Hacking NanoBSD for Fun and Profit
Building a Manageable Hosting Platform

Patrick M. Hausen
EuroBSDCon 2010

punkt.de

What is NanoBSD?

It‘s a shell script:

```
$ more /usr/src/tools/tools/nanobsd/
nanobsd.sh

#!/bin/sh
#
# Copyright (c) 2005 Poul-Henning Kamp.
# All rights reserved.
# ...
```

What does it do?

Build a customized FreeBSD image that runs really well on this:

- 500 Mhz AMD CPU (i386)
- 512 MB RAM
- 1 GB Compact Flash
- DC current
- No moving parts
- Less than 20 Watts power

Other environments

Why would I want to put it on this?

- 2–4 Core Intel CPU (amd64)
- 4–16 GB of RAM
- 2 SATA hard disks (mirrored)
- AC current
- 100–150 Watts power

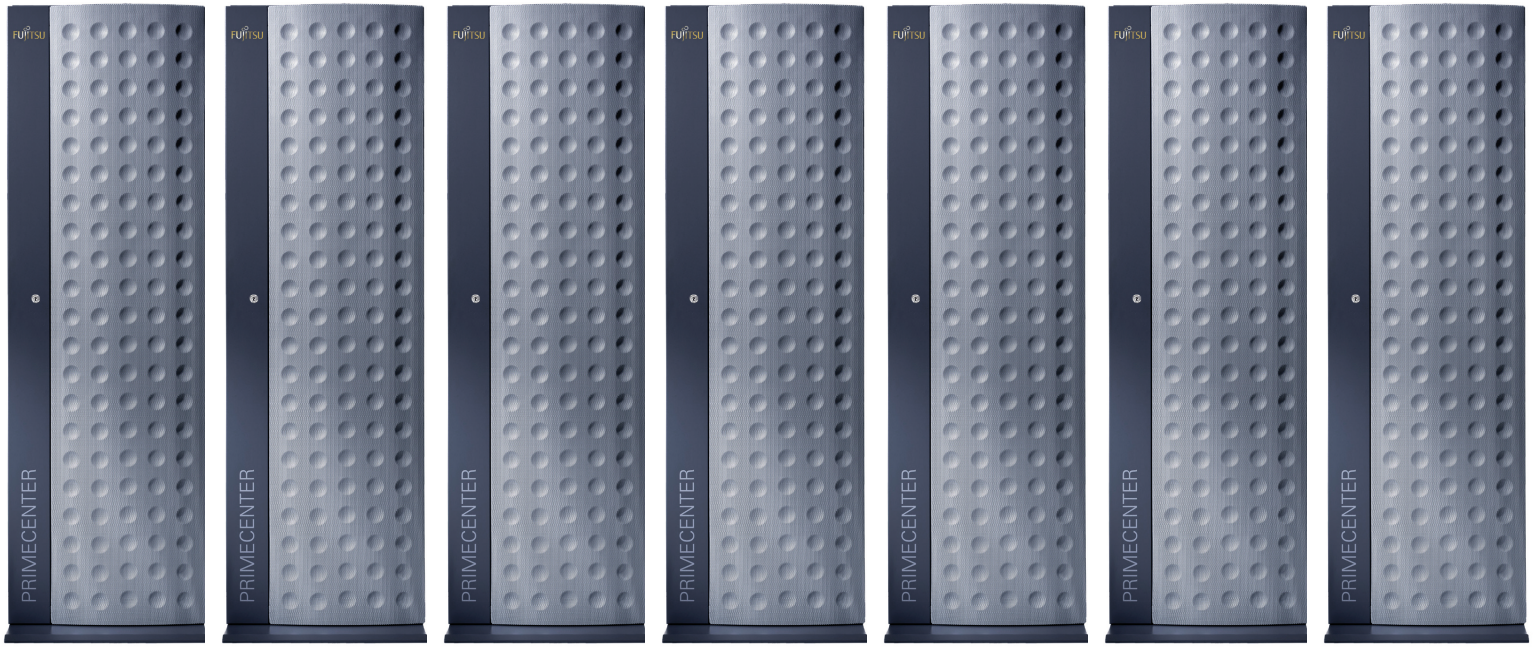
Maintenance

I‘t easy to install and update this:


```
# make buildworld buildkernel ...
# portupgrade ...
```

