Linux Clustering

Dr. Andreas Müller

afm@othello.ch

Slide 1

Linux Conference, 27. Juni 2000

Agenda

1. Warum Clustering? Ziele?
2. Anwendungszustand
3. Freie Lösungen
4. Was braucht es in Zukunft?

Slide 2

Warum Clustering?

1. **Rechenleistung:** 1 Problem zu gross für 1 Rechner
2. **Lastverteilung:** viele Probleme zusammen zu gross für 1 Rechner
3. **Verfügbarkeit:** Tod eines einzelnen Rechners zu wahrscheinlich

Slide 3

Rechenleistung: Parallelisierbarkeit

1. **Ulam Folge:** $x_n = u(x_{n-1})$ mit

$$u: \mathbf{N} \to \mathbf{N} : x \mapsto \begin{cases} \frac{x}{2} & x \text{ gerade} \\ 3x+1 & x \text{ ungerade} \end{cases}$$

   Für welches n ist $x_n = 1$: keine Parallelisierbarkeit bei gegebenem x_0

Slide 4

2. **Kryptologische Probleme:** zum Beispiel Brute Force Angriff auf RC5-64 durch `distributed.net`
   (a) ca. 250000 CPUs
   (b) bisher rund 1000 Tage, geschätzte Dauer: nochmals über 1000 Tage
   (c) Rate: 113 GKeys/s, Zuwachs: 143 MKeys/s pro Tag

3. **Ray Tracing:** Je glänzender die Flächen, desto "glänzender" die Parallelisierbarkeit
4. **Physikalische Probleme:** approximative Entkoppelung wegen
   (a) $c < \infty$, keine Wirkung breitet sich schneller als die Lichtgeschwindigkeit aus

Slide 5

   (b) Geometrie. Aber: je höher die Dimension, desto schwieriger die Entkoppelung.

Rechenleistung: Kommunikation

1. Topologie der Teilprobleme (wer muss mit wem kommunizieren) bestimmt Kommunikationsarchitektur des Clusters
2. n Knoten haben $\frac{n(n-1)}{2}$ mögliche Verbindungen: nicht skalierbar

Slide 6

3. k-dimensionaler Würfel: 2^k Knoten haben $2k$ Verbindungen
4. Ziel: Rechenleistung und Bandbreite gleichmässig auslasten

Lastverteilung

Slide 7

1. **Netzwerk Link:** Bandbreite zweier Links optimal ausnutzen
2. **Internet Proxy:** Memory- und/oder I/O-Kapazität durch einen zweiten Rechner erweitern
3. **Web Server:** CPU- oder TCP-Engpässe durch zweites System abfangen
4. **File Server:** Netzwerkdistanz zum Klienten minimieren

Verfügbarkeit

Slide 8

1. Backup-System übernimmt bei einem Ausfall
2. Kann kombiniert sein mit Load Sharing
3. **Cold Standby:** Failover durch Netzschalter
4. **Hot Standby:** Failover durch "Stöpseln" oder automatisch

Anwendungszustand

1. Beim Failover muss der (serverseitige) Anwendungszustand migriert werden
2. Serverseitiger Zustand beeinträchtigt Skalierbarkeit
3. $\Rightarrow$ möglichst wenig serverseitiger Zustand

Slide 9

4. Aber: Klientenseitiger Zustand beeinträchtigt Robustheit und Portabilität
5. Klientenseitiger Zustand braucht Bandbreite
6. Zustand muss beim Wiederanlauf wieder hergestellt werden können

