Load-Balancing für den schmalen Geldbeutel

Thomas Heil <heil@terminal-consulting.de>

01.12.2011

Überblick **2**
Einleitung . . . 3
Wer bin ich . . . 4
Wer bin ich . . . 5
Warum dieses Thema . . . 6
Warum dieses Thema . . . 7
Aufgaben und Funktion von Load-Balancing . . . 8
Notwendigkeit von Load-Balancing . . . 9
Notwendigkeit von Load-Balancing II . . . 10
Notwendigkeit von Load-Balancing III . . . 11

LB-Techniken **12**
DNS . . . 13
Layer 2 / 3 / 4 . . . 14
Layer 7 . . . 15

Was sollte der LB noch können **16**
Was noch I . . . 17
Was noch II . . . 18
Was noch III . . . 19
Was noch, Persistence . . . 20
Was noch, Verteilungsverfahren . . . 21

Direct Routing **22**
Direct Server Return . . . 23
Direct Server Return II . . . 24
Direct Server Return . . . 25

NAT **26**
Einfaches DNAT . . . 27
LVS/NAT . . . 28
LVS/Nat . . . 29

IP Tunneling **30**
LVS/IP-Tunneling . . . 31

Reverse Proxy **32**
HAProxy . . . 33
SNAT / HAProxy . . . 34

SNAT TProxy **35**
SNAT TProxy mit HAProxy . . . 36
TProxy SNAT / HAProxy . . . 37

SSL Terminationz.B. mit Wildcard Zertifikat **38**
SSL Termination mit Pound / Stunnel . . . 39
SSL Termination mit Pound II . . . 40
SSL Termination mit Pound und TProxy . . . 41
SSL Termination mit Pound und TProxy . . . 42

Das TIME_WAIT und CONNTRACK ProblemGrenzen von Layer 4 - 7 Proxies **43**
TIME_WAIT. . . . 44
TIME_WAIT II . . . 45

Auswahl des richtigen Load Balancers, Verfahrensoder entsprechender Kombinationen **46**
Kriterien . . . 47

Die grosse Herausforderung **48**
Die Anwendung. . . . 49
Die Anwendung II . . . 50

Der schmale GeldbeuteloderDie OpenSource Philosophie **51**
schmaler Geldbeutel . . . 52
OpenWrt . . . 53
OpenWrt II . . . 54
OpenWrt III . . . 55

Beispiele - / - Anwendungsszenarien **56**
Beispiele . . . 57
Beispiele II . . . 58
Beispiele II . . . 59
Beispiele II-1 . . . 60
Beispiele II-2 . . . 61

Fragen, Hinweise. Danke für Ihre Aufmerksamkeit **62**

Einleitung

- Wer bin ich
- Warum dieses Thema?
- Aufgaben und Funktion von Load-Balancing
- Notwendigkeit von Load-Balancing
- Load-Balancing Techniken
- Wie wähle ich den richtigen LB aus?
  - Hardware vs. Software
  - oder beides? kommerziell oder frei
- Die grosse Herausforderung, Die Anwendung hinter dem LB
- Der schmale Geldbeutel oder die Open Source Philosophie
- Beispiele / Anwendungsszenarien
- Praxisteil, Fazit, Fragen?

Wrt – 3 / 62

Wer bin ich

- Administrator bei der bookandsmile GmbH
  - http://www.billigflieger.de
  - Load Balancing
  - Monitoring
  - Virtualisierung
- OpenSouce Consultant
  - OpenSource Consultant
  - VPN, Proxies, Firewall
  - Virtualisierung, Verteilung, Überwachung
  - Load Balancing, HA

Wrt – 4 / 62

Wer bin ich

- OpenWrt Entwickler
  - Package Maintainer für HAProxy, Openswan, memcached, pound, varnish
- OpenSource Enthusiast

Wrt – 5 / 62

Warum dieses Thema

- erste Versuche mit Debian
  - `HAProxy`, `keepalived`, `iptables`
- OpenBSD
  - `HAProxy`, `CARP`, `PF`
- Installation, Betrieb und Wartung äusserte sich zeitintensiv und schwer nachvollziehbar
- Einarbeitung der Kollegen oft schwierig
- Neuinstallation nach Hardwarschaden benötigte etwa einen Tag
- Suche nach dem absoluten Minimum (bzgl. Installation, Sicherung und Wiederherstellung)
- nachvollziehbaren Anpassungen
- zentralem Accounting, Verkehrsfluss

