Hochvefügbarkeitssysteme mit Linux

von Atif Ghaffar
<atif(at)developer.ch>

Über den Autor:

Atif ist ein Chamäleon. Er verändert seine Rollen von Systemadministrator zu Programmierer, zu Lehrer, zu Projektmanager oder was immer in seinem Beruf benötigt wird. Gelegentlich findet man ihn sogar auf der Toilette mit seinem Laptop wie er gerade dabei ist, Dokumentation zu schreiben. Atif glaubt daß er Linux und der Open-Source Gemeinde sehr viel verdankt. Mehr über ihn findet man auf seiner Homepage

Übersetzt ins Deutsche von:
Guido Socher
<guido(at)linuxfocus.org>

Inhalt:

Zusammenfassung:

Während der Entwicklung eines für sein Unternehmen extrem wichtigen Systems wird man sich follgende Fragen stellen:

- Wie wichtig sind die Dienste, die auf diesen Computern laufen für das Unternehmen?
- Wie viele andere Dienste sind von diesem einzelnen Dienst, der auf dem neuen Computer laufen soll abhängig (denk an NIS/NFS/DB/LDAP Server)
- Was passiert, wenn ein Teil der Maschine versagt? (Stromversorgung, Netzwerk, Festplatte ...)
- Was passiert, wenn die Maschine völlig ausfällt?

Wenn ich mir selbst diese Fragen stelle, dann ist die Antwort fast immer : Man wird mich feuern:)

Andererseits, wenn ich mir die Frage stelle "wird das Betriebssystem versagen?" dann ist die Antwort
Nein. Weil ich keine **32 Bit Erweiterung** für einen **16 bit Patch** auf einem **8 bit Betriebssystem** benutze, daß ursprünglich für einen **4 bit Microprozessor** von einer **2 bit Firma** entwickelt wurde, die nicht ein bit (bißchen) Konkurrenz ausstehen kann.
Der Spruch war von einer .signature Datei

Nun zu einer ernsthaften Diskussion

Warum HA?

Ha steht für High Availability, zu Deutsch Hochverfügbarkeit. Ich habe großes Vertrauen in Linux, aber ich habe wenig Vertrauen in die Firmen, die die Hardware entwickeln. Bei einem Server der Tag und Nacht läuft, wird eines Tages die Stromversorgung, die Netzwerkkarte, das Motherboard, u.s.w... versagen. Dadurch wird mein Server zusammenbrechen. Weil dieser Server versagt, werden unter Umständen weite Teile des Firmennetzes versagen. Z.B:

- Ein Service, von dem ich nicht wußte, daß er existierte, funktioniert plötzlich nicht mehr, weil er die Domain billingSys106.company.com nicht mehr finden kann. Hmmm, was könnte der Grund sein. Oh, ja, auf meinem Server läuft der DNS Server, der den Domainnamen auflösen sollte.

- Jemand kann das SAP System nicht benutzen, weil der LDAP Server nicht antwortet. Oh, Moment mal. Habe ich nicht 3 Monate lang gekämpft, um SAP Autentifizierung auf den LDAP Server zu verlegen?

- ... oder niemand kann sich in die Windows Rechner einloggen. Nur eine Unixkiste funktioniert nicht. Warum sollte NT davon beeinflußt werden. Ja, der NT domain controller läuft auf Linux+Samba mit Autentifizierung durch LDAP.

Dasselbe kann natürlich mit einem Windows Server passieren. Das wird jedoch nicht viel Geschrei verursachen, weil dieser Server ohnehin oft nicht funktioniert und sich die Leute daran gewöhnt haben. Ich warne Dich, wenn dann eines Tages mal der Linux Server versagt, dann wird das Management sagen "wie konnten wir das einem Linux Server anvertrauen?"

- In einer der Firmen, für die ich gearbeitet habe, stellte der NFS Server Dateien für den Webserver, den Intranetserver, den Datebankserver und einige andere Applicationen zur Verfügung. Ein Problem mit dem NFS Server würde die Firma praktisch zum Stillstand bringen.
  Natürlich ist es keine gute Wahl NFS für all diese Dinge zu benutzen, aber benuzten wir diese Firma einfach mal als Beispiel. Der NFS Server wurde als HA Server mit Sun's Clusterlösung ausgestattet. Eine extrem teure Sache.

  Laß uns im folgenden dieses Konzept etwas genauer untersuchen

Was ist HA?

HA (High Availability oder Hochverfügbarkeit) ist, wie der Name sagt, ein System, das immer verfügbar ist.
Das kann wichtig sein für Dienste, die das Funktionieren der Firma garantieren: Beispiel:

- intranet web server
- File server

- Mail server
- DNS service

Diese Dienste können im Prinzip aus zwei Gründen versagen.

