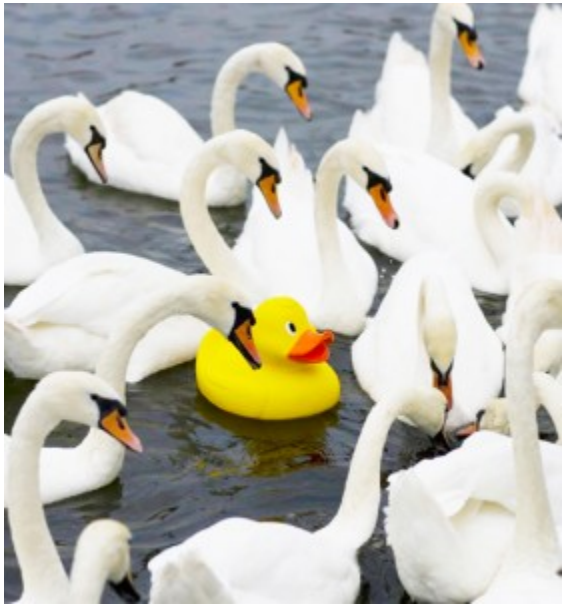
Institutionalizing FreeBSD Isolated and Virtualized Hosts Using `bsdinstall(8)`, `zfs(8)` and `nfsd(8)`

editor@callfortesting.org
@MichaelDexter

BSDCan 2018

Jails and bhyve…



FreeBSD’s had Isolation since 2000 and Virtualization since 2014

Why are they still strangers?

Institutionalizing FreeBSD Isolated and Virtualized Hosts Using `bsdinstall(8)`, `zfs(8)` and `nfsd(8)`

Integrating as first-class features

Institutionalizing FreeBSD Isolated and Virtualized Hosts Using `bsdinstall(8)`, `zfs(8)` and `nfsd(8)`

This example but *this is not* FreeBSD-exclusive

Institutionalizing FreeBSD Isolated and Virtualized Hosts Using `bsdinstall(8)`, `zfs(8)` and `nfsd(8)`

`jail(8)` and `bhyve(8)` “guests”

Application Binary Interface
vs.
Instructions Set Architecture

Institutionalizing FreeBSD Isolated and Virtualized Hosts Using `bsdinstall(8)` `zfs(8)` and `nfsd(8)`

The FreeBSD installer
The best file system/volume manager available
The Network File System

Broad Motivations

Virtualization!
Containers!
Docker!
Zones!
Droplets!
More more more!

My Motivations

2003: Jails to mitigate “RPM Hell”
2011: “bhyve sounds interesting...”
2017: Mitigating Regression Hell
2018: OpenZFS EVERYWHERE

A Tale of Two Regressions

Listen up.

Regression One

FreeBSD Commit r324161

“MFV r323796: fix memory leak in [ZFS] `g_bio` zone introduced in r320452”

Bug: r320452: June 28th, 2017

Fix: r324162: October 1st, 2017

3,710 Commits and
3 Months Later

June 28th through October 1st

BUT

July 27th, FreeNAS MFC

Slips into FreeNAS 11.1
Released December 13th

Fixed in FreeNAS January 18th

3 Months in
FreeBSD HEAD

36 Days in
FreeNAS Stable

TEST ALL THE THINGS!

Regression Two

FreeBSD Commit r317064

“Optimize pathologic case of `telldir()` for Samba.”

r235647: July 29th, 2014

to

r317064: April 17th, 2017

81,417 Commits and
3 *Years* Later

July 16th, 2014	FreeBSD 9.3
July 29th, 2014	Bug Introduced
January 20th, 2014	FreeBSD 10.0
November 14th, 2014	FreeBSD 10.1
December 31st, 2016	9.3 End of Life
April 17th, 2017	Resolved in FreeBSD
July 26th, 2017	Resolved in FreeBSD

The Regression Gap

November 14th, 2014 FreeBSD 10.1

December 31st, 2016 9.3 End of Life

July 26th, 2017 FreeBSD 11.1

Seven Months Off The Radar
Nine Months Of My Investigation

“Any effort spend in the past is deprived from CURRENT”

– Former FreeBSD Release Engineer

“The moment a regression is end-of-lifed, it becomes default behavior and infinitely more difficult to locate”

– Michael Dexter

Paleophobia Counseling

Don’t fear the past! Embrace it! It’s Static!!!

