Linux Clustering mit Failover

Mark Akermann,
November 2001

1 Was versteht man unter Clustering?

Clustering bezeichnet die Integration mehrerer Computersysteme zu einem Rechnerverbund, der von außen betrachtet wie ein einzelner Rechner erscheint. Das Clustering verfolgt, je nach Einsatzgebiet, mehrere Ziele:

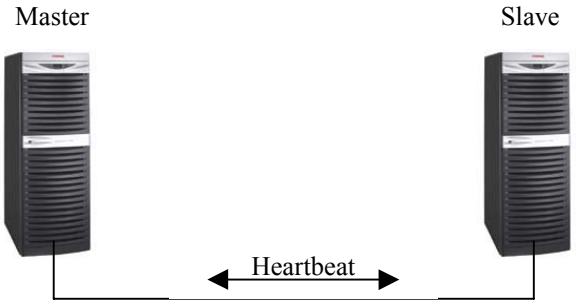
- **Rechenleistung:** Die von den verschiedenen Prozessoren in einem Rechnerverbund aufgebrachte Leistung soll zur gemeinschaftlichen Lösung eines Problems zusammengefasst werden. Das einzelne Problem ist derart umfangreich, dass die Lösung auf einem einzelnen Rechner zu lange dauern würde.

- **Lastverteilung:** Es ist eine grosse Zahl von kleineren Problemen zu lösen, deren Gesamtheit ein einzelnes System überfordern würde. Ein Rechnerverbund stellt sich den Klienten gegenüber als ein einziges System dar, in welchem Teil des Verbundes die Anfrage des Klienten behandelt, wird ist unwesentlich.

-**Verfügbarkeit:** Eigentlich ist ein einzelnes System durchaus in der Lage, die gesamte Last aufzunehmen, doch soll durch ein bereitstehendes und automatisch aktiviertes System die Verfügbarkeit erhöht werden.

Bei Failover handelt es sich um eine Software zur Bildung von Hochverfügbarkeitsclustern.
In der Fachliteratur wird in diesem Zusammenhang oft von ein High-Availability System(HA) gesprochen. Hochverfügbarkeitscluster sind technisch von allen Clusterarten am einfachsten zu realisieren . Ziel ist es, z.B. einen Server ausfallsicher zu machen. Im Gegensatz zu reinen Hardwarelösungen, wie z.B. redundanter Hardware, dient bei einer Softwarelösung ein weiterer Rechner im Falle des Ausfalls als Ersatz für den anderen. Dabei ist es nebensächlich, ob dieses Backup-System ein eigenständiger Computer ist, der mit identischer Konfiguration ausgestattet ist wie der andere Rechner, oder ob ein oder mehrere Systeme die Ersatzfunktion ‚nebenbei' übernehmen.
Grundsätzlich lassen sich HA-Cluster auch als Loadbalancer einrichten, dies bedeutet aber einen höheren Hardwareaufwand.

Nicht alle Anwendungen lassen sich in einem Hochverfügbarkeitscluster realisieren. Entscheidend ist dabei, inwiefern die Anwendungen von einem Sitzungsstatus abhängen. Sitzungslose Dienste, wie z.B. HTTP oder DNS, eignen sich dabei am meisten[1].



[1] Weitere Informationen zu dieser Thematik befinden sich im Linuxclustering Paper von Dr. Andreas Müller

Die einfachste Art eines HA-Clusters besteht aus zwei Rechnern, die eine gemeinsame IP-Adresse verwenden und sich über ein sog. Heartbeat Protokoll über die gegenseitige Verfügbarkeit austauschen. Die geteilte IP-Adresse befindet sich dabei zunächst bei einem der beiden Rechner und wechselt bei dessen Ausfall zum anderen über. Durch die Aussendung von sog. gratuitous ARP-Paketen wird den anderen Teilnehmern im Netzwerk die neue Zuordnung der IP-Adresse zur MAC-Adresse des Backup-Systems mitgeteilt. Der gesammte Vorgang des IP-Adressen Tauschs wird ferner als IP-Failover bezeichnet.

Weitergehende HA-Lösungen enthalten Mechanismen zur Überprüfung von Prozessen und Ressourcen. So ist z.B. ein Webserver, dessen httpd-Dienst abgestürzt ist , genausowenig verfügbar, wie ein Server bei dem der Strom ausgefallen ist. Durch spezielle Programme kann der Zustand eines Dienstes, z.B. durch Abfragen überwacht werden . Im Falle eines Fehlers übernimmt ein anderer Rechner im Cluster jene Funktion.
Dasselbe kann auch für Ressourcen, wie z.B. Speicherkapazität, durchgeführt werden, wenn diese einen Grenzwert unter- bzw. überschreiten.

Je nach Komplexität des Aufbaus und der Software der Cluster Lösung können die Ressourcen auch von mehreren Rechnern geteilt werden. Dabei werden einzelne Rechner in Resourcengruppen eingeteilt.

2 Linux HA-Clustering Projekte im Überblick

Für Linux gibt es derzeit mehrere Cluster Lösungen. Viele der Einzelprojekte haben sich zum Linux-HA Projekt zusammengeschlossen. Übriggeblieben sind vier eigenständige Projekte.
Alle der unten aufgeführten Programme sind als Open-Source Software verfügbar. Daneben gibt es inzwischen auch kommerzielle Lösungen.

