Linux-HA-Cluster mit Heartbeat und DRBD

Reservespieler

Ein Hochverfügbarkeits-Cluster unter Linux ist auch mit einfachen Mitteln zu realisieren. Der Einstieg in die Oberliga für Server benötigt nur Standardkomponenten und die beiden freien Programme Heartbeat und DRBD – und schon steht beim Absturz des Mittelstürmers ein Reservespieler bereit. Andreas Sebald

Hohe Ausfallsicherheit ist schon durch Einsatz zweier billiger PCs möglich, auch wenn die Hersteller edler Highend-Hardware dabei mit den Zähnen knirschen. Der häufigste Anwendungsfall für HA-Cluster (High Availability) dieser Art dürften Standarddienste wie Samba, Apache oder eine Datenbankapplikation sein. Diese Art Cluster stellt ein Backup-System (Hot Standby) für Dienste zur Verfügung.

Im Gegensatz zu Computing- oder Load-Balancing-Clustern ist einer der Server in diesem Fall passiv und wird erst tätig, wenn der Arbeitsserver ausfällt. Die von der Anwendung benötigten Daten werden dabei über das Netz gespiegelt, ein gemeinsamer Datenspeicher (Shared Storage) ist nicht nötig: Der Linux HA-Cluster ist ein Share Nothing Cluster. Da beide Rechner komplett ausgestattet sind und sich keine gemeinsamen Ressourcen teilen, fällt ein Single Point of Failure weg. Das sieht zunächst nach höheren Investitionen aus, aber die speziellen SCSI- oder Firewire-Komponenten für Shared Storage Cluster sind teuer, die Share Nothing Cluster kommen mit Standardkomponenten aus.

Komponenten besser redundant

In einer Produktivumgebung ist es zu empfehlen, Komponenten wie Netzteile, Hubs, Switches und Festplatten von guter Qualität einzusetzen und sie redundant auszulegen. Für einfache HA-Cluster genügen jedoch schon zwei einfache PCs mit jeweils zwei Netzwerkkarten, auf denen eine übliche Linux-Distribution läuft, im beschriebenen Beispiel Suse 9.0 Professional.

In ihr sind die zur Clusterbildung benötigten Pakete – Heartbeat und DRBD – bereits enthalten. Der Admin installiert sie über Yast und startet sie über den Runlevel-Editor. Eine neuere Linux-Version einzusetzen ist derzeit nicht ratsam, DRDB in den zu Redaktionsschluss aktuellen Releases 0.6xx arbeitet noch nicht mit Kernel 2.6 zusammen. Von Version 0.7 gibt es nur eine Pre-Release.

Bevor es losgeht

Beide Systeme bekommen die übliche Partitionsaufteilung, am besten möglichst gleichartig, also zum Beispiel »/swap«, »/« und »/boot«. Für die Daten, die der Cluster zur Verfügung stellen soll, ist jeweils noch eine separate Partition notwendig. Sie sollte auf beiden Rechnern etwa gleich groß sein und darf keinen Mountpoint erhalten und nicht beim Systemstart angebunden werden; diese Optionen sind über den Schaltknopf »Fstab-Optionen« im Yast-Menü »Partitionen bearbeiten« zugänglich (siehe **Abbildung 1**).

Als Dateisystem ist ein Journaling Filesystem wie Ext 3 oder ReiserFS zu empfehlen. XFS hingegen wird von DRDB erst ab Version 0.7 unterstützt. Die beiden Netzwerkkarten bekommen feste IP-Adressen. Im Beispiel für Karte 1 (»eth0«) die 140.10.0.1 auf dem primären Knoten, auf dem anderen Server bekommt die Karte 1 (»eth0«) die Adresse 140.10.0.2 als sekundärer Knoten. Das sind auch die Adressen, die öffentlich sichtbar sind.

Dabei ist an dieser Stelle eine deutliche Warnung auszusprechen: Die Clustersoftware nimmt tief gehende Manipulationen im Netz vor. Da beide Server abwechselnd mit der gleichen IP-Adresse im Netz sein werden, kann ein Fehler leicht alle aktiven Netzwerkkomponen-