1. Wenn kein Zustand vorhanden ist, muss auch nichts migriert werden
2. Je mehr ich auf dem Server speichern muss, desto mehr Resourcen muss ich für die Speicherung und das wiederfinden allozieren
3.
4. Ein Windows Klient ist weniger stabil, und Abhängigkeiten vom Browser sind das Kreuz der Web-Programmierer.
5. Bei grossen Datenmengen für den Zustand ist ein dauernder Hin- und Hertransport zwischen Klient und Server nicht praktikabel.
6. Zum Wiederherstellen des Zustandes sind viele verschiedene Vorgehensweisen denkbar:
   (a) Der Zustand wird zusammen mit der Benutzeridentifikation in einem persistenten Speicher (Filesystem, Datenbank, Verzeichnisdienst) gespeichert. Bei Neuidentifikation des Benutzers kann er wiederhergstellt werden. Damit ist nicht nur die Verfügbarkeit besser, für Klient und Server wird auch Roaming möglich. Trotzdem geht jener Zustand verloren, der zu nicht committeten Transaktionen gehört.
   (b) Angefangene Operation werden in ein Transaktionslog geschrieben. Beim Wiederanlauf kann durch nachfahren des Transaktionslog auch der “hängige” Zustand wiederhergestellt werden.

Zustandslose Dienste

Slide 10

1. Routing. Aber: Routingprotokoll bauen grosse Mengen an Zustand in den Routingtabellen auf. Kann jederzeit wiederhergstellt werden.
2. DNS
3. NIS
4. LDAP, nur connectionless LDAP. Normales LDAP braucht `bind`.
5. RPC

Wie entsteht Zustand?

Slide 11

1. Authentisierung
   (a) TCP Sequenz Nummern: TCP socket state (`telnet`, `ftp`)
   (b) TCP Sequenz Nummern und Sitzungsschlüssel (`ssh`)
   (c) Basic Authentication Header (`http`)
   (d) Cookies (`http`)
   (e) URL-Komponenten (hidden fields, `http`)
   (f) Tickets (Kerberos)
   (g) SSL Session Keys (`https`, `simap`)

   Die Trennung von Authentisierung und Anwendungszustand ist Voraussetzung für Single Sign On.

2. Anwendungsspezifischer Zustand
   (a) Offene Files und Locks in einem Netzwerk Filesystem
   (b) Hängige Transaktionen in einer Datenbank
   (c) Temporäre Objekte in einer Corba-Anwendung
   (d) Benutzer-History

Slide 12

Stateless Design

1. Ermöglicht problemlosen Failover
2. Workaround: Nur zustandslose Komponenten clustern
3. TCP basierte Anwendungen: der TCP Zustand geht immer verloren, es braucht automatischen Sitzungswiederaufbau (SMB)

Slide 13

4. Firewalls: je mehr Zustand, desto besser kontrollierbar

Wiederanlauf und Split Brain

Slide 14

1. Wiederanlauf ist meist schwieriger als der eigentliche Failover
2. Nicht replizierte Updates des Masters müssen mit den Updates des Slave in Einklang gebracht werden
3. Split Brain (zum Beispiel durch Kommunikationsprobleme verursacht): Mehrere Knoten aktiv
   (a) Wiederholte Anwendungsfehler wegen ARP-Inkonsistenz
   (b) Gleichzeitige Updates auf zwei Servern

Freie HA Projekte

Slide 15

1. Fake
2. failoverd
3. Heart
4. Linux Virtual Server
5. Linux-HA
6. `failover`

IP Address Take Over

Verwende ARP um die Zuordnung von Hardware-Adresse zu IP-Adresse zu beeinflussen:

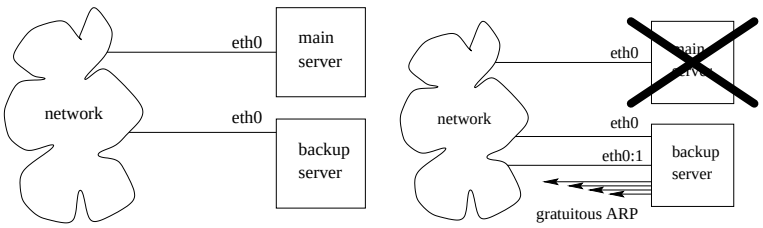
1. Keine statischen ARP Einträge
2. ARP Antwort mit einer anderen Zuordnung als gegenwärtig im ARP Cache $\Rightarrow$ Update
3. Gratuitous ARP: ARP Anfrage eines Rechners nach seiner eigenen Hardware-Adresse $\Rightarrow$ Update in allen Rechnern, die einen ARP Eintrag für diese Adresse haben

