Fehlertolerantes Raid mit Enhanced Network Block Devices

Blockhaus in der Ferne

Hochverfügbarkeit klingt gut und teuer. Unter Linux reichen aber auch bereits Standardkomponenten für ein ausfallsicheres Speichersystem aus vernetzten, räumlich verteilten Blockdevices. Lord Hess, Arne Wiebalck

Wenn eine Serverkomponente ausfällt, ist es in einem von zwei Fällen eine Festplatte. Die Hardware ist meist für ein paar hundert Euro ersetzbar, aber schon bei kleinen Unternehmen sind die gespeicherten Daten im Schnitt Hunderttausende wert [1]. Gegen deren Verlust schützt ein Backup. Das rettet die Daten, kostet aber Zeit fürs Recovery und auch der damit verbundene Stillstand kann hohe Kosten verursachen.

Deshalb beugen hochverfügbare Systeme auch dem Zeitverlust vor. Eine ihrer Maximen ist: Der Ausfall eines Teils muss für das Ganze folgenlos bleiben. Ein Speichersystem nach diesem Grundsatz kann unter Linux aus relativ preiswerten Standardkomponenten aufgebaut werden. Das Schlüsselwort dafür heißt ENBD.

Netz-Disks

ENBD ist das Kürzel für Enhanced Network Block Device [2] und bezeichnet eine Erweiterung des Network Block Device [3] von Pavel Machek, das seit der Version 2.1.101 in den Linux-Kernel integriert ist. Beiden gemeinsam ist das Prinzip, Platten (oder nur Partitionen, ja selbst einzelne Dateien) von entfernten, jedoch via Ethernet erreichbaren Rechnern so zu mounten, dass sie wie lokale Disks erscheinen.

Anders als zum Beispiel NFS exportiert ENBD ein Device, kein Filesystem. Das Besondere daran ist, dass sich diese virtuellen Platten wie physikalisch vorhandene auch zu Raid-Gruppen unterschiedlicher Level zusammenfassen lassen. Das Attribut „enhanced“ bezieht sich unter anderem auch auf die zusätzliche Fähigkeit von ENBD, eine unterbrochene Verbindung zwischen Client und Server selbstständig wiederherzustellen.

Mit dieser Technik sind Fileserver möglich, die gar keine eigenen Platten haben und stattdessen ihre Disks über voneinander unabhängige Netzwerkverbindungen von verschiedenen Rechnern als ENBD-Laufwerke importieren. Auf diese Weise verteilt sich der Server über mehrere Räume oder Gebäude. Ein solcher Aufbau könnte überleben, selbst wenn durch einen Brand oder Wassereinbruch ein Teil total zerstört würde.

Der ENBD-Server, zuständig für den Export des Device, läuft als normaler Prozess im Userspace. Er kommuniziert über einen frei wählbaren Port mit einem ENBD-Client, der von verschiedenen Seiten solche Devices importiert. Der Datenverkehr lässt sich mit SSL verschlüsseln. Im ENBD-Client teilen sich ein Kernelmodul »enbd.o« und ein Daemon den Job (siehe Abbildung 1). Der Grund dafür ist, dass dem Modul allein nur 1 GByte Speicher zur Verfügung stünde, während sein Kompanion den gesamten physikalisch vorhandenen Speicher benutzen kann.

Doppelt gesichert

Im hier vorgestellten Beispiel gibt es drei Backend-Rechner mit jeweils sechs 200-GByte-Disks, von denen fünf zu einer Raid-5-Gruppe zusammengeschlossen sind, während die sechste als Hot-Spare-Platte dient. Beim Ausfall einer Platte aus der Raid-Gruppe springt sie automatisch ein. Der eigentliche Fileserver – das Frontend – mountet diese drei Raid-Gruppen (netto jeweils knapp 800 GByte) und schließt sie seinerseits in einem Raid 5 zusammen.

So wird doppelte Redundanz erreicht: In jedem Backend-System darf eine Platte und im Gesamtverbund ein komplettes Backend-System ausfallen, ohne dass der Betrieb unterbrochen würde oder gar Daten verloren gingen. Um die gleiche Zuverlässigkeit wie bei Geräten von etablierten Herstellern zu erreichen, musste das Projekt allerdings ein eigenes USB-Zusatzgerät entwickeln. Es basiert auf einem Mikroprozessor und einer LCD-Anzeige. Der Mikrokontroller überwacht die Drehzahlen aller Lüfter sowie die Temperaturen an sechs verschiedenen Stellen im Gehäuse.

