tutorials.de (*http://www.tutorials.de/forum/*)
- **Linux - Tutorials** (*http://www.tutorials.de/forum/linux-tutorials/*)
- - **Hochverfügbare Dienste mit CARP im Linuxcluster**
(*http://www.tutorials.de/forum/linux-tutorials/279181-hochverfuegbare-dienste-mit-carp-im-linuxcluster.html*)

Sinac 26.06.07 14:05

Hochverfügbare Dienste mit CARP im Linuxcluster

1. Einführung
Um Dienste in einem Netzwerk hochverfügbar und ohne Ausfallzeit anbieten zu können genügt es oft nicht einfach ein System redundant aufzusetzen oder einen StandBy Server bereitzuhalten. Das Ersatzsystem muss trotzdem ins Netzwerk gebracht und entsprechend konfiguriert werden um z.B. die gleiche IP Adresse wie der ausgefallene Server zu haben. Auch sind auf Seiten der Clients in diesem Fall oft ARP Nachrichten nötig um die neue MAC Adresse zu publizieren. Dies ist besonders kritisch im Falle von Routern und anderen Gateways wo ein Ausfall unter Umständen die gesamte Standortanbindung in Gefahr bringt.

2. Warum CARP?
Für solche Ansprüche gibt es zur Zeit 3 Protokolle:

HSRP (Hot Standby Router Protocol)
VRRP (Virtual Router Redundancy Protocol)
CARP (Common Address Redundancy Protocol)

Erstere sind properitäre Protokolle die patentrechtlich unter andrem CISCO gehören. CARP dagegen ist frei und wurde vom OpenBSD Team entwickelt. Es ist nativ unter OpenBSD, FreeBSD und NetBSD einsetzbar und durch einen Userlandport namens UCARP auch für Linux 2.4 und 2.6 verfügbar. CARP bietet gegenüber den Konkurrenten noch weitere Vorteile wie z.B. Protokollunabhängigkeit und verschlüsselte Kommunikation.

3. Funktionsweise
CARP stellt eine virtuelle IP und MAC Adresse zur Verfügung über welche die Clients mit dem CARP Cluster kommunizieren. Diese ist auf einen Master-Node gerichtet, also den momentan aktiven Server. Fällt Dieser aus, so wird die Kommunikation innerhalb weniger Augenblicke auf einen des Slaves übertragen indem Dieser die virtuelle IP und MAC Adresse annimmt. Natürlich ist es nötig, dass beide System gleich aufgesetzt sind den gleichen Datenbestand mit den entsprechnden Diensten halten. Um z.B. auf Firewalls einen unterbrechungsfreien Cluster zu realisieren müsen noch andere Dinge beachtet werden, z.B. müssen die Server ihre Tabellen mit Informationen über Stateful-Connections austauschen damit keine Verbindung abbricht.

4. Umgebung
Für dieses Tutorial wurde Debian GNU/Linux 4.0 Systeme in virtuellen Maschinen unter VMWare ESX genutzt. Um wirklich einen hochverfügbaren Cluster aufzubauen würde man in der Praxis natürlich physikalisch getrennte Systeme nutzen die optimalerweise in räumlich getrennten Brandabschnitten stehen.

Als Dienst zum Test der Hochverfügbarkeit wird ein einfacher Apache Webserver aufgesetzt.

5. Installation
Unter Debian gestalltet sich die Installation via APT denkbar einfach. Als root angemeldet langt folgender Befehl:

Code:

```
apt-get install ucarp
```

Allerdings sollte das Ganze auch ohne APT kein Problem sein:

Code:

```
wget http://download.pureftpd.org/pub/ucarp/ucarp-1.2.tar.bz2
tar jxf ucarp-1.2.tar.bz2
cd ucarp-1.2
./configure
make
make install
```

Und wo wir schon dabei sind installieren wir auch gleich den Webserver:

Code:

```
apt-get install apache2
```

6. Einrichtung von UCARP

Der Hauptteil der Einrichtung ist sowohl auf dem Master als auch auf den Slaves gleich, alle Unterschiede werden explizit erwähnt.