Rephrased: “I wouldn’t be looking into the past if you didn’t hide the regressions there!”

– Also Michael Dexter

The 25 Year Old BSD Bug

posted by Thom Holwerda on Sat 10th May 2008 20:27 UTC, submitted by rosebug

1983. The year of the IBM PC XT, the Apple Lisa, Pioneer 10 leaving the solar system, and Hooters opening up shop in Florida. It's also the birthyear of a 25 year old BSD bug, squashed only a few days ago.

A few days ago, Marc Balmer, OpenBSD developer, received an email from an OpenBSD user. The email claimed that SAMBA would crash when serving files off an MS-DOS filesystem. Balmer got into contact with a few SAMBA developers who claimed that SAMBA uses a special workaround in order to function properly on BSD systems: the code for reading directories in *all* BSDs was flawed.

Understandably, Balmer's first reaction was disbelief. *"Of course my first reaction was to blame Samba,"* he writes. Despite his initial reaction, he decided to dig deeper into this case, and he uncovered a bug that had been sitting in the code of all BSDs (including Mac OS X), including a lot of old releases. He confirmed the bug was already in 4.2BSD, released in August of *1983*.

FreeBSD 1.0 arrived in 1993…
UNIX V4 move to C was 1973…
A 25 ~ 45 Year Window!

Hypervisors to the rescue!

Incorporate them into your development and testing

Ideally over 45 years...
(But 15 will have to do)

See: Isolated Build Environments

`/boot/kernel` layout arrived in 5.0
and boots in `bhyve(8)`

Retroactive `bsdinstall(8)`
if repackaged

...which arrived in 9.0

Two habits must change...

DECOUPLE INSTALLATION VERSIONS FROM INSTALLERS

DECOUPLE INSTALLATION PROCEDURES FROM NEW HARDWARE


ARRÊT
STOP
Windows 10
Disk 1 of 2079
Windows Series
Disk format High-density (1.44mb)
Version 10
Microsoft
Setup instructions
1. Inser the disk in drive A
2. Type and press ENTER
3. Type and Setup and press ENTER
©1985 - 2015 Microsoft Corporation
All Rights Reserved

`bsdinstall(8)` Hacks:

Avoid zpool name collision
Add ZFS-booted Host support
Optionally keep destinations mounted
Optionally pull boot blocks from destination
Remove some `dialog(1)` dependencies
Support “nested” boot environments

`bsdinstall(8)` is the Official FreeBSD Installer

Pros:

Largely `/bin/sh`, C for UFS
Supports many partitioning schemes
Supports UFS and ZFS, GELI
Supports simple `jail(8)` guests
Suddenly Supports FreeBSD 5.0 onward

`bsdinstall(8)` Cons:

Assumes a fresh installation
Assumes host revision = guest revision
Dependence on `bsdconfig(8)`
Dependence on `dialog(1)`
C-based components are complex
Traps `/bin/sh 'exit'` statements

Nested Boot Environments

```
# zfs list
zroot/ROOT/default                1.04M   195G    96K  /
zroot/ROOT/default/tmp              88K   195G    88K  /tmp
zroot/ROOT/default/usr             352K   195G    88K  /usr
zroot/ROOT/default/usr/home         88K   195G    88K  /usr/home
zroot/ROOT/default/usr/ports        88K   195G    88K  /usr/ports
zroot/ROOT/default/usr/src          88K   195G    88K  /usr/src
zroot/ROOT/default/var             528K   195G    88K  /var
zroot/ROOT/default/var/audit        88K   195G    88K  /var/audit
zroot/ROOT/default/var/crash        88K   195G    88K  /var/crash
zroot/ROOT/default/var/log          88K   195G    88K  /var/log
zroot/ROOT/default/var/mail         88K   195G    88K  /var/mail
zroot/ROOT/default/var/tmp          88K   195G    88K  /var/tmp
```

