SpeedPartner

Web-Performance-Optimierung mit varnish

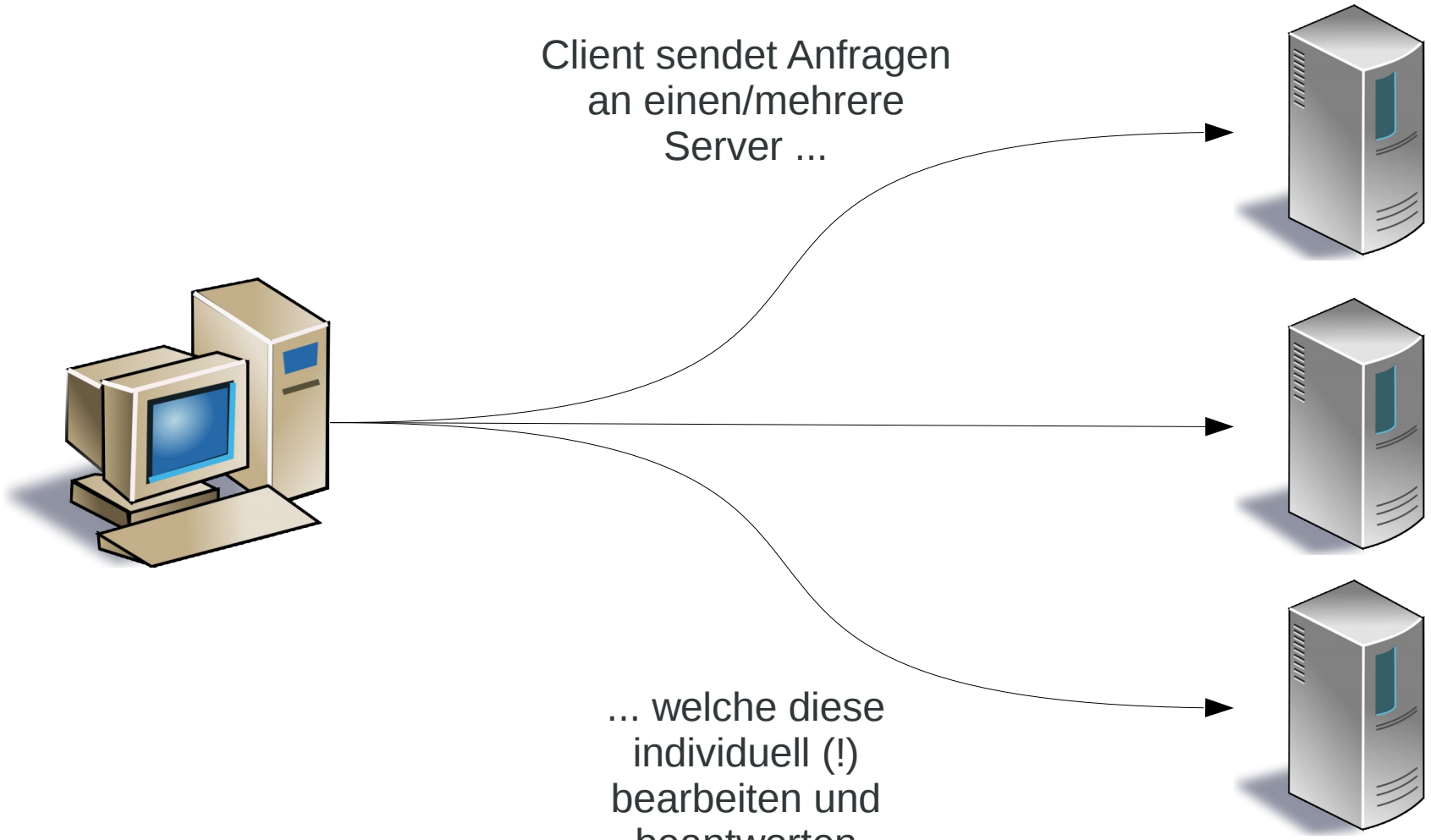
Aufbau / Ziele:

- Einführung / Überblick
- Varnish-Tools (Administration, Logging und Statistik)
- Konfigurationssprache
- Konfigurationsbeispiele aus der Praxis
- Edge-Side-Includes
- TYPO3: Automatische Löschung Varnish-Cache bei Änderungen
- Links / Hilfen

