Schreibgeschütztes NFS-Root für eine Gruppe von plattenlosen NFS-Clients

Ignatios Souvatzis
Institut für Informatik der Universität Bonn
ignatios@cs.uni-bonn.de

DECUS Symposium Bonn
18. April 2002

– Typeset by FoilTEX –

Struktur des Vortrags

1. technische und soziale Systemumgebung
2. warum shared root / read-only root?
3. Schwierigkeiten bei der Implementierung
4. unsere Methoden
5. unsere Implementierung (Skripte...)
6. Gedanken über einen alternativen Ansatz

Systemumgebung [1]

- Server:
  - UltraSparc 10, physikalisch sicher aufgestellt
  - OS: Solaris (Problem aber OS-unabhängig)

Systemumgebung [2]

- Clients:
  - 10 Netzwerkcomputer (DNARD), physikalisch sicher aufgestellt
  - CPU: Strongarm SA/110 bei 233 MHz, 64 MB RAM, Ethernet
  - OS: NetBSD 1.4 – 1.5 *(– 1.6)*
  - Problem OS-unabhängig innerhalb der Unix-Familie, unsere Implementierung OS-abhänging, aber vermutlich grossteils übertragbar
  - Nutzer: u.a. halbjährlich wechselnde Studierende (Parallelrechnerpraktikum), einloggen vom Pool oder zu Hause

Warum ein shared/read-only Root? [1]

- Vereinfachte Administration
  - Konfiguration nur einmal fuer alle Clients vom Server aus
- Plattenplatzeinsparung

```
theory 196 # du -ks bin sbin etc
5795    bin
12760   sbin
606     etc
```


Warum ein shared/read-only Root? [2]

- Sicherheit des Servers
  - Root-Bereich *muß* mit Root-Rechten exportiert werden
  - Ein Einbruch in einen Client, selbst als root, lässt sich durch `reboot` beheben
    aber: Benutzerdateien!

Schwierigkeiten bei der Implementierung [1]

- beim Bootprozess
  - Netzwerkkonfiguration
  - Swap-Konfiguration
- im Betrieb
  - Logdateien
  - `*.pid`-Dateien
  - Sockets (`/dev/log, /dev/printer, ...`)
  - Zeitstempel auf Geräteknoten (`/dev`)
  - `ssh_host_key`
  - Softwareinstallation (`pkg`-System)

Schwierigkeiten bei der Implementierung [2]

- beim Shutdown
  - `/etc/nologin`
    wird beim Start von shutdown auf der ersten Maschine angelegt und verhindert weiters Einloggen auch auf den anderen Maschinen, falls `/etc` wirklich das gleiche Verzeichnis ist.

Keine Probleme

- `ssh_host_key` ist für alle diese Clients gleich
- Netzwerkkonfiguration über DHCP (IPv4) und stateless address autoconfiguration (IPv6)
- `syslog` schickt statt auf Dateien an den Server

Sockets, *.pid

- zur Bootzeit wird ein Memory File System für `/var/run` erstellt *(auslagerfähig!)*
- `/dev/log` auf /var/run/log verschoben, inzwischen NetBSD-Standard
- `/dev/printer` auf /var/run/printer verschoben, NetBSD-Standard
- `.pid`-Dateien, bei anderen OS in `/etc`, in NetBSD in `/var/run`

Geräteknoten: Problem

Warum sind Geräteknoten ein Problem?

- Terminal-Knoten ändern beim Login und Logout ihren Besitzer
- Terminal-Knoten ändern ihren Zeitstempel (`atime`) bei jedem Lesevorgang (idletime-Erkennung!)
- bei Workstations auch bei Maus, Tastatur, Konsolausgabe (Bildschirmschoner!)

Geräteknoten: Lösung

- Memory File System für `/dev`
- `/dev` (für Clients) auf dem Server enthält nur `/dev/console` (für `/sbin/init`
- Geräteknoten werden zur Bootzeit durch `MAKEDEV` erzeugt (ca. 20 Sekunden länger).
- *Ausblick:* In NetBSD-1.6 wird das ganze von `/sbin/init` durchgeführt, falls kein /dev/console vorhanden ist. Motivation hier: Installations- oder Demosysteme auf MS-DOS-Filesystemen oder CD-ROM o.ä.

swap, /var

Zur Startupzeit wird durch den Aufruf von `/bin/hostname` der Rechnername aus dem Kernel erfragt, den der Client per DHCP erhalten hat. Daraus werden einige Namen erzeugt:

- Für jeden Client existiert auf dem Server ein exportierter Baum `var-`*name* (für übriggebliebene Logdateien, (Drucker-) Spoolbereich etc.
- Für jeden Client existiert auf dem Server eine exportierte Swap-Datei, die ebenfalls den Namen des Clients enthält.

server:/etc/dhcpd.conf

Nichts wirklich spezielles:

```
option domain-name "cs.uni-bonn.de";
option domain-name-servers 131.220.4.211, 131.220.4.1;
deny unknown-clients;
use-host-decl-names on;
subnet 131.220.4.0 netmask 255.255.255.0 {
        option routers 131.220.4.3;
        group {
                option lpr-servers theory.cs.uni-bonn.de;
                server-name "theory";
                next-server theory;
```

/etc/dhcpd.conf [2]

```
                filename "netbsd-SHARK-1.5.3_20020114";
                option root-path "/BSD/root2/1.5";
                host rechner1 {
                        hardware ethernet 10:20:30:40:50:60;
                        fixed-address rechner1.cs.uni-bonn.de;
                }
                ...
        }
}
```


/etc

- Problem:

  Einzelne Dateien werden in `/etc` noch verändert (`/etc/nologin`, `/etc/motd ...`)

  In einzelnen Fällen (motd) ist das Beschreiben abschaltbar, in anderen jedoch nicht ohne Codeänderung.