Warum dieses Thema

- ein LB gehört in jede Landschaft, die Webserver bzw Webservices anbietet, selbst bei sehr kleinen Setups
- auch intern sind LB durchaus hilfreich
  - Balancing von Mysql-Slaves unter Berücksichtigung von `Slave Delays`
- vollständige Realisierung mit freien Bausteinen (OpenSource)
- damit jeder Zugang zu Technologie und Verständnis erlangen kann
- Geld sollte hauptsächlich in menschliche Realisierung, anstatt Hardware und Lizenzen fliessen
- Aufwand für Anpassungen / Konfigurationen fallen auch bei Anschaffung von Hardware und Lizenzen an
- es sollte immer ein entsprechendes Test-System in der Planung berücksichtigt werden

Aufgaben und Funktion von Load-Balancing

- Verteilung hoher Anfragelasten auf nachgelagerte Server

```
1 das bedeutet in erste Linie Scaling nicht Hochverfügbarkeit
```

- Realisierung dieser Aufgabe über bestimmte Verfahren / Layer
  - Layer 2 (Bonding)
  - Layer 3 und Layer 4 (Direct Server Return, NAT)
  - Layer 7 (Reverse Proxy, Application Proxy)
- Realisierung von Firewall-Funktionalitäten nach gegebenen Ansprüchen
- Umsetzung von Hochverfügbarkeitskonzepten
- verlässliche Erkennung, ob Anfragen an nachgelagerte Server abgesetzt werden können
- Realisierung von Server Updates im laufenden Betrieb, ohne Downtime

Notwendigkeit von Load-Balancing

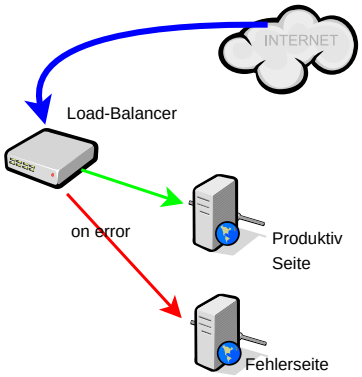
- ein Server kann die Last von Anfragen nicht abarbeiten

```
1 das betrifft vor allem Spitzen, also Zeiten in denen besonders viele Anfragen
2 an den Server gestellt werden
```

- Web-Inhalte sind mehr und mehr dynamisch
- Nutzer verweilen wesentlich länger auf der Webseite bzw. dem Service
- damit belegen sie für einen längeren Zeitraum Ressourcen, die bei statischen Inhalten anderen sofort wieder zur Verfügung gestanden hätten
- kostengünstigerer Ausbau von Ressourcen (kaufe lieber preiswerter)
  - Pizzabox vs. Bolide
  - von jeder Sorte sollten mind. zwei vorhanden sein, um Ausfälle kompensieren zu können
- mehrere Server erhöhen leider auch die Ausfallwahrscheinlichkeit

Notwendigkeit von Load-Balancing II

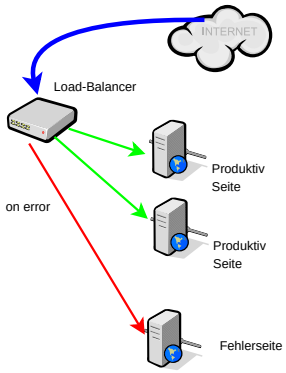
- ein einfaches Fehlerseiten Szenario



- Fehlerseite bsp. über kleine VM
- bei Überlast oder Wartung springt so die Fehlerseite automatisch ein
- immerhin besser als ein Timeout oder eine weisse Seite

Notwendigkeit von Load-Balancing III

- besseres Fehlerseiten Szenario



- Fehlerseite wieder über kleine VM
- bei Überlast oder Wartung springt so die Fehlerseite automatisch ein
- Ausfälle eines Produktiv-Systems führen nicht länger zu einer Fehlerseite

Wrt – 11 / 62

LB-Techniken

12 / 62

DNS

- Verteilung über Round-Robin DNS

```
1 dig -t google.com
2 .root-servers.net.      90288   IN      A       192.58.128.30
3 j.root-servers.net.     90288   IN      AAAA    2001:503:c27::2:30
4 l.root-servers.net.     90288   IN      A       199.7.83.42
5 k.root-servers.net.     90288   IN      A       193.0.14.129
6 ...
```

- Ausfall eines Servers schliesst einen nicht unerheblichen Teil der Nutzer aus
- Scale not HA
- geographisches LB, aka akamai
- auch hier: Anzahl der Server steigert die Ausfallwahrscheinlichkeit