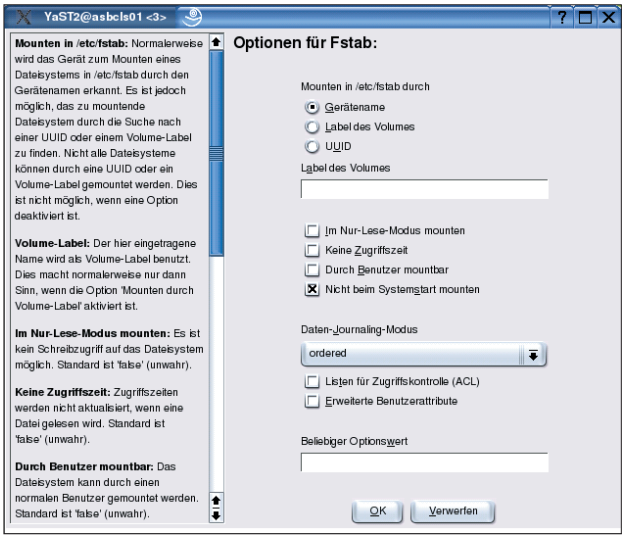

Abbildung 1: Die Option »Nicht beim Systemstart mounten« muss aktiviert sein. Das Mounten ist Aufgabe der Clustertools. Als Device wird später die DRBD-Ressource in die Datei »/etc/fstab« eingetragen.

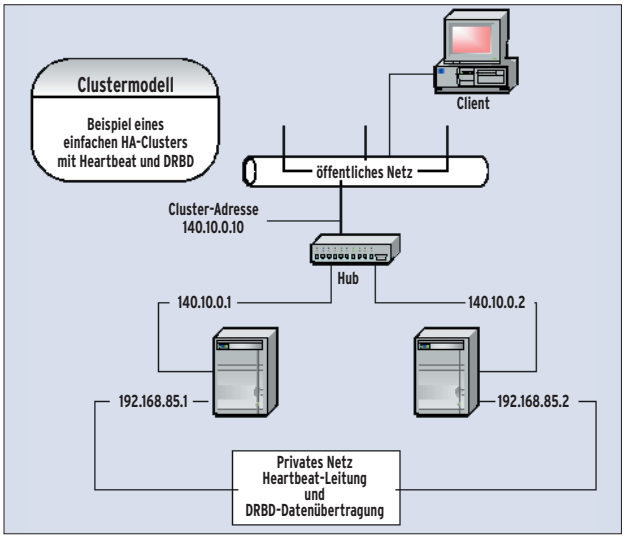

Abbildung 2: Ein einfacher Share Nothing Cluster. Im öffentlichen Netz ist unter der Clusteradresse (140.10.0.10) der aktive Knoten erreichbar. Das Spiegeln der Daten und die Clusterüberwachung per Heartbeat erfolgen über das private Netz.

ten lahm legen. Die jeweils zweite Netzwerkkarte (»eth1«) ist für die private, die Heartbeat-Leitung zuständig. Im vorliegenden Beispiel hat Knoten 1 als IP-Adresse 192.168.85.1 und Knoten 2 die 192.168.85.2. Die Ports beider Netzwerkkarten verbindet der Admin mit einem Netzwerk-Crossover-Kabel oder einem separaten Hub (Abbildung 2).

Howtos schlagen oft ein Nullmodem-Kabel als Heartbeat-Leitung vor, weil Heartbeat dann auch noch mit kaputtem IP-Stack funktioniert. Jedoch ist dieses Argument fragwürdig: Wenn ein Linux-Server seinen IP-Stack verloren hat, ist der Ernstfall eingetreten. Spottbillige 100-MBit-Netzwerkkarten erlauben es heutzutage, außer Heartbeat auch den Datenabgleich über das private Netz laufen zu lassen, also im günstigsten Fall direkt über ein Crossover-Kabel. Das minimiert zum einen die Gefahr eines Angriffs auf die Daten. Zum anderen ist diese Leitung unbelastet vom Zugriffsdatenstrom der Benutzer, die Übertragungskapazität kann so voll ausgeschöpft werden.

Trivial, aber wichtig sind eindeutige Namen für beide Rechner. Es ist eine gute Idee, die Namen für beide, die private und die öffentliche IP-Adresse, jeweils in die »/etc/hosts« einzutragen. Schließlich sollte der sekundäre Server noch seine Zeit per »xntp« mit dem primären Server abgleichen; es ist darauf zu achten, den Zeitunterschied der beiden Server vor der Inbetriebnahme des Clusters auszugleichen.

Listing 1: »/etc/ha.d/ha.cf«

```
01 #
02 # /etc/ha.d/ha.cf
03
04 debugfile /var/log/ha-debug
05 logfile /var/log/ha-log
06 # Wie schnell soll auf eine fehlende Antwort
   reagiert werden?
07 keepalive 2
08 deadtime 30
09 warntime 10
10 initdead 120
11 # Soll bei Restart des primaeren Knotens
12 # sofort zurueckgeschaltet werden?
13 # auto_failback on
14 # oder
15 # nice_failback off
16 # die alte Form "nice_failback off" sollte nicht
   mehr benutzt werden,
17 # aber die Heartbeat-Version von Suse 9.0
   versteht den Parameter noch nicht.
18 # Ist eine serielle Heartbeat-Leitung
   vorhanden?
19 #baud   19200
20 #serial /dev/ttyS0
21 # Ist der Standard UDP belegt?
22 #udpport        694
23 # Ueber welches Interface soll der Heartbeat
   laufen?
24 bcast   eth1
25 # kann sich der Server selbst ueberwachen?
26 #watchdog /dev/watchdog
27 # Ist ein STONITH-Geraet auf der anderen Seite?
28 #stonith baytech /etc/ha.d/conf/stonith.baytech
29 # Welche Maschinen sind im Cluster?
30 node    knoten1
31 node    knoten2
```