Das Gerät wird anstelle des Diskettenlaufwerks eingebaut. Sein Display zeigt Daten wie IP-Adresse, Speichernutzung, Hostname und Ähnliches an. Bei Über-

hitzung, Plattendefekt oder Ausfall eines Lüfters wird der Administrator durch Mail, Syslog oder Ausgabe auf dem Display informiert. Schlimmstenfalls fährt der Rechner herunter, um weiteren Schaden möglichst zu vermeiden.

Zusätzlich ist es möglich, das plattenlose Frontend mehrfach auszulegen und über einen Heartbeat-Cluster zu sichern (Hinweise unter [4] und [5]). Der Preis für die auf diese Weise erreichbare hohe Fehlertoleranz besteht einerseits in der für Hot Spares und Raid-Gruppen erforderlichen Speicherkapazität: Von 3,6 TByte bleiben nur noch 1,6 TByte (44 Prozent) für Nutzdaten übrig. (Wer zum Beispiel statt drei Backends sechs einsetzt, verbessert das Verhältnis auf rund 70 Prozent.) Der zweite Nachteil: Die doppelte Berechnung der Raid-5-Paritäten kostet Zeit, sprich Performance beim Schreiben.

Die Installation eines derartigen Speichersystems ist nicht übermäßig kompliziert. Der Quellcode der ENBD-Software ist unter [6] erhältlich. Vor dem Übersetzen ist lediglich im Makefile der Pfad zu den Kernelquellen zu aktualisieren und bei Bedarf das SMP-Flag zu setzen. Für Kernelversion 2.6 ist bereits ein Patch verfügbar.

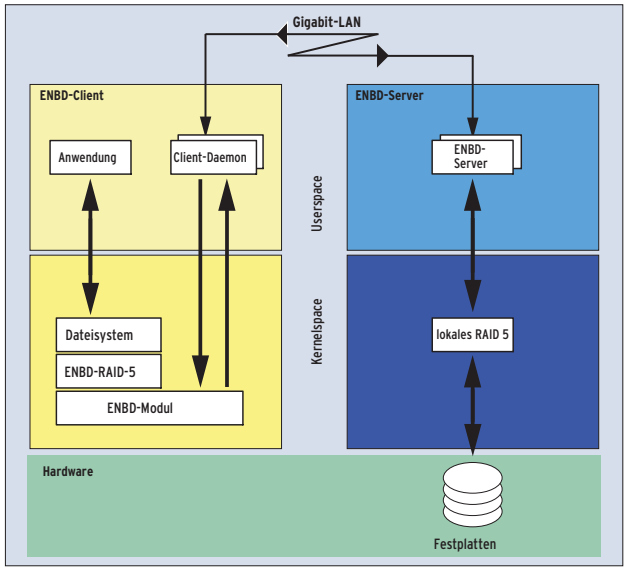

Abbildung 1: Schematische Darstellung der ENBD-Architektur.

Aufbauarbeit

Anschließend folgt das übliche »make; make install«. Mittels »make test« lässt sich die Funktion von ENBD lokal überprüfen. Danach startet der Server mit »enbd-server *Portnummer Device*«. Die

Konfiguration des Testsystems

Herzstück aller vier Rechner (Abbildung 2) ist ein Intel-D865GBF-Mainboard mit einer P4-CPU, getaktet mit 2,4 GHz. Sie verfügen über 1 GByte RAM und ein Gigabit-Ethernet-Interface on Board. Das Frontend verfügt zusätzlich über eine PCI-Gigabit-Karte. Die drei Backends haben je sechs 200-GByte-SATA-Festplatten (Seagate ST3200822AS), von denen fünf über einen Raid-Controller (Highpoint Rocketraid 1820) zu einer Raid-5-Gruppe mit Chunksize 4 KByte verbunden sind. Die verbleibende Platte dient jeweils als Hot Spare.

Alle Rechner laufen unter einem Debian-Linux auf Knoppix-Basis mit Kernel 2.4.26. Sie sind untereinander über einen Cisco-4500-GBit-Switch verbunden. Der Arbeitsspeicher des Frontend wird als Datencache erster Ordnung, der Arbeitsspeicher der drei Backends als Datencache zweiter Ordnung benutzt.

Abbildung 2: Das hier beschriebene Testsystem: Die drei unteren Rechner exportieren die Devices, das obere System mountet sie.

Portnummer ist frei wählbar, statt eines Devicenamens wie »/dev/md0« kann der Admin auch eine gewöhnliche Datei angegeben. Über die zusätzliche Option »-i *Signatur*« erhält das Device eine Kennung, die es dem Client ermöglicht, die importierten Laufwerke seinerseits immer unter denselben Devicenamen anzubieten.