Wrt – 13 / 62

Layer 2 / 3 / 4

- Datalink Ethernet
  - Bonding (z.B. 802.3ad) Link Aggregation
  - Mac based LB (active active CARP[a])
- Layer 3 und 4 (Netzwerk und Transport)
  - Direct Routing (z.B. LVS)
  - Tunneling (z.B. LVS)
  - DNAT oder Full-Nat (z.B. LVS)
- Aufwand sehr gering, da die Abarbeitung im Kernel abläuft
- Kombinationen mit Hochverfügbarkeit möglich
  - VRRP[b])

[a] **CARP** - Common Address Redundancy Protocol
[b] **VRRP** - Virtual Router Redundancy Protocol

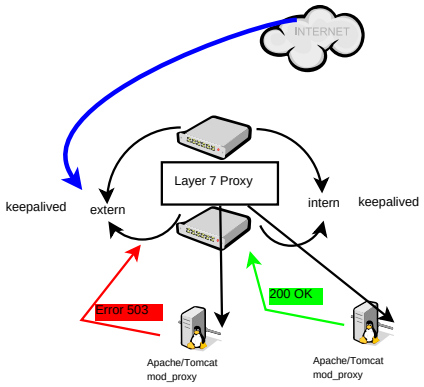
Layer 7

- Application
  - Reverse Proxy basierende Lösungen
  - im Gegensatz zu NLBs[a] brauchen diese wesentlich mehr Ressourcen
  - sie sind langsamer und können wesentlich weniger Traffic verarbeiten
  - können jedoch auf Protokoll Ebene (http, rdp, smtp) arbeiten
  - damit sind bsp. URL Switching u.v.a. mehr möglich
  - Vertreter sind:
    - varnish, haproxy, pound, pen, stunnel
    - apache, lighttpd, nginx
  - Wieviel Traffic haben Sie?

[a] **NLB** - Network Load Balancer

Was noch I

- Health Checks
  - Server antwortet zwar noch auf ICMP und HTTP, liefert jedoch WSOD[a] oder 503
  - Beispiel: HAProxy, Apache, Tomcat



Wrt – 17 / 62

[a] **WSOD** - White Screen of Death

Was noch II

- Health Checks, Direkt versus Indirekt
  - Direkt: check port 80 with 200 OK

```
1 option httpcheck
2 server 01 1.2.3.4:80 cookie 01 maxconn 20 check
```

  - der Server selbst liefert den Status, anderer Port ist möglich
  - INDIRECT: check port xyz with 200OK über einen eigenen TCP Service

```
1 server 01 1.2.3.4:52080 cookie f01 track server-status/f-01 maxconn 100
2 #
3 backend server-status
4 server f-01 1.2.3.5:9201 check inter 20s rise 2 fall 2
5 #
```

  - ein Service mit anderer IP-Adresse / Port liefert den Status eines Servers
  - in HAProxy Worten: benutze den Status eines anderen Backends

Wrt – 18 / 62

Was noch III

- Zusammenfassung Checks
  - oft reicht ein einfacher TCP Check nicht aus (http vs. pure tcp)
  - es muss darauf geachtet werden, dass Checks nicht zu häufig ausgeführt werden
  - nicht das die Server nur noch mit Checks beschäftigt sind
  - Wie verhält sich der Check unter Last Bedingungen?
  - Flapping muss berücksichtigt werden
  - achten Sie darauf, dass die angestrebte Lösung diese Möglichkeiten besitzt
- Persistence
  - der LB muss eine Anfrage von einem Client immer wieder an den gleichen Server schicken
  - z.B. wäre ohne globales Session Management kein Login möglich (PHPSESSION)

Was noch, Persistence

- Persistence
  - der LB muss also wissen oder sich merken, welche Anfrage an welchen Server ausgeliefert wurde
  - was passiert, wenn ein Server unpässlich ist
  - was passiert, wenn einige Pakete verloren gehen und die Anfrage erneut abgesetzt werden muss
  - Je nach Verfahren entsteht ein nicht unerheblicher Aufwand, der vor allem CPU und Speicher kostet
  - Mit wieviel Speicher werden HW-Appliances ausgeliefert? (2-4GB RAM)
  - Ausweg Cookie Insertion (Proxy injiziert Cookie und sendet ihn zurück an den Client)
  - IP-Hashing-basierend, wenn bsp. TCP verwendet wird sind keine Cookies möglich
  - Wie ist die IP Verteilung der Clients (NAT, Proxy etc.)