Nested Boot Environments

```
zroot/ROOT/default              1.04M   195G    96K  /
zroot/ROOT/default/tmp            88K   195G    88K  /tmp
zroot/ROOT/default/usr           352K   195G    88K  /usr
...
zroot/ROOT/current              1.04M   195G    96K  /
zroot/ROOT/current/tmp            88K   195G    88K  /tmp
zroot/ROOT/current/usr           352K   195G    88K  /usr
...
zroot/ROOT/illumos              1.04M   195G    96K  /
zroot/ROOT/netbsd               1.04M   195G    96K  /
...
```

Nested Boot Environments

/etc/rc.d/zfsbe

```
zfs list -rH -o mountpoint,name,canmount,mounted \
      -s mountpoint -t filesystem $_be | \
      while read _mp _name _canmount _mounted ; do
         # skip filesystems that must not be mounted
         [ "$_canmount" = "off" ] && continue
         [ "$_mounted" = "yes" ] && continue
            case "$_mp" in
            "none" | "legacy" | "/" | "/$_be")
                  ;;
            "/$_be/"*)
                  mount -t zfs $_name ${_mp#/$_be}
                  ;;
            *)
                  zfs mount $_name
```

Scripted bsdinstall(8)

```
export BSDINSTALL_DISTDIR="/pub/FBSD/.../12.0-CURRENT"
export ZFSBOOT_DISKS="md0"
export ZFSBOOT_PARTITION_SCHEME="GPT"
export ZFSBOOT_POOL_NAME="zroot"
export ZFSBOOT_BEROOT_NAME="ROOT"
export ZFSBOOT_BOOTFS_NAME="default"
export ZFSBOOT_DATASET_NESTING="1"
export BOOT_BLOCKS_FROM_DISTSET="1"

# Alternative UFS layout
#export PARTITIONS="md0 {512M freebsd-ufs /, \
100M freebsd-swap, 512M freebsd-ufs, /var, \
auto freebsd-ufs /usr }"
```

Scripted `bsdinstall(8)`

```
# mdconfig -t malloc -s 4G
md0
# bsdconfig script <the script>
# sh /usr/share/examples/bhyve/vmrun.sh \
-m 2G -d /dev/md0 vm
```

You *could* wrap the generation of such scripts in a framework

#AchievementUnlocked

`bsdinstall(8)` can suddenly generate block storage-backed virtual machines using the in-base installer

#Institutionalized

#AchievementUnlocked

Add a “`vmtab`”
Add an `rc` script
Rejoice!

#ArguablyInstitutionalized

Bonus: You can already boot a fresh installation with vmrun.sh!

#NotSoFast

AHCI: Only 8.4 onward
(Shorter regression window)
Block devices are limiting
Other OS Support?

I ♥ ZFS

I ♥ Boot Environments

I ♥ *BSD Unix

I ♥ ZFS

Great Storage Architecture

Test Every OpenZFS OS!

… but, only proprietary operating systems care where they boot

Why limit yourself?

Show the thing...

Networked Boot Environments

#WAT?

Root on NFS since day one

Longer than NVMe
Longer than SATA AHCI
Longer than IDE...

Conceptually…

`zfs set sharenfs=on zroot/ROOT/head`

But “`sharenfs`” is fragile

Follow `/etc/rc.d/zfsbe`

Now What?

`mount -t zfs /ROOT/head/` ...
`chroot(8)` or `jail(8)` `/ROOT/head/` ... or ...
Export `/ROOT/head/` over NFS ...

```
# cat /etc/exports
/ROOT/head -maproot=root -network 192.168.2.0 -mask 255.255.255.0
/ROOT/head/tmp -maproot=root -network 192.168.2.0 -mask 255.255.255.0
/ROOT/head/usr/home -maproot=root -network 192.168.2.0 -mask 255.255.255.0
/ROOT/head/usr/ports -maproot=root -network 192.168.2.0 -mask 255.255.255.0
/ROOT/head/usr/src -maproot=root -network 192.168.2.0 -mask 255.255.255.0
/ROOT/head/var/audit -maproot=root -network 192.168.2.0 -mask 255.255.255.0
/ROOT/head/var/crash -maproot=root -network 192.168.2.0 -mask 255.255.255.0
/ROOT/head/var/log -maproot=root -network 192.168.2.0 -mask 255.255.255.0
/ROOT/head/var/mail -maproot=root -network 192.168.2.0 -mask 255.255.255.0
/ROOT/head/var/tmp -maproot=root -network 192.168.2.0 -mask 255.255.255.0
```