- Softwarefehler
- Hardwarefehler

Um Hardwarefehlverhalten zu verhindern, trifft man normalerweise etliche Vorsichtsmaßnahmen schon beim Bestellen der Hardware. Redundante Stromversorgung, Raid 5, etc.
Was oft übersehen wird, ist Fehlverhalten der Software. Ob Du es glaubst oder nicht, ich habe Linuxrechner wegen einer fehlerhaften Netzwerkkarte versagen sehen.
Der Chef ist normalerwiese nicht daran interessiert zu wissen, ob das System abgestürzt ist, weil die Stromversorgung versagt hat oder weil die Netzwerkkarte fehlerhaft war.
Was interessiert, ist die Verfügbarkeit des "**Dienstes**" , den ein Rechner zur Verfügung stellt. Natürlich läuft der "**Dienst**" auf einem Rechner und das Umlenken der Anfragen für diesen "**Dienst**" auf einen intakten Rechner ist die Kunst der Hochverfügbarkeit.

Ein Beispiel einer HA Implementierung

In diesem Beispiel entwickeln wir ein Rechnercluster, auf dem der Apache Webserver läuft. Für diese Cluster benutzen wir eine gute Maschine mit starker CPU und viel Speicher. Eine zweite Maschine hat gerade genug CPU Leistung und Speicher, um den Dienst alleine laufen zu lassen.
Der erste Rechner wird der Hauptrechner sein (master) und der zweite der Ersatzrechner (backup). Die Aufgabe des backup Rechners ist es, dem master den Dienst wegzunehmen, sobald er denkt, daß der Master nicht mehr richtig antwortet.

Wie man das macht

Wie greifen die Benutzer auf den Webserver zu? Sie tippen http://intranet/ in ihren Browser und der DNS Server lenkt das auf die IP Adresse 10.0.0.100 (z.B) um.

Wie wäre es, wenn wir einfach im Falle des Versagens eines Rechners den Eintrag im DNS Server so ändern, daß er auf einen Rechner mit einer anderen IP Adresse geht?
Sicher, das ist eine Möglichkeit, aber DNS wird auf der Client-Seite in einem Zwischenspeicher gehalten (DNS cache) und was ist, wenn wir DNS mit HA laufen lassen wollen?
Eine bessere Möglichkeit ist, daß der backup die IP Adresse des masters übernimmt, falls der master versagt. Diese Methode nennt man IP takeover. All Webbrowser werden weiterhin Anfragen an 10.0.0.100 schicken, aber dabei auf den backup zugreifen.

Wie Cluster miteinander reden

Wie weiß der master/backup, daß der jeweils andere versagt hat?
Sie unterhalten sich sowohl über die serielle Schnitstelle als auch über ein Crossover Ethernetkabel. Man benutzt beides, Ethernetkabel und serielles Kabel aus Redundanzgründen. Sie überprüfen ihren Herzschlag. Ja, wie beim Menschen. Wenn das Herz nicht mehr schlägt, ist der Mensch vermutlich tot.
Die beiden Rechner schicken sich in regelmäßigen Abständen kurze Nachrichten zu. Wenn diese

Nachrichten ausfallen (Herzschlag), dann ist der Rechner ausgefallen. Das Programm, das man dazu braucht, nennt sich, rate mal ..., heartbeat (Herzschlag).
heartbeat gibt es unter www.linux-ha.org/download/.

Das Programm zur Übernahme der IP Adresse nennt sich fake und ist in heartbeat enthalten.

Vorbereiten der Cluster Knoten

Wie schon gesagt werden wir zwei Rechner verwenden, einen leistungsstarken und einen weniger leistungsstarken. Beide Maschinen haben zwei Netzwerkkarten und jeweils eine serielle Schnittstelle. Wir brauchen auch ein crossover cat 5 RJ45 Ethernetkabel und ein null modem Kabel (cross over serial cable)

Das erste Ethernet Interface (eth0) benutzen wir auf beiden Maschinen zum Anschluß an das Netzwerk. Das zweite Ethernet Interface (eth1) wird für das private Heartbeat Netz benutzt, über das wir UDP Pakete schicken.

Hier die Adressen für eth0 auf beiden Rechnern:
clustnode1 ip Adresse 10.0.0.1
clustnode2 ip Adresse 10.0.0.2

Nun reservieren wir eine weitere ip Adresse als eigentliche Serviceadresse. Diese ist 10.0.0.100. Im Augenblick brauchen wir diese Adresse noch keiner von beiden Machine zuzuweisen.

