The bhyve Operator's Manual

Michael Dexter

AsiaBSDCon 2013

OVERVIEW

bhyve is a legacy-free Type-2 hypervisor for FreeBSD that was imported into the mainline FreeBSD development repository in January of 2013 with svn revision r245652. A hypervisor allow for the operation of one or more guest operating systems within a host operating system.

As a legacy-free hypervisor, a bhyve host requires the Extended Page Tables (EPT) feature found on "Nehalem" and newer generations of Intel processors. This requirement eliminates the need for memory management routines that are traditionally implemented in software and yields virtually bare metal guest performance. A bhyve guest requires VirtIO network and block devices, which were already available in FreeBSD 8-STABLE, 9-STABLE and 10-CURRENT at the time of bhyve's import. If these two requirements are satisfied, the bhyve host and guests will operate in the established FreeBSD manner.

HARDWARE REQUIREMENTS

The presence of the Extended Page Table (EPT) feature can be determined by examining the host's demesg(8) output for the presence of the POPCNT (POP Count) feature as the two are coupled but not related. Established dynamic memory and storage requirements apply otherwise with the caveat that there is a 1:1 relationship between the deduction of dynamic memory from the host and its allocation to guests.

SOFTWARE REQUIREMENTS

A FreeBSD 10-CURRENT system from svn revision r245652 onward will include all of the necessary bhyve host components: the vmm(4) kernel module, the libvmmapi library and the bhyveload(8), bhvye(8) and bhyvectl(8) utilities.

A suitable FreeBSD 8-STABLE, 9-STABLE or 10-CURRENT guest can exist in a disk image or any valid storage device and only requires a modified /etc/ttys entry to work. All other options can be specified at runtime at the loader prompt. Permanent configuration changes however are generally desired and will be demonstrated.

Permanent /etc/ttys configuration (can be appended):

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

Boot time or permanent /etc/fstab configuration for a MBR-partitioned device:

# Device	Mountpoint	FStype	Options	Dump	Pass#
/dev/vtbd0s1a	/	ufs	rw	1	1
or for a GPT partitioned des					

or for a GPT-partitioned device:

# Device	Mountpoint	FStype	Options	Dump	Pass#
/dev/vtbd0p1	/	ufs	rw	1	1

Example runtime or permanent /etc/rc.conf networking configuration:

ifconfig vtnet0="DHCP"

Depending on how your guest is built, the /boot/loader.conf may require:

```
virtio_load="YES"
if_vtnet_load="YES"
virtio_pci_load="YES"
virtio_blk_load="YES"
```

BHYVE OPERATION

At a minimum, a bhyve host requires that the vmm.ko kernel module be loaded:

su kldload vmm

To enable guest networking, load the if_tap.ko kernel module and create a tap(4) interface (tap0 on em0 as an example):

```
su kldload if_tap
su ifconfig tap0 create
su ifconfig bridge0 addm tap0 addm em0 up
su ifconfig tap0 up
```

The host and guest are now configured for bhyve operation. The bhyve guest bootstrap process is a two-stage operation not unlike that of the host. The bhyveload(8) utility loads the guest kernel into memory and the bhyve(8) utility executes it. These commands should be run in tight sequence with identical memory, storage device and guest name parameters.

All configuration parameters forward will be highly site-specific.

```
su /usr/sbin/bhyveload -m 256 -M 0 -d mydiskimage myguestname
su /usr/sbin/bhyve -a -A -m 256 -M 0 -I -H -g 0 \
-s 0:0,hostbridge \
-s 1:0,virtio-net,tap0 \
-s 2:0,virtio-blk,mydiskimage \
-S 31,uart,stdio \
myguestname
```

Assuming a successful boot, the guest should behave like a standard FreeBSD system and 'shutdown -p now' will cleanly shut it down.

If you wish to terminate a guest at the loader prompt, simply type 'quit'.

Executing guests can be viewed by name on the host system with:

```
su ls /dev/vmm
```

The above bootstrap procedure will not free the memory allocated to the guest but it can be freed using bhyvectl(8):

```
su /usr/sbin/bhyvectl --vm=myguestname --destroy
```

PRAGMATIC BHYVE GUEST OPERATION

The above procedures do not address the subjective matter of populating and configuring the guest userland or automating the guest operation lifecycle.

Two efforts exist to automate the building and operation of bhyve guests.

bhyve developer Neel Natu provides a script named 'vmrun.sh' that works in conjunction with a FreeBSD 10-CURRENT installation iso image named 'release.iso' to automate the creation of a bootable disk image named 'diskdev' that can be populated using the standard FreeBSD installer. This script provides an 8G disk image and 512M of dynamic memory by default but these parameters can be easily modified at runtime or by modifying the script.

