Practical ZFS For A Common FreeBSD User

Andriy Gapon <avg@FreeBSD.org>

KyivBSD, 2012

Outline

Introduction

- What this talk is *not* about
  - Complex vdev setups
  - "Enterprise-ish" setup: spares, ZIL/log and L2ARC/log devices, deduplication
  - Big iron (controllers, expanders, backplanes, multipath, huge memory, etc)
  - Tiny iron (ZFS in 64MB of RAM)
  - Setup for special purpose servers (file servers, etc), hardcore tuning
- What this talk is about
  - "Normal" setup for workstation / desktop use
  - Why ZFS is worth using in such environments
  - Making best use of ZFS features (without breaking sweat)
  - Missing (usability) features and work in progress

Introduction

- What this talk is *not* about
  - Complex vdev setups
  - "Enterprise-ish" setup: spares, ZIL/log and L2ARC/log devices, deduplication
  - Big iron (controllers, expanders, backplanes, multipath, huge memory, etc)
  - Tiny iron (ZFS in 64MB of RAM)
  - Setup for special purpose servers (file servers, etc), hardcore tuning
- What this talk is about
  - "Normal" setup for workstation / desktop use
  - Why ZFS is worth using in such environments
  - Making best use of ZFS features (without breaking sweat)
  - Missing (usability) features and work in progress

Main benefits of ZFS

- Copy-on-write for safer data and faster recovery
  - No fsck after ungraceful restart
  - zpool import -F for going back in time
- Built-in volume manager
  - Mirror, RAID-Z, etc
  - More flexibility with creating filesystems and volumes
    - No need to precalcualte number of inodes
    - No pain when filesystem usage nears its size
- Snapshots and clones
  - Checkpoint any significant change
  - Automatic snapshots
    - zfsnap, zfs-snapshot-mgmt, zfs-periodic, zfstools
  - Boot Environments
    - sysutils/beadm, google://manageBE

Main benefits of ZFS

- Copy-on-write for safer data and faster recovery
  - No fsck after ungraceful restart
  - zpool import -F for going back in time
- Built-in volume manager
  - Mirror, RAID-Z, etc
  - More flexibility with creating filesystems and volumes
    - No need to precalcualte number of inodes
    - No pain when filesystem usage nears its size
- Snapshots and clones
  - Checkpoint any significant change
  - Automatic snapshots
    - zfsnap, zfs-snapshot-mgmt, zfs-periodic, zfstools
  - Boot Environments
    - sysutils/beadm, google://manageBE

Main benefits of ZFS

- Copy-on-write for safer data and faster recovery
  - No fsck after ungraceful restart
  - zpool import -F for going back in time
- Built-in volume manager
  - Mirror, RAID-Z, etc
  - More flexibility with creating filesystems and volumes
    - No need to precalcualte number of inodes
    - No pain when filesystem usage nears its size
- Snapshots and clones
  - Checkpoint any significant change
  - Automatic snapshots
    - zfsnap, zfs-snapshot-mgmt, zfs-periodic, zfstools
  - Boot Environments
    - sysutils/beadm, google://manageBE

Main benefits of ZFS

- Copy-on-write for safer data and faster recovery
  - No fsck after ungraceful restart
  - zpool import -F for going back in time
- Built-in volume manager
  - Mirror, RAID-Z, etc
  - More flexibility with creating filesystems and volumes
    - No need to precalcualte number of inodes
    - No pain when filesystem usage nears its size
- Snapshots and clones
  - Checkpoint any significant change
  - Automatic snapshots
    - zfsnap, zfs-snapshot-mgmt, zfs-periodic, zfstools
  - Boot Environments
    - sysutils/beadm, google://manageBE

My setup

- Hardware
  - Off-the-shelf desktop hardware (Core2 Duo, Athlon II X2)
  - Chipset-integrated controllers (ICH9, SB750)
  - Paired 512GB HDDs in ZFS Mirror
  - 4-8GB of RAM, non-ECC
- ZFS configuration and features
  - GPT partitioning, swap is *not* on ZFS
  - Snapshots: automatic and manual
  - Scrub once in two months (via periodic)
  - ZVOLs for VMs, including ZFS pools on ZVOLs
  - Boot Environments using custom scripts / commands
- Why this configuration...

My setup

- Hardware
  - Off-the-shelf desktop hardware (Core2 Duo, Athlon II X2)
  - Chipset-integrated controllers (ICH9, SB750)
  - Paired 512GB HDDs in ZFS Mirror
  - 4-8GB of RAM, non-ECC
- ZFS configuration and features
  - GPT partitioning, swap is *not* on ZFS
  - Snapshots: automatic and manual
  - Scrub once in two months (via periodic)
  - ZVOLs for VMs, including ZFS pools on ZVOLs
  - Boot Environments using custom scripts / commands
- Why this configuration...