- Piranha / LVS

Piranha ist der Oberbegriff der HA-Aktivitäten von Redhat. Es handelt sich dabei im Grundegenommen um 2 seperate Lösungen: Einen loadbalancing Cluster auf Basis des Linux Virtual Server Projekts (LVS) und einen als Failover Service (FOS) bezeichneten HA-Cluster. Redhat bietet Piranha in einer verwaisten Version zum Download an. Die Weiterentwicklung gibt es nur mit dem Produkt „High Availability Server“ von Redhat für 2538 Euro zu kaufen.

- Heartbeat

Heartbeat ist das Ergebnis der Arbeit des Linux HA Projektes. Es entwickelte sich aus dem Zusammenschluß vieler Einzelprojekte. Der Heartbeat kann über ein oder mehrere Netzwerkkarten, sowie serielle Schnittstellen erfolgen. Derzeit unterstützt Heartbeat nur IP-Adressübername , sowie das starten ausgefallener Dienste auf dem Backup-Rechner.
Insgesamt macht das Projekt einen unausgereiften Eindruck. Zudem ist die Dokumentation mit 2 kurzen Texten viel zu wenig.

- Failsafe

Failsafe ist eine hochproffessionelle Clustering Lösung für Linux und IRIX, die bis zu 16 Rechner in einem Verbund verwalten kann. Es ist differenziert konfigurierbar, bietet Ressourcenüberwachung, Serverüberwachung und diverse Monitoringfunktionen. Zudem ist Failsafe sowohl über eine Webschnittstelle als auch ein graphisches Programm administrierbar. Die Dokumentation ist sehr ausführlich gehalten. Zusammenfassend kann man sagen, dass Failsafe keine Wünsche offen läßt, sich aber Aufgrund des hohen Einarbeitungs- und Planungsaufwands nur für große Projekte eignet.
Failsafe wurde von Silicon Graphics Inc. (SGI) für ihr Betriebsystem IRIX entwickelt. Im Rahmen der Open Source Initiative von SGI wurde es später auf Linux portiert.

- Failover

Failover wurde von Dr. Andreas Müller aufgrund der fehlenden HA Lösungen für Linux Proxy Server entwickelt. Es eignet sich darüber hinaus auch zum Clustern anderer Dienste. Es zeichnet sich durch seine einfache Installation und seine Plattformunabhängigkeit aus. Zudem bietet es weitreichende Diagnose- und Monitormöglichkeiten. Weitere Informationen folgen im nächsten Kapitel.

3 Eigenschaften von Failover

Failover eignet sich für Cluster aus 2-3 Rechnern. Es beherrscht IP-Failover und verwendet ebenfalls einen Heartbeat. Der Heartbeat läßt sich mit zusätzlicher Konfigurationsarbeit auch auf beliebig viele Netzwerkkarten verteilen. Zudem kann man bei Verwendung eines Round-Robin fähigen DNS-Servers und einer symmetrischen Konfiguration (2 geteilte IP-Adressen) ein einfaches Loadbalancing erzielen. Jedoch ist der Aufwand für beide dieser Lösungen mit der Einarbeitung in die Skriptsprache TCL verbunden.

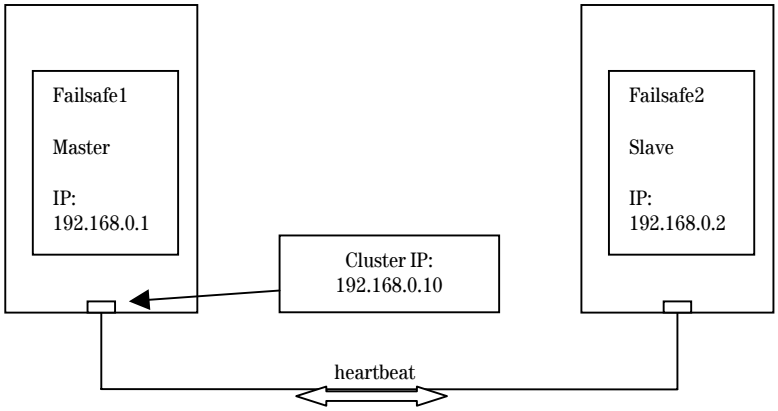
Bei einer Standardinstallation werden 2 Rechner jeweils als Master und Slave eingerichtet. Es gibt hier nur eine geteilte IP-Adresse und der Heartbeat läuft bei jedem Rechner über eine Netzwerkkarte. Dabei muß beachtet werden, das der Heartbeat zusätzliche Last auf dieser Schnittstelle erzeugt. Bei der Berechnung der Netzwerklast für den ausfallsicheren Server muß dieser also berücksichtigt werden.
Desweiteren überwacht und startet Failover nicht die Dienste auf den Systemen auf denen es eingesetzt wird. Es gab diese Funktion ab der Version 0.4.0 der Failover Distribution in Form eines zusätzlichen Programmes mit dem Namen Health-checker, aufgrund einiger Änderungen ist es aber derzeit nicht mehr enthalten. Health-checker wird zur Zeit nicht weiterentwickelt und es ist ungewiss ob es jemals an die neue Version von Failover angepasst wird. Resourcenüberwachung sollte also, wenn benötigt, durch ein weiteres Programm erledigt werden.