UCARP bietet die Möglichkeit beim Aktivieren und Deaktivieren eines Nodes als Master ein Skript auszuführen um z.B. einen Dienst zu starten, die Firewall zu konfigurieren oder den ARP-Cache anzupassen. Wir nutzen die Skripte um das virtuelle Interface des Clusters zu laden und ggf. wieder zu deaktivieren:

Code:

```
mkdir /etc/ucarp
cd /etc/ucarp
touch ./ucarp-up.sh
touch ./ucarp-down.sh
```

Vorher erstellen wir uns eine .conf-Datei für UCARP in der wir die wichtigsten Einstellungen für den Start von UCARP festlegen und auch die Parameter für das virtuelle Interface einstellen. /etc/ucarp/ucarp.conf:

Code:

```
# Configuration file for UCARP
# Physical Network interface
UCARP_INTERFACE=eth0

# Virtual Network Interface
UCARP_IF_ALIAS=eth0:0

# Real IP Address
UCARP_SRCIP=172.16.0.11

# Hosts CARP ID
UCARP_VHID=1

# Advertisement Interval
UCARP_ADVBASE=1

# Password for encryption
UCARP_PASS=geheim

# Virual IP Address
UCARP_ADDR=172.16.0.1

# Netmask
UCARP_MASK=255.255.0.0
```

```
# Upscript
UCARP_UPSCRIPT=/etc/ucarp/ucarp-up.sh

# Downscript
UCARP_DOWNSCRIPT=/etc/ucarp/ucarp-down.sh
```

Die Punkte im Einzelnen sollten relativ klar sein:

Code:

```
UCARP_INTERFACE=eth0
```

Das Netzwerkinterface auf dem UCARP laufen soll, also das LAN Interface.

Code:

```
UCARP_IF_ALIAS=eth0:0
```

Das virtuelle Interface auf dem später die virtuelle IP Adresse des Clusters liegt.

Code:

```
UCARP_SRCIP=172.16.0.11
```

Die echte IP Adresse der oben angegeben Netzwerkkarte. Hier muss bei jeden anderen Node natürlich die entsprechende IP Adresse eingetragen werden.

Code:

```
UCARP_VHID=1
```

Die ID des virtuellen Servers für den diese Konfguration gilt.

Code:

```
UCARP_ADVBASE=1
```

Intervall zum Abgleich der Nodes in Sekunden.

Code:

```
UCARP_PASS=geheim
```

Das Password für die HMAC Verschlüsselung der Kommunikation. Diese erfolgt per SHA1.

Code:

```
UCARP_ADDR=172.16.0.1
```

Die virtuelle IP Adresse unter der unser Cluster später erreichbar sein soll.

Code:

```
UCARP_MASK=255.255.0.0
```

Die Subnetzmaske unseres LAN Segments.

Code:

```
UCARP_UPSCRIPT=/etc/ucarp/ucarp-up.sh
UCARP_DOWNSCRIPT=/etc/ucarp/ucarp-down.sh
```

Der Pfad zu den Skripten die beim Aktivieren und Deaktivieren eines Nodes ausgeführt werden.

Nun bearbeiten wir die zuvor erstellten Up- und Down-Skripte um bei Aktivierung das virtuelle Interface zu erstellen und umgekehrt bei Abgabe des Masterstatus wieder zu löschen.

/etc/ucarp/ucarp-up.sh

Code:

```
#!/bin/bash
source /etc/ucarp/ucarp.conf
ifconfig $UCARP_IF_ALIAS $UCARP_ADDR netmask $UCARP_NETMASK
```

/etc/ucarp/ucarp-down.sh

Code:

```
#!/bin/bash
source /etc/ucarp/ucarp.conf
ifconfig $UCARP_IF_ALIAS down
```

Die ucarp-up.sh wird wie erwähnt beim Aktivieren eines Nodes ausgeführt. Sie nimmt sich die Parameter für den Alias, also die virtuelle Schnittstelle, die virtuelle Adresse und die Netmask aus der ucarp.conf und erstellt das Interface.

Gibt ein Node seinen Masterstatus wieder ab weil z.B. der erste Server wieder erreichbar wird, wird ucarp-down.sh ausgeführt und löscht das Interface wieder.

Als Nächstes erstellen wir uns das Startskript für UCARP. Alternativ kann dafür auch ein Init-Skript erstellt werden. Das Skript legen wir als start.sh unter /etc/ucarp ab:

Code:

```
source /etc/ucarp/ucarp.conf

ucarp /
  --interface=$UCARP_INTERFACE /
  --srcip=$UCARP_SRCIP /
  --vhid=$UCARP_VHID /
  --pass=$UCARP_PASS /
  --advbase=$UCARP_ADVBASE /
  --preempt /
  --addr=$UCARP_ADDR /
  --daemonize /
  --upscript=$UCARP_UPSCRIPT /
  --downscript=$UCARP_DOWNSCRIPT
```

Die erste Zeile stellt uns die Angaben aus der ucarp.conf Datei zur Verfügung, die im Aufruf übergebenen Parameter sollten daher ja klar sein. Die Option --daemonize lässt UCARP als Daemon im Hintergrund laufen. Wichtig ist die Option --preemt, welche nur auf dem Master

Server mit übergeben wird. Diese aktiviert einen Node und wird daher bei den Slaves nicht übergeben.