Scalability

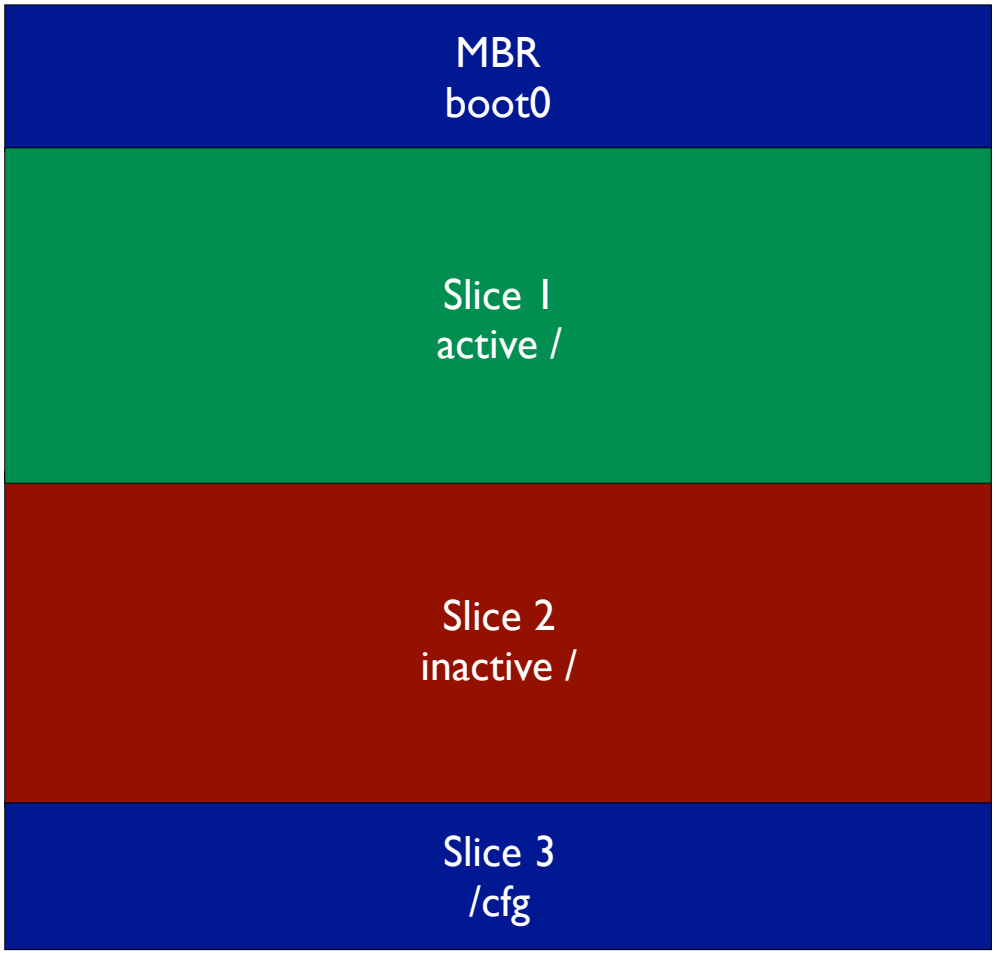
But, what about this?



Advantage NanoBSD

- Unattended build process
- Result: image with OS and all desired packages
- Install and update with dd
- 2 partitions for system, update one, roll back if necessary

On–Disc Structure



Peculiarities

- / is mounted read–only.

  This is a good thing!

- /etc and /var are memory disks, /etc populated at boot time from /cfg.

  But we need persistent storage for the customers‘ applications! (Apache, MySQL, ...) And for /etc, because we frequently create new passwd and group entries for vhosts. (fcgid/suexec)

- We need swap!