Failover zeichnet sich vor allem durch seine einfache Installation und Einrichtung aus. In den folgenden Kapiteln wird darauf näher eingegangen. Ausschlaggebend ist jedoch, dass Failover sehr rasch einsetzbar ist. Im Gegensatz dazu benötigen alle anderen HA Lösungen für Linux viel mehr Planungs- und Einrichtungsaufwand.
Zudem ist Failover nicht ausschließlich auf Linux begrenzt. Es läßt sich auf alle Unix-Derivate portieren. Derzeit gibt es vorkompilierte Distributionen für Linux und Solaris. Der Einsatz von Failover mit FreeBSD ist zudem dokumentiert. Die Interoperabilität der Failovervarianten zwischen den verschiedenen Unix-Systemen ist kein Problem. Es wäre also z.B. ein Linuxrechner als Slave für einen Solarisserver denkbar. Tatsächlich gibt es schon einige solcher Cluster die einwandfrei funktionieren.

Die Dokumentation der Clustering Lösung ist recht kurz gehalten. Damit bietet es aber mehr Dokumentaton als andere Vergleichbare Projekte. Ferner ist es im Vergleich zu den anderen Lösungen am meisten ausgereift. Einem Vergleich mit Failsafe von SGI hält es in diesem Zusammenhang allerdings nicht stand. Dafür ist der Planungsaufwand von Failsafe aber um einiges höher.
Desweiteren überzeugt Failover durch seine Möglichkeiten der Überwachung und Fehlerdiagnose. Im Paket sind die Programme failmon und faildebug enthalten. Failmon ermöglicht die Überwachung des Zustands von Failover auf dem eigenen sowie auf entfernten Rechnern. Faildebug dient zur Steuerung der Logging-Funktionen.
Mit etwas Aufwand kann man den Zustand des Clusters auch per SNMP Traps bestimmen. Die Dokumentaton gibt hierzu eine kurze Hilfestellung. Und als weitere Überwachungsmöglichkeit bietet das Paket eine HTML-Seite mit den aktuellen Zustandsdaten, die sich jede Minute erneuert. Diese kann z.B. mit einem Webserver abrufbar gemacht werden.

4 Grundlegende Konzepte

Failover unterscheidet zwischen Master und Slave Rechnern. Der Master Rechner übernimmt während des Normalbetriebs die Funktion des Clusters (z.B. als Proxyserver), während ein Slaverechner diese Funktion im Falle des Ausfalls des Masters übernimmt. Dabei lassen sich mehrere Master und Slaves definieren, wozu allerdings Kenntnisse in TCL nötig sind. Ein Rechner kann gegenüber anderen also auch mehrere Rollen übernehmen.

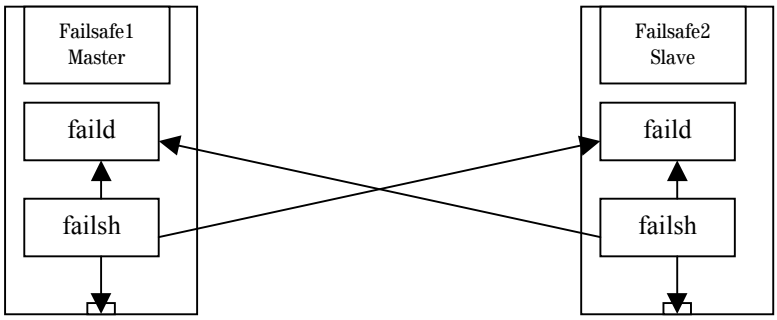


Um den Zustand des Clusters festzustellen verwendet Failover einen Zustandsdämon namens `faild`. `Faild` überwacht verschiedene Dienste, die während des Bootvorgangs gestartet werden. Einer dieser Dienste ist z.B. die IP-Adresse, die von den Cluster Rechnern geteilt wird. Den Zustand verwaltet und erneuert `faild` und hält ihn für Abfragen bereit. Den aktuellen Zustand kann man z.B. mit dem Programm `failstat` abfragen.

Grundsätzlich unterscheidet `faild` dabei 3 verschiedene Zustände: `Up`, `Recover` und `Fail`. Der Zustand `Up` ist immer dann vorhanden, wenn der dazugehörige Dienst seine Funktion voll erfüllt. Unter normalen Bedingungen befindet sich ein Dienst auf einem Masterrechner in diesem Zustand. Ein Dienst im Zustand `Recover` ist derzeitig inaktiv, kann aber jederzeit gestartet werden. Dies ist z.B. bei Diensten auf den Slaverechnern im Normalbetrieb der Fall. Die zweite IP-Adresse schlummert somit auf dem Slaverechner in diesem Zustand, bis der Masterrechner in den Zustand `Fail` übergeht. In diesem Zustand ist der Dienst nicht startbar oder unerreichbar.

Zustand	**Beschreibung**
Up	Dienst ist aktiv
Recover	Dienst kann jederzeit gestartet werden
Fail	Dienst ist nicht startbar
Indoubt	Interner Zustand des Slave: Ausgefallener Master wurde erkannt, es wird aber noch gewartet
Reclaim	Interner Zustand des Masters: Nachdem er erkennt daß Slave up ist fordert er den Dienst zurück

Wie gesagt verwaltet `faild` nur die Zustandsdaten verschiedener Dienste auf dem eigenen Rechner. Woher weiß aber z.B. der Slaveknoten vom Zustand des Masters ? Um diese Überwachung zu realisieren bedient sich Failover eines weiteren Programms: `Failsh`.
`Failsh` ist eine Anwendung, die den Zustand von Diensten auf anderen Rechnern des Clusters in regelmäßigen Abständen abfragt. Die Abfrage erfolgt dabei bei dem `faild`-Dämon des anderen Rechners. Ist der Zustand des Dienstes `fail` oder der andere Knoten nicht erreichbar, dann startet `Failsh` diesen Dienst auf dem eigenen Rechner und teilt dies den anderen Rechnern des Clusters mit. Beispielsweise fragt `Failsh` nach dem Zustand der Cluster-IP-Adresse auf dem Master. Ist dieser unerreichbar dann startet `Failsh` auf dem Slave einen Dienst der die IP-Adresse auf dem Slaverechner einrichtet und die Clients über *gratuitous ARP-Pakete* von dem Wechsel informiert.