Herzklopfen

Das Programm Heartbeat aus dem High Availability Linux Project [3] ist ein Dienst, der seinen Clusterpartner überwacht und im Ernstfall (Failover) die für die Anwender notwendigen Dienste startet sowie die Cluster-IP-Adresse im Netz übernimmt. Welche die notwendigen Dienste sind, hängt natürlich vom Verwendungszweck des Clusters ab. Für die gebräuchlichsten Anwendungen genügen die Start-Skripte aus »/etc/init.d« oder es werden entsprechende Skripte mit Heartbeat geliefert.

Bei der Installation wurde bereits jedem Knoten eine öffentliche IP-Adresse auf »eth0« zugewiesen. Für die Konfiguration von Heartbeat ist noch eine gemeinsame öffentliche IP-Adresse notwendig. Unter dieser virtuellen Adresse ist später der jeweils aktive Knoten im Netz erreichbar. Im Beispiel ist 140.10.0.10 die IP-Adresse (Listing 2). Für diese Adresse sollte auch ein DNS-Eintrag erfolgen, der den Namen des Clusters im Netz bekannt gibt.

Heartbeat wird mit drei Dateien konfiguriert: »/etc/ha.d/ha.cf«, »/etc/ha.d/resources« und »/etc/ha.d/authkeys«. Sie sind intern ausführlich dokumentiert.

Heartbeat kommt mit wenigen Einträgen in der Konfigurationsdatei »/etc/ha.d/ha.cf« aus (siehe Listing 1). Die ausführlichen Kommentare im mitgelieferten Original wurden zugunsten der besseren Übersicht entfernt. Die Datei hält die grundlegenden Eigenschaften von Heartbeat fest, etwa welche Knoten wie am Cluster beteiligt sind.

Heartbeat startet Serverdienste

Die Ressourcendatei »/etc/haresources« legt fest, welche Dienste mit welcher IP-Adresse immer im Netz erscheinen sollen. Listing 2 zeigt ein einfaches Beispiel zur Konfiguration des zustandslosen HTTP-Servers Apache ohne dynamische Webseiten. Ein anderer Dienst, der nicht unbedingt dynamische Daten braucht, wäre DNS.

Die in dieser Datei stehenden Anwendungen dürfen natürlich nicht beim Systemboot geladen werden – ab jetzt ist Heartbeat für den korrekten Start der benötigten Programme zuständig. Erst wenn Heartbeat festgestellt hat, dass es selbst auf dem aktiven Knoten läuft, startet es die benötigten Programme. Die Datei »/etc/ha.d/haresources« muss unbedingt auf beiden Knoten den gleichen Inhalt haben.

Um den Cluster in Gang zu setzen, fehlt noch die »/etc/ha.d/authkeys«-Datei (Listing 3). Sie enthält in Klartext das Passwort für die Heartbeat-Verbindung und muss auf beiden Knoten gleich sein. Der Datei-Inhalt folgt immer dem folgenden Schema, die Reihenfolge der Einträge spielt keine Rolle:

```
auth Nummer
Nummer Auth-Methode [Auth-Schlüssel]
```

Da ein im Klartext lesbares Passwort sicherheitskritisch ist, muss die Datei zwingend die Zugriffsrechte 600 haben. Dabei ist »crc« eigentlich keine Authentifizierung, sondern dient der Erkennung von Übertragungsfehlern.

Die Authentifizierung nach »sha1« ist wohl die sicherste Methode unter den drei gezeigten. Die Datei lässt sich jedoch um andere Authentifizierungen erweitern, da die Methoden dynamisch eingebunden werden. Es kann zwar immer nur eine Methode im Cluster angewendet werden, aber je mehr Methoden

Listing 2: »/etc/ha.d/haresources«

```
01 #
02 # /etc/ha.d/haresources
03
04 # - knoten1 ist der primäre Knoten
05 # der Knotenname muss mit dem ha.cf Eintrag
06 # übereinstimmen und einen echten Server benennen
07 # - der freigegebene Dienst ist Apache
08 # - der Dienst wird unter der IP-Adresse
09 #   140.10.0.10 im Netz angeboten
10 knoten1 140.10.0.10 apache
```

Listing 3: »/etc/ha.d/authkeys«

```
01 auth 3
02 # 1 crc
03 3 md5 meinPasswort
04 # 2 sha1
```

definiert sind, desto schwieriger wird es für Angreifer, im Netz die richtige zu finden. Falls eine sehr exotische Signatur zum Einsatz kommt, sind Brute-Force-Angriffe besonders schwer. Bei einer sicheren Netzverbindung wie einem Crossover-Kabel ist überhaupt keine Schlüsselsignatur nötig; dann reicht »crc« völlig aus.

