

Linux Business: Firmenvernetzung, High Availability und Virtualisierung

LiHAS – LinuxHaus Stuttgart
Adrian Reyer
are@lihas.de

Erwartungen an HA

Einfach:

Alles soll immer funktionieren.

Was ist 'alles'?

Server

Clients

Internes Netzwerk

Internetanbindung

Drucker

Telefon

Menschen

Klassisch betrachten HA-Lösungen aber nur die
Server

Menschenredundanz

Teuer, Plätze müssen doppelt besetzt werden,
obwohl einer reicht.

Frustrierend, man ist ja überflüssig

Besser:

vernünftige Dokumentation und Einweisung eines
Kollegen für den Notfall, also eher ein Backup als
Redundanz.

Telefonredundanz

Bei heutiger Mobiltelefonverbreitung relativ unkritisch. Die Telekommunikationsprovider können recht schnell alle Anrufe an die eigentliche Festnetznummer auf Handynummern oder anders angebundene Festnetznummern umleiten. FAX genügt, wie FAX mit Laptop und Handy geht sollte man allerdings vorher mal geübt haben.

Drucker- / Clientredundanz

Auch kein Problem, die wenigsten Firmen haben nur einen Drucker und nach wie vor kann jeder Laserdrucker die alte Laserjet Emulation, ein Ersatz ist daher schnell aufgebaut.

Ein Arbeitsplatzrechner kostet auch nicht die Welt, da lässt sich schon einer vorhalten.

Redundanz im lokalen Netz

Der logistische Aufwand um das lokale Netzwerk wirklich ausfallsicher zu machen, zwei unabhängige Anschlüsse pro Client, d.h. Verdopplung der Switchports und der zu verlegenden Kabel ist immens.

Den verwendeten Switchtyp als Ersatz im Schrank liegen zu haben ist wohl meist ausreichend.
WLAN als Alternative intressant, insbesondere in Verbindung mit Laptops.

Redundanz Server

- redundante Netzteile
- RAID-Systeme
- Unterbrechungsfreie Stromversorgungen

Problem: nicht alles ist redundant, es gibt genügend einzelne mögliche Fehlerpunkte im Server.

Hier gibt es diverse Anbieter, die hochkommerzielle Produkte hierfür anbieten, z.B. VMware ESX, Microsoft Cluster Server, Citrix mit XEN

Redundante Internetanbindung

- Internetzugang ist wichtig
 - Richtung Staat: Steuererklärungen
 - Richtung Kunde: Projektabgaben
z.B. im Designbereich
- Offizieller redundanter Internetzugang ist teuer und oft nicht zu Ende gedacht
 - 2MBit-Leitung (32 ISDN B-Kanäle gebündelt) mit ISDN-Fallback, der im gleichen Leitungsbündel liegt und vom gleichen Provider versorgt wird