Auf Seite des Clients ist das erwähnte Kernelmodul einzubinden. Das bewerkstelligt »insmod enbd.o«. Der Test mit »cat /proc/nbdinfo« zeigt, ob alles funktioniert. Außerdem empfiehlt es sich, das Modul über die Anweisung

```
echo "merge_requests=512" >/proc/nbdinfo
```

so zu konfigurieren, dass es einzelne Operationen für ein Device zunächst sammelt (im Beispiel sind das bis zu 512 Blöcke) und später gebündelt ausführt. Diese Technik steigert die Performance erheblich.

Nach dem Laden des Kernelmoduls wird der Client gestartet:

```
enbd-client Hostname:port -n Anzahl Instanzen
-b Blocksize dev
```

Dabei bezieht sich »*Hostname*« auf den Namen des exportierenden Rechners, die Anzahl der Serverprozesse ist besonders für Mehrprozessor-Systeme mit Hyper-Threading interessant, bei denen sie der Anzahl der CPUs entsprechen soll. Mehr sind nicht sinnvoll. Die Blockgröße bestimmt den Umfang der Datenpakete. Schließlich ist noch der Name zu spezifizieren, unter dem das importierte Device verfügbar sein soll.

Ein Aufruf von »fdisk« zeigt, ob die konfigurierten Optionen auch richtig angekommen sind:

```
enbd-client Hostname:5000 -n 2 -b 4096↵
/dev/nda
fdisk -l /dev/nda
Disk /dev/nda: 799.9 GB,↵
799990087680 bytes
255 heads, 63 sectors/track, 97259 cylinders
Units = cylinders of 16065 * 512 = 8225280↵
bytes
Disk /dev/nda doesn't contain a valid↵
partition tab.
```

Sicher und schnell?

Nachdem alles aufgebaut und konfiguriert ist, interessiert natürlich besonders die Performance des gesamten Systems. Die Messungen erfolgten mit dem Benchmark Bonnie++ [7]. Die Ergebnisse fassen die beiden Tabellen 1 (Schreibzugriffe) und 2 (Lesezugriffe) zusammen. Der Datendurchsatz wurde jeweils bei verschiedenen Dateigrößen (Spalte 1) ermittelt. Die Spalten zwei und drei enthalten die lokal auf den Backends ermittelten Werte. Auf sie hatte ENBD keinen Einfluss, sie eignen sich deshalb sehr gut als Vergleichsgrößen für die ENBD-Devices.

Durchsatz und CPU-Belastung in den Spalten vier bis sechs gelten für ein einzelnes exportiertes ENBD-Device (also eine Backend-Raid-Gruppe), gemessen auf der Frontend-Seite. Die Werte der letzten beiden Spaltenpaare wurden ebenfalls auf dem Frontend ermittelt, dieses Mal jedoch, nachdem die exportierten Devices dort ihrerseits zu Raid-5-Gruppen verbunden und mit ReiserFS formatiert wurden.

Es zeigt sich besonders beim Schreiben ein deutlicher Vorteil der Entwicklerversion 2.4.32 gegenüber der letzten stabilen Release 2.4.31. Das liegt vor allem daran, dass die neuere Version ankommende Daten puffert und erst dann schreibt, wenn ein Block vollständig übertragen ist. So muss sie – im Gegensatz zu ihrer Vorgängerin – bei teilweise übertragenen Blöcken nicht zuerst die Anschlussstelle ermitteln.

Laptop im Spiegel

Durch die Optimierung weiterer Parameter ließe sich der Durchsatz sicher noch verbessern. Einen praktisch außerordentlich wirkungsvollen Effekt können die vorgestellten Benchmarks aber gar nicht erfassen: Die Fähigkeit von ENBD, nur geänderte Daten tatsächlich zu übertragen. Dazu werden blockweise MD5-Prüfsummen berechnet und beim Zurückschreiben einer vorher eingelesenen Datei verglichen.

So ist es möglich, nur jene Blöcke zu transferieren, die neue Inhalte haben. Dadurch reduziert sich das Volumen der Datenübertragung unter Umständen auf einen Bruchteil. Insbesondere beschleunigt sich die Synchronisation eines Raid-Systems nach temporärem Ausfall einer Komponente enorm.

