05. Mai 1999 <flierl@luga.de>

Konfiguration eines festplattenlosen Linux-Rechners mit SuSE 6.1

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument beschreibt die Konfiguration eines festplattenlosen Linux-Rechners und die dazu nötigen Anpassungen an einem S.u.S.E. 6.1-Linux-Server. Die komplette Konfiguration des Servers dauert ungefähr eine halbe Stunde, die Einbindung eines festplattenlosen Linux-Rechners etwa zwei bis drei Minuten. Als Schmankerl für alle Linux-Geeks befindet sich am Ende dieses Dokumentes noch ein kleiner Abschnitt über `pvmpov` in einem Linux-Cluster.

1.1 Anwendungsgebiete

① In **Schulen** können festplattenlose Linux-Rechner ganze Windows- und Netware-Installationen ersetzen; man könnte jeden Schulrechner mit einer Linux-Bootdiskette ausrüsten und so in eine vollwertige Linux-Workstation umwandeln – ohne dabei Änderungen an dessen lokaler Festplatte vornehmen zu müssen. Der Umstellung auf Linux sollte, dank benutzerfreundlicher Oberflächen wie KDE und GNOME[1], nichts mehr im Wege stehen.

[1] K Desktop Environment: "`http://www.kde.org`" und `http://www.gnome.org`"

② **Kernel Entwickler** nutzen festplattenlose Linux-Rechner, da bei eventuellen Kernel-Crashes keine teuren Festplatten beschädigt werden können und lästige Wartezeiten bei häufigem Booten durch den Wegfall der sonst üblichen Festplatten-Checks minimiert werden.

③ **Preiswerte Parallelrechner** arbeiten häufig mit nur einer einzigen (Server-)Festplatte. Ein spektakuläres Beispiel ist das Beowulf-Konzept zur Parallelschaltung herkömmlicher Standardrechner, das von Thomas Sterling und Don Becker am CESDIS-Institut der NASA entwickelt wurde, die bereits 1994 16 DX4-PCs zusammenschlossen: "`beowulf.gsfc.nasa.gov`"

1.2 Vor- und Nachteile

Vorteile festplattenloser Linux-Rechner:

① **Einfache Administration**: jeder beliebige Rechner kann innerhalb weniger Minuten in eine voll funktionsfähige Linux-Workstation verwandelt werden und belegt dabei auf der Serverfestplatte nur ca. 1.6 MB für seine Konfigurationsdateien.

② **Widerstandsfähigkeit gegen äußere Einflüsse:** verglichen mit PCs, die mit einer lokalen Festplatte arbeiten, sind festplattenlose Linux-Rechner sehr robust gegen häufiges, wahlloses Ein- und Ausschalten.

Nachteile festplattenloser Linux-Rechner:

① **Performanceeinbußen:** Programme mit intensiver Nutzung von I/O-Operationen (X11, netscape, QII) werden **durch die Netzwerkkarte** des festplattenlosen Rechners **gebremst**. Auch fehlerhaft installierte Netzwerke (z.B. Ethernet-Netzwerke mit falscher Terminierung oder internen Reflexionen) können das Arbeiten an einem festplattenlosen Rechner verlangsamen.

② **mangelnde lokale Serversicherheit:** Root-Dateisystem-exportierende NFS-Server mit `no_root_squash`-Option sind **nicht sicher**. Einem Hacker mit physischem Zugang zum lokalen Netzwerk wäre durch einen NFS-Server mit Schreibzugriff auf die Systemdateien des Servers Tür und Tor geöffnet. Ein Hacker müßte lediglich seinen eigenen Rechner mit der IP-Adresse des festplattenlosen Linux-Rechners im Netz anmelden und hätte dann über NFS uneingeschränkten Zugang zu den Systemdateien des Servers.

1.3 Systemvoraussetzungen

- Der **Server** sollte mit einer großen Festplatte (≥1 GB), viel Hauptspeicher (≥32 MB) und einer schnellen Netzwerkkarte[2] ausgerüstet sein. Ich habe folgende Programme (und ihre Abhängikeiten) auf dem Server installiert (S.u.S.E 6.1-Distribution):

Paket	Serie	Beschreibung	Inhalt
`nkita`	`a`	Netzwerk Basissystem	bootp-2.4.3 nfs-2.2beta38
`nkitb`	`a`	Netzwerkprogramme	portmap_5beta rlogin rsh
`shadow`	`a`	Shadow Passwort System	shadow-980724-14
`lx_suse`	`d`	Die Quellen des SuSE Kernels	kernel 2.0.36
`bind`	`n`	Name Server & Utilities	bind-4.9.7

  Anmerkung: Das Paket "`lx_suse`" belegen ca. 60 MB auf der Serverfestplatte; es wird nur zum Erstellen der Bootdisketten für die Clients benötigt und kann später wieder deinstalliert werden.