Traditionelle Lösung

Eine Menge Meetings

Eine Angebotssumme weit über den finanziellen
Möglichkeiten

Um die Finanzierung zu ermöglichen werden Teile
gestrichen

Nach 3 Jahren redet man immer noch darüber, getan
ist aber nichts

Der Linuxweg

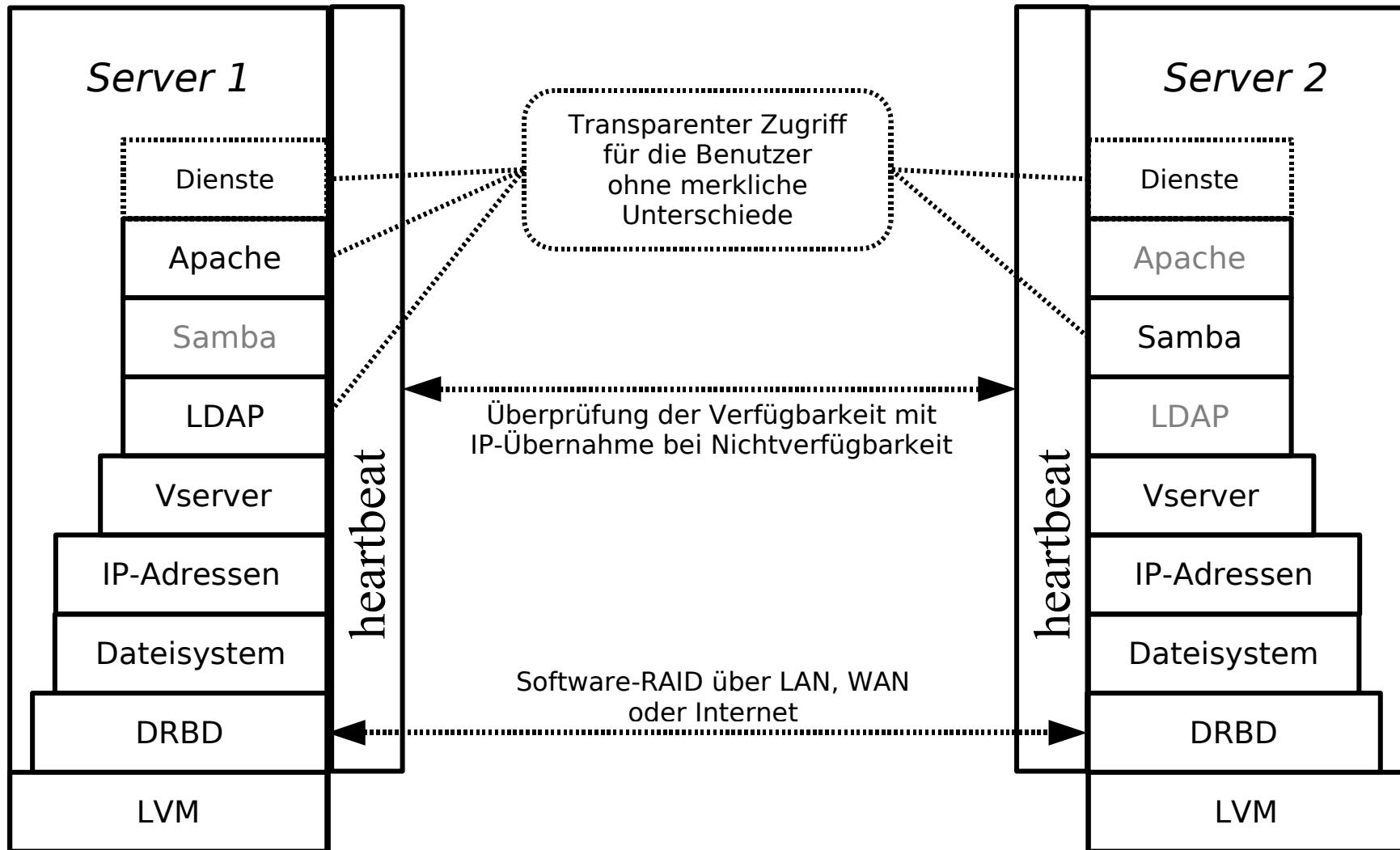
Der UNIX-Ansatz ist, viele kleine Tools zu verwenden, die zusammen die Lösung bringen. Mit Linux kann man diesen Ansatz übernehmen.

DRBD – in etwa RAID-1 über Netzwerk
heartbeat – überprüft ob der andere Server noch
verfügbar ist

linux-vserver – ein praktischer Kontainer um
Konfigurationen zu gruppieren

-> Zusammen gibt das einen HA-Cluster

Aufbau redundanter Server



Vorteile

- Keine Spezialhardware nötig
- Die beiden Server müssen nicht identisch sein
- Die alte Hardware kann weiter verwendet werden (muss aber nicht)
- Debugging funktioniert mit denselben Werkzeugen wie auch bei einem normalen Server

insbesondere:

- Der alte Admin kann weiter verwendet werden, die Administration im vserver unterscheidet sich nicht sehr von einem normalen System und die heartbeat-/drbd-/vserver-konfiguration ist nicht so schwer

Linuxansatz Internetanbindung

- Verschiedene unabhängige Internetzugangsmöglichkeiten nutzen
 - ADSL, SDSL
 - TV-Kabel
 - Funkstrecken
 - UMTS
- Diese Leitungen zusammenschalten, z.B. mittels einem Rootserver, OpenVPN und epl

Hier ist das Ende

Glückwunsch, Ihr habt überstanden

Problemstellung

- Redundante Server für alle Dateibasierten Dienste sind mit Linux mit freier Software realisierbar, z.B. mit der Kombination linux-vsserver, drbd, nagios, heartbeat
- lokale Netzwerke lassen sich durch Hardwareeinsatz redundant auslegen, aber wie geht das mit der Internetanbindung?