My setup

- Hardware
  - Off-the-shelf desktop hardware (Core2 Duo, Athlon II X2)
  - Chipset-integrated controllers (ICH9, SB750)
  - Paired 512GB HDDs in ZFS Mirror
  - 4-8GB of RAM, non-ECC
- ZFS configuration and features
  - GPT partitioning, swap is *not* on ZFS
  - Snapshots: automatic and manual
  - Scrub once in two months (via periodic)
  - ZVOLs for VMs, including ZFS pools on ZVOLs
  - Boot Environments using custom scripts / commands
- Why this configuration...

Tuning

Minimalist tuning on 8GB system (amd64)

```
vfs.zfs.arc_min=402653184 # 384MB
vfs.zfs.arc_max=2147483648 # 2GB
vfs.zfs.arc_meta_limit=1610612736 # 1.5GB
```

- *arc_min* to preserve the most most used data when applications take all memory
- *arc_max because* I do not want ZFS to over-cache and compete with apps for memory[1]
- *arc_meta_limit* because default is $\frac{1}{4}$ of ARC, too low for ports, FreeBSD source trees (svn, git)[2]

[1] but it's rather large to cache source tree(s) where I hack

[2] Perhaps *arc_meta_limit* == *arc_max* wouldn't be so bad in general case.

Useful dataset properties

- mountpoint, canmount - control if dataset is mounted and where
- atime - turn this off :-)
- compression - makes sense for ports (but not distfiles), source trees and similar
- exec, setuid, readonly - control access (may improve time spent on security checks)
- volsize, volblocksize - control ZVOLs
- primarycache, secondarycache - for some volumes or dataset it may make sense to skip caching

Warnings

- Be extra careful when modifying pool device configuration
  - Greatest risk to your data is here

zpool add

Never ever confuse zpool add and zpool attach!
Check and re-check documentation 7 times before hitting enter.
There is no going back from zpool add!

- If you can ECC, do ECC
- There is no zfsck, recovering when things go bad is very hard
  - But possible
- Swap on ZFS - should work, but there are conflicting reports

FreeBSD®

Warnings

- Be extra careful when modifying pool device configuration
  - Greatest risk to your data is here

zpool add

Never ever confuse zpool add and zpool attach!
Check and re-check documentation 7 times before hitting enter.
There is no going back from zpool add!

- If you can ECC, do ECC
- There is no zfsck, recovering when things go bad is very hard
  - But possible
- Swap on ZFS - should work, but there are conflicting reports