Das Herz beginnt zu schlagen

Suse erlaubt es zwar, Heartbeat bequem mit dem Runlevel-Editor zu starten, bei den ersten Tests ist die gute alte Shell jedoch die weitaus bessere Wahl, da sich die Startmeldungen leichter verfolgen lassen:

```
#~ /etc/init.d/heartbeat start
Starting High-Availability Services  done
```

Sobald Heartbeat auf dem ersten Rechner läuft, arbeitet dieser als primärer und damit aktiver Knoten. Um einen Cluster zu bilden, ist Heartbeat nur noch auf dem anderen Server zu starten. Auf diesem stellt es dann fest, dass der andere Knoten auf seine Anfrage antwortet, und startet die Resource-Dienste nicht lokal, sondern wartet auf den Ernstfall. In der Datei »/var/log/ha-log« ist der Verlauf des Starts auf den jeweiligen Servern nachzulesen. Nun ist der gewünschte Dienst – im Beispiel Apache – im Netz mit der virtuellen IP-Adresse erreichbar.

Auf dem passiven Knoten versucht Heartbeat permanent den anderen Knoten zu erreichen. Sobald die Verbindung abbricht, macht Heartbeat den Server, auf dem es noch läuft, zum aktiven Knoten: Der Failover-Fall ist eingetreten, der sekundäre Knoten übernimmt die Aufgabe, die Cluster-Dienste als Server im Netz bereitzustellen. Die Umschaltzeit beträgt mit den Standardeinstellungen etwa 20 Sekunden. Diese Zeit braucht der Knoten, um sicher zu sein, dass sein Partner auf der anderen Seite wirklich nicht mehr arbeitet und nicht nur etwa wegen zu hoher Netzlast ein paar IP-Pakete verschwunden sind.

Listing 4: »/etc/drbd.conf«

```
#
# drbd.conf Beispiel
#
# Parameter die anzupassen sind:
# - resource name, hier drbd0
# - hostname
# - device
# - disk,
# - IP-Adresse
# - port in der "on <hostname> {}" section
# - disk-size.
#
resource drbd0 {
  protocol = C
  fsckcmd  = fsck.ext3 -y
  disk {
    # Wenn der lower level device einen io-error meldet, haben wir ein Problem
    # und der andere Knoten soll uebernehmen
    do-panic
    # maximale Devicegroesse in KByte runtergerundet auf 4
    disk-size = 1194300k
  }
  net {
    sync-min    = 500k  # syncer versucht ueber dieser Rate zu bleiben
    sync-max    = 100M  # maximale durchschnittliche syncer Bandbreite
    tl-size     = 5000  # transfer log Groesse
    timeout     = 60    # 0.1 Sekunden
    connect-int = 10    # Sekunden
    ping-int    = 10    # Sekunden
  }
  on knoten1 {
    device  = /dev/nb0     # virtueller Block-Device, steht auch in /etc/fstab
    disk    = /dev/hdb1    # logische Partition, wird von YAST oder manuell angelegt
    address = 192.168.85.1 # IP-Adresse fuer die DRBD-Daten
    port    = 7788         # Port fuer die DRBD-Daten
  }
  on knoten2 {
    device  = /dev/nb0
    disk    = /dev/hdb2
    address = 192.168.85.2
    port    = 7788
  }
}
```

Listing 5: »/etc/fstab

```
/dev/hda1          /                 reiserfs    defaults                1 1
/dev/hdb1          swap              swap        pri=42                  0 0
devpts             /dev/pts          devpts      mode=0620,gid=5         0 0
proc               /proc             proc        defaults                0 0
usbdevfs           /proc/bus/usb     usbdevfs    noauto                  0 0
/dev/cdrecorder    /media/cdrecorder auto        ro,noauto,user,exec     0 0
/dev/cdrom         /media/cdrom      auto        ro,noauto,user,exec     0 0
/dev/fd0           /media/floppy     auto        noauto,user,sync        0 0
/dev/nb0           /usr1             ext3        noauto                  0 0
```

Failover – Machtübernahme im Ernstfall

Entschließt sich der sekundäre Knoten dazu, die Macht im Netz zu übernehmen, dann zwingt er mit einem speziellen IP-Broadcast alle aktiven Netzwerkkomponenten und Computer in seinem Subnetz, ihren ARP-Cache zu leeren, und meldet sich mit der virtuellen Cluster-IP-Adresse. Clients verlieren dabei zwar die Verbindung, merken aber nach dem neuen Verbindungsaufbau keinen Unterschied.