Slide 16

1. Diese Bedingung lässt sich zum Beispiel bei in gewissen NAT-Situationen mit Firewall-1 nicht durchhalten. Probleme auch mit Proxy-ARP Situationen.
2. Diese Forderung ist in vielen Implementationen nicht erfüllt, und wird manchmal auch von Switches verhindert.
3. Gratuitous ARP wird von gewissen Systemen nicht ausgewertet: Firewall-1 und (angeblich) von gewissen Windows-Versionen

Fake

Slide 17



Normalbetrieb Backupbetrieb

failoverd

1. Überwacht Interface auf Aktivität
2. Failover (mit IP Address Takeover) beim Ausfall des Paketstroms vom Master
3. Perl-Implementation

Slide 18

Heart

1. Heartbeatprotokoll für den Aufbau eines Clusters
2. UDP-basiert
3. “Konfiguriert sich selbst”
4. niedrige Latenz angestrebt

Slide 19

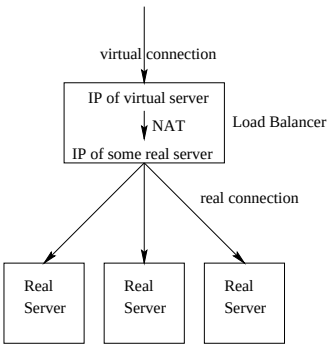
5. Entwicklung eingestellt, aufgegangen in Linux-HA

Linux Virtual Server

1. Alle Anfragen an virtuelle Adresse
2. LVS verteilt mittels NAT auf reale Server

Slide 20

3. verschiedene Scheduling Algorithmen
4. Limitiert durch NAT-Fähigkeiten und Scheduler



Linux-HA

1. Ambitiöses Projekt mit den im Linux-HA-Howto formulierten Zielen
2. Heart wird als Heartbeat verwendet
3. IP Address Takeover

Slide 21

4. Stand der Entwicklung: Master/Slave-Konfiguration mit zwei Knoten
5. Work in Progress

failover

1. IP Address Take Over
2. Implementation in C
3. für Solaris und Linux
4. SNMP Trap Support

Slide 22

5. Monitor Interface (Text, HTML, Curses)

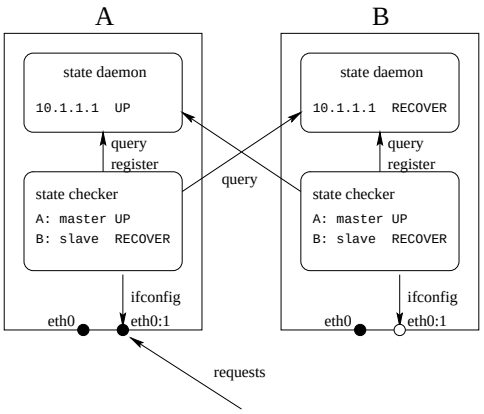
failover: Architektur

Slide 23

1. Zustandsdämon (`faild`)
   (a) `UP`: Dienst ist aktiv
   (b) `RECOVER`: Dienst kann jederzeit gestartet werden
   (c) `FAIL`: Dienst ist nicht startbar
2. Zustandsklient (`failsh`)
   (a) Fragt alle Dämons des Clusters via RPC ab
   (b) Vollführt Zustandsänderungen
   (c) Datiert lokalen Dämon auf
   (d) Zustandslos, restart jederzeit möglich, da der Zustand aus `faild` wieder hergestellt werden kann