- Lösung:

  Wir erzeugen ein Memory File System, welches als Union-Mount über /etc gelegt wird.

Installation mit dem pkg-System

- einem Client wird zeitweise Schreibzugriff gegeben
- Die Environment-Variable PKG_DBDIR wird auf /usr/pkg/libdata/pkgdb gesetzt, bevor pkg_add oder ein anderes der pkg-Tools benutzt wird.
- nach Installation wird der Export-Eintrag (auf dem Server) wieder auf readonly gestellt.
- Sicherere Lösung wäre das Installieren direkt auf dem Server, erfordert aber zusätzliche Arbeit an den pkg_xxx - Tools.

/etc/rc.d/shroot

```
#!/bin/sh

# PROVIDE: shroot
# REQUIRE: root
# BEFORE: mountcritlocal

. /etc/rc.subr

name="shroot"
start_cmd="shroot_start"
stop_cmd=":"
required_files="/sbin/MAKEDEV /sbin/MAKEDEV.local"
```

```
shroot_start () {

hostname=‘/bin/hostname‘

case "$shroot_pfx_var" in
"") ;;
*) /sbin/mount -t nfs ${shroot_pfx_var}${hostname} /var ;;
esac
case "$shroot_pfx_swap" in
"") ;;
*) /sbin/mount -t nfs ${shroot_pfx_swap}${hostname} /swap\
&& /sbin/swapon /swap ;;
esac
```



```
/sbin/mount -t mfs -o -i=256 -o -s=512 swap /dev
/sbin/mount -t mfs -o -i=256 -o -s=512 -o union swap /etc
/bin/chmod 755 /dev /etc
echo -n "creating device nodes...";
/bin/cp /sbin/MAKEDEV /sbin/MAKEDEV.local /dev
(cd /dev; sh MAKEDEV all)
echo done.
}

load_rc_config $name
run_rc_command "$1"
```

/etc/fstab

NetBSD arbeitet verschiedene Teile von /etc/fstab zu verschiedene Zeitpunkten ab (root zuerst), so dass das `shroot`-Skript nach dem Mounten von `/`, aber vor dem mounten von `/var/games` stattfindet:

```
theory:/BSD/root2/1.5     /           nfs ro                0 0
theory:/BSD/root/vargames /var/games  nfs rw                0 0
swap                      /tmp        mfs rw,-s=32768       0 0
swap                      /var/run    mfs rw,-i=512,-s=256 0 0
```


/etc/rc.conf

```
rc_configured=YES
shroot_pfx_var=theory:/BSD/root/var-
shroot_pfx_swap=theory:/BSD/swap/
critical_filesystems_beforenet="/var /var/run"
update_motd=NO
rpcbind=YES # nfs
domainname=theory.cs.uni-bonn.de # NIS
ypbind=YES
amd=YES amd_dir=/var/amdroot
nfs_client=YES
ip6mode=autohost
defaultroute=131.220.4.3
sshd=YES postfix=YES ntpd=YES
```

```
lpd=YES lpd_flags=-s
inetd=YES       # ntalk, (c)fingerd
```


Die Filesysteme im Betrieb:

```
> mount
theory:/BSD/root2/1.5 on / type nfs (read-only)
theory:/BSD/root/var-bagpipe on /var type nfs
theory:/BSD/swap/bagpipe on /swap type nfs
mfs:34 on /dev type mfs (asynchronous, local)
mfs:37 on /etc type mfs (asynchronous, local, union)
mfs:1000 on /var/run type mfs (asynchronous, local)
theory:/BSD/root/vargames on /var/games type nfs
mfs:1085 on /tmp type mfs (asynchronous, local)
pid1121@bagpipe:/home on /home type nfs
131.220.4.18:/opt/export/home/stud/user on \
/var/amdroot/131.220.4.18/opt/export/home/stud/user \
type nfs (nodev, nosuid)
```

```
theory:/export/home/2 on /var/amdroot/theory/export/home/2 \
type nfs (nodev, nosuid)
```


Alternativer Ansatz: memory disk

- wie beim ersten Ansatz: keine Änderungen werden zurückgeschrieben
- die meisten mfs-Spielereien können wir uns sparen

aber:

- mehr Hauptspeicher dauerhaft belegt, da ganzes root ausgepackt wird (selbst bei komprimiertem Kernel)
- mehr Aufwand bei der Änderung der Konfiguration

Evtl. hybrider Ansatz?

WWW

- Parallelrechnerlabor am Institut für Informatik der Uni Bonn: http://theory.cs.uni-bonn.de/info5/system/parlab/
- NetBSD: http://www.netbsd.org/