Das Schlimmste, was in einer Failover-Situation passieren kann, ist, dass der primäre Knoten noch munter weiter am Netz hängt, während der sekundäre Knoten plötzlich auch mit derselben IP-Adresse im Netz auftaucht. Dieses Split-Brain-Problem ist nur durch Stonith zu lösen, das ist ein Akronym mit der martialischen Bedeutung „Shoot the other node in the head“.

Stonith funktioniert nur mit spezieller Hardware, die bei Failover dem anderen Server den Strom abdreht oder einen vorgeschalteten Hub stilllegt, indem ein Befehl eine schaltbare Steckerleiste (Power Switch) abschaltet [5]. Dazu muss ein entsprechender Treiber in der »/etc/ha.d/ha.cf« konfiguriert sein.

Datenpumpe über Heartbeat

Um dynamische Daten im Cluster auf beiden Servern redundant zu halten, ist eine Art Raid-System erforderlich, das alle Schreibzugriffe der Festplatte des aktiven Knotens über das Netzwerk auch auf dem sekundären Knoten spiegelt. Dazu gibt es zwei Programme die Ähnliches leisten. Zum einen E-NBD, das Extended Network Block Device, eine nicht sehr effiziente Methode, die

ohne besondere Partitionen auskommt. Zum anderen DRBD (Distributed Replicated Block Device), das von dem Österreicher Philipp Reisner als Diplomarbeit entwickelt wurde [2].

Der Kern von DRBD ist ein Daemon, der beim Booten startet, während das Mounten der Clusterpartition normalerweise später ein Heartbeat-Skript übernimmt. DRBD benötigt jeweils eine eigene Partition auf jedem Clusterknoten. Die Partition wird mit eigenen Devices angesprochen, die in der Suse-Distribution bereits angelegt sind, ansonsten sind sie mit

```
#~ for i in 0 1 2 3 4 ; ➥
      do mknod /dev/nb$i b 43 $i ; ➥
      done
```

schnell erstellt. Die Partitionen kann nur der aktive Knoten lesend und schreibend mounten, der passiven Knoten sollte sie nicht anfassen; manche Filesysteme verhindern sogar den Lesezugriff.

Für DRBD muss nur die Datei »/etc/drbd.conf« bearbeitet werden. Der Start erfolgt wie bei Heartbeat über den Runlevel-Editor oder über »/etc/init.d/drbd start«. DRBD ist ein Programm, das unabhängig von Heartbeat arbeitet, aber als Clusterdienst von ihm verwaltet werden kann. Die Konfiguration von »/etc/drbd.conf« (Listing 4) erfordert etwas mehr Aufwand als Heartbeat.

Ein unscheinbarer, aber wichtiger Eintrag ist der Protokolltyp. Philipp Reisner hat sich beim Schreiben von DRBD auch Gedanken zur Übertragungsperformance und -Sicherheit gemacht. So gibt es drei Übertragungsprotokolle, die sich in der Behandlung der zu schreibenden Datenblöcke unterscheiden:

- A: Eine Schreiboperation ist auf dem Primärknoten beendet und bestätigt, sobald die Daten dem Netzwerk übergeben wurden.
- B: Eine Schreiboperation ist dann auf dem primären Knoten beendet, wenn die Daten auf dem sekundären Knoten angekommen sind.
- C: Eine Schreiboperation ist erst dann auf dem primären Knoten beendet, wenn die Daten auf dem sekundären Knoten angekommen sind und der primäre eine Schreibbestätigung des sekundären Knotens hat.

Für einen Cluster, der ja für höhere Zuverlässigkeit sorgen soll, kommt Protokoll A nicht in Frage. Der Einsatz von Protokoll B ist ein Kompromiss, aber ein ruhiges Gewissen im täglichen Betrieb gewährt nur Typ C. Die Datei »/etc/drbd.conf« legt auch die logische Partitionsgröße fest. Natürlich kann diese nicht größer sein als die physikalische Partition, die etwa mit »fdisk -l« feststellbar ist. Die Größenangaben erfolgen in Byte oder KByte und sollten wegen der Blockgröße von 4 KByte auf ein Vielfaches von vier abgerundet werden.