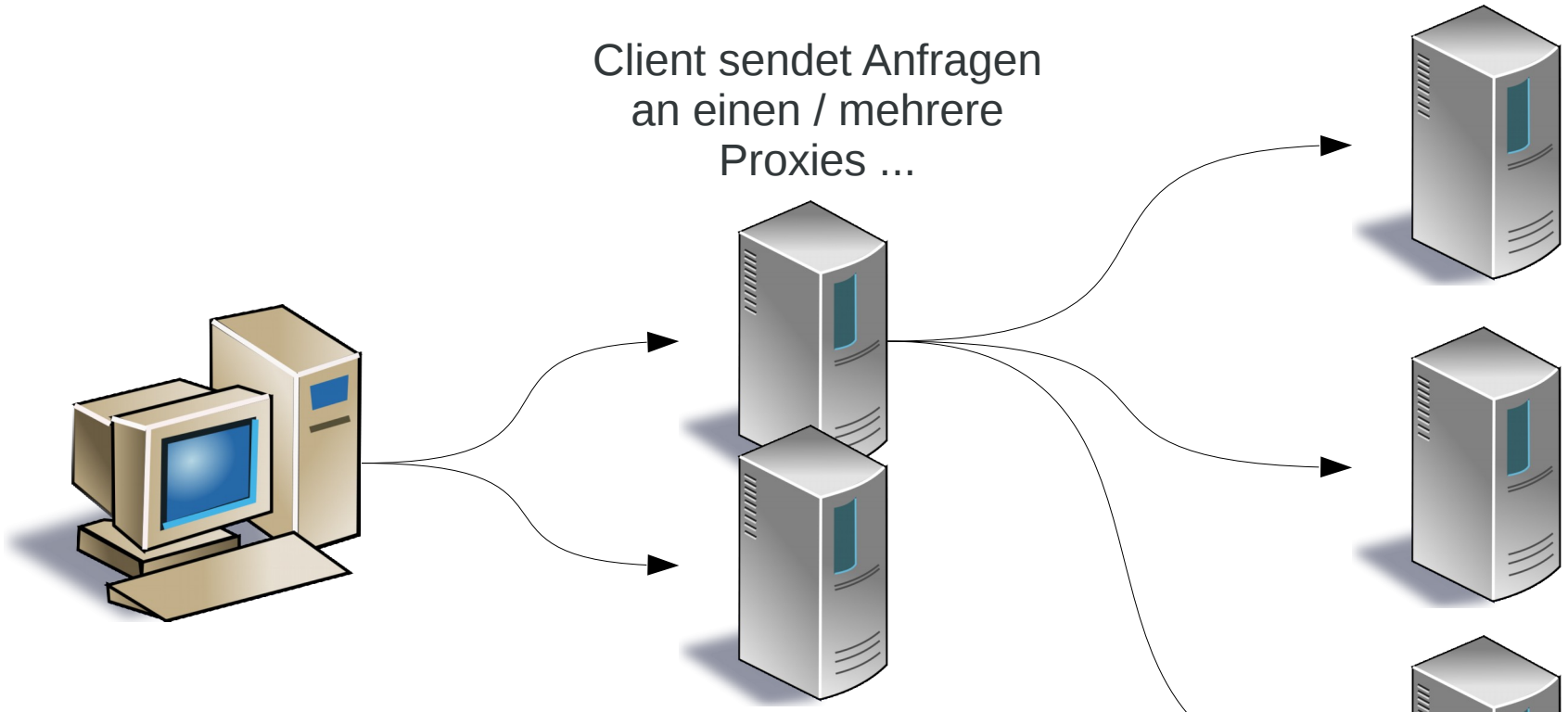
Über mich

- Stefan Neufeind
- Mit-Geschäftsführer der SpeedPartner GmbH aus Neuss
  ein Internet-Service-Provider (ISP)
  - Individuelle TYPO3-Entwicklungen
  - Hosting, Housing, Managed Services
  - Domains / Domain-Services
  - IPv6, DNSSEC, ...
- Aktive Mitarbeit im Community-Umfeld (PHP/PEAR, TYPO3, Linux)
- Freier Autor für z.B. t3n, iX, video2brain, ...

Klassischer Fall für eine direkte http-Anfrage



Eine http-Anfrage per Proxy



... welche diese nach Regeln

- selbst beantworten
- weiterleiten
- bearbeiten
- zwischenspeichern

Arten von Web-Proxies

Web-Proxy in Client-Nähe bzw. „im Netz“:

- Zwischenspeicherung (Cache)
- Authentifizierung
- Filterung
  - Zugriffssteuerung
  - „Werbefilter“
  - Virenscanning
  - SSL-Terminierung:
    Ermöglichen von Prüfung verschlüsselter Inhalte

z.B. Squid („leistungsstarker Klassiker“)

Web-Proxy in Server-Nähe:

- Zwischenspeicherung (Cache)
- Entlastung Server
  - Client-Verbindungen (z.B. Keepalives)
  - Selbständige Bearbeitung von Anfragen aus dem Cache
  - Vermeidung gleichartiger paralleler Anfragen an die Webserver
- Verteilung über mehrere Server
  - Round-Robin, Random, ...
  - In Abhängigkeit von Anfragen (URL, Cookies, Client-IP, ...)
- Bearbeitung der Anfragen
- Bearbeitung der Antworten
  - z.B. Edge-Side-Includes
- SSL-Terminierung ← Nicht als direkter Teil von Varnish!

z.B. Varnish (optimiert als Reverseproxy)

Stärken von Varnish:

- Konzeptioniert als Reverseproxy
- Optimiert auf Durchsatz / Last
- Minimalistischer, fokusierter Funktionsumfang
- Flexible Konfiguration / komplexe Regelwerke möglich
- Aktive, selektive Cache-Leerung durch Applikationen möglich
  - Erlaubt bei Änderungen Erneuerung bestimmter Inhalte anhand URLs, Header-Zeilen, ...
- Zusammenstellen von Antworten aus Teilen (Edge-Side-Includes [ESI])