The 'vmrun.sh' script requires only a guest name such as 'vm1':

su vmrun.sh vml

Mr. Natu's script and instructions can be obtained from:

```
http://people.freebsd.org/~neel/bhyve/vmrun.sh
http://people.freebsd.org/~neel/bhyve/bhyve_instructions.txt
```

A suitable 10-CURRENT 'release.iso' image can be retrieved from:

http://ftp4.us.freebsd.org/pub/FreeBSD/snapshots/amd64/amd64/ISO-IMAGES/10.0/

Note that the versioning prefix must be removed for the iso to work.

Alternatively, the bhyve.org web site provides a series of scripts that facilitate the creation and operation of bhyve guests in a highly-customizable manner.

While evolving, at the time of writing these scripts include:

0-make-softdevice.sh	Create and mount a disk image or zvol
1-format-device.sh	Format the disk image or zvol
1-format-zvol-gpt.sh	Format a zfs zvol with a GPT layout
2-install-guest.sh	Populate the disk image with the OS
3-host-prep.sh	Prepare the host for bhyve and networking
4-boot-guest.sh	Boot the guest
5-cleanup-guests.sh	Clean up after the guest
mount-diskdev.sh	Mount a specified disk image

In addition to the basic functionality provided by 'vmrun.sh', these scripts create and format disk images in MBR and GPT format, format zvol, iSCSI and hardware devices, populate FreeBSD 8-STABLE, 9-STABLE and 10-CURRENT userlands from the official mirrors and facilitate guest operation.

Any and all jail(8) and NanoBSD configuration strategies should be applicable to bhyve and the creation of "thick" and "thin" bhyve hosts and guests.

THE BHYVE DEVELOPMENT ENVIRONMENT

As a native FreeBSD 8-STABLE and later on FreeBSD 10-CURRENT computing environment, bhyve provides the ideal infrastructure for all categories of FreeBSD development. The hard partitioning provided by the vmm virtual machine manager allows the "unstable" FreeBSD 10-CURRENT environment to be used for "stable" FreeBSD 8 and 9 development with far fewer potential risks than emulated or jailed environments. Common development nuisances such as library incompatibilities are eliminated by the virtual machine divide.

The inclusion of dtrace(1M) in FreeBSD 10-CURRENT provides an unprecedented level of system introspection that should prove invaluable to FreeBSD, bhyve and future foreign guest operating systems.

Some rudimentary DTrace examples:

```
su kldload dtraceall
su dtrace -n 'io:::start /execname == "bhyve"/ { @[ustack()] = count(); }'
              libc.so.7`0x800ed3fda
             bhyve`pci_vtblk_qnotify+0x59
             bhyve`pci_vtblk_write+0x220
              bhyve`pci_emul_io_handler+0x16e
             bhyve`emulate inout+0x197
             bhyve`vmexit_inout+0x155
             bhyve`vm_loop+0x118
             bhyve`fbsdrun start thread+0x87
              libthr.so.3`0x800c53413
              505
```

The sysutils/DTraceToolkit port provides FreeBSD-specific DTrace scripts.

<pre>su /usr/share/dtrace/toolkit/hotuser -p `pgre</pre>	p -n bhyve	9	
Sampling Hit Ctrl-C to end. ^C			
FUNCTION		COUNT	PCNT
bhyve`vm_loop		1	1.8%
libvmmapi.so.5`0x800838920		1	1.8%
libthr.so.3`0x800c59491		1	1.8%
libthr.so.3`0x800c58df1		1	1.8%
libvmmapi.so.5`0x800838950		1	1.8%
libthr.so.3`0x800c5992f		1	1.8%
libc.so.7`0x800f83bb3		1	1.8%
bhyve`emulate_inout		1	1.8%
libc.so.7`0x800ed3fd0		1	1.8%
libthr.so.3`0x800c594c1		1	1.8%
bhyve`pci_vtblk_write		1	1.8%
bhyve`pci_vtblk_proc		1	1.8%
libvmmapi.so.5`0x800838a78		2	3.6%
libvmmapi.so.5`0x800838a2e		5	8.9%
bhyve`vmexit_inout		6	10.7%
libc.so.7`0x800f8875a		31	55.4%