Quelle und Ziel der Daten

Nun muss DRDB noch wissen, woher und wohin mit den Daten. Dabei spielen die Knotennamen eine ähnliche Rolle wie bei Heartbeat. Jeder Knoten braucht vier Angaben: das logische Device, die Partition, die IP-Adresse des Knotens und die Portnummer, mit der die Übertragung arbeiten soll.

Zum einfacheren Verständnis zeigt das Beispiel nur eine DRBD-Partition (»drbd0«), in Wirklichkeit kann man mehrere Partitionen konfigurieren und simultan betreiben. Die Datei »/etc/drbd.conf« sollte in den wesentlichen Merkmalen auf beiden Knoten gleich sein, nur das Startverhalten beim Booten kann je nach geplanter Knotenrolle unterschiedlich eingestellt werden.

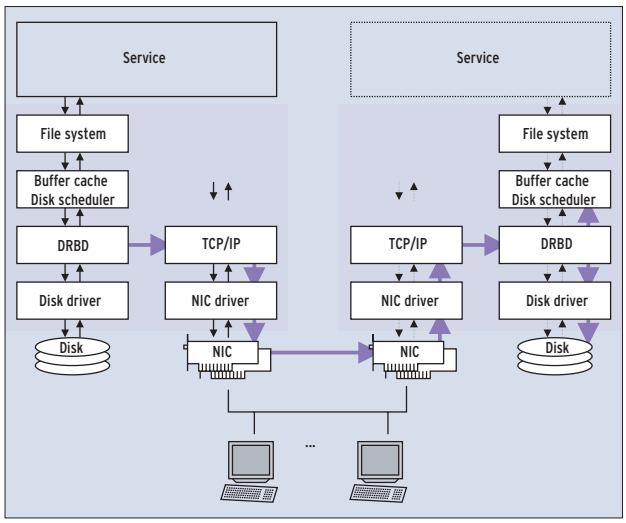

Abbildung 3: DRBD als Prinzipskizze. Das Spiegeln der Daten erfolgt transparent übers Netzwerk, DRDB übernimmt die Verwaltung der beteiligten Partitionen komplett, die Anwendung sieht keinen Unterschied. (Quelle: [2])

Der erste Start

Für die grundlegende Funktion von DRBD ist der Einsatz von Heartbeat nicht notwendig, alle Tests für die Datenübertragung funktionierten auch unabhängig einwandfrei. Als Block-Device (»/dev/nb0«) ist mit einem Editor die DRBD-Ressource, wie sie in »/etc/drbd.conf« definiert wurde, in die Datei »/etc/fstab« einzutragen. Die Datenpartition bekam eine eigene kleine Festplatte spendiert. Der andere Knoten im Cluster muss nicht zwangsläufig die gleiche Partitionsaufteilung haben. Die DRBD-Ressource ist in diesem Fall »/dev/nb0« und als Ext-3-Dateisystem auf »/usr1« mountbar.

Das Tool »drbdsetup« hilft beim schrittweisen Aufbau des Systems übers Netzwerk. Für den ersten Test und für alle User, denen Suses Yast das DRBD-Paket nicht eingebaut hat, hier die Maßnahmen für den manuellen Start einer DRBD-Session. Voraussetzung ist der korrekte Eintrag von »/dev/nb0« in der Datei »/etc /fstab« auf jedem Knoten. Zuerst wird der sekundäre Knoten vorbereitet:

```
insmod drbd.o
drbdsetup /dev/nb0 disk /dev/hdbx
drbdsetup /dev/nb0  net 192.168.85.1 ➥
   192.168.85.2 B
```

Wobei »/dev/hdbx« die verwendete Partition ist und das »B« hinter der Knoten-IP-Adresse das Übertragungsprotokoll einstellt. Auf dem primären Knoten ist einzugeben:

```
insmod drbd.o
drbdsetup /dev/nb0 disk /dev/hdbx
drbdsetup /dev/nb0  net 192.168.85.1 ➥
   192.168.85.2 B
drbdsetup /dev/nb0 primary
```

Damit besteht jetzt eine Verbindung zwischen den beiden Knoten und mit Hilfe von

```
mkfs -t  ext3 /dev/nb0
mount /dev/nb0 /usr1
```

wird die neue Partition formatiert und eingebunden. Damit kann jetzt auch das Spiegeln aller Daten, die auf dem primären Knoten nach »/usr1« geschrieben werden, auf den anderen Knoten erfolgen. Auf jenen Servern, bei denen Yast das Kernelmodul bereits eingebunden hat, entfällt das Laden des Moduls »drbd.o« natürlich.