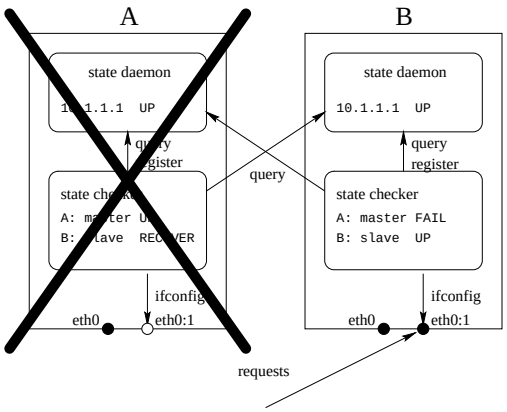
Failover Normalbetrieb

Slide 24



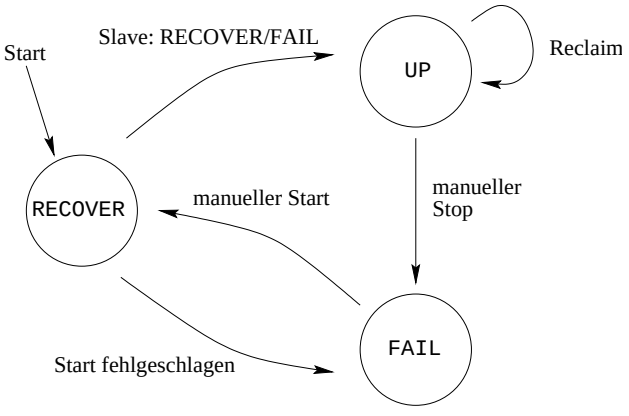
Failover Backupbetrieb

Slide 25



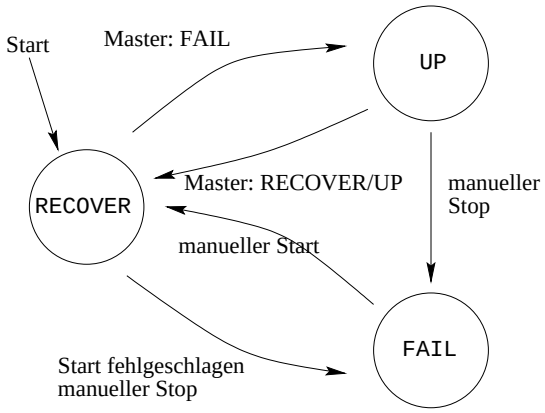
Failover Zustandsdiagramm Master

Slide 26



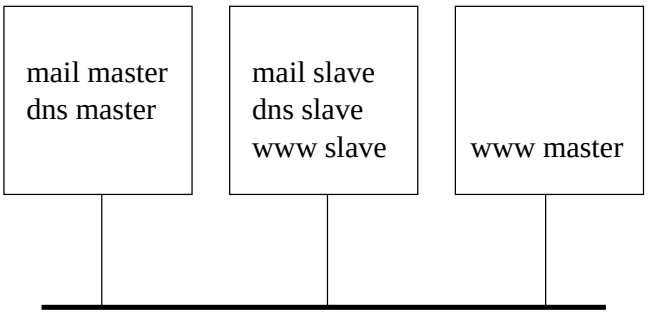
Failover Zustandsdiagramm Slave

Slide 27



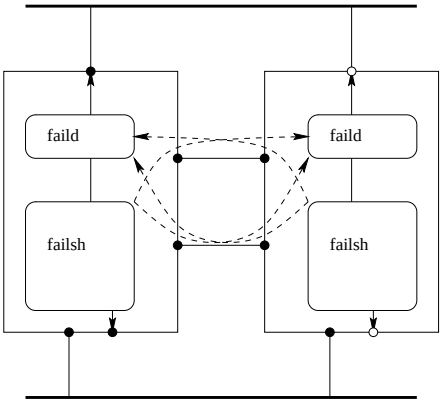
Failover Dreieckskonfiguration

Slide 28



Failover Firewall

Slide 29



Was braucht es in Zukunft?

Slide 30

1. **Überwachung von Anwendungen:** APIs oder Dienste, die die Entscheidung zum Failover und den Failover vereinfachen
2. **Single Master und Multi Master:** Anwendungen sollten auf Multi Master Replication für den Anwendungszustand vorbereitet werden
3. **Cluster Filesysteme:** Ansätze vorhanden in Network Block Device, DRBD, ODR. Produktionsqualität?
4. **Cluster Datenbanken:** Single Master Replication in MySQL in 3.23.15 Alpha. Konzeptionell am weitesten: Mariposa.
5. **Zustandslose Client-Server-Anwendungen**

Slide 31

Think Cluster!