- Die **Clients** starten ihr Betriebssystem von einer Bootdiskette und mounten ihr komplettes Dateisystem über NFS vom Server; sie benötigen keine eigene Festplatte. Um zügiges Arbeiten zu gewährleisten, sollte jeder Client über möglichst viel Hauptspeicher (≥32 MB) und eine schnelle Netzwerkkarte verfügen.

[2] Ich bevorzuge für vier festplattenlose Linux-Clients aus ökonomischen Gründen eine Lösung mit 10Mbit-Ethernet-Netzwerkkarten; falls mehr als vier festplattenlose Rechner im Netzwerk angeschlossen werden, empfehle ich eine Lösung mit Fast-Ethernet oder FDDI, da das Arbeiten mehrerer Benutzer in einem Netzwerk sonst unerträglich langsam wird (allein der Neustart eines festplattenlosen Linux-Clients erzeugt ca. 17 MB Traffic!).

2 Konfiguration des Clients

2.1 Erstellen der Bootdiskette

Mit folgenden Befehlen erstellen Sie als Benutzer "`root`" eine auf die Hardware des Clients angepasste Bootdiskette mit ROOT-NFS-Unterstützung[3]:

Erstellen einer Bootdiskette mit ROOT-NFS-Unterstützung

```
cd /usr/src/linux && make menuconfig                            Linux-Kernel-Konfiguration
make dep clean bzImage modules modules_install      Kompilieren des Kernels und der Module
mknod /dev/boot255 c 0 255                                 Erstellen des Dummy-Root-Device
dd if=/usr/src/linux/arch/i386/boot/bzImage of=/dev/fd0  Kopieren des Kernels auf Diskette
rdev /dev/fd0 /dev/boot255                         Kopieren des Dummy-Root-Device auf Diskette
```

Das *Dummy-Root-Device* muß auf die Diskette kopiert werden, damit der Kernel das Root-Dateisystem beim Booten des Clients über das Netzwerk mountet.

Bei der Linux-Kernel-Konfiguration sollte noch folgendes beachtet werden:

- Der Netzwerkkartentreiber darf nicht als Modul kompiliert, sondern muß fest in den Kernel gelinkt werden, da er gleich beim Booten benötigt wird.
- Der Kernel muß das `BOOTP`-Protokoll und `ROOT-NFS` unterstützen;
  - Diese Optionen sind bei **Kerneln bis zur 2.1er Serie** in der Linux-Kernel-Konfiguration unter "`NFS filesystem support`" und ihrer Unteroptionen "`Root file system on NFS`" und "`BOOTP support`" zu finden.
  - In **Kerneln ab der 2.1er Serie** ist die `BOOTP`-Option bei den "`Networking options`" als Unterpunkt: "`IP: Kernel level autoconfiguration`", die "`ROOT-NFS`"-Option unter "`Filesystems`" → "`Network File Systems`" → "`NFS filesystem support`" → "`Root file system on NFS`" zufinden.

2.2 Booten des Clients

Im `BIOS` des Client-PCs sollten Sie die *Boot Sequence* auf "`A,C,SCSI`" stellen, um sicherzugehen, daß der Kernel auch von der Bootdiskette geladen wird. Wenn der PC als *headless node*[4] in einem *Beowulf-Cluster* betrieben werden soll, muß im `BIOS` noch die Option "`Halt on no error`" aktiviert werden, damit der PC später beim Booten nicht stehen bleibt und "meckert", weil keine Tastatur angeschlossen ist. Beachten Sie, daß inzwischen einige ATX-Motherboards auf dem Markt sind, die nicht booten, wenn kein Monitor an der Graphikkarte angeschlossen ist. Solche Motherboards sind für den Aufbau eines *Beowulf-Clusters* nicht geeignet.
Nachdem der Kernel von der Bootdiskette geladen wurde, können Sie die Hardwareadresse des Clients notieren, die später bei der Server-Konfiguration benötigt wird:

```
ne2k-pci.c: PCI NE2000 clone 'RealTek RTL-8029' at I/O 0xe800, IRQ 10.
eth0: PCI NE2000 found at 0xe800, IRQ 10, 00:00:B4:55:7E:DA.
Sending BOOTP requests....
```

2.3 Häufige Probleme

Nach dem Booten taucht häufig eines der drei folgenden Probleme auf:

- *M$ Windoze* startet
  Der PC startet anscheinend nicht den Linux-Kernel von der Bootdiskette. Stellen Sie sicher, daß die *Boot Sequence* im `BIOS` auf "`A,...,C`" gestellt wurde.