Was noch, Verteilungsverfahren

- der erste der antwortet
  - muss keine gute Auslastung / Verteilung bedeuten, da viele Faktoren die Antwortzeiten bestimmen
- der die wenigstens Verbindungen hat
  - sinnvoll, wenn lange Sessionzeiten gebraucht werden
- round robin
  - nicht deterministisch, aber sorgt dennoch für eine gute Verteilung auf alle beteiligten Server
  - sehr einfach umzusetzen
- alle Verfahren lassen sich mit Wichtungen verbinden um bsp. vorhandene Hardwareklassen berücksichtigen zu können (schwache vs. starke Server)
- und natürlich IP-Hashing

Direct Server Return

- Funktionsweise
  - Umschreiben der Ziel-Mac-Adresse auf einen oder mehrere Server mittels `ipvsadm` und entsprechender Kernel Module
  - Server sendet die Antwort direkt
  - sehr performant
- Randbedingungen
  - Backend darf auf ARP auf loopback nicht antworten
  - kann damit nicht auf allen Systemen betrieben werden

```
ipvsadm -A -t <VIP>:80 -s wrr
ipvsadm -a -t <VIP>:80 -r 192.168.1.11:80 -g
ipvsadm -a -t <VIP>:80 -r 192.168.1.12:80 -g
sysctl -w net.ipv4.ip_forward=1
sysctl -w net.ipv4.conf.lo.arp_ignore=1
sysctl -w net.ipv4.conf.lo.arp_announce=2
#
```

  - LB und Server müssen sich innerhalb des gleichen Netzwerkes befinden
  - Anpassungen auf den Servern notwendig

Direct Server Return II

- Erweiterung mittels `keepalived`

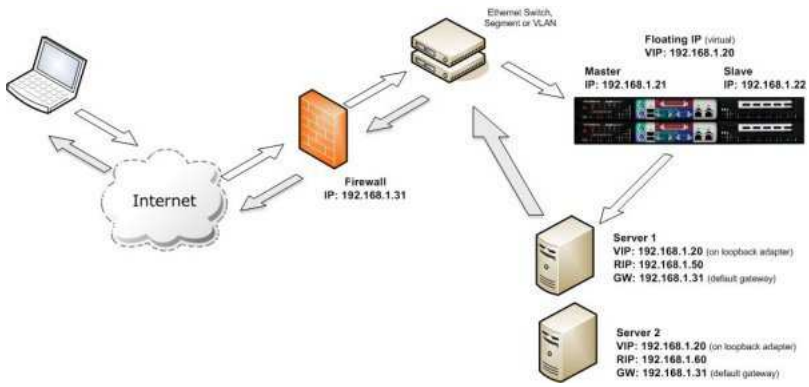
```
#on the balancer
virtual_server 192.168.1.10 80 {
        delay_loop 30
        lb_algo wrr
        lb_kind DR
        persistence_timeout 500
        protocol TCP
        real_server 192.168.1.11 80 {
                weight 1
                TCP_CHECK {
                        connect_port 80
                        connect_timeout 3
                }
        }
        real_server 192.168.1.12 80 {
                weight 2
                TCP_CHECK {
                        connect_port 80
                        connect_timeout 3
                }
        }
}
```

```
#auf den Webservn
#iptables-mod-nat-extra
iptables -t nat -A PREROUTING -p tcp -d  192.168.1.10 --dport 80 -j REDIRECT
```

- Service checks dank keepalived (Framework um ipvsadm)

Direct Server Return

- Schema von loadbalancer.org



NAT

Einfaches DNAT

- Beispiel:
  - Umschreiben der Zieladdresse auf einen Netzbereich

```
1 iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to 172.16.0.2-172.16.0.3
```

- PRO
  - einfache Realisierung
- CON
  - keine Einflussmöglichkeiten
  - Hosts müssen aufeinander folgen
  - Hosts müssen immer verfügbar sein

```
1 nicht zu gebrauchen
```