Housekeeping

github.com/stblassitude/boot_root_nfs

```
# bhyveload -h /ROOT/head \
-e boot.netif.name=vtnet0 \
-e boot.netif.hwaddr=02:01:02:03:04:05 \
-e boot.netif.ip=192.168.2.202 \
-e boot.netif.netmask=255.255.255.0 \
-e boot.nfsroot.server=192.168.2.1 \
-e boot.nfsroot.nfshandle=X631083b5dea37b8... \
-e boot.nfsroot.nfshandlelen=28 \
-e boot.nfsroot.path=/ROOT/head \
-e vfs.root.mountfrom=nfs:192.168.1.1:/ROOT/head \
-e vfs.root.mountfrom.options=rw \
-m 1024 head
```

Housekeeping

/ROOT/head/etc/fstab

```
192.168.2.1:/be/head/tmp /tmp nfs rw,noatime,async 0 0
192.168.2.1:/be/head/usr/home /usr/home nfs rw,noatime,async 0 0
192.168.2.1:/be/head/usr/ports /usr/ports nfs rw,noatime,async 0 0
192.168.2.1:/be/head/usr/src /usr/src nfs rw,noatime,async 0 0
192.168.2.1:/be/head/var/audit /var/audit nfs rw,noatime,async 0 0
192.168.2.1:/be/head/var/crash /var/crash nfs rw,noatime,async 0 0
192.168.2.1:/be/head/var/log /var/log nfs rw,noatime,async 0 0
192.168.2.1:/be/head/var/mail /var/mail nfs rw,noatime,async 0 0
192.168.2.1:/be/head/var/tmp /var/tmp nfs rw,noatime,async 0 0
```

But That’s Hard!

/ROOT/head...

Boot bare metal thanks to `zfsbe`

Mount and contain with `chroot(8)`

Mount and boot with `jail(8)`

Export/boot w/ `bhyveload(8)`/`bhyve(8)`

(Add `TFTPd` and `DHCPd` and ...)

Boot with `bhyve(8)` UEFI-GOP PXE

Boot with Xen PXE or ...

