BSD Multiplicity: An applied survey of BSD multiplicity and virtualization strategies from chroot to BHyVe

Michael Dexter

BSD.lv Project – Call For Testing

EuroBSDCon 2011

Related Materials

Presentation and How-To's: multiplicity.bsd.lv

`sendbug`: editor@callfortesting.org

Updates and Corrections Welcome

BSD Multiplicity: Context

The non-conflicting *plurality* of conventionally-singular configured *execution environments...*

...each the *context* that each layer of the Unix model provides to the layer above it

Specifically within Berkeley Unix systems and BSD-licensed utilities

BSD Multiplicity: Context

Conventional Unix Layers

(NETWORK) HOST – RFC's

(USERLAND) INSTANCE – API's
KERNEL – ABI
(HARDWARE) MACHINE – ISA

Instruction Set Architecture – Application Binary Interface
Application Programming Interfaces – “Request For Comments”
TCP/IP-based network protocols

BSD Multiplicity: Context

Pluralized Unix Layers

HOST – HOST – HOST...

INSTANCE – INSTANCE – INSTANCE...
KERNEL – KERNEL – KERNEL...
MACHINE – MACHINE – MACHINE...

BSD Multiplicity: Context

A Sea of Choices

VMWare, Linux KVM, chroot, FreeBSD jail, Solaris Zones, VirtualBox, sysjail, EC2, Xen, Dragonfly BSD vkernel, Parallels, Linux Vserver, UserModeLinux, SIMH, Microsoft Hyper-V, QEMU, Virtuoso, GXemul, Linux OpenVZ, Virtual PC, Sun xVM and of course...

BSD Multiplicity: Context

Cloud Computing

Image courtesy of the US Army

BSD Multiplicity: Context

My Motivations

1991 – Desire for my own box

1998 – `'rpm -ivh *.rpm'`

(Oops! Re-install the OS)

BSD Multiplicity: Context

The solution in the broadest sense:

The separation, compartmentalization, containment, imprisonment or *isolation* of filesystems, applications and/or users.

BSD Multiplicity: Context

Additional Motivations

Cross-platform development
and system administration

The *consolidation* of systems,
even if dissimilar

BSD Multiplicity: Context

Long-term Motivation

1991: Hundreds of simultaneous users on a 33MHz Sun 4/490 (my first dexter@)

Future: A private instance for thousands of users on commodity hardware

BSD Multiplicity: Considerations

Administrative:

Storage Devices

Network Devices

Console Devices

Kernels

Userland

BSD Multiplicity: Considerations

Administrative: Storage Devices

```
# dd if=/dev/zero of=1GB.img
    bs=1024 count=1000000
```

BSD Multiplicity: Considerations

Administrative: Network Devices

Is a network address provided from within or outside a system?

To what real or virtual hardware device?

BSD Multiplicity: Considerations

Administrative: Guest Console

Host Console? Xwindow?
SSH? Serial Port? VNC?

BSD Multiplicity: Considerations

Administrative: Kernels

Is the kernel provided from within or outside a system?

Is it stock? Modified?

BSD Multiplicity: Considerations

Administrative: Userland

Is it stock? Reduced? Crunched?

make.conf SKIPDIR site.tgz...

TinyBSD – NanoBSD – miniBSD

BSD Multiplicity: Survey Criteria

A rigorous analysis of theory, taxonomy, security and performance are beyond the scope of this survey

Many papers address these but "Virtualization" and now "Cloud" have been hijacked by marketing departments to represent just about anything

BSD Multiplicity: Survey Criteria

Goal of this presentation:

Inspire your experimentation with these strategies by conveying their relative strengths and weaknesses and establishing expectations relating to their configuration, administration and use

MACHINE – KERNEL – INSTANCE – HOST

Hardware Multiplicity

Easy: Buy more machines

Excellent isolation

Disadvantage: High cost

MACHINE – KERNEL – INSTANCE – HOST

Honorable mention:

The blade server: “a stripped down server computer with a optimized to minimize the use of physical space and energy.”