Als nächstes konfigurieren wir die zweite Netzwerkkarte und geben ihnen Adressen aus einem Bereich, der noch frei ist. Z.B:
clustnode1 ip Adresse 192.168.1.1
clustnode2 ip Adresse 192.168.1.2

Jetzt können wir die seriellen Schnitstellen verbinden und testen, daß der hearbeat funktioniert. (Es ist einfacher, wenn man den gleichen seriellen Port auf beiden Rechnern benutzt.) Näheres unter http://www.linux-ha.org/download/GettingStarted.html

Installation von heartbeat

Die Installation ist ganz einfach. heartbeat ist als rpm und tar.gz File verfügbar. Wenn Du Probleme mit der Installation hast, dann solltest Du besser nicht die Verantwortung für ein HA System übernehmen (... oder es wird vielleicht ein NA [not available] System). Eine gute Anleitung findet sich unter "Getting Started with Linux-HA" auf der linux-ha.org Seite.

Konfiguration des Clusters

Konfiguration von hearbeat

Die Konfigurationsdateien sind in /etc/ha.d
Editiere /etc/ha.d/authkeys und schreibe folgendes:

```
#/etc/ha.d/authkeys
```

```
auth 1
1 crc
#end /etc/ha.d/authkeys
```

Man kann später auf md5 oder sha umstellen. Im Moment ist der Autentifizierungsmechanismus mit 1 gut gewählt.

edit /etc/ha.d/ha.cf

```
debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility     local0
deadtime 10
serial  /dev/ttyS3  #change this to appropriate port and remove this comment
udp     eth1      #remove this line if you are not using a second network card.
node    clustnode1
node    clustnode2
```

editiere die Datei /etc/ha.d/haresources

```
#masternode ip-address service-name
clustnode1 10.0.0.100 httpd
```

Das legt fest, daß clustnode1 der master ist. Wenn er ausfällt übernimmt backup und wenn er wieder funktioniert, wird er den Dienst wieder erhalten.
Der zweite Eintrag definiert die IP Adresse die übernommen werden soll und der dritte Eintrag ist der dienst. Wenn clustnode2 übernimmt, wird versucht, den Befehl

```
/etc/ha.d/httpd start
```

auszuführen und wenn das versagt, wird

```
/etc/rc.d/init.d/httpd start
```

ausgeführt.
Dasselbe passiert für den stop Befehl, wenn der Dienst wieder zurückgegeben wird.

```
/etc/ha.d/httpd stop
```

und falls das nicht geht:

```
/etc/rc.d/init.d/httpd stop
```

Wenn man fertig mit der Konfiguration von clustnode1 ist, kann man die Dateien nach clustnode2 kopieren.
In dem Verzeichnis /etc/ha.d/rc.d findet sich das Script ip-request. Dieses übernimmt die Zuweisung der Service IP Adresse.
Nun starte einfach /etc/rc.d/init.d/heartbeat auf beiden Rechnern. Es empfiehlt sich außderdem zu Testzwecken verschiedene Webseiten auf beiden Servern zu haben. Dadurch kann man die Rechner leicht unterscheiden.
Wenn Du mit der Konfiguration auf clustnode1 fertig bist, kannst Du die Dateien nach node2 kopieren.

Es ist außerdem sicherzustellen, daß der Service httpd nicht automatisch beim Booten gestartet wird. Dazu enfernt man die Links in den etc/rcN Verzeichnissen oder man Verschiebt die Datei httpd (oder apache. Das ist unterschiedlich je nach Distribution) von /etc/rc.d/init.d/ nach /etc/ha.d/rc.d/. Wenn alles funktioniert hat, dann hat clustnode1 die Adresse 10.0.0.100 und beantwortet http Anfragen.
Zum Test kann man jetzt clustnode1 mit shutdown anhalten und innerhalb von 10 Sekunden sollte

clustnode2 übernehmen.
Der maximale Betriebsausfall wird daher 10 Sekunden betragen.

Wie sieht es mit der Integrität von Daten aus?

Wenn der Service httpd von clusternode1 nach clusternode2 geht, dann sieht man dort nicht dieselben Daten. Alle Dateien fehlen und die Daten der CGI-Bins.

Zwei Antworten:
1) Man sollte niemals von CGI's in Dateien schreiben. Stattdessen benutzt man eine Netzwerkdatenbank. MySQL ist sehr gut.
2) Man kann beide Clusternodes an ein zentrales externes SCSI Plattensystem anschließen. Hierzu muß man sicherstellen, daß nur ein Rechner immer mit dem SCSI Speichersystem redet und daß beide Rechner unterschiedliche SCSI IDs haben (z.B 6 und 7)
Bei Adaptec 2940 SCSI Karten kann man zum Beispiel die SCSI ID des Hostadapters ändern. Billigere Karten lassen das nicht zu.
Einige Raid Controller werden als cluster-aware Controller verkauft. Hier ist vorher zu prüfen, ob diese sich einstellen lassen, ohne daß man zuerst das Microsoft cluster kit kaufen muß.
Ich hatte zwei NetRaid Controller von HP und diese unterstützen definitiv nicht Linux. Solches Zeug sollte man vermeiden.