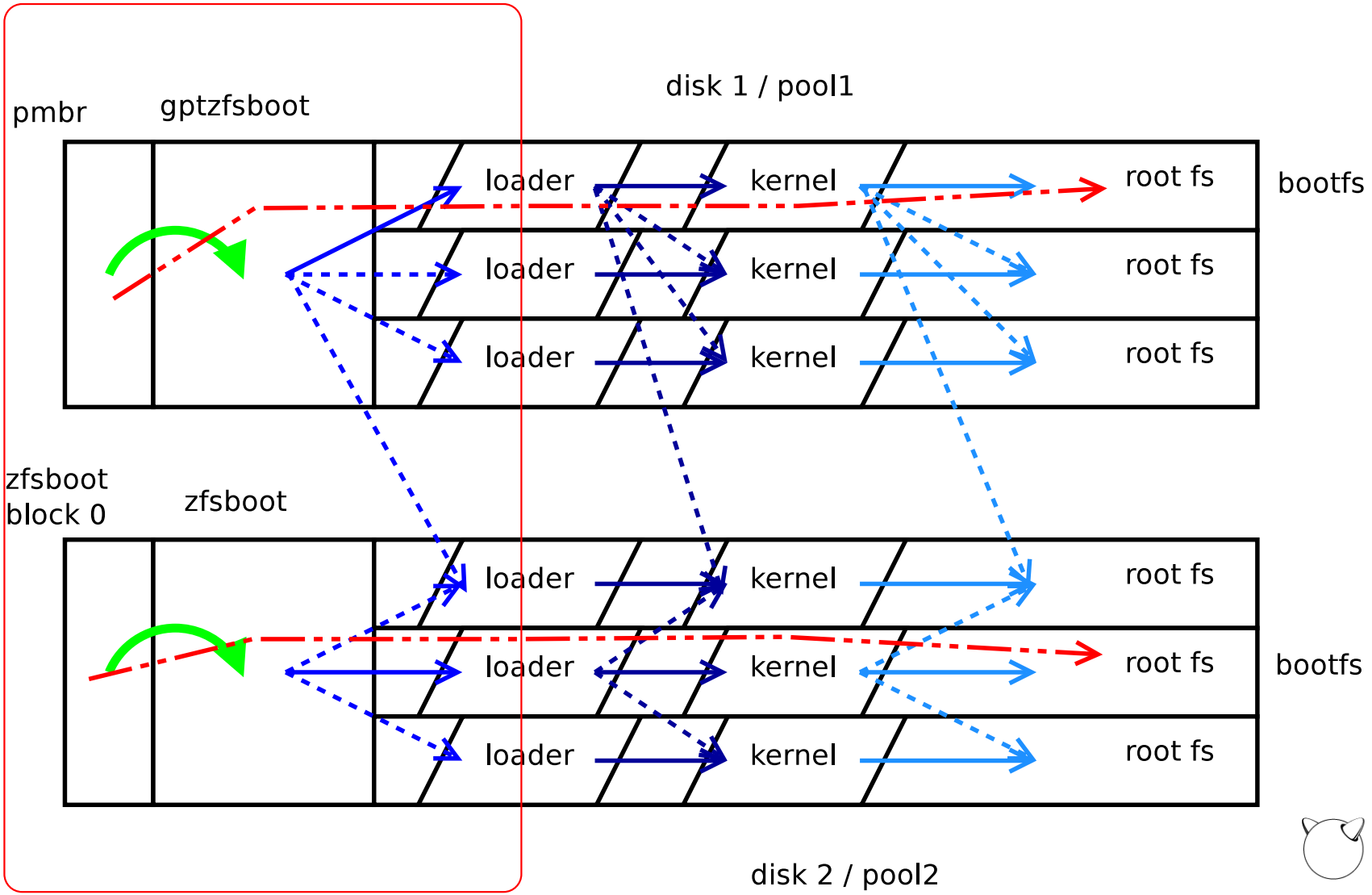
FreeBSD®

Boot Environments

- Snapshot - you can rollback to it, but you can not boot it
- Snapshot+clone - now you can boot
- Dataset layout for BE
  - pool/ROOT/$tag is a convention from Solaris
  - What to include into BE
    - BE root dataset can have dependent datasets
    - Dataset hierarchy is just a convenience for managing their properties
- What to modify - a clone or original?
  - Clone is a convention from Solaris
- Working with clone: DESTDIR, chroot, freebsd-update -b <basedir>

FreeBSD®

One picture...


disk 1 / pool1
pmbr
gptzfsboot
loader
kernel
root fs
bootfs
loader
kernel
root fs
loader
kernel
root fs
zfsboot
block 0
zfsboot
loader
kernel
root fs
loader
kernel
root fs
bootfs
loader
kernel
root fs
disk 2 / pool2