K&R and the CSRG did not have this option

Quotation courtesy of Wikipedia

MACHINE – KERNEL – INSTANCE – HOST

Honorable mention:

Xen: (GPL licensed kernel) with very good NetBSD host support

Good effort at a Popek and Goldberg Virtual Machine Monitor

Formal Requirements for Virtualizable Third Generation Architectures - 1974

MACHINE – KERNEL – INSTANCE – HOST

Xen shortcomings:

x86 MMU has performance issues

Fidelity issues: Highest performance requires a modified guest OS (Paravirtualized mode)

MACHINE – KERNEL – INSTANCE – HOST

Example configuration

```
kernel  = "/root/netbsd-5.1-XEN3_DOMU.gz"
#kernel  = "/root/netbsd-5.1-INSTALL_XEN3_DOMU.gz"
Memory  = 64
name    = 'NetBSD'
vif     = [ 'mac=00:16:3e:00:00:11, bridge=bridge0' ]
disk    = [ 'file:/root/netbsd.img,0x1,w' , \
'phy:/dev/cd0a,ioemu:hdc:cdrom,r' ]
root = "xbd0"
```

MACHINE – KERNEL – INSTANCE – HOST

Detailed NetBSD Xen instructions are at

multiplicity.bsd.lv

Nice! Disk images from privileged instances can be shared with QEMU

MACHINE – KERNEL – INSTANCE – HOST

Honorable mention:

FreeBSD and NetBSD on Amazon EC2

www.daemonology.net/freebsd-on-ec2/

wiki.netbsd.org/amazon_ec2/

MACHINE – KERNEL – INSTANCE – HOST

Honorable mention:

Software Virtual Machines

(Emulators)

SIMH, GXemul

MACHINE – KERNEL – INSTANCE – HOST

SIMH Vax

```
load -r /usr/pkg/share/simh/ka655.bin
set cpu 64m
at nvr openbsd.nvram
deposit rq qtime 1000000
set rq0 ra92
at rq0 vax.img
set rq1 cdrom
at rq1 install43vax.iso
set rq2 cdrom
at rq2 floppy43.fs ...
```

MACHINE – KERNEL – INSTANCE – HOST

```
at xq0 vr0
boot cpu
exit
>>>boot dua2:
```

Retro – Interactive – Finicky – Slow

Requirements vary by machine

Wide range of machines incl. PDP-11

MACHINE – KERNEL – INSTANCE – HOST

GXemul - DECstation 5000/200

```
startx
gxemul -e 3max -d nbsd_pmax.img -d \
b:pmaxcd-4.0.iso -M 64
gxemul -X -Y 2 -e 3max -d pmax.img \
-d b:pmaxcd-4.0.iso
```

Familiar syntax – Slow

ARM – MIPS – PowerPC – SuperH

MACHINE – KERNEL – INSTANCE – HOST

Honorable mention:

QEMU/kQEMU (GPL-licensed)

```
startx
qemu -hda i386.img -cdrom install49.iso \
-boot -d -m 64
qemu -hda i386.img -boot c -m 64
```

Flexible – Proven

MACHINE – KERNEL – INSTANCE – HOST

Honorable mention:

qemu-img utility

```
qemu-img info -f raw guest.vmdk
qemu-img convert guest.vmdk -O raw guest.img
```

BSD Licensed!

MACHINE – KERNEL – INSTANCE – HOST

Honorable mention:

VirtualBox (GPL-licensed)

Now owned by Oracle

Fast – Flexible – Professional

MACHINE – **KERNEL** – INSTANCE – HOST

BHyVe BSD Hypervisor

"Your work with BHyVe is the first independent validation of our code base."

– Neel Natu

MACHINE – **KERNEL** – INSTANCE – HOST

BHyVe BSD Hypervisor

A type-2 “Hosted” Hypervisor for FreeBSD that is under active development with the goal of hard logical partitioning

Supports PCI pass-through for storage and network devices

See also BSDCan 2011 BHyVe Presentation

MACHINE – **KERNEL** – INSTANCE – HOST

BHyVe BSD Hypervisor

Requirements:

VMX (VT-x) and EPT (Nested Page Tables)

vmm.ko kernel module on the host

A modified guest at this time

MACHINE – **KERNEL** – INSTANCE – HOST

BHyVe Host Configuration

```
pkg_add -r subversion-freebsd binutils
svn co http://svn.freebsd.org/base/projects/
  bhyve /usr/src/
```

(Make and build GENERIC kernel and world on host)