LVS/NAT

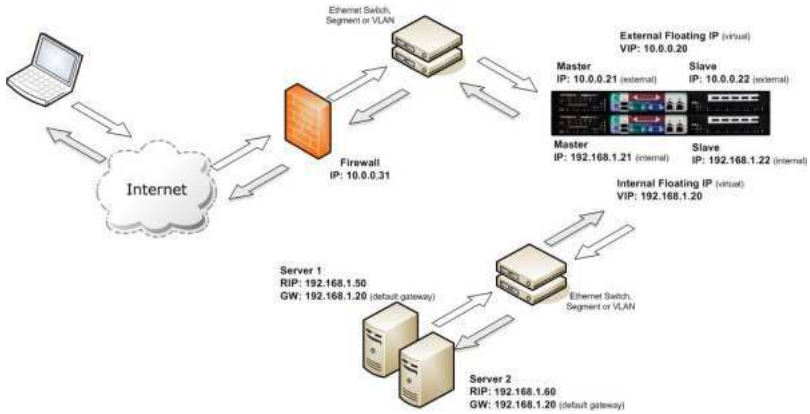
- Beispiel:
  - Umschreiben der Zieladdresse auf einen oder mehrere Server mittels `ipvsadm` und entsprechender Kernel Module

```
1 ipvsadm -A -t <VIP>:80 -s wrr
2 ipvsadm -a -t <VIP>:80 -r 192.168.1.11:80 -m
3 ipvsadm -a -t <VIP>:80 -r 192.168.1.12:80 -m
```

- PRO
  - einfache Realisierung
  - keine Anpassung der Zielrechner notwendig
- CON
  - keine Service Checks
  - LB muss Default Gateway für alle beteiligten Server sein
  - oder entsprechend konfiguriertes Policy Routing

LVS/Nat

- Schema von loadbalancer.org



LVS/IP-Tunneling

- Beispiel:
  - arbeitet wie Direct Routing
  - die VIP wird auf jedem Backend konfiguriert (nur tun (ip tunnel) Interface)
  - LB sendet Anfragen über entsprechenden IP Tunnel (ipip), jedoch mittels DNAT

```
1 ipvsadm -A -t <VIP>:80 -s wrr
2 ipvsadm -a -t <VIP>:80 -r 192.168.1.11:80 -i
3 ipvsadm -a -t <VIP>:80 -r 192.168.1.12:80 -i
```

- PRO
  - Server können durch Tunnel in fast beliebigen Netzen stehen
- CON
  - keine Service Checks
  - Anpassung auf Servern notwendig (IP Tunnel)

Wrt – 31 / 62

HAProxy

- Beispiel:
  - Anfrage wird vom Proxy entgegengenommen und dann nach speziellen Regeln an einen oder mehrere Server weitergeleitet

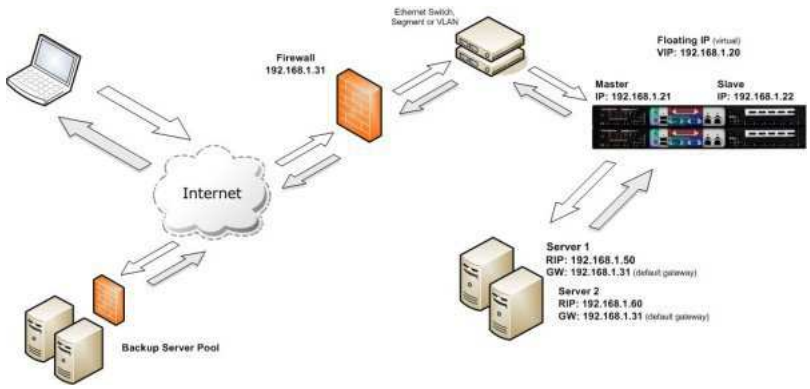
```
1 listen http:80
2   mode http
3   option xforwardfor
4   server 01  192.168.1.1:80
5   server 02  192.168.1.2:80
```

- PRO
  - einfach zu realisieren, Service Checks eingebaut
- CON
  - Server sieht nicht die Absender IP des Clients
  - Anpassung auf Servern für Logging und Access-Control mit `X-Forwarded-For`

Wrt – 33 / 62

SNAT / HAProxy

- Schema von loadbalancer.org



SNAT TProxy

SNAT TProxy mit HAProxy

- Beispiel:
  - Anfrage wird vom Proxy entgegengenommen und im Transparent Modus an einen oder mehrere Server weitergeleitet

```
1 listen http:80
2   mode http
3   source 0.0.0.0 usesrc clientip
4   server 01  192.168.1.1:80
5   server 02  192.168.1.2:80
```

- PRO
  - keine Modifikation an den Servern notwendig (Proxy wird quasi unsichtbar)
- CON
  - LB muss Default Gateway sein
  - LB muss TProxy und Policy Routing unterstützen