/---- /-bara/dtraca/toolkit/batugar n `ngron n bhuuo`

su /usr/share/dtrace/toolkit/hotkernel

Sampling Hit Ctrl-C to end. ^C		
FUNCTION	COUNT	PCNT
vmm.ko`vm_guest_msrs	1	0.0%
vmm.ko`vcpu_set_state	1	0.0%
vmm.ko`vmx_setreg	1	0.0%
vmm.ko`vm_nmi_pending	1	0.0%
vmm.ko`vm_get_register	1	0.0%
vmm.ko`lapic_intr_accepted	1	0.0%
vmm.ko`vm_lapic	1	0.0%
vmm.ko`vlapic_op_mem_read	2	0.0%
vmm.ko`lapic_mmio_write	2	0.0%
vmm.ko`vlapic_intr_accepted	2	0.0%
vmm.ko`vm_set_register	2	0.0%
vmm.ko`lapic_pending_intr	3	0.0%
vmm.ko`vmm_fetch_instruction	3	0.0%
vmm.ko`vmmdev_ioctl	3	0.0%
vmm.ko`vmmdev_ioctl	3	0.0%
vmm.ko`vm_exitinfo	3	0.0%
vmm.ko`vcpu_stats	3	0.0%
vmm.ko`vmm_emulate_instruction	4	0.0%
vmm.ko`vmx_getreg	4	0.0%
vmm.ko`lapic_timer_tick	5	0.0%
vmm.ko`vm_gpa2hpa	7	0.0%
vmm.ko`vlapic_pending_intr	7	0.0%
vmm.ko`vlapic_op_mem_write	7	0.0%
vmm.ko`vmm_decode_instruction	19	0.0%
vmm.ko`vmcs_read	20	0.0%
vmm.ko`ept_vmmmap_get	21	0.0%
vmm.ko`vlapic_timer_tick	29	0.0%
vmm.ko`vm_run	30	0.0%
vmm.ko`restore_guest_msrs	32	0.0%
vmm.ko`restore_host_msrs	42	0.0%
vmm.ko`vlapic_update_ppr	142	0.1%
vmm.ko`vmx_run	33275	16.3%
kernel`acpi_cpu_c1	168750	82.6%

In addition to the DTrace suite, the bhyvectl(8) command provides extensive bhyve-specific profiling information:

su bhyvectlget-lowmemvm=guest0 lowmem 0x0000000000000000/2684	35456
su bhyvectlget-statsvm=guest0 vcpu0	
number of ticks vcpu was idle	5244097
number of NMIs delivered to vcpu	0
vcpu migration across host cpus	1186676
vm exits due to external interrupt	2229742
number of times hlt was ignored	0
number of times hlt was intercepted	2158532
vcpu total runtime	288974908299

BHYVE REFERENCE

The following options are available to the bhyveload(8) and bhyve(8) commands:

```
Usage: bhyveload [-d <disk image path>] [-h <host filesystem path>] [-m
<lowmem>] [-M <highmem>] <vmname>
Usage: bhyve [-aehABHIP] [-g <gdb port>] [-z <hz>] [-s <pci>] [-S <pci>]
[-p pincpu] [-n <pci>] [-m lowmem] [-M highmem] <vm>
       -a: local apic is in XAPIC mode (default is X2APIC)
       -A: create an ACPI table
       -q: qdb port (default is 6466 and 0 means don't open)
       -c: # cpus (default 1)
       -p: pin vcpu 'n' to host cpu 'pincpu + n'
       -B: inject breakpoint exception on vm entry
       -H: vmexit from the guest on hlt
       -I: present an ioapic to the quest
       -P: vmexit from the quest on pause
       -e: exit on unhandled i/o access
       -h: help
       -z: guest hz (default is 100)
       -s: <slot,driver,configinfo> PCI slot config
       -S: <slot,driver,configinfo> legacy PCI slot config
       -m: lowmem in MB
       -M: highmem in MB
       -x: mux vcpus to 1 hcpu
       -t: mux vcpu timeslice hz (default 200)
```

FUTURE DEVELOPMENTS

bhyve contains experimental PCI device pass-through support and is scheduled to include:

- AMD Virtualization Extensions
- Foreign Guest Operating System Support
- ACPI Guest Suspend and Resume
- Thin Provisioning of Memory
- Generalization of CPUID Features for Guest Migratability
- Sparse Image Support such as QCOW, VDI and VMDK
- Porting to other Host Operating Systems

bhyve is arguably the most compelling FreeBSD development since jail(8) and continues FreeBSD's tradition of providing innovative multiplicity options to operators and developers. jail(8) and bhyve are by no means mutually-exclusive technologies and should provide value when used in conjunction with one another in parallel or via encapsulation. bhyve also promises to make the most of recent FreeBSD features such as the DTrace, the ZFS filesystem and FreeBSD's virtual network stack. The fundamental point to remember about bhyve is that a FreeBSD bhyve system is simply a FreeBSD system that will fully leverage your FreeBSD administration and development experience.