boot0 and boot2 stages

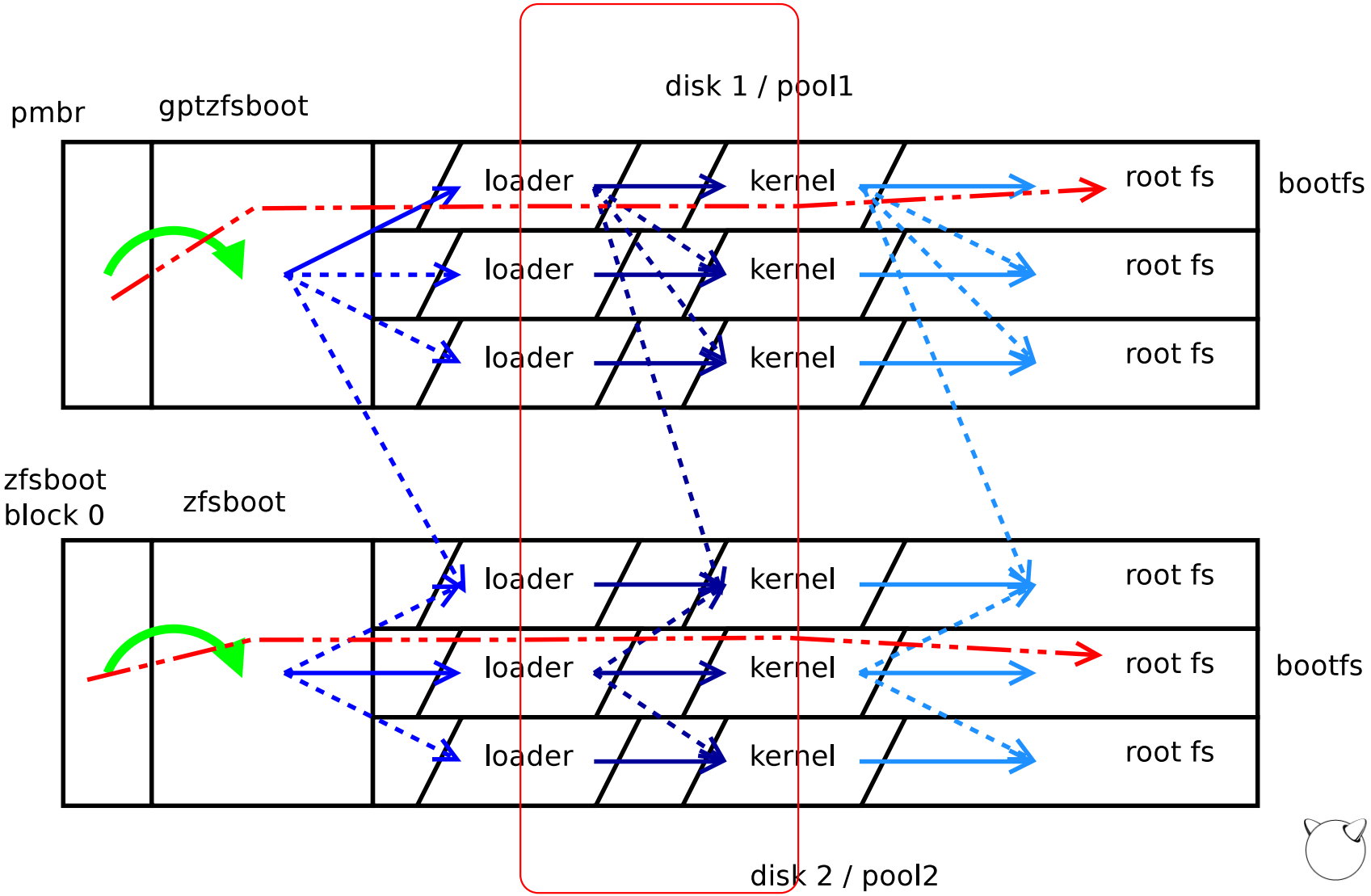
- boot0-like stage always takes boot2-like stage from the same disk using simple rules
- boot2-like stage probes all disks and partitions it can understand for ZFS pools
- allows to select a different pool, a specific filesystem in the pool and a specific loader

zfsboot prompt

```
FreeBSD/x86 boot
Default: pool1:ROOT/test1:/boot/zfsloader
boot: pool2:ROOT/knowngood:/boot/zfsloader.old
```

- default pool is the first pool detected by probing which starts at boot disk
- default filesystem is determined by bootfs property

One picture...


disk 1 / pool1
pmbr
gptzfsboot
loader
kernel
root fs
bootfs
loader
kernel
root fs
loader
kernel
root fs
zfsboot
block 0
zfsboot
loader
kernel
root fs
loader
kernel
root fs
bootfs
loader
kernel
root fs
disk 2 / pool2