Die beim Verbinden gestartete Synchronisation dauert einige Zeit, da die ganze Partition, auch der Bereich ohne Daten, erst kopiert werden muss. Die eher konservative Standardeinstellung »sync-max = 1M« in der Datei »/etc/drbd.conf« lässt sich für Verbindungen über ein Crossover-Kabel auch auf »100M« setzen. Damit verkürzt sich die Synchronisationszeit von etwa 20 Minuten pro GByte auf rund drei Minuten – wenn auf beiden Seiten eine 100-MBit-Netzwerkkarte vorhanden ist. Der Fortschritt der Synchronisation lässt sich mit dem folgenden Befehl anzeigen:

```
cat /proc/drbd
```

Es handelt sich um eine statische Anzeige, daher muss der Befehl zur Aktualisierung wiederholt werden.

Listing 6: »/etc/ha.d/haresources« mit DRBD

```
01 #
02 # /etc/ha.d/haresources
03 #
04 # Wichtig: Diese Datei muss auf beiden
05 # Cluster-Knoten gleich sein!
06 # Einfaches Beispiel einer funktionierenden
07 # Ressourcenzuteilung
08 # - knoten1 ist der primäre Knoten
09 # der Knotenname muss mit dem ha.cf Eintrag
10 # übereinstimmen und einen echten Server benennen
11 # - der freigegebene Dienst ist Apache
12 # - der Dienst wird unter der IP-Adresse
13 #   140.10.0.10 im Netz angeboten
14 # - datadisk ist das Script,
15 #   das DRBD startet und stopt
16 knoten1 140.10.0.10 datadisk::drbd0 apache
```

Listing 7: MySQL-Start

```
01 cd /var/lib
02 /etc/init.d/mysql stop
03 mv mysql /usr1/mysql
04 ln -s /usr1/mysql ./mysql
05 chmod 700 /usr1/mysql
06 chown mysql /usr1/mysql
07 chgrp daemon /usr1/mysql
08 /etc/init.d/mysql start
```

Das Starten eines Clusters, der Daten mit DRBD spiegelt ist etwas heikel; wer nicht aufpasst, riskiert einen inkonsistenten Datenbestand.

Vorsicht beim Neustart

Grundsätzlich sollte der Server, der als Letzter noch aktiv im Netz war und somit den aktuellen Datenbestand hält, als Erster wieder starten. Fährt der Server mit dem älteren Datenbestand zuerst hoch, muss er als sekundärer Server auf den anderen warten, sonst startet erzwungenerweise ein primärer Server mit eventuell veraltetem Datenbestand.

Wenn die DRBD-Konfigurationsdatei nicht explizit festlegt, dass nach einem Timeout der DRBD-Dienst als Primary oder Secondary starten soll, wartet der Server sicherheitshalber beim Booten auf eine Eingabe. Sie kann entweder »yes« (»zes« bei deutschem Tastaturlayout) für den Start als primärer Server oder »no« für das Warten auf den anderen Partner sein. Niemals dürfen beide Knoten primärer Server sein.

Ist sichergestellt, dass ein Server läuft, bekommt der zweite ein »no« als Eingabe. Ein automatischer Quicksync gleicht dann die geänderten Blöcke an. Ist der als Zweiter gestartete Knoten der Heartbeat-Primärknoten, sorgt Heartbeat dafür, dass der DRBD-Dienst dort weiterläuft. Sind aus irgendeinem Grund beide Server heruntergefahren, muss der Admin mit Vorsicht den Spiegel wieder herstellen. Wie immer sollte er den Server als Ersten starten, der als Letzter noch im Netzwerk war. Er ist nach dem Start als primärer Server mit der Eingabe von »yes« zu starten, damit ist gewährleistet, dass die Benutzerdienste sofort und mit aktuellen Daten hochfahren.

Automatischer Abgleich

Beim Start des Clusterpartners gleicht DRBD automatisch die gespiegelten Partitionen vollständig ab. Dieser Abgleich ist wie der initiale Abgleich langsam. Wurde der aktive Knoten korrekt heruntergefahren und hat der andere Knoten den Datendienst komplett übernommen, startet der Knoten auch wieder ohne nach Bestätigungen zu fragen. Es läuft nur der Quicksync mit der Meldung »SyncingAll, but I have the good data. No need to wait«. Die Benutzerdienste stehen sofort wieder bereit. Der ideale Zustand ist also erreicht, wenn immer mindestens ein Knoten läuft.

Wenn beide Clusterdienste einwandfrei funktionieren, kann DRBD in Heartbeat integriert werden. Dazu erhält in der Datei »/etc/haresources« die Zeile der zu startenden Dienste einfach zusätzlich den Aufruf des Skripts »datadisk::drbd0« (Listing 6). Dabei ist der Parameter »drbd0« der Name der DRBD-Ressource, die in der Datei »/etc/drbd.conf« definiert ist. Da nachfolgend startende Dienste in der Regel auf die Daten zugreifen, muss das »datadisk«-Skript in der Reihenfolge sofort nach der Cluster-IP-Adresse folgen.