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

(For the guest kernel)

```
/boot/loader.conf
hw.physmem="0x100000000"
```
(Limit host's RAM to 4GB)

MACHINE – **KERNEL** – INSTANCE – HOST

BHyVe Execution

```
kldload vmm
  kldload if_tap
  ifconfig tap0 create
  kldload bridgestp
  kldload if_bridge
  ifconfig bridge0 create
  ifconfig bridge0 addm em0
  ifconfig bridge0 addm tap0
  ifconfig bridge0 up
  ifconfig tap0 up
```

```
cd /usr/share/vm1
```
(Preconfigured guest with a disk image)

```
  sh vmrun.sh vm1
```

MACHINE – **KERNEL** – INSTANCE – HOST

```
Launching virtual machine "vm1" with 768MB memory
  below 4GB and 2048MB memory above 4GB …
  Consoles: userboot

FreeBSD/amd64 User boot, Revision
  1.1(neel@freebsd.org, Sun Sep 25 22:19:14 PDT
  2011)Loading /boot/defaults/loader.conf /boot//
  kernel/kernel text=0x41e94f
  data=0x57ac0+0x273590
  syms=[0x8+0x737b8+0x8+0x6abe3]/boot//kernel/
  virtio.ko size 0x4ad8 at 0xbc8000/boot//kernel/
  if_vtnet.ko size 0xac80 at 0xbcd000/boot//
  kernel/virtio_pci.ko size 0x56c0 at 0xbd8000/
  boot//kernel/virtio_blk.ko size 0x4f60 at
  0xbde000
```

(Enters boot screen)

MACHINE – **KERNEL** – INSTANCE – HOST

BHyVe Guest Image Preparation

mkdir /usr/share/vm1

```
  mkdir /usr/share/vm1/boot
  mkdir /usr/share/vm1/boot/kernel

cd /usr/share/vm1
  cp ${OBJDIR}/sys/boot/userboot/userboot/
  userboot.so .

# create the 32MB virtio backing disk device
  dd if=/dev/zero of=diskdev count=32768 bs=1024

cd /usr/share/vm1/boot
  cp /boot/*.4th boot
  cp -a /boot/defaults .
  cp /boot/loader.help .
  cp /boot/loader.rc .
  cp /boot/menu.rc .
```

MACHINE – **KERNEL** – INSTANCE – HOST

Guest `loader.conf`

```
kernel="/kernel"

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

kern.hz="100"
  hw.pci.enable_msix="0"
  hw.pci.honor_msi_blacklist="0"bootverbose="1"

mfsroot_load="YES"
  mfsroot_type="mfs_root"
  mfsroot_name="mdroot"
```

(Finally copy in built guest kernel and modules)

MACHINE – **KERNEL** – INSTANCE – HOST

BHyVe TO DO

- Support other operating systems such as Linux and Windows
- Emulation of legacy devices (UART, VGA, IDE) and possibly BIOS INT call emulation (Works around the BIOS emulation requirement for FreeBSD by modifying the loader to run on top of the hypervisor but may not have the same luxury for other OS's.)
- IOAPIC emulation and instruction emulation.
- Need AMD/SVM support in BHyVe (work in progress).
- Better integration with the host's scheduler, virtual memory system and to possibly allow the host to be more aware of virtual CPU threads.

MACHINE – **KERNEL** – INSTANCE – HOST

BHyVe TO DO

- Implement memory over-commit. “The KVM hypervisor supports overcommitting CPUs and memory. Overcommitting is the process of allocating more virtualized CPUs or memory than there are physical resources on the system. CPU overcommit allows under-utilized virtualized servers or desktops to run on fewer servers which saves power and money.”

Currently, memory has to be stolen from FreeBSD at boot-time. It would be useful to a) grab free pages from the system to build the guest's address space, and b) somehow hook this into the user-space bhyve process to allow overcommit. Important to FreeBSD for hosting virtual machines.

Quotation courtesy of Wikipedia

MACHINE – **KERNEL** – INSTANCE – HOST

BHyVe TO DO

- The same goes for CPU resource allocation: CPU. BHyVe will use 100% of a CPU even when the guest is idle. Great for hard logical partitioning but not for running a bunch of VM's.
- There are also packaging aspects of it that we have not yet explored (such as an embedded hypervisor packaged with VM creation and management tools).