[3] nach `/usr/src/linux/Documentation/nfsroot.txt`
[4] *headless node* ist die Bezeichnung für einen Rechner, der ohne Monitor und Tastatur in einem *Beowulf-Cluster* arbeitet.

- Fehlermeldung: "`Unable to open at leat one network device`"
  Der Kernel hat die Netzwerkkarte Ihres PCs nicht erkannt. Dieses Problem entsteht, wenn Sie entweder den Netzwerkkartenteiber nicht in den Kernel gelinkt haben, oder der Netzwerkkartentreiber nicht den richtigen I/O-Port der Karte gescannt hat.
  Im ersten Fall müssen Sie den Kernel mit integriertem Netzwerkkartentreiber neu kompilieren; im zweiten Fall nehmen Sie im Source-Code des Netzwerkkartentreibers einfache Änderungen vor. Ich hatte anfangs z.B. das Problem, daß eine Netzwerkkarte einen unüblichen I/O-Port belegte; ich mußte also in der Datei "`/usr/src/linux/drivers/net/ne.c`" diesen I/O-Port (0x2A0) hinzufügen, um eine (sehr alte) NE2000 zum Laufen zu bringen:

```
/* A zero-terminated list of I/O addresses to be probed. */
static unsigned int netcard_portlist[] =
 0x300, 0x2A0, 0x280, 0x320, 0x340, 0x360, 0;
```

- Fehlermeldung: "`Unable to contact NFS-server for root fs, using /dev/fd0 instead`"
  Ihr Linux-Client schickt `BOOTP`-Anfragen in das lokale Netzwerk und bekommt keine Antwort von einem `BOOTP`-Server; Sie sollten überprüfen, ob der Client mit dem Server verbunden ist und der Server bereits in der Lage ist `BOOTP`-Anfragen zu beantworten. Falls er das noch nicht kann, beginnen Sie jetzt mit der Konfiguration des Servers.

3 Konfiguration des Servers

3.1 DNS – Domain Name System

Unter Linux gibt es zwei Lösungen für die Namensauflösung eines Rechners: Die "`/etc/hosts`-Lösung" ist für den Testbetrieb oder in kleineren Netzwerken geeignet. Die zweite Lösung, ein *Domain Name Server*, ist leistungsfähiger, sollte allerdings nur von einem erfahrenen Systemadministrator installiert werden, da hier bereits kleine Fehler verheerende Folgen auf den Traffic des Internet haben können.

3.1.1 DNS mit der `/etc/hosts`-Datei

Hier das Beispiel einer "`/etc/hosts`"-Datei; Sie sollten in dieser Datei entweder die für Ihre Rechner reservierten IP-Adressen oder die für private Netzwerke reservierten IP-Adressen aus dem Bereich `192.168.0.1` bis `.255` eintragen. Bei privaten, nicht registrierten Netzwerken verwenden Sie einen noch nicht vergebenen Namen für Ihre Domain (hier z.B. "`joof.de`"), um spätere Konflikte bei der Namensauflösung zu vermeiden.

/etc/hosts

```
#IP-Adresse   kompletter Hostname kurzer Hostname Beschreibung
127.0.0.1     localhost.joof.de   localhost
192.168.0.1   n1.joof.de          n1              # k6-2 350, meine Linux Box
192.168.0.2   n2.joof.de          n2              # k6-2 450, tjfs erster w98 PC
192.168.0.3   n3.joof.de          n3              # Celeron 450, tjfs zweiter w98 PC
```

In der Datei "`/etc/host.conf`" sollten folgende zwei Einträge zu finden sein:

/etc/host.conf

```
order hosts
multi on
```

Wenn Sie S.u.S.E. Linux verwenden, so dürfen Sie die Datei "`/etc/host.conf`" nicht direkt editieren, sondern müssen alle Änderungen mit Hilfe von `YaST` vornehmen:
"`System administration`" → "`Network configuration`" → "`Configuration nameserver`" → "`Do you want to acces a nameserver?`" → "`No`"

3.1.2 DNS mit `BIND`

Die Konfiguration eines Nameservers ist eine eigene Wissenschaft. Deswegen empfehle ich das Buch "*BIND und DNS*" aus dem O'Reilly-Verlag für ein tieferes Studium dieser Materie. Ich werde im folgenden nur die einzelnen Dateien auflisten und zeigen, was man alles editieren muß, um einen funktionsfähigen *Domain Name Server* zu installieren; mein *Domain Name Server* ist *Primary-Server* für die Domain "`joof.de`" und *Cache-Server* für alle anderen Domains. Meine Konfigurationsdateien für `BIND` (Version 4.9.7) sehen wie folgt aus:

/etc/named.boot

```
directory /var/named
primary joof.de               db.joof
primary 0.168.192.in-addr.arpa db.192.168.0
primary 0.0.127.in-addr.arpa   db.127.0.0
cache . db.cache
```

/var/named/db.joof

```
@ IN SOA n1.joof.de. root.joof.de. (
 1
 10800
 3600
 604800
 86400 )
IN NS n1.joof.de.
n1          IN A 192.168.0.1
localhost   IN A 127.0.0.1
n2          IN A 192.168.0.2
n3          IN A 192.168.0.3
n4          IN A 192.168.0.4
n5          IN A 192.168.0.5
n6          IN A 192.168.0.6
n7          IN A 192.168.0.7
n8          IN A 192.168.0.8
n9          IN A 192.168.0.9
www         IN CNAME n1
ftp         IN CNAME n1
secure_zone IN TXT "192.168.0.0:255.255.255.0"
            IN TXT "127.0.0.1:H"
```

/var/named/db.192.168.0

```
@ IN SOA n1.joof.de. root.joof.de. (
1
10800
3600
604800
86400 )
    IN NS n1.joof.de.
1   IN PTR n1.joof.de.
2   IN PTR n2.joof.de.
3   IN PTR n3.joof.de.
4   IN PTR n4.joof.de.
5   IN PTR n5.joof.de.
6   IN PTR n6.joof.de.
7   IN PTR n7.joof.de.
8   IN PTR n8.joof.de.
9   IN PTR n9.joof.de.
```

/var/named/db.127.0.0

```
@ IN SOA localhost.joof.de. root.joof.de. (
1
10800
3600
604800
86400
   IN NS n1.joof.de.
1  IN PTR localhost.joof.de.
```

Die Datei “`/var/named/db.cache`” ist in jeder `BIND`-Distribution zu finden oder kann bei “`ftp://ftp.internic.net`” bezogen werden; sie sollte immer auf dem aktuellen Stand gehalten werden, damit Ihr *Domain Name Server* einwandtfrei funktioniert.
Den *Domain Name Server* starten Sie in der S.u.S.E.-Distribution als Benutzer “`root`” mit dem Befehl “`/sbin/init.d/named start`”. Danach sollten Sie mit dem Befehl “`tail -f /var/log/messages`” ungefähr folgende Meldungen in der Syslog-Datei lesen können:

tail -f /var/log/messages

```
Apr  5 5:28:38 n1 named[1613]: starting
Apr  5 5:28:38 n1 named[1613]: primary zone "joof.de" loaded (serial 1)
Apr  5 5:28:38 n1 named[1613]: primary zone "0.168.192.in-addr.arpa" loaded (serial 1)
Apr  5 5:28:38 n1 named[1613]: primary zone "0.0.127.in-addr.arpa" loaded (serial 1)
Apr  5 5:28:38 n1 named[1613]: cache zone "" loaded (serial 0)
Apr  5 5:28:38 n1 named[1614]: Ready to answer queries.
```

Oft wird in den Konfigurationsdateien des Servers an wichtiger Stelle ein Punkt vergessen und der Server startet mit einer anderen als oben angegebenen Meldung; um sicherzustellen daß Ihr *Domain Name Server* einwandfrei funktioniert, sollten Sie die Meldungen des Servers genau überprüfen! Benutzen Sie auch das Tool `nslookup` und testen, ob die Namensauflösung für alle Hosts in ihrer Domain funktioniert (z.B. mit “`nslookup n1`”, “`nslookup 192.168.0.1`”, “`nslookup n1.joof.de`”).
Überprüfen Sie auch die Dateien “`/etc/host.conf`” und “`/etc/resolv.conf`”:

/etc/host.conf

```
order hosts bind
multi on
```

/etc/resolv.conf

```
search joof.de
nameserver 192.168.0.1
```

Die letzten beiden Dateien werden bei S.u.S.E. Linux über `YaST` konfiguriert:
“`System administration`” → “`Network configuration`” → “`Configuration nameserver`” → “`Do you want to acces a nameserver?`” → “`Yes`”

3.2 SYSLOGD – System Logging Daemon

Sie können die Systemmeldungen aller Clients auf den Server umleiten (dies ist zum Beispiel bei *Beowulf-Clustern* sehr hilfreich, wenn man nicht ständig von einem Rechner zum anderen Springen will, um nachzusehen, ob auf ihm alles geschmiert läuft.); dazu müssen Sie den `SYSLOG`-Daemon des Servers mit der Option “`-r -s joof.de`” starten und bei jeder “/**etc**/**syslog.conf**” der Clients den Eintrag “`*.* @<IP Adresse des Servers>`” aufnehmen.
Unter S.u.S.E. Linux editieren Sie auf dem Server den Eintrag “`SYSLOGD_PARAMS=¨r -s joof.de¨`” in der Datei “`/etc/rc.config`” und starten als Benutzer `root` den `SYSLOGD` mit dem Befehl “`/sbin/init.d/syslog reload`” neu.