Problembehandlungsmöglichkeiten

- ignorieren, im Zweifelsfall dem ISP oder Hardwarehersteller die Schuld geben
- offizielle und allgemeine Lösung des Problems durch betreiben eines eigenen AS und damit Teilname am Internetrouting mit mehreren Standleitungen mit festen IPs zu verschiedenen Providern
- verschiedene Breitband-/Funkinternetleitungen + mindestens 2 Rootserver + OpenVPN + Cloud

Ansatz 'ignorieren'

- gebräuchliche Lösung
- funktioniert nicht
- unfair

AS Ansatz (I)

- teuer
 - Die Zuteilung einer Autonomous System Number (ASN) kostet
 - mindestens zwei Leitungen zu verschiedenen Uplinks mit festen IPs nötig
 - unter 2 /24 (Klasse C) Netzen wird das Routing von den großen im Netz oft ignoriert und kommt für kleine Firmen oder gar Privatpersonen damit nicht in Frage

AS Ansatz (II)

- aufwändig
 - offizieller Papierkram mit dem RIPE und mehreren Providern
 - wirklich gute Routing und Routingrouting Kenntnisse sind nötig

OpenVPN-Ansatz

- Benötigt mindestens zwei beliebige Internetzugänge, gerne auch mit dynamischen IPs, z.B. DSL, Kabel, UMTS (ab ca. 25EUR * Leitung / Monat)
- Benötigt mindestens einen, zur Redundanz besser zwei Rootserver mit jeweils 2 festen IPs (ab ca 40 EUR * Server / Monat)
- von Außen sind zwei verschiedene IPs zu verwenden (kann z.T. auf Dienstebasis entschärft werden)

Systembeschreibung

- mit dem EQL-Device werden 2 OpenVPN-Verbindungen gebündelt, jeweils auf beiden Seiten
- die OpenVPN-Verbindungen werden mittels iptables/FWMARK und iproute2 auf die beiden Leitungen verteilt

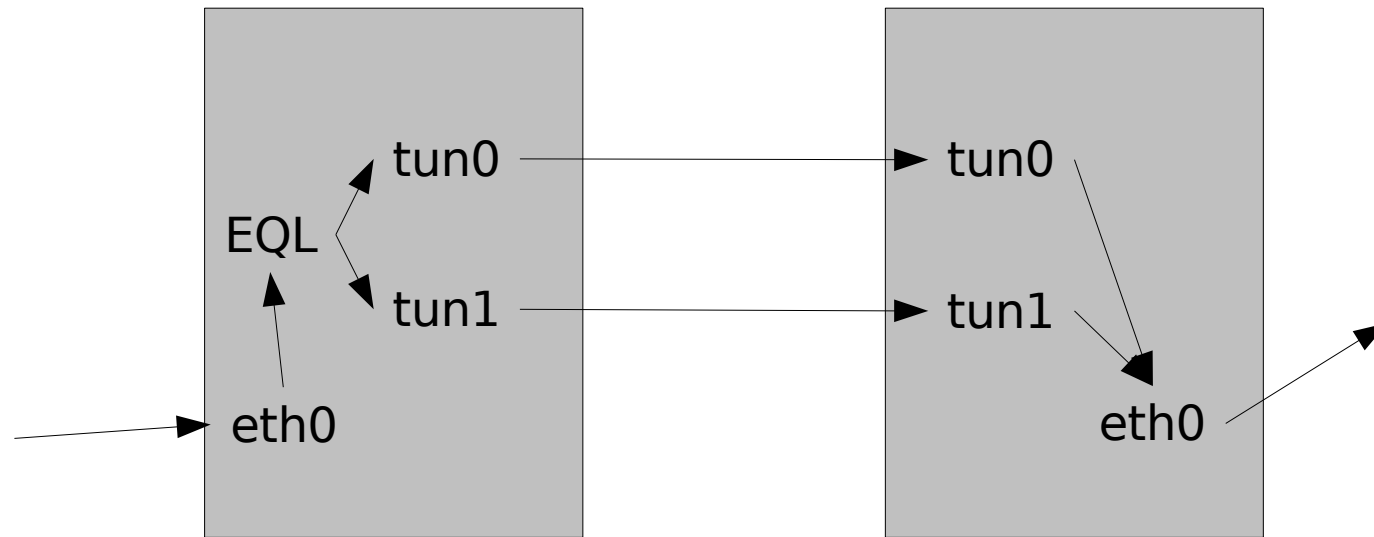
Probleme

- Es gehen gleiche IP-Adressen über verschiedene Leitungen aus dem gleichen Rechner
 - OpenVPN mit tun-Devices (IP-Tunnel) ist sehr wählerisch bezüglich der IP-Adressen im Tunnel -> tap-Devices (Ethernet Tunnel) verwenden und viel Ärger sparen
 - IP-Spoofprotection des Kernels muss abgeschaltet werden, per Default reagiert der Linuxkernel allergisch auf IPs, die am 'falschen' Interface ankommen