Current work is on high-performance paravirtualized I/O.

MACHINE – **KERNEL** – INSTANCE – HOST

BHyVe TO DO

- Suspend/resume/pause support. Requires adjusting the TSC value that the guest sees rather than have it free-run. Not sure how to resume on a machine that has a different frequency, though it may not be too hard.
- Hiding CPUID features. It's currently pass-through (once again, great for hard partitioning where you want full use of the underlying hardware), but for migration, features need to be hidden since they may not exist on the machines the VM is being moved to.
- Probably much more. The most important thing is to get people interested and working on it to fill the gaps: way too many for just a few people.

MACHINE – **KERNEL** – INSTANCE – HOST

BHyVe TO DO

- Boot loader improvements: the boot-loader runs as a separate process in user-space. For legacy OS's, it might be worthwhile to pull in some of the old BSD dosrun BIOS emulation and allow a sector-0 boot. This would require either 16-bit software emulation for the early part of the boot or AMD/SandyBridge 'unrestricted guest' support.
- The virtio block code uses raw disk images. It would be useful to have a sparse filesystem representation such as Qemu COW2 or Vmware VMDK support.
- The virtio net code can only interface to a tap device backend. This should be configurable as can be done with QEMU.

MACHINE – **KERNEL** – INSTANCE – HOST

BHyVe Status

“As far as the bigger picture is concerned, we would like people to start hacking on it. The code is pretty simple and we'll be more than happy to help anybody get going.”

– Neel Natu

“Running the modified “BHYVE” kernel config on a FreeBSD host works very well. I did notice that SMP with a 9.0 guest is currently broken beyond 2 vCPUs with a TSC sync issue (that doesn't exist in 8.1; 8 vCPUs works there), but other than that it should be fine.”

– Peter Grehan

MACHINE – **KERNEL** – INSTANCE – HOST

Honorable mention:

DragonflyBSD `vkernel(7)`

```
./boot/kernel -m 64m -r \
root.img -I auto:bridge0
```

MACHINE – **KERNEL** – INSTANCE – HOST

Honorable mention:

NetBSD/usermode

“What I've done is created a port of NetBSD which runs as a userspace application and has full access to libc.”

– Jared D. McNeill

MACHINE – KERNEL – INSTANCE – **HOST**

BSD Breeding Ground

`chroot`, FreeBSD/PC-BSD `jail`
DragonFly Jail, `sysjail`, kauth Jail,
mult process jailing

MACHINE – KERNEL – INSTANCE – **HOST**

`chroot(8)`

Committed by Bill Joy on March 18th 1982

17 months before 4.2BSD

Proved useful for building the system

MACHINE – KERNEL – INSTANCE – **HOST**

`chroot(8)`

"The chroot system call was first added to provide an alternate build environment for the system. It was later adapted to isolate anonymous **ftp** access to the system.

The original intent of chroot was not to ensure security. Even when used to provide security for anonymous **ftp**, the set of operations allowed by **ftp** was carefully controlled to prevent those that allowed escape from the chrooted'ed environment."

– McKusick, Neville-Neil

(The Design and Implementation of the FreeBSD Operating System)

MACHINE – KERNEL – INSTANCE – **HOST**

So began nearly 30 years of wack-a-mole...

MACHINE – KERNEL – INSTANCE – **HOST**

"Change root" modifies the vnode of a given directory from the perspective of a provided command. Any library dependencies of that command must be satisfied *within* the directory as the command will not see outside of it. Such dependencies can be determined with the `ldd` command.

MACHINE – KERNEL – INSTANCE – **HOST**

...not to mention *functional* isolation. One or more restrictions are applied to create a restrictive context where none existed.

Privacy concern: the one-way mirror

The chrooted binary can't see out but everyone else can see in.

Alas, some moles can break out of jail.

MACHINE – KERNEL – INSTANCE – **HOST**

Filesystems, not disks.

Though a mounted disk image could be used for a chrooted directory

Used daily by millions of chrooted daemons and institutionalized in OpenBSD

MACHINE – KERNEL – INSTANCE – **HOST**

```
root@test.bsd.lv:/root$ ldd `which sh`
/bin/sh:
        -ltermcap.0 => /lib/libtermcap.so.0
        -ledit.2 => /lib/libedit.so.2
        -lc.12 => /lib/libc.so.12
```

Our first userland concerns...

