Maps & Policy Delegierung

Ralf Hildebrandt

2. Mailserver Konferenz 2005
Magdeburg

Was sind Maps?

- Postfix benutzt Tabellen um Informationen schreiben und lesen zu können:
  - Zugriffskontrolle (access)
  - Adreßmanipulation (address rewriting)
  - und sogar für Filterung
- alle Tabellen beinhalten (Schlüssel, Wert)-Paare
- Telefonbuchanalogie
  - Schlüssel: Name
  - Wert: Telefonnummer

Was sind Maps?

- Das (Schlüssel, Wert) Interface verbirgt die Komplexität einer LDAP und SQL Abfrage vor Postfix
- Die ist ein gutes Beispiel für die Verbindung komplexer Systeme über einfache, klar definierte Schnittstellen

Wo werden Maps benutzt?

- Aliases:

```
alias_maps = hash:/etc/postfix/aliases
```

- Adreßmanipulation:

```
virtual_alias_maps = hash:/etc/postfix/virtual
```

- Routing:

```
transport_maps = hash:/etc/postfix/transport
```

- Inhaltsfilterung:

```
header_checks = regexp:/etc/postfix/header_checks
```

Was für Typen von Maps gibt es?

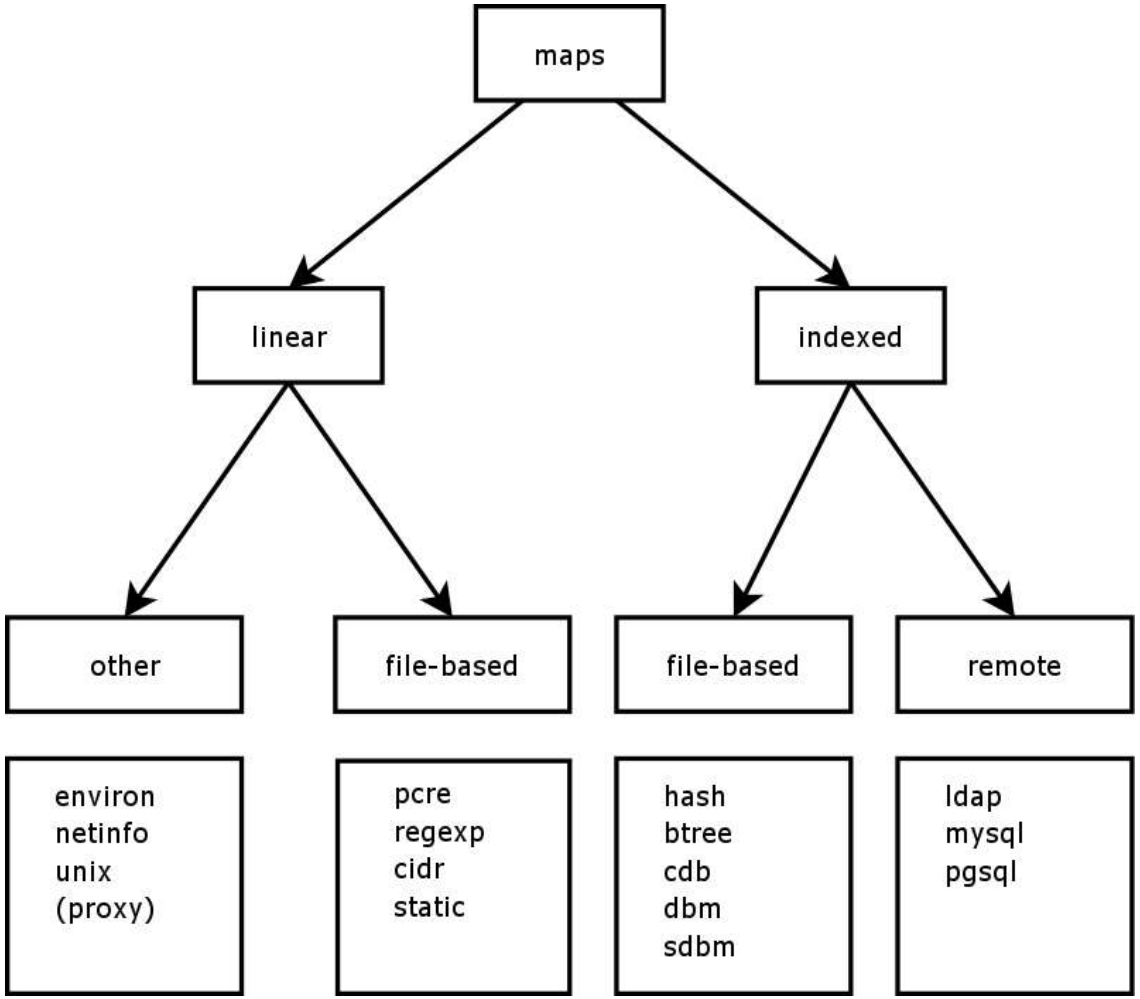
indizierte Maps

- `hash`
- `btree`
- `cdb`
- `*sql`
- `ldap`

Lineare Maps

- `pcre`
- `regexp`
- `cidr`
- `environ`
- `nis`
- `nisplus`

Übersicht über die Typen



Wozu *SQL oder LDAP?

- dateibasierte Maps werden geöffnet bevor der Daemon ins `chroot()` geht
- damit eine Änderung aktiv wird, muß der Daemon sich beenden – und liest beim Neustart die neuen Daten

Was passiert, wenn sich so eine Map oft ändert?

```
Mar  5 12:10:05 mail postfix/smtpd[10388]:\
table cdb:/etc/postfix/no_internet_mail has changed \
-- restarting
```

Wie funktioniert ein Lookup?

- Maps finden nur genaue Übereinstimmungen
- Wie unterschieden sich die Lookups in den verschiedenen Type von Maps?
  - linear
  - indiziert

Lookup in indizierten Maps

- Erzeuge neue Abfragen (basierend auf der Syntax aus den manual Seiten) durch Aufbrechen der Eingabe:
  - `user@sub.domain.tld` wird aufgespalten in `user` und `sub.domain.tld` Anteile
  - `sub.domain.tld` dann weiter in `domain.tld` und `tld`
- `parent_domain_matches_subdomains` steuert, ob der Eintrag `domain.tld` in einer Map auch für `sub.domain.tld` gilt (üblicherweise ist die Form dafür `.domain.tld`).