Probleme (II)

- Wenn DSL direkt vom Rechner aus gemacht wird und nicht über Router muss bei jeder Neueinwahl der NAT-Code aktualisiert werden, bei der Verwendung von Routern ist das nicht nötig
- eql selbst mag Neustarts nicht, das Modul muss neu geladen werden

Routingbeschreibung (I)



Routingbeschreibung

- Ein Paket das über eine der Leitungen soll geht zunächst an eth0 des Gateways, dort ist das Default GW auf das eql-Device (verhält sich wie ein PPP Link) gesetzt, dieses verteilt das Paket auf einen der beiden Tunnel. Auf der anderen Seite kommt das Paket aus dem Tunnel und verlässt den Rechner via eth0, geht also eben nicht über ein dortiges eql-Device
- Umgekehrt geht das Paket auf dem Rootserver durch das eql und auf dem lokalen Rechner nicht

Details – ip route

Neue Routingtabellen dem ip-route bekannt machen durch hinzufügen von

189 vpn1189

190 vpn1190

191 vpn1191

192 vpn1192

zu `/etc/iproute2/rt_tables` (reicht für 4 verschiedene Leitungen)

Details – openvpn

Eine Konfigurationsdatei

`/etc/openvpn/clientX.conf` pro

Internetanschluss, gegenüber einer normalen `client.conf` kommen folgende Zeilen hinzu:

`dev tun2`

`up /usr/local/sbin/lihasvpn-tun.sh`

'`dev tunX`' ist eine laufende Nummer, 1 Device pro Internetanbindung

das `up`-Skript wird benötigt um die OpenVPN-Verbindung dem `eq1` Unterzuordnen

/usr/local/sbin/lihasvpn-tun.sh

```
#!/bin/bash
tun_dev=$1; tun_mtu=$2; link_mtu=$3; ifconfig_local_ip=
$4; ifconfig_remote_ip=$5; mode=$6
if [ "$mode" == "init" ]; then
  case "$tun_dev" in
    tun2)
      # ADSL-Leitung, 500 UP, 6000 DOWN
      eql_enslave eql $tun_dev 500
      logger -t lihasvpn $*: eql_enslave eql $tun_dev 500
      ;;
    *)
      logger -t lihasvpn $*: no device with this name handled
      ;;
  esac
else logger -t lihasvpn $*: mode $mode; fi
```

/etc/network/interfaces

```
iface eql inet static
    address 172.21.1.2
    pointopoint 172.21.1.1
    netmask 255.255.255.255
    broadcast 172.21.1.2
    mtu 1500
```

/etc/ppp/ip-up.d/lihasvpn

```
#!/bin/bash
PATH=/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin
export PATH

PPP_IFACE="$1"; PPP_TTY="$2"; PPP_SPEED="$3"; PPP_LOCAL="$4";
PPP_REMOTE="$5"; PPP_IPPARAM="$6"

export PPP_IFACE PPP_TTY PPP_SPEED PPP_LOCAL PPP_REMOTE \
      PPP_IPPARAM

PORT_eth1=1189; PORT_ppp0=1190; PORT_ppp1=1191; \
      PORT_ppp2=1192; PORT_ppp3=1193

IPTABLES=iptables; IPROUTE=ip
VPNSERVERS="serverip1 serverip2"
VPNPORT=`eval echo \${`echo PORT_"\$PPP_IFACE"``

logger -t lihasvpn Handling Startup $PPP_IFACE, $PPP_LOCAL, \
      VPN-Port $VPNPORT
```

```
( set -x
for server in $VPNSERVERS; do
$IPTABLES -I OUTPUT -t mangle -p udp -d $server --dport $VPNPORT \
-j MARK --set-mark $VPNPORT

$IPTABLES -I PREROUTING -t mangle -p udp -d $server --dport $VPNPORT \
-j MARK --set-mark $VPNPORT

$IPTABLES -I POSTROUTING -t nat -p udp -d $server --dport $VPNPORT \
-j SNAT --to-source $PPP_LOCAL

$IPTABLES -I OUTPUT -t nat -p udp -d $server --dport $VPNPORT -j SNAT \
--to-source $PPP_LOCAL

$IROUTE route add table vpn$VPNPORT $server dev $PPP_IFACE
done
$IROUTE route add table vpn$VPNPORT $PPP_REMOTE dev $PPP_IFACE \
proto kernel scope link src $PPP_LOCAL

$IROUTE rule add fwmark $VPNPORT table vpn$VPNPORT

echo -n 1 > /proc/sys/net/ipv4/route/flush ) 2>&1 | logger -t lihasvpn
```