MACHINE – KERNEL – INSTANCE – **HOST**

FreeBSD/PC-BSD/DragonFly BSD Jail

Super `chroot`

Adds networking, processes, users,
a *root* user

MACHINE – KERNEL – INSTANCE – **HOST**

If your user account feels inadequate,

this is what you really want.

```
jail /usr/jail/ myjail 192.168.1.10 \
/bin/sh /etc/rc
```

A directory – A hostname – An IP – `rc`! - `sshd`!

(`jail` brought me back to BSD in 2002)

MACHINE – KERNEL – INSTANCE – **HOST**

Essentially a “real” host with as many filesystem, disk image, network and userland knobs you can turn.

MACHINE – KERNEL – INSTANCE – **HOST**

The official `jail` management mechanisms introduced in FreeBSD 5.0 (not available in DragonFly BSD). PC-BSD introduced the Warden `jail` management tool.

A script that loosely follows `adduser` syntax, uses disk images and creates `rc` scripts can be found at multiplicity.bsd.lv

MACHINE – KERNEL – INSTANCE – **HOST**

Honorable mention:

`sysjail`

As the `systrace` device offers the process interception that characterizes functional isolation, a faithful `jail` clone could be build atop it for use with OpenBSD and NetBSD.

MACHINE – KERNEL – INSTANCE – **HOST**

`sysjail` has been theoretically and perhaps practically undermined for security purposes by `systrace` vulnerabilities yet remains useful for trusted isolation.

See Robert Watson's “woot” talk on `systrace` vulnerabilities for more information.

MACHINE – KERNEL – INSTANCE – **HOST**

Honorable mentions:

A `kauth`-based Jail for NetBSD

Process jailing for OpenBSD

Debian/kFreeBSD `jail`

compat_linux may support Linux `rc` in `jail`

MACHINE – KERNEL – **INSTANCE** – HOST

What would *instance* multiplicity look like?

A single kernel *could* support multiple `init` processes booted from separate storage devices, provided that context was given to conventionally-global constructs such as the process table and user table.

MACHINE – KERNEL – **INSTANCE** – HOST

What would *instance* multiplicity look like?

Entire instances from the `init` process on down need only be placed in a structure not unlike that of a jail, provided the *appropriation* of said global constructs, plus some form of management mechanism for the resulting federated instances.

MACHINE – KERNEL – **INSTANCE** – HOST

The mult project explores this approach using NetBSD 3.1

The result is *logical,* rather than *functional* isolation

MACHINE – KERNEL – **INSTANCE** – HOST

Originally intended for grid computing but is applicable to public, private and anonymous “cloud” computing.

mult.bsd.lv

MACHINE – KERNEL – INSTANCE – HOST

Very Honorable Mention:

Network Stack Virtualization

wiki.freebsd.org/Image/VNETSamples

www.openbsd.org/papers/f2k9-vrf/

MACHINE – KERNEL – INSTANCE – HOST

Review: machine multiplicity

HOST – HOST – HOST

INSTANCE – INSTANCE – INSTANCE
KERNEL – KERNEL – KERNEL
MACHINE – MACHINE – MACHINE

MACHINE – **KERNEL** – INSTANCE – HOST

Review: kernel multiplicity

HOST – HOST – HOST

INSTANCE – INSTANCE – INSTANCE
KERNEL – KERNEL – KERNEL
MACHINE

MACHINE – KERNEL – INSTANCE – **HOST**

Review: host multiplicity

HOST – HOST – HOST

INSTANCE
KERNEL
MACHINE

MACHINE – KERNEL – **INSTANCE** – HOST

Review: instance multiplicity

HOST – HOST – HOST

INSTANCE – INSTANCE – INSTANCE
KERNEL
MACHINE

MACHINE – KERNEL – **INSTANCE** – HOST

Review: instance multiplicity

HOST – HOST – HOST

INSTANCE – INSTANCE – INSTANCE
KERNEL
MACHINE

MACHINE – KERNEL – INSTANCE – HOST

Thank you!

Please explore these!

Thank you EuroBSDCon organizers!