLogin yes" >> /mnt/etc/ssh/sshd_config
```

How You Get There

```
bhyve-menu.sh                    All in One

prep_src-user.sh                 unionfs /usr/src
patch_as.sh                      10-CURRENT as(1)
build_as.sh
copy_as.sh
loader_prep.sh                   /etc/loader.conf
```

Legitimate Concerns

“x86 virtualization is about basically placing another nearly full kernel, full of new bugs, on top of a nasty x86 architecture which barely has correct page protection. Then running your operating system on the other side of this brand new pile of sh*t.

You are absolutely deluded, if not stupid, if you think that a worldwide collection of software engineers who can't write operating systems or applications without security holes, can then turn around and suddenly write virtualization layers without security holes.”

– Theo de Raadt

Legitimate Concerns

The BHyVe heavy lifting is done by:

`/usr/src/sys/amd64/vmm/intel/vmx.c`

1,300 LOC

My pseudo-package: 259K

The Future

Soon, in the words of Peter Grehan October 18th, 2012:

- ACPI Tables
- Guest Idle Detection (100% CPU usage) – Fixed in 10
- AHCI Device Emulation
- VirtIO MSIx Support

On the horizon:

- Takuya Asada's BIOS Emulation Work
- Build with CLANG – Simple fix in progress
- AMD Support – In progress

The Future

On the horizon:

- Better Integration with Host Scheduler
- Memory Over-Commit
- Suspend and Resume
- Generalization of CPUID Features for Guest Portability
- Sparse Images (QCOW, VDI, VMDK, ZVOL?)
- Non-tap VirtIO Back-End

My TO DO List

VT-d PCI Pass-Through

Probe and reject host device 1/0/0 for use in slot 5 in the BHyVe guest:

```
pciconf -lv | grep em0

em0@pci0:0:25:0:    class=0x020000 card=0x21ce17aa
chip=0x15028086 rev=0x04 hdr=0x00
```

```
/boot/loader.conf

pptdevs="0/25/0"                          ← Device ID
blackhole_load="YES"
```

```
bhyve ... -s 5,passthru,0/25/0 ...
```

My TO DO List

Serial Console Output:

`'boot h'` at the loader prompt or:

```
/boot/loader.conf
boot_serial="YES"

bhyve ... -S 2,uart,stdio ...
```

My TO DO List

NanoBHyVe

Mmmm... Dogfood

This presentation was brought to you using:

A Refurbished Lenovo ThinkPad T420

PC-BSD 9.0

LibreOffice (Broken spell check!)

Questions?

Thank you MeetBSD 2012 organizers,
sponsors and YOU!

Happy Hacking!