3.3 NFSD – Network File System Server

Der NFS-Server ermöglicht den Dateisystemzugriff über das Netzwerk. Das Einrichten eines NFS-Servers ist einfach: die Datei “`/etc/exports`” enthält Einträge, die dem NFS-Server sagen, welcher NFS-Client mit welchen Rechten auf die Festplatte des NFS-Servers zugreifen darf; Clients, die ihr Root-Dateisystem von einem NFS-Server mounten, müssen besondere Rechte vom NFS-Server erhalten; die Option “`no_root_squash`” sagt dem NFS-Server, daß Root-Rechte nicht umgesetzt werden[5]:

/etc/exports

```
/        192.168.0.2(rw,no_root_squash,link_absolute)
```

[5] Standardmäßig wird die Option “`root_squash`” verwendet, die bewirkt, daß der Benutzer `root` des NFS-Clients keine für `root` typischen Sonderrechte auf dem Dateisystem des Servers hat. Erreicht wird dies, indem Zugriffe mit der User-ID 0 auf die User-ID 65534 (-2) umgesetzt werden. Diese User-ID ist dem Benutzer `nobody` zugewiesen.

Die Datei “/etc/exports” wird vom NFS-Server gelesen. Werden an ihr Änderung vorgenommen, so muß der NFS-Server neu gestartet werden. Mit S.u.S.E.-Linux wird dies mit dem Befehl “/sbin/init.d/nfsserver restart” erreicht. Außerdem sollte die Datei “/etc/rc.config” noch den Eintrag “NFS_SERVER="yes"” enthalten, damit der NFS-Server bei jedem Neustart hochgefahren wird.

Zwei häufige Probleme mit NFS: Greifen 32, 64 oder mehr Linux-Clients auf den NFS-Server zu, kann es mit dem in der aktuellen S.u.S.E.-Distribution enthaltenen NFS-Server zu einem der zwei folgenden Probleme kommen:

- Der NFS-Server liefert die Daten mit Verzögerungen;
  Lösung: man sollte einen zweiten oder dritten NFS-Server im Netz installieren.
- Es kommt zu Laufeitfehlern, wenn viele Linux-Clients gleichzeitig auf die gleichen symbolischen Links zugreifen;
  Lösung: man sollte die symbolischen Links der Dateien im jeweiligen Root-Dateisystem der Clients eventuell durch physische Kopien dieser Dateien ersetzen. Dies erfordert zusätzlich rund 20 MByte Festplattenplatz je Linux-Client.

3.4 BOOTP – Internet Boot Protokoll Server

Sobald ein BOOTP-Server eine BOOTP-Anfrage von einem Client herhält, schickt er an ihn, falls seine Hardwareadresse in der Datei “/etc/bootptab” aufgelistet ist, alle nötigen Informationen, um sein TCP/IP-Protokoll zu konfigurieren; damit der BOOTP-Server zu aktiviert wird, muß in der Datei “/etc/inetd.conf” das #-Zeichen vor dem Eintrag für den BOOTP-Server entfernt werden und die Datei “/etc/bootptab” existieren:

/etc/inetd.conf

```
# Tftp service is provided primarily for booting.  Most sites
# run this only on machines acting as "boot servers."
#
#tftp    dgram   udp     wait    nobody  /usr/sbin/tcpd  in.tftpd /tftpboot
bootps  dgram   udp     wait    root    /usr/sbin/bootpd        bootpd -c /tftpboot
```

“inetd”, der *Internet Super Server* muß nach Anpassung von “/etc/inetd.conf” noch neu gestartet werden; bei S.u.S.E-Linux geschieht das mit “/sbin/init.d/inetd reload”.
Die Zeilen 1 bis 9 der Datei “/etc/bootptab” enthalten die für alle Clients gültige Standardkonfiguration. In den nachfolgenden Zeilen werden die für jeden Client individuellen Einstellungen aufgelistet; hier muß man “<kurzer_Hostname>:<IP-Adresse>:<Hardwareadresse (ohne Doppelpunkte!)>:<Standardkonfigurationsname>:” angeben. Beispiel:

/etc/bootptab

```
1  .default:\                                        #Standardkonfiguration für alle Clients
2          :ht=ether:\                               #Hardwartyp
3          :dn=joof.de:\                             #Domainname
4          :ds=192.168.0.1:\                         #DNS-Server IP-Adresse
5          :sm=255.255.255.0:\                       #Subnetz-Maske
6          :ns=192.168.0.1:\                         #IEN-116 Name-Server Adressliste
7          :gw=192.168.0.1:\                         #Gateway IP-Adressse
8          :vm=rfc1048:\                             #Vendor magic cookie selector
9          :hn:to=-18000:                            #Send clients hostname to client
10 n2:ip=192.168.0.2:ha=0000b4591716:tc=.default:    #hostname, IP- und Hardwareadresse
11 n3:ip=192.168.0.10:ha=0000b48c2c8f:tc=.default:
```

Ein Client, der in /etc/bootptab eingetragen ist, sollte beim Booten automatisch seine IP-Konfiguration erhalten und folgendes anzeigen:

```
Sending BOOTP requests....<7>ARP: arp called for own IP address OK
Root-NFS: Got BOOTP answer from 192.168.0.1, my address is 192.168.0.2
Root-NFS: Got file handle for /tftpboot/192.168.0.2 via RPC
VFS: Mounted root (nfs filesystem).
INIT: version 2.75 booting
```

3.5 Einrichten des NFS-Root-Dateisystems

Im Verzeichnis "/tftpboot/<IP_Adresse_des_Clients>/" befindet sich das jeweilige Root-Dateisystem eines Clients. Hier werden alle Client-speziefischen Anpassungen gemacht; das Einrichten eines NFS-Root-Dateisystems ist nicht einfach, weshalb ich ein an die S.u.S.E. 6.1-Distribution angepasstes Skript (add_node) geschrieben habe[6]. Dieses Script sollte nach "/usr/sbin/" kopiert werden und nur vom Systemadministrator ausführbar sein. Beispiel:
"/usr/sbin/add_node n2 192.168.0.2 0000B48C2C8F" erzeugt ein NFS-Root-Dateisystem für den Linux-Client "n2" mit Hardwareadresse "00:00:B4:8C:2C:8F" und IP-Adresse "192.168.0.2" und macht noch ungefähr folgende (hier unvollständig zusammengefasste) Anpsassungen:

- Erzeugen des Verzeichnisses "/tftpboot/192.168.0.2" und der symbolischen Links "n2" und "n2.joof.de" darauf.
- Erzeugen der Unterverzeichnisse "home,root,mnt,tmp,opt,usr"
- Kopieren einiger Dateien aus dem /usr-Verzeichnis, die beim Booten noch vor dem Mounten des richtigen /usr-Verzeichnisses benötigt werden ("/usr/sbin/rpc.ugidd,zic", "/usr/bin/grep", "/usr/share/zoneinfo/MET").
- Hardlinks aller Dateien der Verzeichnisse "/sbin, /bin, /dev"
- Erzeugen eines speziell angepassten "/var- und /etc"-Verzeichnisses:
  - "/etc/HOSTNAME"
  - "/etc/rc.config"
  - Hardlinks auf "/etc/passwd", "/etc/shadow", "/etc/gshadow", "/etc/issue", "/etc/issue.net", "/etc/profile", "/etc/csh.cshrc", "/etc/hosts"
  - Servername in "/etc/hosts.equiv" aufnehmen.
- Update der Dateien "/etc/exports", "/etc/hosts.equiv", "/etc/bootptab" und Neustart des inetd und nfsd.

4 Povray auf einem Parallelrechner

Wenn man auf mehreren Rechnern ein Linux-System laufen hat, ist es natürlich reizvoll, die Rechenpower aller Rechner parallel zu nutzen; dazu gibt es die *Parallel Virtual Machine* (*PVM*), ein Programm, das Prozesse auf mehrere Rechner verteilen kann. Jetzt muß man nur noch nach einem Programm suchen, das so viel Rechenpower benötigt, daß es sich lohnt dafür extra einen Parallelrechner einzurichten; eines dieser resourcenverschlingenden Programme nennt sich Povray, ein freier Raytracer (vgl. http://www.povray.org). Für diesen Raytracer gibt es einen *PVM*-Patch[7], der ursprünglich von Brad Kline (Cray Research Inc.) entwickelt wurde. Weitere Anpassungen wurden später von Andreas Dilger und Harald Deischinger gemacht.
Bei S.u.S.E.-Linux bleibt einem die Patch-Arbeit erspart, da *PVM*-Povray als eigenes Paket (pvmpov, Serie beo) mitgeliefert wird.