Die Abfrageergebnisse und die Bewertung des anderen Rechners kann mit dem Programm `failmon` überwacht werden.

Ein entscheidender Gesichtspunkt dabei ist, dass die IP-Adresse auf dem virtuellen Interface als Dienst zu betrachten ist. Das heißt, es wird von einem speziellen TCL-Skript generiert und gesteuert. Bei der simplen Einrichtung von Failover übernimmt ein Shellskript alle notwendigen Aufgaben der Einrichtung, so dass man sich nur bei weiterführenden Konfigurationen mit dieser Thematik beschäftigen muß.

5 Die Installation und Einrichtung von Failover

Die Installation von Failover gestaltet sich sehr einfach. Es bedarf dazu eines Linux-Systems mit funktionierendem Netzwerk. Als Vorraussetzung für die Installation des Failover-Pakets müssen die GNU Portable Threads[2] (pth) installiert sein. Dabei handelt es sich um eine einheitliche Thread-Architektur, die auf beliebige Unix Varianten portierbar sein soll. Im Zweifelsfall kann ein entsprechendes rpm-Paket bei rpm-find.net gefunden werden.
Um die GNU Portable Threads zu installieren geht man bei RedHat-basierenden Linux-Distributionen z.B. wie folgt vor:

```
# rpm –i pth-1.3.7-1rh61.i386.rpm
```

Nun kann Failover auf die gleiche Weise installiert werden:

```
# rpm –i failover-0.5.11-1.i386.rpm
```

Jetzt befinden sich alle Dateien, die zum Betrieb von Failover nötig sind auf dem Rechner. Nun muß Failover konfiguriert werden. Auch dies ist bei einer Standardkonfiguration sehr einfach. Dazu befindet sich ein Shell-Skript im Verzeichnis `/usr/local/share/failover`. Nach dem Ausführen des Skriptes müssen einige Fragen beantwortet werden und das Skript schreibt dann die dazugehörigen Parameter automatisch in die Konfigurationsdateien.
Der Vorgang läuft beispielsweise wie folgt ab:

```
# cd /usr/local/share/failover
# sh setup

You are about to configure IP address failover for your Linux system.
You will need the following information to succeed:

      1. An IP address that is going to be failed over, i.e. in
         case of a failure, the address will migrate from one
         machine to the other.
      2. The hostname of a master or slave server that backs up
         your system in case of a failure, or that you back up.

During the configuration, some files are installed in /etc/rc.d/init.d.
Do you want to continue? [y] y
What IP address should be failed over [193.5.25.63]? 192.168.0.10
What netmask does the system use [255.255.255.128]? 255.255.255.0
What broadcast does the system use [193.5.25.127]? 192.168.0.255
What interface should be configured [eth0]? eth0
Which subinterface [:1]? :1
Are we the master host [yes]? yes
What is the slave's name? failsafe2

using the following configuration:
        we are the master,
        the master is named localhost.localdomain,
        the slave is named failsafe2,
        the IP address to fail over is 192.168.0.10,
        the netmask of this network is 255.255.255.0
        and the broadcast address is 192.168.0.255
        This address is configured on interface eth0:1

is this configuration ok [yes]? yes
```

[2] siehe Quellenangabe am Ende

Failover ist nun einsatzbereit. Damit Failover nach dem Bootvorgang oder beim Wechsel des Runlevels gestartet oder gestoppt wird, müssen noch entsprechende Links in den Verzeichnissen der Runlevel angelegt werden. Bei Redhat befinden sich diese unter `/etc/rc*.d`, wobei * den Runlevel bezeichnet. Zu beachten ist hier allerdings, daß viele Distributionen andere Verzeichnisse verwenden. Die zuverlinkenden Startup-Skripte wurden von dem eben ausgeführten Setup-Skript bereits unter `/etc/init.d` angelegt. Es handelt sich um `faild`, `failsh` und ein Skript mit dem Namen der `IP-Adresse` die man vorher angegeben hat.
Wichtig ist ferner, dass die Skripte in der Reihenfolge

- `faild`
- *`IP-Adresse`*
- `Failsh`

ausgeführt werden.

Zunächst kann man die Ordnungsgemäße Funktion der einzelnen Skripte durch gezielten Aufruf testen. Dazu startet man zunächst `/etc/init.d/faild` mit dem Parameter `start` und fragt seinen Status mit `failstat` ab. Mittels `ps` kann man überprüfen ob der Dienst überhaupt gestartet wurde. Der Status sollte fail sein, da noch kein Dienst gestartet wurde. Danach startet man das `IP-Adressen`-Skript und fragt nochmals mit `failstat` den Status ab. Er sollte nun `Recover` sein. Schließlich wird `/etc/init.d/failsh` gestartet und mit `failmon` der Cluster überwacht werden. Funktioniert `failsh` nicht, kann auch `failmon` nicht gestartet werden. Asl Zustand sollte nun `up` angezeigt werden.