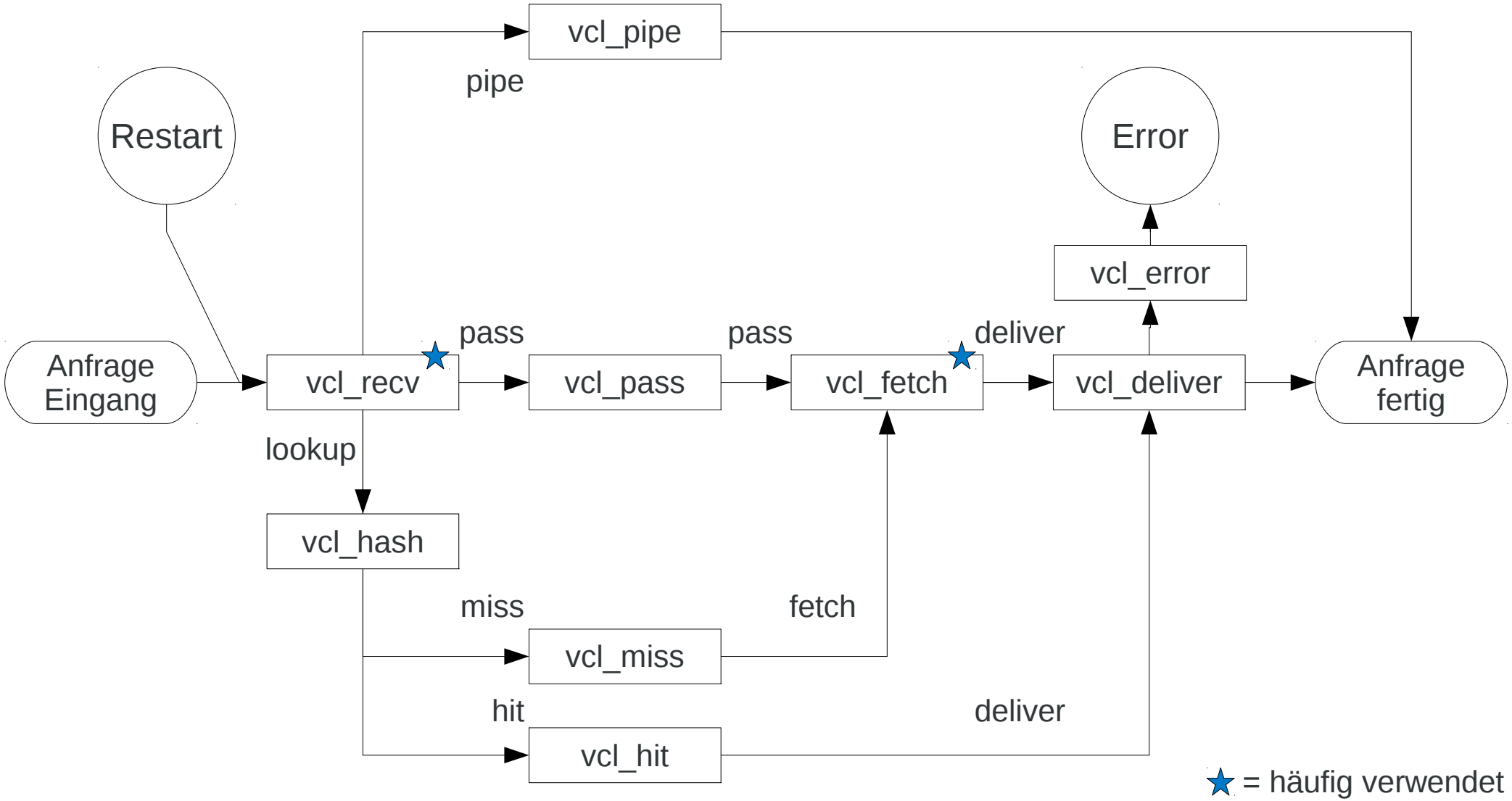
Konfiguration per Varnish Configuration Language (VCL):

- Regelbasierte Bearbeitung von Anfragen
- Domain-spezifische Sprache statt reiner „Konfiguration“
- Interne Übersetzung in Binärcode für optimierte Bearbeitung von Anfragen
- Arbeit mit Objekten und deren Eigenschaften:
  - Anfrage, Antwort, Cache-Objekt
- Zeilenweise Ausführung von Bedingungen / Zuweisungen
- Aufgeteilt in Subroutinen für die verschiedenen Phasen einer Bearbeitung
- Standard-Logik je Subroutine, falls nicht anders definiert oder falls kein Rücksprung vor Ende der Subroutine erfolgt

Minimales Konfigurationsbeispiel:

```
backend default {
  .host = "127.0.0.1";
  .port = "8080";
}
```

Basis-Verlauf einer Anfrage in VCL:



Unterschiedliche Backends je Inhalt + nicht cachebare Inhalte:

- Backends definieren und Regeln für Anfrage / Antwort auswerten

```
backend default {
  .host = "127.0.0.1";
  .port = "82";
}
backend static {
  .host = "127.0.0.1";
  .port = "81";
}
```

Dev hier bewusst nicht gecached

```
sub vcl_recv {
  if (req.http.host == "dev.example.com") {
    set req.backend = default;
    return (pass);
  }
  if (req.url ~ "^/[^?]+\.(jpeg|jpg|png|gif|js|css)$") {
      set req.backend = static;
  } else {
      set req.backend = default;
  }
}
```

vcl_recv = direkt nach Eingang der Anfrage
vcl_fetch = nach Abholen eines Inhalts vom Backend (nicht gecached)

req = Request-Objekt (Anfrage)
obj = Cache-Objekt (Inhalt)

```
sub vcl_fetch {
  if (req.http.host == "dev.example.com") {
    set obj.cacheable = false;
    return (pass);
  }

                ... weitere Bedingungen ...

  if (!obj.cacheable) {
    set obj.http.X-Cacheable = "not cacheable";
    return (pass);
  } else {
    set obj.http.X-Cacheable = "yes";
    return (deliver);
  }
}
```

Anfragen „bereinigen“:

- Nicht cache-relevante Teile einer Anfrage entfernen (1/2)

```
sub vcl_recv {
  # Google Analytics Cookies entfernen
  set req.http.cookie = regsub(req.http.cookie,"__utma=[^;]*;?( |$)","");
  set req.http.cookie = regsub(req.http.cookie,"__utmb=[^;]*;?( |$)","");
  set req.http.cookie = regsub(req.http.cookie,"__utmc=[^;]*;?( |$)","");
  set req.http.cookie = regsub(req.http.cookie,"__utmz=[^;]*;?( |$)","");

  # alternativ: alle Cookies ausser zwei ...
  set req.http.Cookie = ";" req.http.Cookie;
  set req.http.Cookie = regsuball(req.http.Cookie, "; +", ";");
  set req.http.Cookie = regsuball(req.http.Cookie, ";(PHPSESSID|fe_typo_user)=", "; \1=");
  set req.http.Cookie = regsuball(req.http.Cookie, ";[^ ][^;]*", "");
  set req.http.Cookie = regsuball(req.http.Cookie, "^[; ]+|[; ]+$", "");

                                    ... weitere Regeln ...

  # Remove the cookie header if it's empty after cleanup
  if (req.http.cookie ~ "^ *$") {
     remove req.http.cookie;
  }
```

Anfragen „bereinigen“:

- Nicht cache-relevante Teile einer Anfrage entfernen (2/2)

```
# Remove cookies and query string for real static files
if (req.url ~ "^/[^?]+\.(jpeg|jpg|png|gif|ico|js|css|txt|gz|zip|lzma|bz2|tgz|tbz|swf|f4v)(\?.*|)$")
{
   remove req.http.cookie;
   set req.url = regsub(req.url, "\?.*$", "");
}

# Normalize Content-Encoding
if (req.http.Accept-Encoding)
{
    if (req.url ~ "\.(jpg|png|gif|gz|tgz|bz2|lzma|tbz)(\?.*|)$") {
        remove req.http.Accept-Encoding;
    } elsif (req.http.Accept-Encoding ~ "gzip") {
        set req.http.Accept-Encoding = "gzip";
    } elsif (req.http.Accept-Encoding ~ "deflate") {
        set req.http.Accept-Encoding = "deflate";
    } else {
        remove req.http.Accept-Encoding;
    }
 }
}
```

Cache-Zeiten in Varnish abweichend von Angaben für Client setzen:

```
sub vcl_fetch {
  if (beresp.cacheable) {
    /* Expires (absolute Zeitangabe) entfernen */
    unset beresp.http.expires;
    /* Stattdessen relative Gültigkeit setzen */
    set beresp.http.cache-control = "max-age=900";
    /* Haltezeit in Varnish */
    set beresp.ttl = 1w;
    /* Marker für Verarbeitung in vcl_deliver */
    set beresp.http.magicmarker = "1";
  }
}

sub vcl_deliver {
  if (resp.http.magicmarker) {
    /* Marker entfernen */
    unset resp.http.magicmarker;
    /* Bei Abruf durch den Client ist das Objekt immer „frisch“ */
    set resp.http.age = "0";
  }
}
```

Objekt vom Backend geholt; zur Ablage im Cache

beresp = Backend-Response

Objekt zur Auslieferung; vom Backend oder aus dem Cache

resp = Response zum Client

Quelle: https://www.varnish-cache.org/trac/wiki/VCLExampleLongerCaching

Selektive Cache-Löschung:

- Möglichkeit 1: per Varnish-Admin-Port
  - z.B. mittels Hilfsmittel „varnishadm“
  - Authentifizierung über einen „shared secret“
  - Verwendet eine TCP-Klartext-Verbindung für die Kommandos (zzgl. Authentifizierung)

```
# einzelne Seite
varnishadm -T 127.0.0.1:6082 -S /etc/varnish/secret ban.url ^/kontakt.htm$

# ganzes Verzeichnis und nur bestimmer Hostname
varnishadm -T 127.0.0.1:6082 -S /etc/varnish/secret "ban req.url ~ ^/somedirectory/ &&
    req.http.host == www.example.com“
```