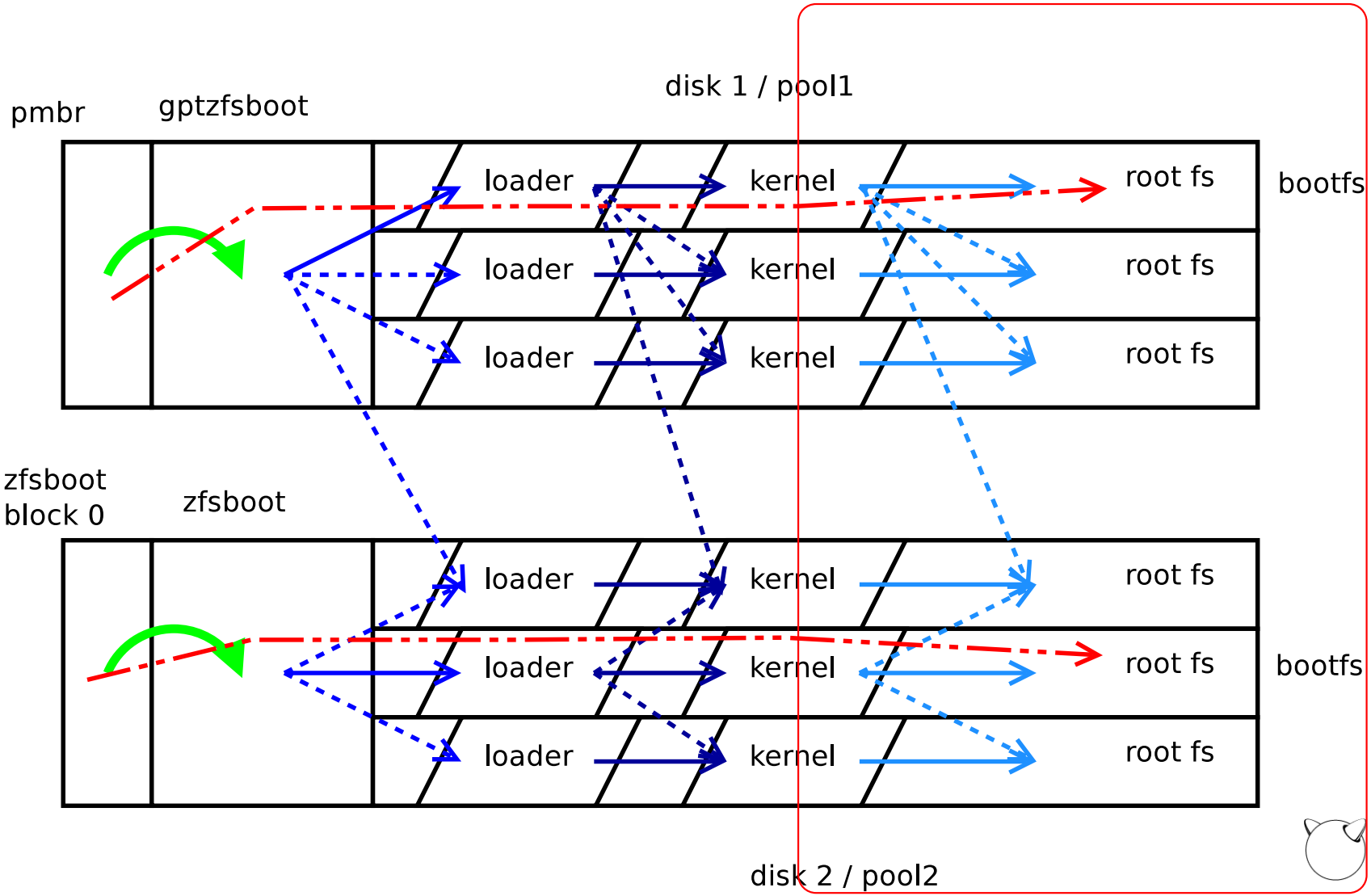
loader

- loader uses boot pool and filesystem passed by boot2
- exposes loaddev and currdev variables
- currdev can be changed, kernel and modules are loaded from currdev by default

loader prompt

```
set currdev=zfs:tank/ROOT/backup:
ls zfs:tank/ROOT/20120929:/boot/modules
```

One picture...


disk 1 / pool1
pmbr
gptzfsboot
loader
kernel
root fs
bootfs
zfsboot
block 0
zfsboot
disk 2 / pool2

Root filesystem

- kernel mounts root from filesystem specified by vfs.root.mountfrom, passed by loader to kenv
- loader tries to set this variable based on /etc/fstab in the filesystem specified by currdev
- can be overriden by explicit setting in loader.conf or at the prompt
- for ZFS a default value is set from value of currdev if the above methods do not produce any value
- for BEs it's best to rely on the default (no root fs in fstab, no vfs.root.mountfrom in loader.conf)

Coming soon

- Easier selection of BE / boot filesystem in loader
  - lszfs patch posted
  - work is starting on menu-driven ZFS filesystem selection!
- Mounting root filesystem from pool not in zpool.cache (exported)
  - patch posted
- nextboot or nextboot-like solution for ZFS
  - ZFS R/W support in loader is *hard*
  - znextboot patch is posted (zfsboot one time configuration)

Nice to have ZFS usability features

- Installation/update tool for zfsboot (dd is not user friendly)
- Full featured BE administration tool(s) in base
- Support for BEs in other tools: system upgrade, packages upgrade, configuration changes, etc
- Automatic snapshot management in base, both periodic and critical point
- ZVOL as dump device
- ZFS monitoring and automatic fault management (zfsd...)
- ZFS administration GUIs
- zfsboot documentation (mea culpa!)
- ZFS support in installer

Thank you!

- Thank you for listening!
- Questions?

FreeBSD®