4.1 Installation und Test von Povray

Zuerst sollte nur das Grundpaket povray Serie gra installiert und getestet werden. Meine Povray-Konfiguration sieht wie folgt aus:

~/.povrayrc

```
Width = 640
Height = 480
Pause_when_Done = off
Bounding_Threshold = 3
Test_Abort=On
Test_Abort_Count=100
```

[6] "http://www.luga.de/~flierl/add_node" (eine auf Red Hat Linux 5.0 angepasste Version ist unter "ftp://ftp.sci.usq.edu.au/pub/jacek/beowulf-utils" zu finden.)
[7] http://www.geocities.com/CapeCanaveral/Lab/6386/pvmpov/pvmpov.html

```
Library_Path=/usr/lib/povray3
Library_Path=/usr/lib/povray3/include
Library_Path=/home/flierl/lego/include
```

Hier ein paar Befehle, die Povray dazu bringen ein Bild zu berechnen und unter X anzuzeigen:

```
flierl@n1:~/ > cd && mkdir pov && cd pov
flierl@n1:~/pov > cp /usr/lib/povray3/pov3demo/recurse/sponge1.* .
flierl@n1:~/pov > x-povray -i sponge1.pov +d +a0.003
```

4.2 Parallel Virtual Machine & PVM-Povray

Falls alles soweit funktioniert, kann man die Pakete **pvm**, **xpvm** und **pvmpov** (Serie beo) und installieren. Beachten Sie, daß Berechnungnen mit PVM-Povray im Cluster aus Sicherheitsgründen nicht von root ausgeführt werden können. Damit man mit PVM-Povray arbeiten kann, muß man zunächst die *Parallel Virtual Machine* starten; damit die *PVM* später funktioniert muß man erst testen, ob man sich als normaler Benutzer mit "**rlogin**" in einen anderen Rechner einloggen kann, **ohne** dabei ein Passwort eingeben zu müssen. Gegebenenfalls sollten Sie noch einmal alle "**/etc/hosts.equiv**"-Dateien überprüfen. Jetzt noch die Parallel Virtual Machine konfigurieren ...

~/.bashrc

```
export PVM_ROOT=/usr/lib/pvm3
export XPVM_ROOT=/usr/X11R6/lib/xpvm
```

... die Rechner mit "**add <Rechnername>**" hinzufügen, die *PVM*-Konfiguration mit "**conf**" anzeigen lassen, ...

```
flierl@n1:~ > pvm
pvm> add n2
1 successful
                    HOST     DTID
                      n2    80000
pvm> conf
2 hosts, 1 data format
                    HOST     DTID     ARCH   SPEED       DSIG
                      n1    40000    LINUX    1000 0x00408841
                      n2    80000    LINUX    1000 0x00408841
pvm>quit
```

... und das gleiche Bild mit PVM-Povray nochmal berechnen:

```
flierl@n1:~/pov > /usr/X11/bin/x-pvmpov -i sponge1.pov +d +a0.003 +N
```

Wichtig ist, daß man *PVM*-Povray immer mit dem kompletten Pfad aufruft. Sollte *PVM*-Povray ein Bild nicht richtig berechnen, so ist es oft hilfreich, die *PVM*-Option mit "-N" zu deaktivieren, um so eventuelle Fehlermeldungen zu analysieren.

4.3 PVM-Povray patchen

Leider enthält der *PVM*-Povray Patch (Version 3.1) noch ein Memory Leak in der Datei "**render.c**", der das Programm bei längeren Berechnungen gerne zum Abstürzen bringt; deswegen sollte man diese Datei noch von Hand patchen; man sollte bei S.u.S.E.-Linux zunächst die Pakete **povray.spm** und **pvmpov.spm** installieren, dann mit "**cd /usr/src/packages/SPECS && rpm -bp pvmpov.spec**" den Sourcecode entpacken, patchen, mit "**cd ../BUILD/povray3/sources**" in den Povray Source Code wechseln und folgende, mit **/*ME*/** gekennzeichneten Anpassungen machen ...

/usr/src/packages/BUILD/povray3/source/render.c

```
void Initialize_Renderer PARAMS((void))
{
  char **Grid;
  int i, xi, yi, Grid_Size;
  int Standard_Sample_Grid_Size;
  size_t size;
  DBL x, y, len;
  DBL T1;
  VEC2 *Standard_Sample_Grid;

  maxclr = (DBL)(1 << Color_Bits) - 1.0;

  size = (Frame.Screen_Width + 1) * sizeof(COLOUR);

/*ME*/
  if (Previous_Line!=NULL) POV_FREE(Previous_Line);
  if (Current_Line!=NULL) POV_FREE(Current_Line);
/*ME*/
  Previous_Line = (COLOUR *)POV_MALLOC(size, "previous line buffer");
  Current_Line  = (COLOUR *)POV_MALLOC(size, "current line buffer");

  for (i = 0; i <= Frame.Screen_Width ; i++)
  {
    Make_ColourA(Previous_Line[i], 0.0, 0.0, 0.0, 0.0, 0.0);
    Make_ColourA(Current_Line[i],  0.0, 0.0, 0.0, 0.0, 0.0);
  }

  if (opts.Options & ANTIALIAS)
  {
    size = (Frame.Screen_Width + 1) * sizeof(char);

/*ME*/
  if (Previous_Line_Antialiased_Flags!=NULL)
POV_FREE(Previous_Line_Antialiased_Flags);
  if (Current_Line_Antialiased_Flags!=NULL)
POV_FREE(Current_Line_Antialiased_Flags);
/*ME*/

    Previous_Line_Antialiased_Flags = (char *)POV_MALLOC(size, "previous
line flags");
    Current_Line_Antialiased_Flags  = (char *)POV_MALLOC(size, "current
line flags");
    for (i = 0; i <= Frame.Screen_Width ; i++)
    {
      Previous_Line_Antialiased_Flags[i] = 0;
      Current_Line_Antialiased_Flags[i]  = 0;
    }
  }

  /* If PVM, this is already enough */
  if(PvmTasks && !PvmSlave)
      return;
```