Varnish 2.x: purge
Varnish 3.x: ban

Selektive Cache-Löschung:

- Möglichkeit 2: Verwendung von http-Headerzeilen in Antworten sowie VCL
  - Verwendung für Tagging von Seiten
  - Verwendung für tag-basierte Cache-Löschung durch http-Antwort und etwas VCL

```
<?php
  header('x-invalidated-by: tag-a,tag-b', false);
  header('cache-control: s-maxage=86400');
  // ... reguläre Ausgaben ...
```

Tags setzen

```
<?php
  header('x-invalidates: tag-a', false);
  // ... reguläre Ausgaben ...
```

In Antwort Cache-Lösung triggern, z.B. per Formular mit POST

```
sub vcl_fetch {
     if (beresp.status >= 200 && beresp.status < 400
    && (req.request == "PUT" || req.request == "POST" ||
        req.request == "DELETE")) {
        ban("obj.http.x-invalidated-by ~ " + beresp.http.x-invalidates);
    }
}
```

VCL

In Anlehnung an: http://blog.kevburnsjr.com/tagged-cache-invalidation

Selektive Cache-Löschung:

- Möglichkeit 3: Verwendung von http-Requests sowie VCL

```
acl purge_acl {
    "localhost";
    "192.168.1.1";
}

sub vcl_recv {
    if(req.request == "PURGE") {
        if(!client.ip ~ purge_acl) {
            error 405 "Not allowed";
        } else {
            ban_url(req.url);
            error 200 "Purged";
        }
    }
}
```

PURGE nur von bestimmten Clients zulassen

Request-Typ beliebig, jedoch ist PURGE „üblich“

Angefragte URL clearen. Evtl. auch Hostname, Wildcards oder zusätzliche Header berücksichtigen

```
<?php
  $curl = curl_init("http://www.example.com/pageToPurge");
  curl_setopt($curl, CURLOPT_CUSTOMREQUEST, "PURGE");
  curl_exec($curl);
```

Anfrage mit selbst gewähltem Typ an varnish schicken

Varnish-Logdaten:

- Logging erfolgt in einen shared-memory-Bereich
- Zugriff per Tool „varnishlog“
- Daemon-Betrieb möglich, welcher dann binäre Logdateien zur späteren Auswertung schreibt
- Logdaten für Kommunikation varnish mit Backend und / oder Client
- Durch shared-memory auch zeitgleicher Zugriff mehrerer Tools auf Livedaten möglich

Varnish-Boardmittel für Logging und Statistiken: (1/2)

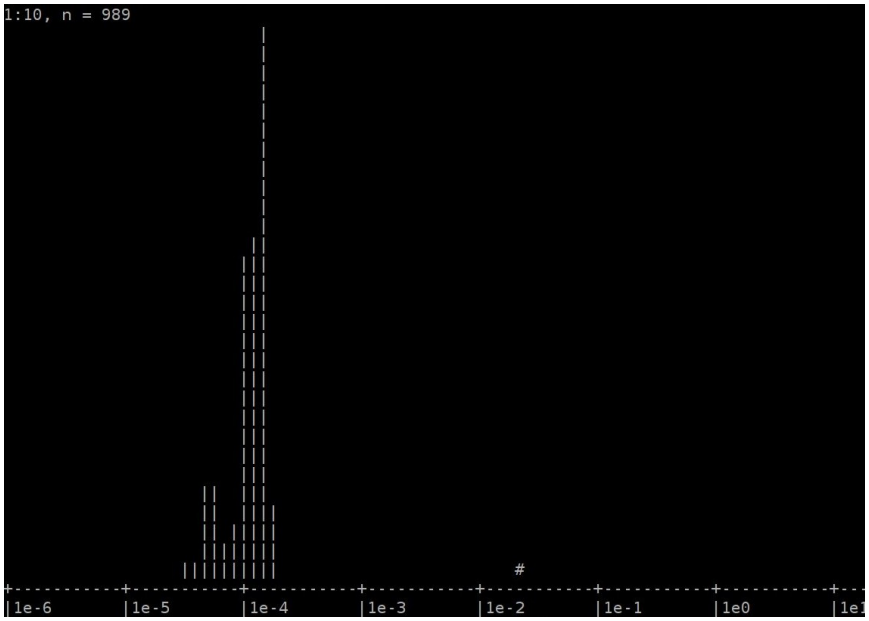
- varnishlog
  - Anzeige oder binäres Logging
  - für alle Anfrage- und Antwort-Header
  - zu Backend und / oder Client

- Varnishncsa
  - Logging in NCSA-/Apache-kompatiblem Text-Format
  - Kann z.B. für klassische Besucherstatistik-Werkzeuge o.ä. Verwendet werden

- varnishtop
  - Auswertung Headerdaten nach Häufigkeit (häufigste URLs, Anfragetypen, Client- oder Backend-Merkmale, ...)