/etc/ppp/ip-down.d/lihasvpn

ist das Gegenstück zu /etc/ppp/ip-up.d/lihasvpn, alles was an Regeln dort erstellt wird, muss hier wieder gelöscht werden.

Router statt PPPoE/PPP?

Wenn der Rechner selbst die DSL-Verbindung nicht herstellt, sondern ein Router verwendet wird, muss die entsprechende Netzwerkkarte statt PPP entsprechend konfiguriert werden. Da die IP zwischen den verschiedenen Routern und dem Client Rechner immer dieselbe ist, können die SNAT-Tricks weggelassen werden.

Lösungsansatz

- komplette Hardware doppelt vorhalten
- Daten auf beiden Systemen vorhalten
 - rsync / unison: klassisch, zeitverzögert, eher Backup als Redundanz
 - Shared Storage (SAN, Netzwerkdateisysteme): wieder einzelne Komponente
 - drbd: vergleichbar RAID-1 über Netzwerk

Folgeproblematik

- Wer führt die Umschaltung durch?
- Wer konfiguriert Dienste um?
- Wer passt Start-/Stoppskripte an die Clusterregeln an?
- Wie können Updates eingespielt werden ohne alles wieder von vorne konfigurieren zu müssen?

Umschaltung

- Wer führt die Umschaltung durch?
 - manuell: langsam
 - automatisch mit heartbeat: schnell, Fehlerkennungen eines Ausfalls möglich.
 - Verbindungen zwischen Rechnern mehrfach redundant auslegen.
 - Möglichst den anderen Server im Fall einer Übernahme abschalten lassen

Umkonfiguration

- zusätzliche Startskripte für Clusterdienste und Erweiterung bestehender Skripte / Konfigurationsdateien um die für den Clusterbetrieb notwendigen Unterscheidungen
 - mühsam und fehleranfällig
 - nach einem Update ist potentiell wieder alles kaputt
- besser: virtuelle Server als Dienst