... dann mit ...

```
cd /usr/src/packages/BUILD/povray3/source/pvm
aimk newunix; aimk newxwin; aimk newsvga
cd ../..
install -m 755 source/pvm/LINUX/pvmpov /usr/bin
install -m 755 source/pvm/LINUX/x-pvmpov /usr/X11R6/bin
install -m 755 source/pvm/LINUX/s-pvmpov /usr/bin
DOCDIR=/usr/doc/packages/pvmpov
```

```
DOC="README.pvmpov pvm_install.doc pvmpov.doc"
install -d -m 755 \${DOCDIR}
install -m 644 ${DOC} ${DOCDIR}
cd /usr/src/packages/BUILD
rm -r povray3
```

... *PVM*-Povray neu installieren. Fertig!

4.4 POVmark – Povray benchmarking

Um die Effizienz des Clusters zu testen, kann man unter "`http://www.haveland.com/povbench/`" die Beispieldatei "`skyvase.pov`" downloaden und testen. Die Web-Seite führt die Resultate anderer Systeme auf, mit denen man die selbst erzielten Ergebnisse vergleichen kann.

PVM-Povray Benchmark

```
/usr/X11/bin/x-pvmpov -i skyvase.pov +h480 +w640 +FT +v1 -d +a0.300 -q9
-mv2.0 -b1000 -nw32 -nh32
```

4.5 MPEG-Animation mit PVM-Povray

- Den Video-Encoder von "`ftp://havefun.stanford.edu/pub/mpeg/MPEGv1.2.1.tar.Z`" downloaden und installieren.
- Den MPEG-Player "`mpeg_play`" (Serie `gra`) und die "`PPM-Tools`" (Serie `gra`) mit "`YaST`" installieren.
- Eine nette Povray-Datei und eine Ini-Datei mit speziellen Befehlen für den MPEG-Encoder erstellen, z.B.:

  test.pov

```
#include "colors.inc"
global_settings { assumed_gamma 2.2 }
camera { location <sin(clock*2*pi)*10, 10, cos(clock*2*pi)*10> look_at  <0, 0, 0> }
light_source { <10, 10, 10> colour White }
sky_sphere { pigment { gradient y color_map {[0.0 Gray50][1.0 Gray15]}} }
plane { y, 0 texture { pigment {Gray75} } translate -y }
sphere { <0, abs(sin(clock*4*pi)*5), 0>, 0.5 texture {pigment {Red}} }
```

  test.ini

```
+w320
+h240
Input_File_Name=test.pov
Initial_Frame=1
Final_Frame=60
Initial_Clock=0
Final_Clock=1
Cyclic_Animation=on
Pause_when_Done=off
Output_to_File=On
Output_File_Type=p
Pre_Frame_Command=mkdir -p %s_mpeg
Post_Frame_Command=ppmtoyuvsplit %s_mpeg/`expr %n + 0` %o && rm %o
Post_Frame_Return=f
Post_Scene_Command=mpeg -PF -h %w -v %h -a 1 -b $(expr %n - 1) %s_mpeg/ -s %s.mpeg && rm -fr %s_mpeg
Post_Scene_Return=f
```

- ... und mit folgenden Befehlen berechnen und anzeigen lassen:

```
flierl@n1:~/pov > /usr/X11/bin/x-pvmpov test.ini +d
flierl@n1:~/pov > mpeg_play -dither color test.mpeg
```

5 Links

- NFS-Root Mini-Howto Andreas Kostyrka, andreas@ag.or.at: "`ftp://metalab.unc.edu/pub/Linux/docs/HOWTO/mini/NFS-Root`"
- Kernelpatches für ein Swap-Dateisystem über NFS: "`http://www-math.math.rwth-aachen.de/~LBFM/claus/nfs-swap/`"