Eine weitere Möglichkeit ist Fibrechannel Karten, Fibrechannel hub und Fibrechannel Storage. Diese Lösung ist gut, aber erheblich teurer als zwei SCSI Karten.
Eine sehr gute Lösung ist GFS (Global File System, siehe unten) über Fibrechannel. Damit kann man auf die Dateien zugreifen, als ob sie auf localen Platten lägen.

Wir benutzen GFS in einer Produktionsanlage mit 8 Rechnern, wovon 2 in einer wie oben beschriebenen HA Konfiguration laufen.

Wie sieht es mit active/active Clustern aus

Man kann sehr einfach einen Active/Active Server bauen, wenn man ein gutes Plattenspeichersystem hat. Fibrechannel und GFS sind hier eine gute Wahl.
Wenn du mit NFS vertraut bist, kann man auch das benutzen, aber ich würde eher davon abraten.

Jedenfalls kann man serviceA dem clustnode1 zuweisen und serviceB clustnode2. In meine haresource file steht z.B:

```
clustnode2 172.23.2.13 mysql
clustnode1 172.23.2.14 ldap
clustnode2 172.23.2.15 cyrus
```

Ich benutze GFS und deshalb gibt es kein Problem mit gleichzeitigem Zugriff auf die Daten.
Hier ist clustnode2 der master für mysql + cyrus und clustnode1 der master für ldap.
Wenn clustnode2 versagt, übernimmt clustnode1 alle Dienste.

Resourcen

Linux-HA.org
: Die Homepage von Linux HA

kimberlite clustering technology
: Ein Kimberlite Cluster unterstützt 2 Server mit shared SCSI oder Fibrechannel als Plattenspeichersystem in einer active-active failover Umgebung. Kimberlite ist entwickelt worden für höchste Dateninegrität und ist sehr robust. Es ist geeignet für HA unter Linux, ohne daß man die Applikationen ändern muß.

ultra monkey
: Ultra Monkey ist ein Projekt, daß load balanced highly available services in einem LAN entwickelt. Es wird Open Source und Linux verwendet. Im Moment arbeitet man an einer skalierbaren HA Web-Farm. Die Technologie kann problemlos auf E-mail und andere Sachen wie z.B FTP übertragen werden.

Linux Virtual Server
: Der Linux Virtual Server ist ein hochverfügbarer skalierbarer Server, gebaut aus einem Cluster von realen Servern. Mit einem load balancer auf Linuxbasis. Die Architektur des Clusters ist für den Benutzer völlig transparent. Der Endbenutzer sieht nur einen einzigen Server.

4U cluster / 4U SAN
: 4U cluster und 4U SAN ist ein HA cluster und eine SAN Implementation von unserer Firma, 4Unet.
Wenn du ein ISP, ein Netzbetreiber, eine Telekomunikationsfirma oder einfach eine Firma bist, die High Availability braucht, dann ist 4Unet der richtige Ort für Fragen.
Beachte: 4Unet setzt Lösungen um. Wir verkaufen keine Cluster oder SANs, wir implementieren sie für unsere Kunden. All Technologien, die von uns verwendet werden, sind Open Source.

Global File System
: Das Global File System (GFS) ist ein shared disk cluster filesystem für Linux. GFS unterstützt journaling und recovery von fehlerhafen clients. GFS cluster teilen sich physikalisch ein Plattenspeichersystem unter Verwendung von Fibre Channel oder shared SCSI. Das Filesystem erscheint auf jedem Knoten wie ein lokales Filesystem und GFS synchronisiert den Dateizugriff im ganzen cluster. GFS ist voll symmetrisch. Alle Knoten sind gleich. Es gibt keinen einzelnen Server oder irgendeinen Flaschenhals. GFS hat Lese/Schreib-caches und volle UNIXfilesystem Semanik.

Talkback für diesen Artikel

Jeder Artikel hat seine eigene Seite für Kommentare und Rückmeldungen. Auf dieser Seite kann jeder eigene Kommentare abgeben und die Kommentare anderer Leser sehen:

Talkback Seite

Der LinuxFocus Redaktion schreiben

LinuxFocus.org
Einen Fehler melden oder einen Kommentar an LinuxFocus schicken

Autoren und Übersetzer:
en --> -- : Atif Ghaffar <atif(at)developer.ch>
en --> de: Guido Socher <guido(at)linuxfocus.org>

2002-02-24, generated by lfparser version 2.25