Desweiteren geht man zum einrichten der Startup-Skripte wie folgt vor:

```
# cd /etc/rc3.d
# ln -s ../init.d/faild S91faild
# ln -s ../init.d/193.5.25.62 S92-193.5.25.62
# ln -s ../init.d/failsh S93failsh
```

Dieses Beispiel richtet die Skripte im Runlevel 3 ein. Entsprechend muß man wenn nötig bei den anderen Runleveln vorgehen. Natürlich kann man auch graphische Tools, wie z.B. `tksysv` oder `Ksysv`, je nach Vorliebe zum Einrichten verwenden.

Bevor die Konfiguration abgeschlossen wird, sollte man sich vergewissern, das der andere Rechner erreichbar ist und die Namensauflösung funktioniert. Eventuell kann man dies durch einen Eintrag in `/etc/hosts` von Hand erledigen.
Die Funktionsfähigkeit der Startup-Skripte kann man durch einen Runlevelwechsel oder einen Reboot testen.
Die Installation erfolgt bei dem Slave Rechner auf analoge Weise.

6 Testen des Clusters

Nachdem der Slave-Rechner eingerichtet ist und man sich von der Funktionsfähigkeit beider Systeme überzeugt hat kann man nun überprüfen, ob der Cluster funktioniert. Vorher sollte man zur Sicherheit beide Computer neu starten. Nach dem Reboot sollten jetzt beide Rechner ihren Status kennen und sich mit dem anderen darüber austauschen.

Durch Aufruf des Überwachungs-Programmes `failmon` auf beiden Rechnern, kann man während des Tests den Verlauf des Statuswechsels verfolgen.

Als erstes sollte man von einem Client-Rechner aus überprüfen, ob der Cluster überhaupt erreichbar ist. Dazu sendet man mit `Ping` fortlaufend ICMP-Pakete an die gemeinsame Cluster-IP-Adresse. Funktioniert alles wie gewollt, hat man mehrere Möglichkeiten um den Wechsel der IP-Adresse vom Master zum Slave herbeizufügen.

Zum Beispiel kann man durch absichtliches Abschalten des `IP-Adressen`-Dienstes einen Failover erzeugen. Dazu bringt man den Dienst auf dem Master Rechner durch folgendes Kommando in den Zustand `fail`:

```
# /etc/init.d/192.168.0.10 fail
```

Der Slave Rechner sollte nun den Ausfall bemerken und nach ca. 20 Sekunden in den Zustand up wechseln. Die IP-Adresse des Clusters ist nach dieser Zeit auch wieder durch `ping` von den Clients erreichbar. Nun kann der Wiederanlauf getestet werden.

```
# /etc/init.d/192.168.0.10 recover
```

Nach dieser Eingabe auf dem Master, sollte der Dienst dort zunächst in den Zustand `recover` und dann in den Zustand `up` wechseln. Auf dem Slave sollte der Zustand nun `recover` sein. Natürlich sollte nun auch ein `Ping` von den Clients aus möglich sein. Somit ist wieder der Normalzustand erreicht.

Der Ausfall des Netzwerkes oder der Netzwerkkarte des Masters läßt sich durch ziehen des Netzwerkkabels simulieren. Nun sollte der Zustand bei beiden `up` sein, denn der beide Rechner gehen nun davon aus, der einzige zu sein, der den Dienst anbietet. Nach dem Einstecken des Netzwerkkabels sollte der Slaverechner wieder in den Zustand `recover` gehen. Der Master wechselt vom Zustand up in `reclaim` und schließlich wieder in den Zustand `up`.

Hartgesottene können einen Stromausfall durch ziehen des Netzsteckers simulieren. Dabei sollten sich die Zustände wie im letzten Beispiel verhalten.

An dieser Stelle noch ein Überblick über die Monitoring- bzw. Diagnosemöglichkeiten von Failover:

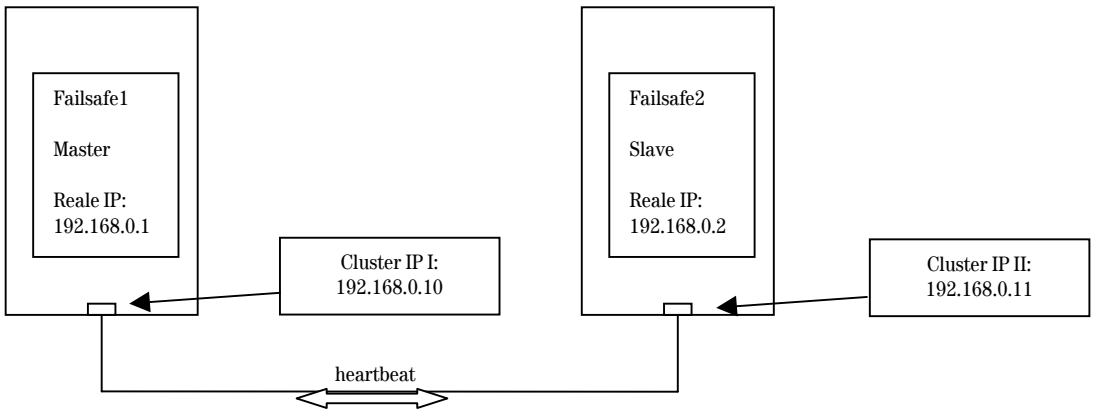
Programm	Beschreibung
Failmon	Dient zur Darstellung der Zustandsdaten von failsh. Funktioniert auch über Netzwerke.
Failstat	Stellt die Daten von faild dar. Ebenfalls Netzwerkfähig.
faildebug	Dient zur Einstellung des Log-Levels (syslog)
failstress	Belastet Failover mit 1000 Anfragen