Lookups in linearen Maps

- jedes Muster wird auf die gesamte Eingabe angewandt.
- je nach Anwendungsbereich (Routing, Aliases, ...) ist die Eingabe:
  - ein ganzer Hostname,
  - eine ganze IP Adresse, oder
  - eine ganze Mailadresse

D.h. es wird **nicht** nach Elterndomänen oder Elternnetzwerken gesucht, Adressen der Form `user@sub.domain.tld` werden **nicht** in ihren User- und Domänenteil aufgespalten und auch "`user+foo`" wird nicht in `user` und `foo` zerlegt.

Wonach wird zuerst geschaut?

canonical/virtual

```
user@domain    address
user           address
@domain*       address
```

access

```
user@domain    OK
domain.tld     OK
user@          OK
```

***: Ersetze `user@site` durch address wenn `site` in `myorigin`, in `mydestination`, in `inet_interfaces` oder in `proxy_interfaces` ist.**

Lookup in indizierter `access(5)`

`Ralf.Hildebrandt@charite.de` soll in einer `access(5)` Map gesucht werden (z.B. indem man: `check_sender_access hash:/etc/postfix/sender_access` nutzt):

1. `Ralf.Hildebrandt@charite.de` als Schlüssel
2. `charite.de`
3. `.charite.de`
4. `de`
5. `.de`
6. `Ralf.Hildebrandt@`

Also brauchen wird für `Ralf.Hildebrandt@charite.de` zwischen 1 und 6 individuelle Anfragen an die Datenbank.

Lookup in regexp: access(5)

Nehmen wir diese Map mit regulären Ausdrücken:

```
/Mailserver 2005/      Rules!
/^ralf/                REJECT
```

- `Ralf.Hildebrandt@charite.de` wird in einer `access(5)` Map gesucht (z.B. wegen: `check_sender_access regexp:/etc/postfix/sender_access`)
- Paßt „`Ralf.Hildebrandt@charite.de`“ auf `/Konferenz 2005/`?
- Paßt „`Ralf.Hildebrandt@charite.de`“ auf `/^ralf/`?

Ja, denn die Loookups sind standardmäßig case-insensitiv!

Fallstricke

- Bitte keine `regexp/pcre` Maps nehmen, wo ordinäre `hash/btree` Maps ausreichend sind:
  `/^pattern@domain$/` in `regexp` ist äquivalent zu `pattern@domain` in `hash/btree/...`
- denn jede Zeile der `regexp/pcre` Map wird durchprobiert bis ein Treffer gefunden wird oder das Ende erreicht ist!
- Das ist auch der Grund für die massive CPU Nutzung durch cleanup bei großen Maps für `header_checks` oder `body_checks`.