ENBD ist nicht nur für größere, verteilte Speichersysteme interessant, auch Notebook-Besitzer können davon profitieren. Speziell für deren Bedürfnisse hat einer

Tabelle 1: Schreibzugriffe

Größe der Testdatei [MB]	Bonnie++ auf lokalem Raid 5 [MB/s]	CPU-Last in %	Ein exportiertes ENBD-(2.4.31)-Device [MB/s]	CPU-Last	Ein exportiertes ENBD-(2.4.32)-Device [MB/s]	CPU-Last	Raid 5 über drei ENBD-(2.4.31)-Devices [MB/s]	CPU-Last	Raid 5 über drei ENBD-(2.4.32)-Devices [MB/s]	CPU-Last
2048	58,26	37 %	14,97	8 %	27,89	16 %	21,81	15 %	36,81	32 %
4096	51,31	35 %	12,18	7 %	24,95	15 %	18,93	13 %	30,88	25 %
8192	48,50	34 %	10,65	6 %	22,02	13 %	16,98	11 %	28,29	22 %

Tabelle 2: Lesezugriffe

Größe der Testdatei [MB]	Bonnie++ auf lokalem Raid 5 [MB/s]	CPU-Last	Ein exportiertes ENBD-(2.4.31)-Device [MB/s]	CPU-Last	Ein exportiertes ENBD-(2.4.32)-Device [MB/s]	CPU-Last	Raid 5 über drei ENBD-(2.4.31)-Devices [MB/s]	CPU-Last	Raid 5 über drei ENBD-(2.4.32)-Devices [MB/s]	CPU-Last
2048	117,42	31 %	81,86	22 %	82,00	21 %	90,00	40 %	88,91	41 %
4096	111,34	31 %	73,63	23 %	69,85	21 %	77,21	35 %	79,60	35 %
8192	111,83	32 %	71,14	22 %	68,33	22 %	76,00	34 %	79,56	36 %

der ENBD-Autoren, Peter T. Breuer, das Fast-Raid-Modul (FR 1) entwickelt. Mit FR 1 können Besitzer mobiler Rechner aus einer Datei, Platte oder Partition des Notebooks und einer externen Datenquelle einen Raid-1-Spiegel bilden. Dieser Spiegel synchronisiert sich automatisch, sobald sich der Laptop wieder mit dem LAN verbindet. Eine interne Look-up-Tabelle sorgt für die Beschleunigung dieses Prozesses, indem sie sich die seit der letzten Synchronisation geänderten Blöcke merkt.

Eine Reihe alternativer Techniken versammelt sich zusammen mit ENBD unter dem Stichwort Storage Over IP. Im Kommen ist beispielsweise ISCSI, ein von der IETF entwickelter Standard, um SCSI-Befehle über IP-Netzwerke zu übertragen und damit Speichersysteme anzubinden.

Verwandtes

Unter Linux befindet sich die entsprechende Server-Software jedoch durchweg noch in einem frühen Stadium und ist entweder in der Größe der exportierbaren Devices auf 2 GByte beschränkt oder nicht SMP-fähig. Eine weitere Alternative wäre eventuell DRDB. Allerdings unterstützt DRDB nur Raid 1, was bedeutet, dass man in jedem Fall in doppelt so viele Platten investieren muss, als effektiv nutzbar sind. Außerdem schneidet diese Technik beim Vergleich der Performance mit ENBD 2.4.32 schlechter ab [6].

Fazit: Mit ENBD verfügt Linux über eine Technologie zum Aufbau hochverfügbarer, verteilter Speichersysteme aus Standardkomponenten. Zusätzlich lassen sich damit auch Datenbestände zwischen Notebooks und Desktop-Rechnern oder zentralen Servern auf elegante und effiziente Art synchronisieren. *(jcb)* ■

Die Autoren

Lord Hess und Arne Wiebalck arbeiten am Kirchhoff-Institut für Physik der Universität Heidelberg. Lord Hess beschäftigt sich mit Fehlertoleranz großer Cluster. Arne Wiebalck arbeitet auf Basis von ENBD an verteilten Raid-Systemen.

Infos

[1] MORI (Market & Opinion Research International): [http://www.mori.com/polls/2000/euro-sme.shtml]
[2] ENBD-Homepage von Peter T. Breuer: [http://www.it.uc3m.es/~ptb/nbd/]
[3] NBD als Kerbelbestandteil: [http://sourceforge.net/projects/nbd/]
[4] Das Linux-HA-Projekt: [http://www.linux-ha.org]
[5] ENBD im Heartbeat-Cluster: [http://wellquite.org/redundant/]
[6] ENBD-Quellen: [ftp://oboe.it.uc3m.es/pub/Programs/]
[7] Ausführliche Messergebnisse: [http://www.kip.uni-heidelberg.de/ti/ClusterRAID/linux-magazin/bonnie/ms/]
[8] Andreas Sebald, „Reservespieler“: Linux-Magazin 07/04, S. 60