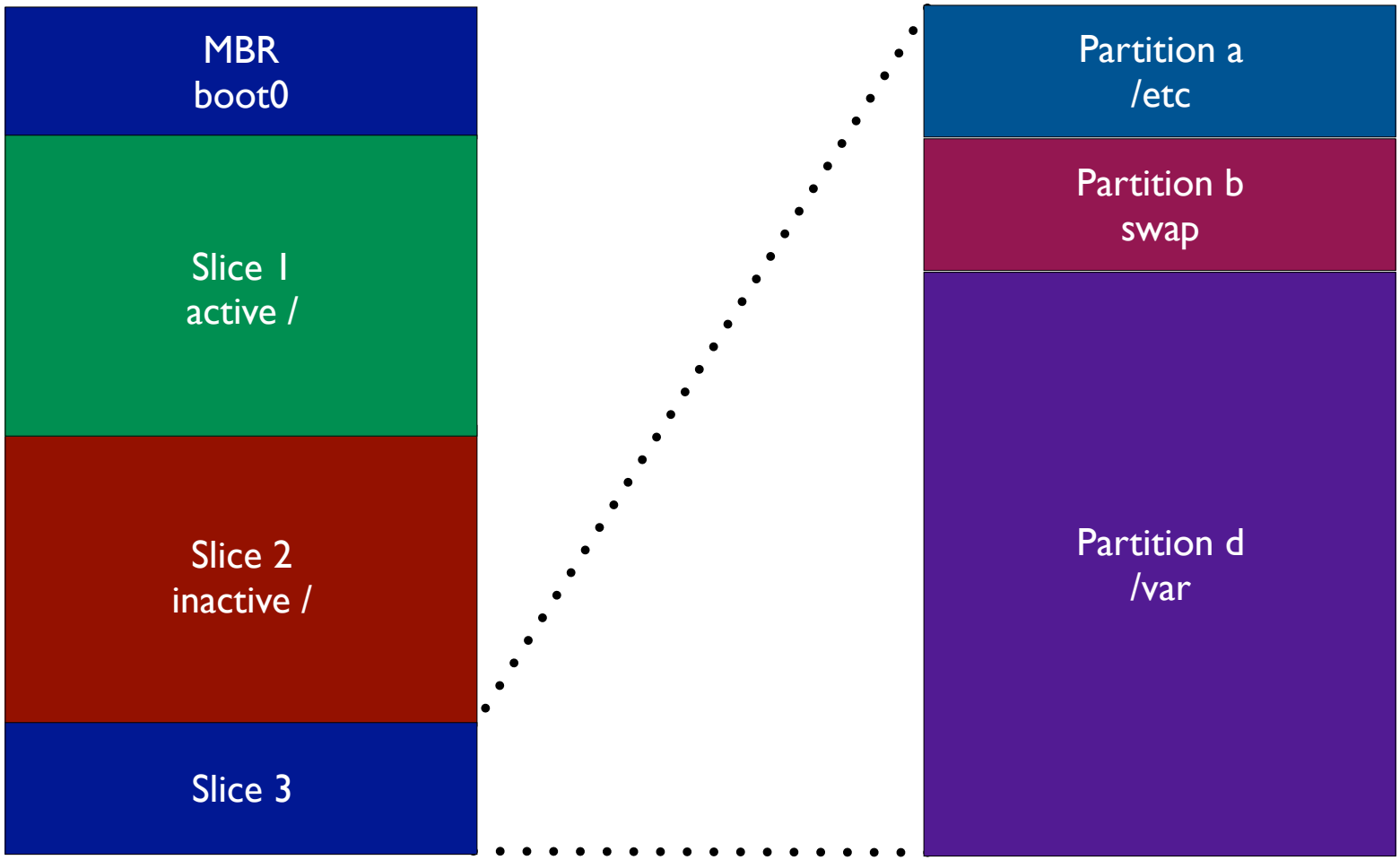
- We want all this on top of a gmirror(8).
  Actually, this is not a problem:
  NANO_DRIVE=“mirror/m0“

Regular Boot

- Create memory disks for /etc and /var
- Populate from /conf/base
- Merge /cfg

You have to copy all changes in /etc to /cfg to store them permanently!