Performance

- Diese Art Lookups zu machen kann Probleme mit sich bringen!
- hohe Latenz – `connect()`, `bind()`, Abfrage, Antwort
- hohe Parallelität – viele Postfix Prozesse stellen Abfragen gleichzeitig ans Backend
- eine DoS oder Wörterbuchattacke trifft das Backend voll

Lösungsansätze

- LDAP/*sql DB in eine dateibasierte Map exportieren um die Performance zu erhöhen
  - die Datenbank kann die hohe Parallelität nicht bewältigen
  - dateibasierte Map hat weniger Latenz
  - sofortige Updates braucht man bei Email selten
- `proxymap`
  - konsolidate die Anzahl der offenen Maps
  - konsolidiert die Anzahl gleichzeitiger Anfragen ans Backend

Was ist proxymap?

- `proxymap` für die Konsolidierung von Anfragen:
  - austricksen von `chroot()`
  - Konsolidierung der Anzahl offener Maps (eine offene Map wird von mehreren Prozessen genutzt)

- 1000 `smtpd` Prozesse fragen
- 1 Sql Server gleichzeitig

- 1000 `smtpd` Prozesse fragen
- 20 `proxymap` Prozesse, die dann
- 1 Sql Server gleichzeitig fragen

Nachteile!

- `proxymap` öffnet nur Maps die durch den Parameter `proxy_read_maps` freigegeben sind
- `proxymap` ist kein Daemon, dem Postfix vertraut, d.h. er darf nicht für sicherheitsrelevante Maps eingesetzt werden – wie z.B. User oder Group IDs, Datei- oder Verzeichnisnamen von Mailboxen oder externe Programme

Sie können ihn nicht für `aliases` benutzen.

Vorteile von CDB

CDB (Constant Data Base)

- Vorteile
  - indiziertes, dateibasiertes Dateiformat, entworfen von Daniel Bernstein.
  - CDB ist optimiert für Lesezugriff
  - garantiert, daß jeder Eintrag nach höchstens zwei Plattenzugriffen gelesen werden kann.

Das ist ja gar nicht so schlecht!

Nachteile von CDB

- inkrementelle Updates sind nicht möglich
- kein Einfügen oder Löschen einzelner Einträge!
- CDB Datenbanken können nur modifiziert werden, indem sie neu erzeugt werden. Daher auch “constant”.
- d.h. man kann sie nicht für die Maps benutzen, die geschrieben werden müssen, z.B. von `verify(8)`

Sie brauchen `tinycdb` (Version 0.5 oder aktueller) von Michael Tokarev:
http://www.corpit.ru/mjt/tinycdb.html

BerkeleyDB Pro & Contra

BerkeleyDB

- Vorteile
  - Sie ist schon installiert!
- Nachteile
  - BerkeleyDB version 4 wird erst ab Postfix 2.0 unterstützt
  - viele kommerzielle UNIXes haben keine BerkeleyDB
  - einige Linux Systembibliotheken nutzen BerkeleyDB, ebenso wie third-party Bibliotheken (SASL).

Nachteile BerkeleyDB II

- Wenn Sie Postfix mit einer anderen Berkeley DB Version kompilieren, dann wird jeder Postfix Daemon abstürzen, weil entweder die Systembibliothek, die SASL Bibliothek oder Postfix selber die falsche Version benutzt.

Und Sie dachten der Wechsel von Windows würde Sie vor der DLL Hölle erretten? Weit gefehlt!

- Aktuelle BerkeleyDB Versions haben eine configure Option "`--with-uniquename`", die die Symbolnamen eineindeutig macht, sodaß mehrere Versionen von BerkeleyDB nebeneinander koexistieren können.

LDAP

- Wichtig: Postfix unterstützt LDAP v1 nicht mehr

(siehe http://www.postfix.org/LDAP_README.html)

```
alias_maps = ldap:/etc/postfix/ldap-aliases.cf
```

und in /etc/postfix/ldap-aliases.cf haben Sie:

```
server_host = ldap.my.com
search_base = dc=my, dc=com
```

LDAP II

Wenn Postfix Mail für die lokale Adresse "ldapuser" empfängt, passiert folgendes:

- Postfix wird Verbindung zum LDAP Server auf Port 389 `ldap.my.com` aufnehmen
- sich anonym binden (bind anonymously),
- nach Einträgen suchen, deren `mailacceptinggeneralid` Attribut "`ldapuser`" ist,
- das "`maildrop`" Attribut der gefundenen auslesen und eine Liste derer Maildrops anlegen,
- die als RFC822 Adresses behandelt werden
- an die die Mail dann zugestellt wird.