TProxy SNAT / HAProxy

- Building Blocks

```
iptables -t mangle -N DIVERT
iptables -t mangle -A PREROUTING -p tcp -m socket -j DIVERT
iptables -t mangle -A DIVERT -j MARK --set-mark 1
iptables -t mangle -A DIVERT -j ACCEPT
ip rule add fwmark 1 lookup 100
ip route add local 0.0.0.0/0 dev lo table 100

echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
echo 1 > /proc/sys/net/ipv4/conf/all/send_redirects
echo 1 > /proc/sys/net/ipv4/conf/eth0/send_redirects

opkg install iptables-mod-tproxy iptables-mod-ipopt iptables-mod-conntrack iptables-mod-nat
    ip6tables
```

- Prerouting schiebt TCP nach Divert
- DIVERT bekommt FWMARK 1 und sorgt bei der Beanwortung einer Anfrage dafür, das HAProxy die Anfrage wieder bekommt

SSL Termination
z.B. mit Wildcard Zertifikat

SSL Termination mit Pound / Stunnel

- Beispiel:
  - Anfrage wird von Pound per https entgegengenommen und dann an einen Reverse Proxy weitergeleitet

```
ListenHTTPS
        Port 443
End
Service
        BackEnd
          Port 10443
        End
End
```

- PRO
  - auf internen Server wird http gesprochen
  - Inspektion des Traffics möglich
- CON
  - SSL kann den LB schnell auslasten
  - Die Anwendung muss mitspielen z.B. `AddHeader "X-SSL-Request: 1"`

SSL Termination mit Pound II

- Beispiel:

```
frontend poundl_443_auf10443
  bind :10443
  mode http
  option forwardfor except 127.0.0.1
  acl acl_web1 -f /etc/haproxy-regex-web1.lst
  acl acl_web2 -f /etc/haproxy-regex-web2.lst
  use_backend web1 if acl_web1
  use_backend web2 if acl_web2
  default_backend devnulll
backend web1
  .. server ...
backend web2
  .. server ...
```

- je nach Regex auf Hostname wird ein anderes Backend angesprochen
- `devnull` liefert timeout, damit gibt es per se keine nicht konfigurierten Hosts
- `viele *.wildcarddomain.de` möglich
- vor allem für kleinere Shops oder Test-Instanzen sinnvoll
- Kombinationen mit Direct Routing sinnvoll

SSL Termination mit Pound und TProxy

- Änderungen:
  - es ist auch möglich pound als transparenten Proxy zu konfigurieren
  - der wiederum transparent an HAProxy weiterleiten kann

```
http://blog.loadbalancer.org/transparent-proxy-of-ssl-traffic-using-pound-to-haproxy-backend-patch-and-howto/
```

- PRO:
  - keine Anpassungen an den Servern notwendig
  - End to End transparent
- CON:
  - komplexe Konfiguration
  - LB muss Default Gateway sein
  - limitierte Sessions, da SSL und iptables Mehraufwand

SSL Termination mit Pound und TProxy

```
 ListenHTTPS
        Address 0.0.0.0
        Port 443
        MaxRequest 16384
End
Service
        BackEnd
                Address 127.0.0.1
                Port 10443
                TProxy 1
        End
End

iptables -t mangle -N DIVERT
iptables -t mangle -A PREROUTING -p tcp -m socket -j DIVERT
iptables -t mangle -A DIVERT -j MARK --set-mark 1
iptables -t mangle -A DIVERT -j ACCEPT

iptables -t mangle -A OUTPUT -s 127.0.0.1 -p tcp -sport 10443 -j DIVERT
iptables -t mangle -A OUTPUT -d 127.0.0.1 -p tcp -dport 10443 -j DIVERT

ip rule add fwmark 1 lookup 100
ip route add local 0.0.0.0/0 dev lo table 100
```

- ausgehender Verkehr muss ebenfalls nach Divert, so das Pound die Anfrage auch zurückbekommt

Das TIME_WAIT und CONNTRACK Problem
Grenzen von Layer 4 - 7 Proxies

TIME_WAIT

- Erläuterung
  - da der LB auf TCP aufsetzt, wird eine Session die erledigt ist in den `time_wait` Zustand übergehen
  - dort verbleibt sie solange wie es der Kernel (sysctl) vorsieht; (15-60 Sekunden) sind das absolute Minimum
  - Systeme, die sehr viele Session zu bearbeiten haben, brauchen entsprechend grosse Session Tabellen

```
 1000 session / s, Session dauer 0,1s  = 0,1 / (1 / 1000) = 100 aktive Sessions
  60s / ( 1 /1000) = 60000 TIME_WAITS
```

  - `TIME_WAIT` ist sehr preiswert, allerdings müssen entsprechende Session Tabellen im Arbeitsspeicher vorgehalten werden.

```
# tcp-time-wait buckets pool size
# net.ipv4.tcp_max_tw_buckets = 180000
```