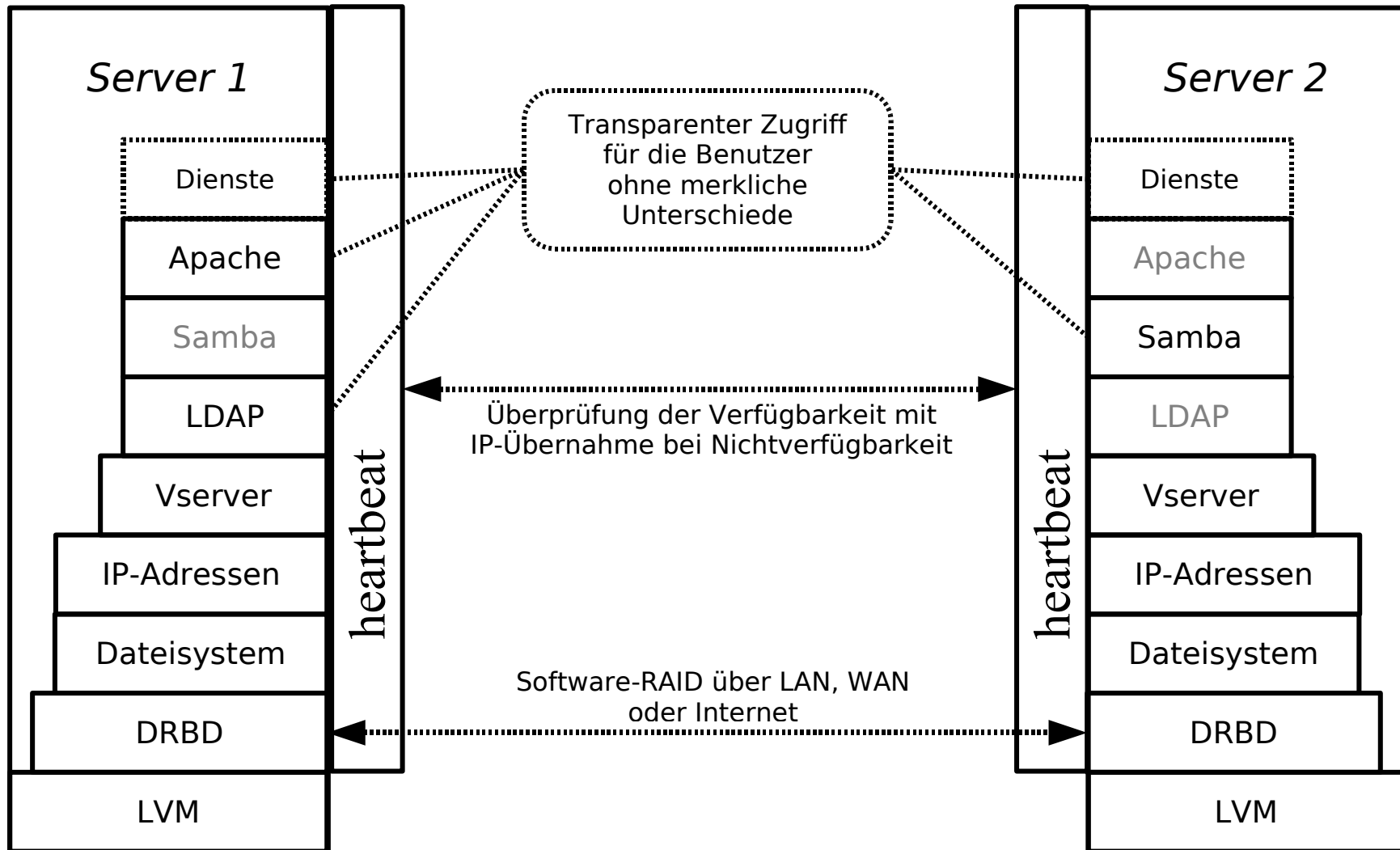
Virtuelle Server – vserver

- vergleichbar chroot, aber ausbruchsicher
- geringer Ressourcenverbrauch (~0.5% gegenüber Standardserver)
- Möglichkeit einen vserver in den Ressourcen einzuschränken
- vserver kommt komplett mit dem eigentlichen Dienst und den Helferdiensten wie z.B. cron, MTA, logrotate

Virtuelle Server – vserver (2)

- hat nur bestimmte vom Hauptserver vorgegebene Rechte
- Hardwareunabhängig (nicht Architekturunabhängig)
- Im vserver sind keine Kenntnisse über den Cluster notwendig
- Abzüglich der Hardware administrierbar wie ein Einzelserver

Aufbau redundanter Server



LVM

- dynamische Verwaltung von Festplattenplatz, auch im laufenden Betrieb
- keine (relevante) Beschränkung der Anzahl der Partitionen

DRBD

- vereinfacht: Netzwerk RAID 1
- schreibende Anwendung bekommt erst dann das OK, wenn beide Seiten die Daten geschrieben haben.
- Synchronisationsrichtung kann einfach gedreht werden.
- <http://www.drbd.org/>

IP Aliase

- Die Hardware im Cluster braucht eine IP
- In den meisten Fällen sind die redundanten Dienste Netzwerkdienste
 - sie brauchen damit eigene IPs
 - Switches müssen IPs mit mehreren MAC-Adressen auf verschiedenen Ports unterstützen

linux-vserver

- Virtualisierungslösung auf dem gleichen Kernel wie das Hardwaresystem
- Benutzung vergleichbar `chroot()` - im Inneren kann ein fast normales Linuxsystem administriert werden
- Berechtigungen Systemeigenschaften im Kernel zu ändern sind üblicherweise aus (Netzwerkkonfiguration, Mounts)
- <http://www.linux-vserver.org/>

heartbeat

- überprüft die Erreichbarkeit der anderen Maschine
- schaltet Dienste an und aus
- Dienste:
 - IPs
 - Richtung der DRBD-Synchronisation
 - vsyncd
- <http://www.linux-ha.org/>