New disk layout



/etc/fstab

```
/dev/mirror/m0s1a / ufs ro 1 1
/dev/mirror/m0s3a /etc ufs rw,noauto 2 0
/dev/mirror/m0s3b none swap sw 0 0
/dev/mirror/m0s3d /var ufs rw 2 2
```

How to mount /etc?

- NanoBSD relies on the diskless framework to do it‘s magic.
- If we do not use the /etc memory disk, then /etc/rc.conf is alway loaded from the read–only root, even if /etc is remounted differently later. (Actually it‘s read twice.)
- This mechanism is hardwired.

  At least I didn‘t find a suitable knob to twist.

Only thing that is executed before te first reading of rc.conf and can safely be messed with: /etc/rc.initdiskless

```
#!/bin/sh
#
# Hack to mount /etc early in the boot process,
# even _before_ /etc/rc.conf is read and evaluated.
#
# The /etc filesystem should be listed with the "noauto" option
# in /etc/fstab, otherwise the later automatic mounting of all
# local filesystems will fail.
#
# This script replaces the stock FreeBSD rc.initdiskless
#

. /etc/rc.subr

trap : 3

/sbin/fsck -p /etc && /sbin/mount /etc

if [ "$?" -ne "0" ]
then
    stop_boot
fi
```

Example File (1)

```
NANO_NAME=rx100-hosting

MAINDIR=/home/nanobsd
NANO_SRC=/usr/src
NANO_OBJ=$MAINDIR/obj/$NANO_NAME
NANO_KERNEL=GENERIC
NANO_DRIVE=mirror/m0
NANO_ARCH=amd64
NANO_PACKAGE_DIR=$MAINDIR/build/$NANO_NAME/packages

NANO_MEDIASIZE=`expr 255 \* 63 \* 1023`
NANO_HEADS=255
NANO_SECTS=63
NANO_IMAGES=1

VAR_DRIVE="$NANO_DRIVE"

. ../common/common.nano
```

Example File (II)

```
customize_cmd       cust_nobeastie
customize_cmd       cust_dualconsole
customize_cmd       cust_gmirror
customize_cmd       cust_ipmi


../common/common.nano:

#
# enable gmirror
#
cust_gmirror()
(
     echo "geom_mirror_load=\"YES\"" \
        >> ${NANO_WORLDDIR}/boot/loader.conf
)
```

Predefined Functions

```
FlashDevice ()
UsbDevice ()
cust_comconsole ()
cust_allow_ssh_root ()
cust_install_files ()
cust_pkg ()
```

Roll your own ...

```
#
# install TYPO3 source
#
cust_install_typo3()
(
        mkdir ${NANO_WORLDDIR}/usr/local/typo3 && \
        cd ${NANO_WORLDDIR}/usr/local/typo3 && \
        cp ${MAINDIR}/typo3/index.html . && \
        for source in ${MAINDIR}/typo3/typo3_src-*.tar.gz
        do
                tar xvfz $source
        done && \
        chown -R root:wheel *

        mkdir -p ${NANO_WORLDDIR}/var/punkt-tools/mastersites/TYPO3 && \
        cd ${NANO_WORLDDIR}/var/punkt-tools/mastersites/TYPO3 && \
        tar xvfz ${MAINDIR}/typo3/mastersites.tar.gz
)
```

Pitfalls

- Packages that insist on installing interactively like Java
- Packages that want network connections during install (mostly PECL for us)
- NanoBSD itself does some last minute modifications that potentially overwrite customize_cmd actions. (e.g. /etc/fstab) There is now a late_customize_cmd statement for that. (RELENG_7 and up)

Workarounds

```
#
# Install JDK
#
cust_java()
{
   mkdir ${NANO_WORLDDIR}/java
   cp java/* ${NANO_WORLDDIR}/java

   chroot ${NANO_WORLDDIR} sh -c \
      'cd java && yes | pkg_add -F *' || true

   rm -rf ${NANO_WORLDDIR}/java
}
```