7 Weiterführende Konfigurationsmöglichkeiten

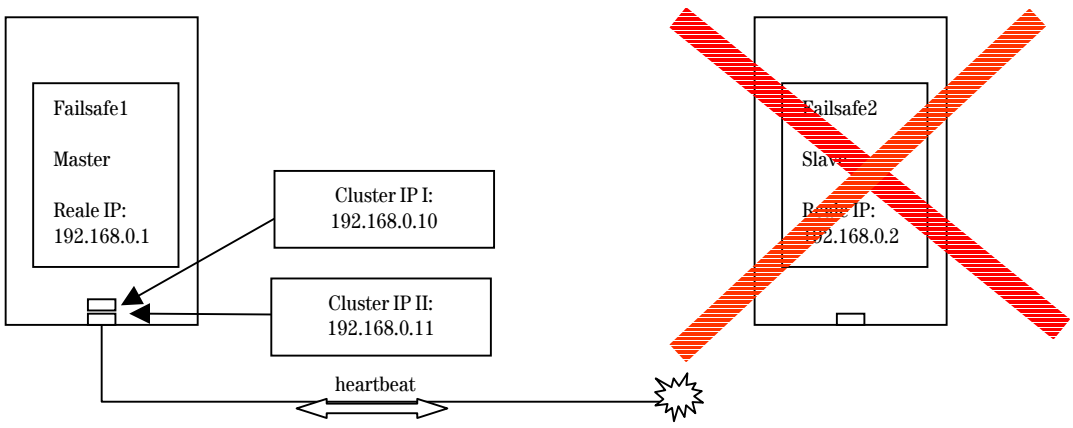
Die Vorangegangenen Erläuterungen haben die einfachste Konfiguration von Failover behandelt. Dabei wird nur eine IP-Adresse zwischen den zwei Rechnern des HA Clusters übernommen. Failover kann darüber hinaus aber noch viele andere Dienste hochverfügbar machen, SNMP-Traps aussenden oder sich als einfacher Loadbalancing Cluster einrichten.
Für die meisten dieser fortgeschrittenen Konfigurationen ist die Kenntnis der Sprache TCL Vorraussetzung.

Symmetrische IP-Adressen , Loadbalancing

Durch die Einrichtung einer weiteren freischwebenden IP-Adresse und der Benutzung eines Round-Robin-fähigen DNS-Servers oder einer Firewall läßt sich Failover als simpler Loadbalancer einsetzen.



Dabei besitzt jede Maschine zusätzlich zu ihrer realen IP-Adresse eine virtuelle, welche sich auf einem logischen Interface befindet. Für diese zusätzliche IP-Adresse übernimmt der Rechner die Rolle als Masterrechner. Der jeweils andere Rechner übernimmt im Gegenzug für diese IP-Adresse die Rolle des Slaves. Der DNS-Server verteilt die Anfragen gleichmäßig an die beiden virtuellen Adressen. Fällt einer der beiden aus, so über nimmt der übriggebliebene die virtuelle IP-Adresse des Gegenübers und bindet diese an ein zweites virtuelles Interface. Er besitzt dann 3 IP-Adressen, eine reale und zwei virtuelle Schnittstellen.



Um symmetrische IP-Adressen einzurichten muß man die Datei `config` im Verzeichnis `/usr/local/share/failover/` mittels eines Text-Editors bearbeiten. Die vorhandenen Kommentare geben Hinweise welche Einräge notwendig sind. Normalerweise müssen die ersten drei Einträge nicht verändert werden. Alle Einträge, die mit required gekennzeichnet sind müssen bearbeitet werden.

```
# failover configuration file, please assign the values you need
# and run setup
#
#directory where failover executables can be found, set only
#if different from defaults
BiNdIr=BINDIR

#directory where failover s-executables can be found, set only
#if different from defaults
SbInDiR=SBINDIR

#directory where pid files and dump files are kept, specify
#only if different from defaults
RuNdIr=PKGDATADIR

#directory containing the tcl scripts (required)
CoNfIgDiR=/usr/local/lib/failover

#name of the interface on each system (required)
InTeRfAcE1=eth0
InTeRfAcE2=eth1

#netmask of the common network (required)
NeTmAsK=255.255.255.0

#broadcast address of the common network (required)
BrOaDcAsT=192.168.0.255

#Host name and failover address of first host (required)
HoSt1=Clusterrechner1
AdDrEsS1=192.168.0.10

#Host name and failover address of second host (required)
HoSt2=Clusterrechner2
AdDrEsS2=192.168.0.11
```

Nach dem Bearbeiten der Datei wird, wie bei der Standardinstallation, das Setup-Skript aufgerufen. Das Skript erstellt nun 2 Verzeichnisse mit den Namen der zwei Clusterrechnern. In diesen Verzeichnissen befinden sich die Start-up-Skripte für beide Rechner. Diese 4 Skripte müssen nun für beide Rechner nach /etc/init.d oder /sbin/init.d kopiert und anschließend für die betreffenden Runlevel verlinkt werden (/etc/rc*.d).