Heartbeat und DRBD miteinander verheiraten

Das Skript »/etc/ha.d/resource.d/datadisk« ist in der Installation von Heartbeat enthalten. In der Datei »/etc/drbd.conf« muss der Befehl zum Filesystem-Check mit allen Parametern zum verwendeten Dateisystem der Clusterpartition passen; von Haus aus ist Ext 2 eingetragen. Das Skript mountet die Partition mit Schreib- und Lesezugriff sowie den Parametern, die in der Datei »/etc/fstab« eingetragen sind.

Komplexere Anwendungen wie die Datenbanken MySQL oder PostgreSQL lassen sich problemlos in das Gespann Heartbeat und DRBD integrieren. Wenn kein Resource-Skript für Heartbeat vorhanden ist, kommt das Originalskript aus »/etc/init.d« zum Einsatz. Periodisch über »cron« startende Skripte müssen immer erst prüfen, ob die zu behandelnden Daten überhaupt auf diesem Server lokal greifbar sind. Eine Datensicherung ist somit nur auf dem aktiven Knoten sinnvoll.

Die relevanten Datenbereiche sollte der Admin über eine symbolische Verknüpfung vom Ort der ursprünglichen Installation in der Clusterpartition ablegen, damit bleibt die logische Datenstruktur erhalten und die Daten liegen trotzdem im gespiegelten Bereich. Auf diese Weise können ganze Datenbanken in die Clusterpartition gelinkt werden. MySQL in eine Cluster-Anwendung verwandeln ist

mit wenigen Aktionen erledigt, wie Listing 7 zeigt. Um MySQL als Clusterdienst zu starten, muss man in der Datei »/etc/haresources« in der Zeile der Dienste lediglich noch »mysql« vor »apache« einfügen:

```
knoten1 140.10.0.10 datadisk::drbd0 ➥
 mysql apache
```

Eine auf diese Weise verschobene Datenbank lässt sich unter Suse nicht mehr mit dem Runlevel-Editor starten, das ist aber auch nicht nötig, denn im Clusterbetrieb erledigt Heartbeat den Job.

Fazit

Das einfache Beispiel zeigt, wie leicht es gelingt, eine hochverfügbare Serverumgebung aufzusetzen. Sowohl Heartbeat als auch DRBD haben noch mehr an Konfigurationsmöglichkeiten zu bieten und eignen sich für viele Szenarien. So ließe sich durch eine zweite Heartbeat-Leitung über die serielle Schnittstelle die Verfügbarkeit steigern. Selbstverständlich sind auch Shared-Storage-Lösungen mit Heartbeat zu realisieren, wenn es die Situation erfordert [5]. Ein Shared Nothing Cluster bietet jedoch für Standardanwendungen wie die hier beschriebenen deutliche Vorteile. So sind zum Beispiel Wartungsarbeiten sowohl am Server als auch am Datenspeicher problemlos durchführbar, während Anwendungen und Daten weiterhin für die Benutzer erreichbar sind.

Einer der wichtigsten Vorteile ist die räumliche Unabhängigkeit der beiden Clusterknoten, eigentlich könnten die beiden Server auf verschiedenen Kontinenten stehen. Cluster mit Shared Storage sind meist auf eine SCSI- oder Firewire-Verbindung angewiesen, die einer Längenbeschränkung unterliegen.

Der technische Aufwand für ein Heartbeat-DRBD-Cluster ist vergleichsweise gering und mit kostengünstigen Standardgeräten zu realisieren. Auch beide Programme sind als Open-Source-Projekte kostengünstig und zudem erprobt und etabliert. Sie fördern den guten Ruf von Linux als stabiles und robustes Betriebssystem. *(uwo)* ■

Infos

[1] Praktische Heartbeat-Anleitung: [http://linux-ha.org/download/GettingStarted.html]

[2] DRBD-Diplomarbeit von Philipp Reisner, TU Wien 2000: [http://www.drbd.org/publications.html]

[3] Heartbeat im Linux-HA-Projekt: [http://www.linux-ha.org]

[4] Jörg Fritsch, Theo Schlossnagle, „Dienstbarer Maulwurf“: Linux-Magazin 07/03, S. 40

[5] André Bonhôte, „Die Kraft der zwei Herzen“: Linux-Magazin 07/03, S. 46

Der Autor

Andreas Sebald ist leitender Systemverwalter einer Behörde. Aus der Unix-Welt kommend ist er darum bemüht, Linux als wirtschaftliches und stabiles Betriebssystem zu vertreten.