LDAP III

- Sie können die Anzahl paralleler ldap Abfragen durch Postfixs `proxymap(8)` Daemon reduzieren
- Die Daten sind da draußen...
- alles was Sie brauchen ist die korrekte Abfrage :)

*SQL

- der mysql Maptyp erlaubt es Postfix, eine MySQL Datenbank abzufragen
- Mehere mysql Datenbanken können genutzt werden:
  - eine für eine `virtual(5)` Map,
  - eine andere für eine `access(5)` Map,
  - und eine ganz andere für `aliases(5)`
  - ...wenn Sie wollen
- Sie können mehrere Server für dieselbe Datenbank angeben, sodaß Postfix einen funktionierenden Server abfragen kann, wenn ein anderer ausgefallen ist.

mySQL Performance

- Beschäftigte Mailserver mit mysql Maps erzeugen viele gleichzeitige mysql Anfragen, sodaß der oder die mySQL Server korrekt dimensioniert sein müssen!
- Sie können die Anzahl der parallelen Anfragen auch hier wieder durch Postfixs `proxymap(8)` Daemon verringern.
- `http://www.postfix.org/MYSQL_README.html`

postgresql

- Dasselbe in grün

Policy Delegierung

Ab Version 2.1 kann Postfixs `smtpd` policy Entscheidungen an einen Server delegieren der außerhalb von Postfix läuft, z.B.:

- `postgrey` von David Schweikert
  http://isg.ee.ethz.ch/tools/postgrey/
- SPF Policy Server von Meng Weng Wong
  http://spf.pobox.com/

Aber wozu?

- Der Entscheidungsträger kann auf einer anderen Maschine laufen
- er kann Entscheidungen anhand aller Attribute treffen, anstatt nur eines
- Perl oder Python schreiben sich leichter als C

Das `policyd` Protokoll ist einfach

- Der Client sendet eine Anfrage als Folge von `name=value` Attributen, getrennt durch Zeilenumbrüche und terminiert durch eine Leerzeile.
- Die Antwort des Servers ist ein `name=value` Attribut und auch diese wird durch eine Leerzeile terminiert.

policyd Beispiel

```
request=smtpd_access_policy
protocol_state=RCPT
protocol_name=SMTP
helo_name=some.domain.tld
queue_id=8045F2AB23
sender=foo@bar.tld
recipient=bar@foo.tld
client_address=1.2.3.4
client_name=another.domain.tld
instance=123.456.7
sasl_method=plain
sasl_username=you
sasl_sender=
ccert_subject=solaris9.porcupine.org
ccert_issuer=Wietse Venema
ccert_fingerprint=C2:9D:F4:87:71:73:73:D9:18:E7:C2:F3:C1:DA:6E:04
size=12345
[Leerzeile]
```

Beispiel II

- Der policy Server darf jede Aktion zurückgeben, die in einer `smtpd access(5)` Map gültig ist.
- Beispiel:

```
action=450 Service temporarily unavailable
[Leerzeile]
```

- Bei Problemen darf der policy Server keine Antwort senden!
- Stattdessen muß er eine Warnung loggen und die Verbindung abbauen
- Postfix versucht es später nochmal

lokal und übers Netz

Postfix kann den policy Server über einen TCP Socket oder einen UNIX-Domain Socket kontaktieren.

Beispiele:

```
check_policy_service inet:127.0.0.1:9998
check_policy_service unix:/some/where/policy
check_policy_service unix:private/policy
```

master.cf und main.cf

- master.cf:

```
policy  unix  -  n  n  -  -  spawn
  user=nobody
  argv=/usr/bin/perl /etc/postfix/postfix-policyd.pl
```

- main.cf:

```
    smtpd_recipient_restrictions =
      ...
      check_policy_service unix:private/policy
```

Solaris UNIX-domain Sockets sind unzuverlässig...

Nehmen Sie stattdessen TCP Sockets

Selbst ist der Mann?

- Grundgerüst für einen policy Server in Perl (schon wieder) von Michael Tokarev:
  http://www.corpit.ru/mjt/smtpd-policy-template.pl

Fragen

Jetzt die Fragen stellen!

Quellen

- Postfix Webseite
  http://www.postfix.org/
- The Book of Postfix
  http://www.postfix-book.com/