Zudem befinden sich nun 2 Dateien mit den Namen der beiden Rechner und der Dateinamenerweiterung .tcl in dem Verzeichnis. Diese müssen nun für jeden Rechner in das in der Config-Datei angegebene Verzeichnis (`CoNfIgDiR`) kopiert werden. Es wird also

beispielsweise die Datei Clusterrechner1.tcl in das Verzeichnis `/usr/local/lib/failover` auf Clusterrechner1 kopiert. Die Datei clusterrechner2.tcl landet analog dazu in das gleiche Verzeichnis auf Clusterrechner2.
Somit kann der Cluster nun mit den gleichen Tests wie bei der Standardinstallation getestet werden.

Web Interface

Durch die Benutzung der Option `-h` von `failmon` läßt sich eine HTML-Seite generieren, die den Zustand des Clusters wiederspiegelt. Die Seite enthält dieselben Informationen, die `failmon` liefert. Sie erneuert sich zudem jede Minute von selbst . Durch Platzierung der Datei im Dokumentenverzeichnis eines Webservers läßt sich der Zustand von der Ferne überwachen. Desweiteren kann man die Ausgabe von failmon –h auch in eine PHP-Page oder in ein CGI-Programm einbauen um sekundengenaue Überwachung zu ermöglichen. Die Einrichtung bei einer simplen HTML-Seite :

1. Zunächst erstellt man im Verzeichnis `/usr/local/bin/` ein Skript mit dem Namen `failpage`.

2. Dieses füllt man mit diesem Inhalt:

```
#! /bin/sh
      PATH=${PATH}:/usr/local/bin
      export PATH
      cd /usr/local/html
      failmon -h >status.new
      mv status.new status.html
      exit 0
```

3. Wird das Skript mittels einem Eintrag in `crontab` jede Minute ausgeführt.

```
* * * * * [ -x /usr/local/bin/failpage ] && usr/local/bin/failpage
```

SNMP-Traps

`Failsh` kann so eingerichtet werden, das es bei jeder Zustandsänderung einen SNMP-Trap absendet. Dazu muß beim Master-Rechner die Datei `/usr/local/lib/master.tcl` und beim Slave-Rechner entsprechend die Datei `slave.tcl` erweitert werden. Zudem muß das im gleichen Verzeichnis befindliche comonityfile bearbeitet werden.Weitere Informationen dazu befinden sich in der Datei `ws-handouts.pdf` auf der Homepage von Failover[3].

Mehrere Master und Slaves, mehrere Interfaces für den Heartbeat

Auch hier muß die Datei `master.tcl` bzw. `slave.tcl` angepasst werden. Man definiert für den Slave-Rechner mehrere Master, für jede Heartbeat-Adresse einen . Der Failover wird nur dann ausgeführt, wenn keiner der anderen einen Dienst im Zustand `up` hat. Prinzipiell ist es gleichgültig auf welchen Computern die jeweiligen Masters bzw. Slaves verteilt sind.

[3] Siehe Anhang

Optimierung des Timings

Hierzu gibt es in `master.tcl` und `slave.tcl` eine Zeile mit einem Eintrag namens `Interval`:

```
failsvc failsafe1 interval 5
```

An dieser Stelle wird der Wert für das Abfrageintervall in Sekunden angegeben. Im konkreten Fall geschieht die Abfrage alle 5 Sekunden. Der Eintrag muß auf Master und Slave gleich sein. Ansonsten kommt es beim Failback zu Problemen.

Health-Checker

Dieses Programm diente in früheren Versionen von Failover der Überwachung von Diensten. Es konnte diese auch auf entfernten Rechnern überwachen und mit Failover kommunizieren (z.B. um eine Statusänderung zu erzwingen). Seit der Umstellung von Failover von Sun RPCs zu GNU Portable Threads ist es nicht mehr im Verbund mit Failover einsetzbar.

Anhang

Wichtige Verzeichnisse von Failover:

`/usr/local/bin`	Enthält die Programme `failc, faildebug, failmon, failstat, failstress`
`/usr/local/lib/failover`	Enthält die Dateien `comunityfile` und `master.tcl/slave.tcl`
`/usr/local/sbin`	Enthält die Dämonen `faild, failsh`, sowie `farp` und `send_arp`
`/usr/local/share/failover`	Enthält das `Setup-Skript`, dessen `Config-Datei`, sowie die Dokumentation

Weiterführende Dokumentation:

Failover:

Sämtliche Dokumentation befindet sich unter: http://failover.othello.ch/

Empfehlenswert sind vor allem der Getting-Started Guide (engl), Das Linux Clustering Paper (deutsch), sowie zur Beschäftigung mit den erweiterten Fähigkeiten von Failover das WS-Handout.

Andere HA Projekte:

Linux HA Project (Heartbeat):	http://www.linux-ha.org/
FailSafe von SGI:	http://oss.sgi.com/projects/failsafe/
Red Hat HA-Server Project (Piranha):	http://ha.redhat.com/
Linux Virtual Server Project (LVS):	http://www.linuxvirtualserver.org/

Software:

Failover:	http://failover.othello.ch/
GNU Portable Threads (Pth):	ftp://ftp.gnu.org/gnu/pth/
Beliebige rpm-Pakete auffinden	http://rpmfind.net

Fehlersymptome und ihre Ursachen:

1. Cluster Ping endet im timeout

Ursache: Beide Dienste sind `up`.

→ funktoniert Heartbeat?
→ läuft Heartbeat über die richtigen Interface?
→ Intervalle auf Master und Slave gleich ?
→ ARP-Chache vom Client auf dem neuesten Stand?