Makefile

```
MAINDIR=        /home/nanobsd
NANOBSD_SH=     /usr/src/tools/tools/nanobsd/nanobsd.sh

OBJDIR=         $(MAINDIR)/obj/$(PLATFORM)
NANOIMAGE=      $(OBJDIR)/_.disk.image
NANOFULL=       $(OBJDIR)/_.disk.full
NANOWORLD=      $(OBJDIR)/_.w
NANOCOMMON=     $(MAINDIR)/build/common/common.nano
GZIMAGE=        $(PLATFORM).img.gz
ETCTAR=         $(PLATFORM).etc.tar.gz
VARTAR=         $(PLATFORM).var.tar.gz
PKGTAR=         $(PLATFORM).pkg.tar.gz

TARGETS?=       $(GZIMAGE) $(ETCTAR) $(VARTAR) $(PKGTAR)

all:            $(TARGETS)

$(GZIMAGE):     $(NANOIMAGE)
                gzip -c $> > $@

$(ETCTAR):      $(NANOWORLD)
                (cd $(NANOWORLD)/etc && tar cfz - .) > $@

$(VARTAR):      $(NANOWORLD)
                (cd $(NANOWORLD)/var && tar cfz - .) > $@

$(PKGTAR):      $(NANOWORLD)
                (cd $(NANOWORLD)/var/db/pkg && tar cfz - .) > $@

$(NANOIMAGE):   $(NANOWORLD) $(NANOCOMMON)
                sh $(NANOBSD_SH) -b -c $(PLATFORM).nano

$(NANOWORLD):   $(NANOBSD_SH) /usr/src/UPDATING $(PLATFORM).nano
                sh $(NANOBSD_SH) -c $(PLATFORM).nano

install:        all
                mkdir -p $(MAINDIR)/images/$(PLATFORM)/`date +%Y%m%d` && \
                cp $(TARGETS) $(MAINDIR)/images/$(PLATFORM)/`date +%Y%m%d`

clean:
                rm -f $(TARGETS) $(NANOIMAGE)

distclean:
                chflags -R noschg $(OBJDIR) || true
                rm -rf $(OBJDIR) $(TARGETS)
```

Result

- rx100-hosting.etc.tar.gz — /etc prototype
- rx100-hosting.img.gz — Image proper
- rx100-hosting.pkg.tar.gz — /var/db/pkg
- rx100-hosting.var.tar.gz — /var prototype

Installation

- Unattended PXE boot install on first disk
- Login, create mirror on second disk
- fdisk mirror to setup slices
- bsdlabel slice 3
- newfs /etc and /var
- dd image to slice 1
- Extract /etc and /var prototypes to their respective partitions
- Boot from second disk
- Insert first disk into mirror

Updating

- dd new image to inactive slice
- Change active slice

  (It‘s more difficult than you would think!)
- Reboot
- Test
- If successfull, update /var/db/pkg
- Update /etc (?)

Bugs (FreeBSD‘s)

- boot0cfg doesn‘t work on a mirror – should be fixed in RELENG_8, need to test.
- boot0cfg doesn‘t reliably change the slice that is booted.
  Probably MBR and boot0 flag not in sync.
  Use boot0cfg **and** fdisk on the raw disks.
  Enable „foot shooting“ first.
- Sometimes there is no rollback — e.g. when updating gmirror from RELENG_6 to RELENG_7

Bugs (mine)

- Having complete /etc on hard disk was a mistake.
- Every time we do major updates, we need to do something similar to mergemaster.
- But we wanted to get rid of that in the first place!
- /conf/base/etc mechanism is actually quite clever — gets updated with every new image.

Upcoming changes

- Update to RELENG_8 — Oct 2010
- Rework /etc setup — Oct 2010
  - back to memory disk again
  - automagically sync changes to /cfg
- Investigate consequences of raising the securelevel, so read–only really is read–only. — End 2010
- Investigate and eventually implement ZFS root — End 2010

Syncing /etc and /cfg

- With cronjob:
  - save_cfg (part of NanoBSD)
  - cfgsync by Paul Schenkeveld
- Instantly:
  - Port lsyncd from Linux to kqueue
  - Implement using sysutils/wait_on

Questions?

Thank you very much.

Presentation and sample scripts on the conference website RSN ;-)

(Today! Promise!)