TIME_WAIT II

- weitere Parameter

```
 net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_max_orphans = 8192
net.ipv4.ip_local_port_range = 16384    64000
net.ipv4.tcp_sack=0
net.ipv4.tcp_window_scaling=0

net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_tw_recycle = 1

net.ipv4.tcp_max_tw_buckets = 16777216
net.ipv4.tcp_max_syn_backlog = 262144
net.core.netdev_max_backlog = 15000
net.core.somaxconn = 262144

net.core.rmem_max  = 8738000
net.core.wmem_max = 6553600
net.ipv4.tcp_rmem = 8192 873800 8738000
net.ipv4.tcp_wmem = 4096 655360 6553600
vm.swappiness=0

fs.file-max=400000
fs.nr_open=400000

#+ nf_conntrack part
```

- Erlaube sehr viele `Waits`, sonstige Verbindungen
- dimensioniert auf 16GB RAM oder mehr

Auswahl des richtigen Load Balancers, Verfahrens oder entsprechender Kombinationen

Kriterien

- Durchsatz
  - ☐ Frage in die Runde: Wieviel Durchsatz brauchen Sie?
  - ☐ Wieviele Session (maximal) werden erwartet?
  - ☐ Nicht alle sind Slashdot oder Google
- Bedienbarkeit
  - ☐ Wie einfach bzw. wie schwer ist es Änderungen vorzunehmen?
  - ☐ verstehen die Entwickler / Programmierer wie das LB funktioniert?
    Bsp: Reverse Proxy: Download ohne Content-Length
- Pros und Cons
  - ☐ muss es Hardware sein, oder geht auch Software
  - ☐ muss es ein kommerzielles Produkt sein?
    Hilft das den Admins / Entwicklern
  - ☐ Wie hoch liegt die TCO wirklich

Die Anwendung

- Trennung von statischen und dynamischen Inhalten
- `/static/img` vs. `/.*?`
- statische Inhalte lassen sich beispielsweise viel schneller mittels `lighttpd` ausliefern
- Realisierung mittels `URL Switching`
  - Tuning von Apache und dynamischen Inhalten
  - deaktivieren von keepalive (wirkt sich auch sehr negativ beim Einsatz von `URL Switching` aus)
- Lasttests der Anwendung
  - Die Anwendung sollte vor dem Start ein Profiling erfahren
  - oder: Auslastung zu Anfang sollte `<= 40 %` betragen

Wrt – 49 / 62

Die Anwendung II

- beliebte Fehler
  - dynamische Downloads ohne Content-Length
  - zu große Post oder Get Requests
  - Verwendung von absoluten URLs oder Protokollen
- Fehlerszenarien
  - Eruieren von Do‘s and Dont‘s
  - Dokumentation (z.B. wie wird ein Server aus dem Grid entfernt für Downtime oder Updates
  - Automatisierung und zentrale Verwaltung der Webserver Konfigurationen (z.B. puppet)
- Sonstiges
  - Überwachung der Anwendung, trotz LB
  - Implementierung von Langzeitstatistiken und vor allem Logging

Wrt – 50 / 62

schmaler Geldbeutel

- absolutes Minimum
  - Realisierung von Load-Balancing Mechanismen
  - Firewall Funktionen
  - VPN Verbindungen
    - OpenVPN, OpenSwan, tinc, Strongswan ...
  - Überwachung / Starten von Diensten, Statistiken
    - monit, xinetd, curl, collectd, luci ...
  - alles auf Basis von OpenWrt
    - X86 bare metal, KVM, XEN, Hyper-V, VMWare, Virtualbox
- Building Blocks
  - OpenWrt Buildchain, ein paar Patches, ein paar Modifikationen, ein paar neue Tools
  - im Augenblick auf 32Bit begrenzt

Wrt – 52 / 62

OpenWrt

- SMP und PAE Unterstützung
- threaded OpenSSL, neue Kernel Module
- gcc, varnish, pound, acpid, keepalived u.v.a.
- rudimentäre initrd Unterstützung (z.B. booten vom USB Stick)
  - uclibc basierend (Basis Image 45MB)
  - busybox als bash Ersatz
  - hotplug statt udev (udev zu gross)
- klein oder gross
  - gleiches Framework egal ob Seagate Dockstar oder X86 Server
  - Test versus Produktiv
  - mit wenig Hardwareaufwand lassen sich komplexe Netzwerk Szenarien realisieren