Varnish-Boardmittel für Logging und Statistiken: (2/2)

- varnishhist
  - Histogram-Darstellung
  - Verteilung Anfragen (Y-Achse)
    nach Antwortzeit (X-Achse, logarithmisch)
  - Pipe-Symbol: gecachte Anfrage,
    Raute: Backend-Anfrage



- varnishstat
  - Statistische Auswertung verschiedener Kriterien,
    insb. Auch Cache-Hit-Ratio

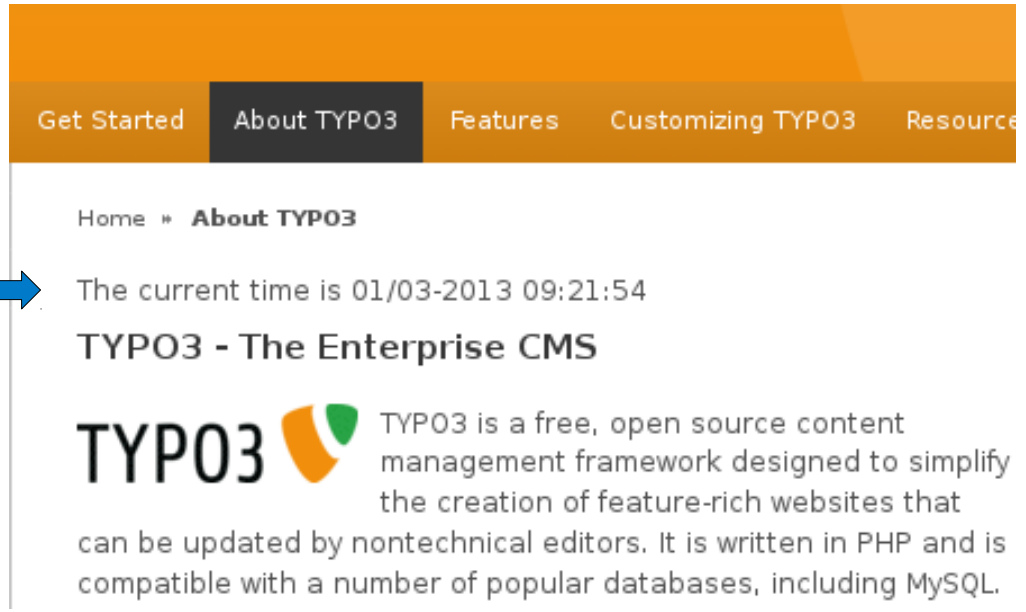
```
Hitrate ratio:        10      100      103
Hitrate avg:      0.9480   0.9240   0.9240

      41425        23.00         8.60 Client connections accepted
     176411        48.00        36.62 Client requests received
     152218        43.00        31.60 Cache hits
       1550         0.00         0.32 Cache hits for pass
      16516         3.00         3.43 Cache misses
      17306         4.00         3.59 Backend conn. success
       7136         1.00         1.48 Backend conn. reuses
        269         0.00         0.06 Backend conn. was closed
       7406         1.00         1.54 Backend conn. recycles
         45         0.00         0.01 Fetch head
       8036         1.00         1.67 Fetch with Length
      16089         5.00         3.34 Fetch chunked
        241         0.00         0.05 Fetch wanted close
         24         0.00         0.00 Fetch zero len
        147            .            .   N struct sess_mem
         72            .            .   N struct sess
       6611            .            .   N struct object
       5530            .            .   N struct objecthead
      15402            .            .   N struct smf
       2263            .            .   N small free smf
```

Einbindung von Blöcken:

- Cachen von Antworten mit speziellen ESI-Tags
  - separat cachebar

```
<esi:include src="/?id=51&type=978&
key=INT_SCRIPT.e6e3f4bc683c3ff8f57f2f46ab
8f9a80&identifier=87b909c18277ab58a6006d2b
7d96e5df&from_varnish=1" />
```



- Auswertung von Einbindungs-Anweisungen durch varnish
- Client erhält Antwort inkl. Ersetzungen

- Ersetzungen müssen in Varnish aktiviert werden (per VCL)

```
sub vcl_fetch {
  #Respect force-reload
  if (req.http.Cache-Control ~ "no-cache") {
    set beresp.ttl = 0s;
    #Make sure ESI includes are processed!
    set beresp.do_esi = true;
    return (deliver);
  }
```