Als letzten Schritt machen wir unsere Skripte noch als Root ausführbar:

Code:

```
chmod 0700 /etc/ucarp/*.sh
```

Um das Ganze nun zu testen konfigurieren wir uns einen Master und eine Slaveserver wie oben beschrieben, ich habe dazu in der Testumgebung einfach die fertige MasterVM geclont und folgende Änderungen vorgenommen:
- IP-Adresse
- Hostname
- UCARP_SRCIP in /etc/ucarp/ucarp.conf

Zusätzlich habe ich in der Apachedatei /var/www/apache2-default/index.html einen Vermerk gemacht auf welchem Server ich mich befinden um das Failover zu testen.

7. Start und Funktionstests mit Apache

Um die Funktion von CARP zu testen starten wir nun auf dem Master und auf dem Slaveserver UCARP über das Skript /etc/ucarp/start.sh . Da wie den Parameter --daemonize angegeben haben erfolgt keine Ausgabe und UCARP läuft als Daemon im Hintergrund. Über ifconfig sollten wir auf dem Master bereits das virtuelle Interface eth0:0 mit der virtuellen IP Adresse sehen:

Code:

```
eth0      Protokoll:Ethernet  Hardware Adresse 00:50:56:82:35:2C
          inet Adresse:172.16.0.11  Bcast:172.16.255.255  Maske:255.255.0.0
          inet6 Adresse: fe80::250:56ff:fe82:352c/64
Gültigkeitsbereich:Verbindung
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:402618 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9019 errors:0 dropped:0 overruns:0 carrier:0
          Kollisionen:0 Sendewarteschlangenlänge:1000
          RX bytes:34064624 (32.4 MiB)  TX bytes:623570 (608.9 KiB)
          Interrupt:177 Basisadresse:0x1400

eth0:0    Protokoll:Ethernet  Hardware Adresse 00:50:56:82:35:2C
          inet Adresse:172.16.0.1  Bcast:172.20.255.255  Maske:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:177 Basisadresse:0x1400
```

Auf dem Backupserver sehen wir unter ifconfig natürlich nichts, da das Interface erst bei Aktivierung des Nodes erstellt wird. Den Status von Nodes schreibt UCARP in die /var/log/Syslog. Auf dem Masterserver:

Code:

```
Jun 27 03:49:17 vmdebian01 ucarp[2327]: [WARNING] Switching to state: MASTER
Jun 27 03:49:17 vmdebian01 ucarp[2327]: [WARNING] Spawning
[/etc/ucarp/ucarp-up.sh eth0]
```

Auf dem Slaveserver:

Code:

```
Jun 27 03:50:24 vmdebian02 ucarp[3802]: [WARNING] Switching to state: BACKUP
Jun 27 03:50:24 vmdebian02 ucarp[3802]: [WARNING] Spawning
[/etc/ucarp/ucarp-down.sh eth0]
```

Rufe ich nun den Webserver über die virtuelle IP Adresse auf werde ich sofort auf den aktiven Masterserver geleitet. Nachdem ich diesen einfach vom Netz gezogen habe, wechselt der Slaveserver innerhalb weniger Augenblicke in den aktiven Zustand, lädt das Interface, welches sofort unter der gleichen MAC und IP Adresse zu erreichen ist und übernimmt die Aufgabe des

Masters. Ein Refresh des Browsers zeigt mir nun an dass ich auf dem 2. Server gelandet bin. Das Syslog vermerkt Dies mit der gleichen Meldung wie zuvor der Masterserver. Eine Reaktivierung des ersten Server kehrt das Ganze wieder um und bringt den zweiten Server wieder in den Slavestatus.

8. Ausblick

Natürlich war das nur eine Minimalkonfiguration von CARP selber. Um einen vernünftigen Cluster zu betreiben muss in den meisten Fällen zusätzlich die Anwendungsschicht synchronisiert werden, sei es nun ein Webserver auf dem Sessioninformationen abgeglichen werden müssen oder eine Firewall die Zustandstabellen repliziert.

9. Quellenangaben

http://www.ucarp.org/project/ucarp
http://de.wikipedia.org/wiki/Common_...dancy_Protocol
http://de.wikipedia.org/wiki/HSRP
http://de.wikipedia.org/wiki/VRRP
http://www.pweb.cz/a/37/ucarp-failov...r-v-praxi.html
http://gentoo-wiki.com/HOWTO_Setup_I...ver_with_UCARP

Bei Fragen oder Anregungen einfach melden!

Greetz...
Sinac