Wrt – 53 / 62

OpenWrt II

- Warum OpenWrt
  - einfach zu installieren und dabei sehr klein
  - damit wird jedem der Zugang zu dieser Technologie ermöglicht
  - Test-Setups kosten fast nichts mehr
  - aufgrund der Größe kann auch einfach die Image Datei gesichert werden
  - oder eben nur die Konfigurationen und die Paketlist

```
1 daraus lässt sich mittels imagebuilder ein neues Image erzeugen
2 Neuinstallation == neues Image
```

  - Kosten für die Realisierung und Anpassung eines Setups fließen vollständig in Hardware und Dienstleistungen ein
  - Was wir verkaufen ist Zeit und Know-How
  - Skalierung jederzeit möglich
  - kein Cluster Framework notwendig

OpenWrt III

- Wir arbeiten an einer NDA[a]
  - Mixture aus LVS, Pound, stunnel, HAProxy ...
  - Web-GUI‘s
  - Installation, Backup, Upgrade und Recovery Prozeduren
  - am `sysctl` Tuning
  - an der Farbe des Servers
- Und die Kosten
  - entstehen nur für Beratungsleistungen und spezielle Features
  - kommen zu einem nicht unwesentlichen Teil dem OpenWrt Projekt zugute
- Timeframe
  - Mitte bis Ende 2012

[a] **NDA** - Network Distribution Appliance

Beispiele

- MySQL Slave LB
  - mehrere Slave werden unter Berücksichtigung von `Seconds behind Master` angesprochen
- redundanter LB für Apache
  - zwei LB via VRRP intern und extern und vielen Webservern
- redundanter LB für IIS mit TProxy Snat
  - mehrere IIS mit Service Checks hinter einem HAProxy
  - LB == Gateway
- redundanter LB für Streaming Server
  - CRTMPSERVER auf 1935 (TCP, TProxy SNAT)
  - LB == Gateway

Wrt – 57 / 62

Beispiele II

- SSL Termination mit Pound
  - aufbrechen des SSL Traffics (transparent) oder per SNAT
- Direct Server Return und SSL-Termination
  - Wenn der Terminator nicht mehr ausreicht (SSL ist sehr CPU hungrig)

Wrt – 58 / 62

Beispiele II

- Beispiel Session TPROXY

```
opkg install iptables-mod-tproxy iptables-mod-ipopt iptables-mod-conntrack iptables-mod-nat
opkg install haproxy
#/etc/firewall.user
iptables -t mangle -N DIVERT
iptables -t mangle -A PREROUTING -p tcp -m socket -j DIVERT
iptables -t mangle -A DIVERT -j MARK --set-mark 1
iptables -t mangle -A DIVERT -j ACCEPT
#/etc/rc.local
ip rule add fwmark 1 lookup 100
ip route add local 0.0.0.0/0 dev lo table 100
```

Wrt – 59 / 62

Beispiele II-1

```
#machine one
config 'interface' 'wan'
        option 'ifname' 'eth0'
        option 'proto' 'static'
        option 'type'   'none'
        option 'ipaddr' '192.168.1.10'
        option 'gateway' '192.168.1.14'
        option 'dns' '141.1.1.1'
        option 'netmask' '255.255.255.0'

config 'interface' 'lan'
        option 'ifname' 'eth1'
        option 'proto' 'static'
        option 'type'   'bridge'
        option 'ipaddr' '192.168.0.10'
        option 'netmask' '255.255.255.0'

#machine two
config 'interface' 'lan'
        option 'ifname' 'eth0'
        option 'proto' 'static'
        option 'type'   'none'
        option 'ipaddr' '192.168.0.11'
        option 'gateway' '192.168.0.10'
        option 'netmask' '255.255.255.0'
```


Beispiele II-2

```
global
        uid 0
        gid 0
        daemon

defaults
       mode    http
       retries 5
       maxconn         120000
       contimeout      130000
       clitimeout      130000
       srvtimeout      130000
       option  redispatch

listen app1
       bind :8080
       mode http
       #transparent do not use transparent until your 100 percent sure
       maxconn 200
       stats enable
       stats uri /

frontend  http-port-80
       bind :80
       mode http
       maxconn 5000
       option dontlognull
       default_backend www

backend www
        mode http
        balance roundrobin
        source 0.0.0.0 usesrc clientip
        server server01 192.168.0.11:80 maxconn 200 check
        server server01 192.168.0.12:80 maxconn 200 check
```


Fragen, Hinweise. Danke für Ihre Aufmerksamkeit 62 / 62