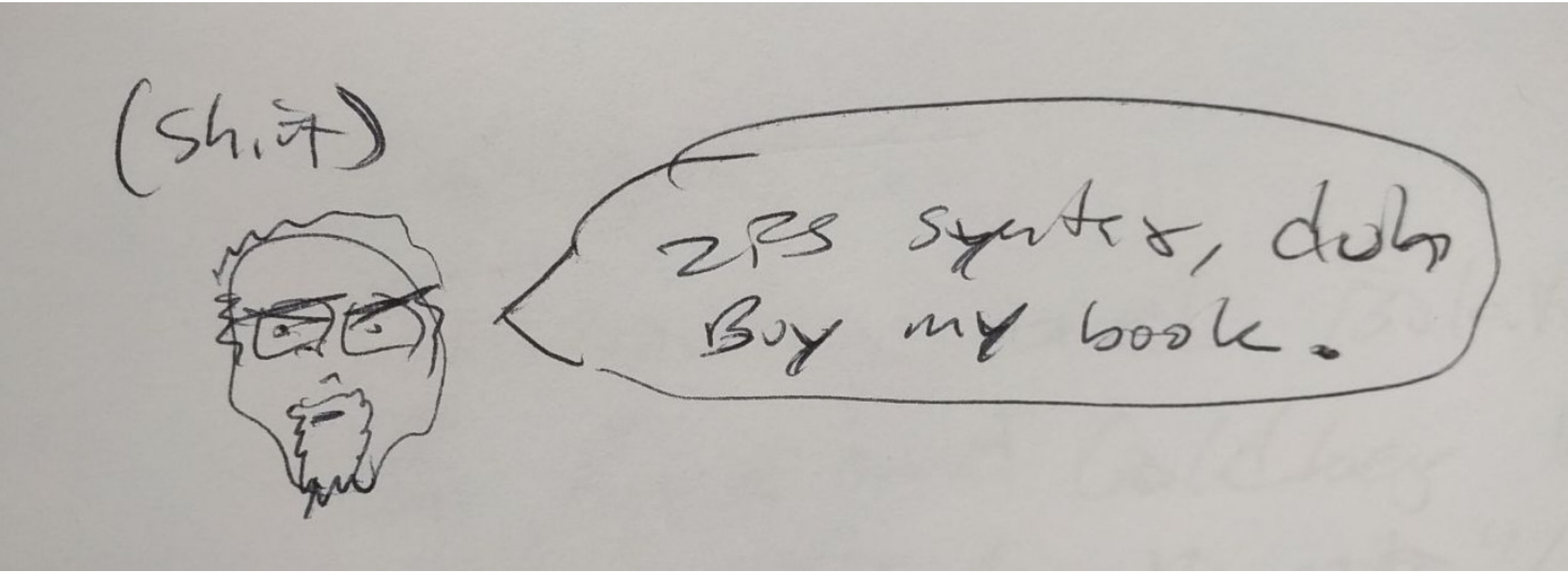
Boot bare metal over the LAN via PXE

Oh, the Places You'll Go!

File-level virtual machines!

Proof of Concept

be(8)


(Sh.it)
ZFS syntax, duh
Buy my book.

```
# be create -l freebsd bd/be/test
# be mount bd/be/test
# be install -p /pub -o FreeBSD \
-a amd64 -b release -r 11.1 bd/be/test
# be sharenfs bd/be/test
# be bootnfs bd/be/test
...
# be sharepxe bd/be/test
# be bootpxe bd/be/test
...
# be WoL 02:01:02:03:04:05
...
```

```
# be create -l flat bd/be/files
# be sharenfs bd/be/files
```

Whoops!

A ZFS-aware NAS system
in two commands

Sorry about that!

Challenges

NFS: “Not a File System”

“Database” Locking

NFS Locking Solutions

FreeBSD 6.0+ “diskless”

GSoC? Audit r/o and root on NFS

Device Support

8.4 Onward VirtIO
8.0 Onward AHCI
5.2 Onward New e1000
5.1 Downward ne2000
ATA Emulation Fail...

Next Time...

bd(8)

Block Device Utility

Block Devices

+

File-Level OS

=

Installer

(and NAS?)

Philosophical Challenges

Oh No! Not `/bin/sh` !

You can *only* write it in…

C

~~Python~~

~~Ruby~~

~~Go~~

~~Lua~~

Rust

...

sh, sed, awk…

Twenty years of installer/configurator refinement *sure* would've been nice...

And… would support FreeBSD 1.0 ~ 12.0

Forklift upgrades should be a warning

They’re called Run Control Scripts for a reason

Let the big iron do the heavy lifting and get out of the way

Lessons learned from

Seven lucky years of user feedback...

The Network Engineer

“I need infinitely-configurable networking, but make storage and applications brain-dead simple.”

The Storage Engineer

“I need infinitely-configurable storage, but make networking and applications brain-dead simple.”

The Software/DevOps Engineer

“I need infinitely-configurable applications, but make networking and storage brain-dead simple.”

Sane Defaults
Plus Overrides
WORKS

Configuration files are great but the command line works on read-only file systems

vmrun.sh win, VBox fail

EYES ON THE PRIZE

Regression Hunting

```
cat releases.txt | while read release
  do
    be create -l flat bd/be/r$release
    be jail bd/be/rel$release & (run tests)
  done
```

Regression Hunting

Is the manual page ratio improving or regressing?

How far will each release build ahead and behind?

Bisect to hunt individual regressions...

More Housekeeping

Improve `ftp-archive.freebsd.org`

Repackage 5.0 Onward (Done!)

r/o and NFS Audit (GSoC?)

`src.conf` Audit (90% Done!)

Packaged Base! (4 Unique Efforts!)

Why are you doing this? Seriously?

Scripted Installer

+

Hardware/Software-Agnostic Hosts

+

`chroot(8)`/`jail(8)` Isolation

+

`bhyve(8)`/Xen/vmm Virtualization

+

Configurable Userland (`src.conf`)

=

Most Docker-y stuff using entirely in-base Unix tools

or…

Institutionalized Isolated and Virtualized Hosts

Raising the question…

Does the Container movement expose ***flaws*** in the Unix computing model, or ***misunderstandings*** of the Unix computing model?

Thank You!

Any Questions?

editor@callfortesting.org
@michaeldexter