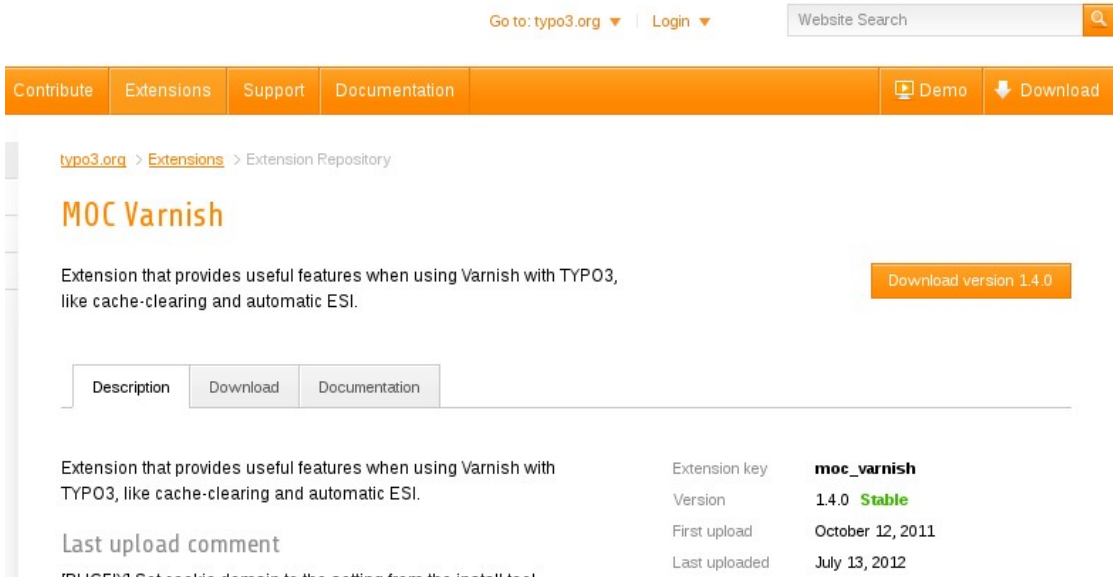
varnish 2.1: esi;

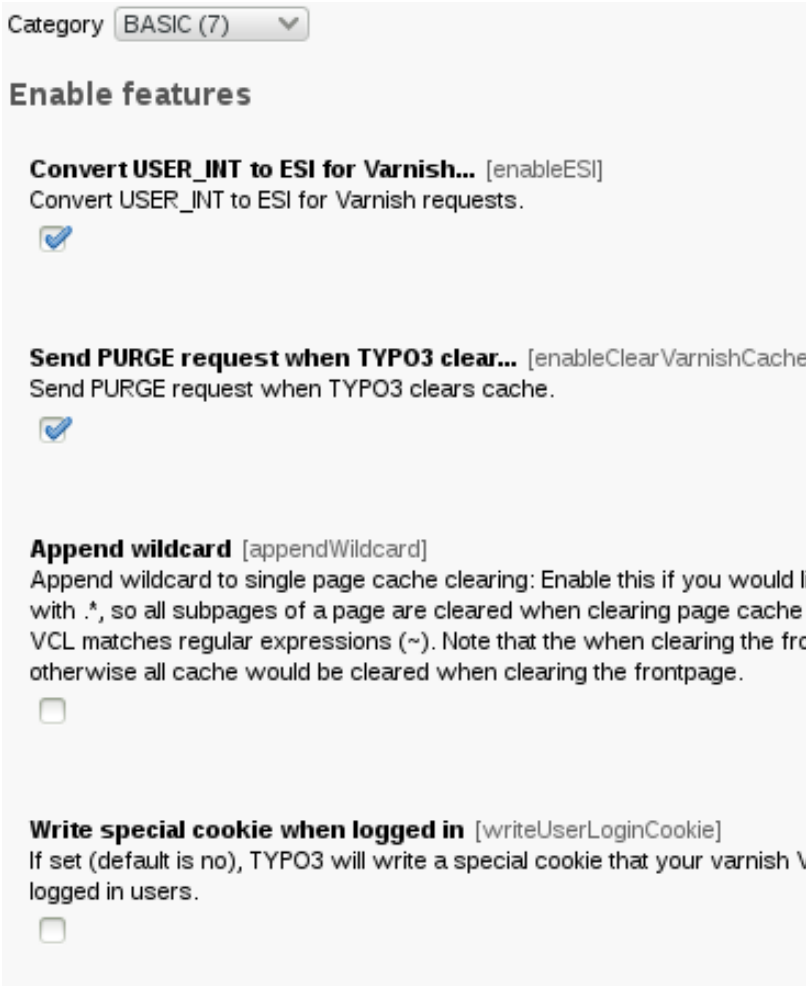
```
  //Force cache for 5 seconds for ESI responses
  if (beresp.http.X-ESI-RESPONSE) {
    set beresp.ttl = 0s;
  } else {
    set beresp.ttl = 24h;
  }
  return (deliver);
}
```

TYPO3: Cache-Löschung bei Änderungen

TYPO3-Extension „MOC Varnish“:

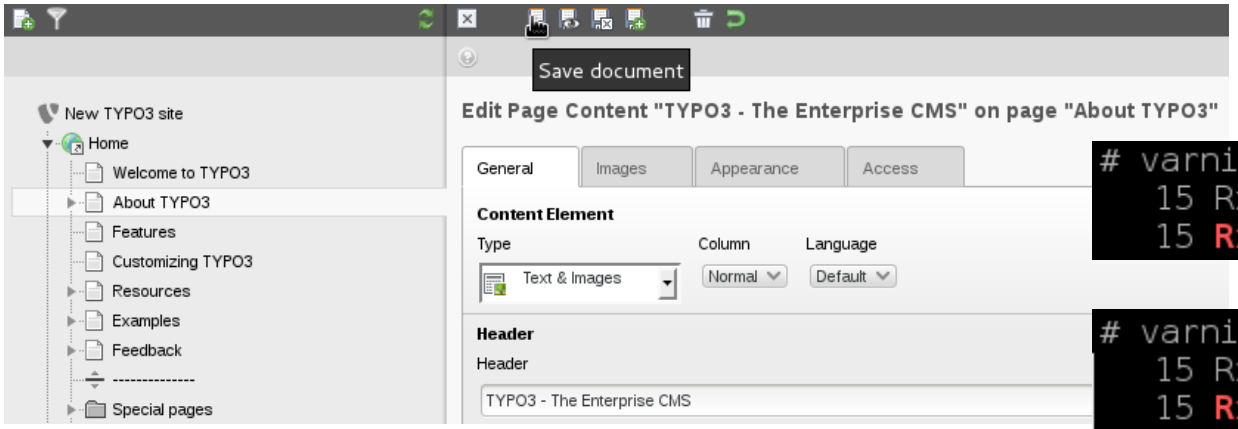
- Rüstet varnish-Unterstützung mit einfachen Mitteln nach
  - automatischen PURGE-Requests bei Änderungen
    → Änderungen trotz langer Cache-Haltezeiten zeitnah sichtbar
  - automatische Umwandlung von USER_INT-Objekten (per Definition nicht cachebar) in ESI-Statements
    → Cachung umgebender Seiten





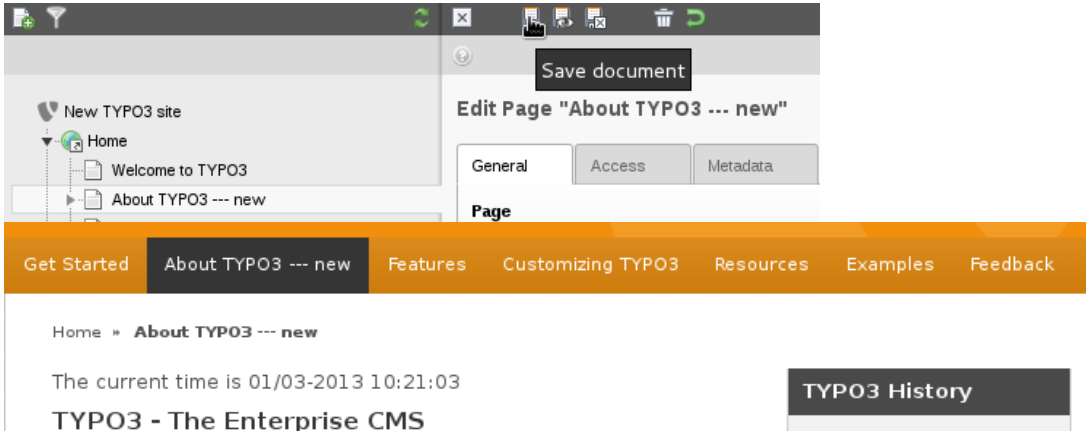
TYPO3-Extension „MOC Varnish“:

- automatischen PURGE-Requests bei Änderungen
  - Bei Änderung an Inhalt Cache der Seite erneuern




```
# varnishlog -c | grep -E PURGE\|RxURL
   15 RxRequest    c PURGE
   15 RxURL        c /about-typo3/
```

  - Bei Änderung an Seite wegen Navigation Aktualisierung dieser und anderer Seiten




```
# varnishlog -c | grep -E PURGE\|RxURL
   15 RxRequest    c PURGE
   15 RxURL        c /get-started/
   15 RxRequest    c PURGE
   15 RxURL        c /about-typo3/
   16 RxRequest    c PURGE
   16 RxURL        c /features/
   15 RxRequest    c PURGE
   15 RxURL        c /customizing-typo3/
   15 RxRequest    c PURGE
   15 RxURL        c /resources/
   15 RxRequest    c PURGE
   15 RxURL        c /examples/
   15 RxRequest    c PURGE
   15 RxURL        c /feedback/
   15 RxRequest    c PURGE
   15 RxURL        c /
```

Fazit

- Vielfältige Möglichkeiten
- Komplexes Regelwerk möglich
- Logs beobachten!
  - Effektiv gecached?
  - Nur gewünschte Inhalte gecached?
    (Cache-relevante Merkmale, ...)
- Zusammenarbeit mit / Einfluss auf zu cachende Applikationen vorteilhaft
- Tipp: In Snippets / Erfahrungen anderer „Inspiration“ suchen

Links / Hilfen

- Migration Varnish 2.1 auf 3.0 (Syntax-Änderungen)
  https://www.varnish-cache.org/docs/3.0/installation/upgrade.html

- VCL-Beispiele / Snippets
  https://www.varnish-cache.org/trac/wiki/VCLExamples

- TYPO3-Extension für Varnish-Anbindung
  http://typo3.org/extensions/repository/view/moc_varnish

Danke fürs Zuhören

sowie

viel Erfolg und Spaß

„auf der Überholspur“ :-)

Link zu den Slides: http://talks.speedpartner.de/

Bei Fragen stehen wir selbstverständlich gerne zur Verfügung:

Stefan Neufeind, neufeind@speedpartner.de
SpeedPartner GmbH, http://www.speedpartner.de/