2. Faild läuft nicht

→ Startup-Skripte in Ordnung ?
→ Richtiger Runlevel ?

3. IP-Adressen-Skript läuft nicht

→ Läuft faild?

4. Failsh läuft nicht, Cluster IP wird nicht erreicht

→ laufen IP-Adressen-Skript und faild?
→ funktioniert Namensauflösung?

5. Nach dem Starten von failstat kommt folgende Fehlermeldung:

```
# failstat
failstat.c:128(0x805aad0/1140/2): cannot connect to remote system:
Connection refused (111)
```

→ Ursache:faild läuft nicht

6. Nach dem Starten von failmon kommt folgende Fehlermeldung:

```
# failmon
receiver.c:55(0x8063a98/1130/3): cannot connect to 127.0.0.1:1848:
Connection refused (111)

failmon.c:175(0x8063a98/1130/2): cannot create socket: Connection refused
(111)

failmon.c:56(0x8063a98/1130/3): initialization failed
```

→ Ursache: failsh läuft nicht

Kurzanleitung zum Installieren eines symmetrischen Failover-Clusters:

1. config- Datei mit Texteditor anpassen
2. setup-Skript ausführen
3. Inhalt des Verzeichnisses Clusterrechner1 auf diesem nach /etc/init.d kopieren
4. clusterrechner1.tcl in das angegebende Konfigverzeichnis auf Clusterrechner1 kopieren (CoNfIgDiR=...)
5. Inhalt des Verzeichnisses Clusterrechner2 auf diesem nach /etc/init.d kopieren
6. clusterrechner1.tcl in das angegebende Konfigverzeichnis auf Clusterrechner1 kopieren(CoNfIgDiR=...)
7. Symbolische Links zu den Start-up-Skripten auf beiden Rechnern anlegen
8. Qualitätskontrolle (Cluster-Test)

Checkliste Failover Installation

	Vorgang	Kommando	Anmerkung
☐	GNU Portable Threads (pth) installiert	rpm –i pth-1.3.7-1rh61.i386.rpm	aktuellen Paketnamen verwenden
☐	Failover-Paket installiert	rpm –i failover-0.5.11-1.i386.rpm	aktuellen Paketnamen verwenden
☐	Setup-Skript ausgeführt	**wo?** /usr/local/share/failover/setup	Weitere Infos: Dokumentation S.9
☐	Faild zum Test starten	/etc/init.d/faild start	
☐	Faild läuft	ps aux | grep faild; failstat	Status in failstat: fail
☐	IP-Adressenskript zum Test starten	/etc/init.d/*.*.*.* start	*.*.*.*= IP-Adresse
☐	IP-Adressenskript läuft	ps aux | grep *.*.*.*; failstat	Status in failstat: recover
☐	Failsh zum Test starten	/etc/init.d/failsh start	
☐	Failsh läuft	ps aux | grep faild; failmon	Zustand in Failmon (abwarten): up
☐	In das Verzeichnis des Standardrunevels wechseln	z.B. cd /etc/rc3.d	Statt 3 auch jeder andere Runlevel
☐	Runlevel-Skript für faild eingerichtet	ln -s ../init.d/faild S91faild	S=Start, unbenutzte Zahlen
☐	Runlevel-Skript für IP-Adressenskript eingerichtet	ln -s ../init.d/193.5.25.62 S92-193.5.25.62	verwenden, Reihenfolge einhalten!
☐	Runlevel-Skript für failsh eingerichtet	ln -s ../init.d/failsh S93failsh	(faild,IP-Adressenskript, failsh)
☐	Killskripte in den Verzeichnissen der anderen Runlevel angelegt	z.B. ln -s ../init.d/faild **K**91failsh	K=Kill, Reihenfolge: failsh, IP-Adressenskript,faild ; freie Zahlen benutzen
☐	Beide Clusterrechner in /etc/hosts eingetragen?		
☐	Faild, *.*.*.* und failsh laufen nach reboot?		siehe oben

Checkliste Failover Clustertest

	Vorgang	Kommando	Anmerkung
	Vorsichtshalber beide Rechner neu booten		
	beide Rechner können sich gegenseitig pingen	ping <hostname>	keine IP-Adresse verwenden
	ping von einem Client auf die Cluster-IP funktioniert	s.o.	
	failmon auf beiden Rechnern aufrufen	failmon	Mit -> bekommt man eine erweiterte Ansicht
	Variante1: Dienst anhalten		
	IP-Adressenskript auf dem Master anhalten	/etc/init.d/192.168.0.10 fail	ping auf Cluster-IP setzt aus
	Slave hat übernommen		ping auf Cluster-IP funktioniert wieder (dauert ca. 20 Sekunden)
	IP-Adressenskript auf dem Master fortsetzten	/etc/init.d/192.168.0.10 recover	Zustand auf Master: recover, dann up Zustand auf Slave: indoubt, dann recover
	Variante2: Netzwerkkabel ziehen		
	Netzwerkkabel aus Master ziehen		selbsterklärend...
	beide Rechner gehen auf up		
	Cluster-IP wird nach 20 Sekunden übernommen		
	Netzwerkkabel wieder in den Master einstecken		
	Slave geht in recover-Zustand über		Zwischenzustand: indoubt
	Master wechselt von up über recover wieder in up		Endzustand: up
	Cluster-IP wird nach 5 Sekunden übernommen		
	Variante3: Stromkabel ziehen		ohne Erklärung