VarnishQ Kurz-Referenz

March 12, 2015

1 Starten und Stoppen

VarnishQ befindet sich auf unseren Caches unter `/web/apps/varnishq` und kann in diesem Verzeichnis mit `./varnishq` gestartet werden. Beenden erfolgt mit `(quit)`.

2 Requests

Requests sind eine *Folge von Datensätzen* der Form `TAG CODE DATA`. Dabei ist

- `TAG` eines der Varnish-Tags wie RXURL, RXHEADER, TXSTATUS usw. *Eindeutige* TAGs kommen pro Request nur einmal vor (wie `RXURL`), *mehrfache* beliebig oft (wie `RXHEADER`)
- `CODE` hat den Wert `B` für *Backend-Request* (Varnish an Backend) oder `C` für *Client-Request* (Client an Varnish).
- `DATA` Der Datenbereich. Er enthält einen String.

Beispiel:
`TXURL B /cda/fragment/teaser/342987329`
Hier ist der TAG `TXURL`, der *CODE* `B` und `/cda/..` das *DATA* -Element.

3 Datenquellen

Datenquellen liefern eine *Folge von Requests*. Es gibt drei *primäre* Datenquellen

1. `(live)` liefert die auf dem Server aktuell verarbeiteten Requests
2. `(file "dateiname")` liefert die Requests aus einem file, das mit `varnishlog > dateiname` auf der Konsole erstellt wurde.
3. `(vlog "dateiname")` dekodiert die Requests aus den täglichen binären Mitschnitten unter `/web/logfiles/varnish/`. Pfade müssen voll angegeben werden.

4 Requestauswahl mit `where`

Die Requestauswahl erfolgt mit `(where` *<where-bedingung>* *<req-quelle>* `)`. Das Ergebnis von `(where..)` ist wiederum eine *<req-quelle>*. Jedes eindeutige Tag kann in der Bedingung direkt verwendet werden:

```
(where (contains rxurl "/a-") (live))
```

liefert eine neue Request-Quelle, die diejenigen Requests enthält, bei denen der RXURL-Satz im DATA-Bereich den String `/a-` enthält (Zugriffe auf DW-Artikel). Bedingungen können mit `(AND..)`, `(OR..)` und `(NOT ..)` verbunden werden. Für mehrfache TAGs gibt es den Quantor `(ANY..)`. Um allein das *Vorkommen* eines bestimmten Tags zu prüfen, kann es einfach genannt werden:

$\Longrightarrow$ `(where (and hit (any (contains rxheader "akamai"))) (live))`.

Hier wurde nach Req. gefragt, die das Tag HIT enthalten und bei denen irgendeine der Headerzeilen den String `akamai` enthält. Für die (mehrfachen) Header-Tags `RXHEADER`, `TXHEADER` und `OBJHEADER` gibt es eigene Funktionen (`rxheader` <*string*>), (`txheader` <*string*>) und (`objheader` <*string*>):

$\Longrightarrow$ `(where (contains (rxheader "Host") "thebobs") (live)).`

Liefert alle Requests an die Bobs. Reguläre Ausdrücke (Perl-kompatibel) werden mit (`matches` <*regex*> <*datum*>) angewendet. Der `\` muss gedoppelt werden, also `\\d+` statt `\d+` für eine Folge von Ziffern.

5 Requests ausführen

Eine Datenquelle kann mit (`dmp ..`), (`dump ..`) oder (`take` *n* <*quelle*>) befragt werden. Handelte es sich um eine Query, so wird sie „ausgeführt".

$\Longrightarrow$ `(dmp (where (contains rxurl "/s-") (live)))`

6 Zeilenauswahl mit `select`, `select-tag` und `select-tags`

`select-tag` liefert nur einen bestimmten Datensatz:

$\Longrightarrow$ `(dmp (select-tag 'rxurl (live)))`

gibt nur RXURL aus. Das Häkchen ist obligatorisch. Bei Backend-Requests erfolgen nur Leerzeilen, da diese kein `RXURL` haben. Der Output wird schöner mit:

$\Longrightarrow$ `(dmp (select-tag 'rxurl (where (type-is 'c) (live))))`

`select-tags` liefert mehrere Datensätze:

$\Longrightarrow$ `(dmp (select-tags '(rxurl reqend) (live)))`

`select` wählt aufgrund einer Bedingung aus. Diese darf die sich auf die Datenelemente TAG CODE und DATA beziehen.

$\Longrightarrow$ `(dmp (select (contains data "english") (live))).`

Auch hier darf mit `and`, `or` und `not` gearbeitet werden. Alle Datensätze, die „english" enthalten oder deren Tag `RXURL` lautet:

$\Longrightarrow$ `(dmp (select (or (eq tag 'rxurl) (contains data "english")) (live))).`

Dabei dient `(eq ...)` dem Tag-vergleich („equal").

`select`, `select-tag` und `select-tags` liefern ihrerseits *Datenquellen*. Also können sie gemeinsam mit where in belibiger verschachtelt werden:

$\Longrightarrow$ `(dmp (select-tag ... (where ..(select .. (where.. )))))`

7 Datenisolierung mit `nth-num`, und `numb`

(`nth-numb` <*n*> <*string*>) isoliert die *n*-te Zahl aus einem String. Die Verabeitungsdauer eines Requests kann mit

$\Longrightarrow$ `(dmp (where (> (nth-num 4 reqend) 2) (live)))`

liefert die Requests, die länger als 2 Sekunden Render-Zeit hatten.(Die vierte Zahl von `reqend` ist die Laufzeit).

(`numb` <*regex*> <*string*>) liefert eine Zahl aufgrund eines Patterns. Zum Beispiel ist

$\Longrightarrow$ `(numb "max-age=(\\d+)" (txheader "Cache-Control"))`

die max-age-Angabe des Servers.

8 Auswertungen mit `chart` und `avg`

Zur Erstellung eines Charts gibt es (`chart` <*anzahl-sätze*> <*länge*> <*ausdruck*> <*quelle*>).

$\Longrightarrow$ `(chart 1000 10 (rxheader "Host") (live))`

erstellt ein Chart der zehn häufigsten Vorkommen der Hostnamen aus den 1000 nächsten Zugriffen auf das Life-System. Durchschnitte werden mit (`avg..`) berechnet. Die durchschnittliche Verarbeitungszeit von 100 Requests, die kein Hit sind, findet sich mit

$\Longrightarrow$ `(avg (nth-numb 4 reqend) (where (not hit) (live)))`

9 Mehrfach-Tags

BACKEND VCL_ACL GZIP HASH OBJHEADER TTL OBJPROTOCOL OBJRESPONSE RXHEADER TXHEADER VCL_CALL VCL_RETURN

10 Einfach-Tags

BACKENDCLOSE BACKENDOPEN BACKENDREUSE BACKEND_HEALT CLI DEBUG EXPBAN EXPKILL FETCHERROR FETCH_BODY HIT INTERRUPTED REQEND REQSTART RXPROTOCOL RXREQUEST RXRESPONSE RXSTATUS RXURL SESSIONCLOSE SESSIONOPEN STATSESS TXPROTOCOL TXREQUEST TXRESPONSE TXSTATUS TXURL WORKTHREAD

11 Performance

Varnishq schafft rund 150000 Requests/s. Für jedes Gigabyte Logfile sind etwa 3 Minuten Verabeitungszeit anzusetzen. Hängt aber von der Query ab. Bei verknüpften Bedingungungen sollten die einfachsten so weit wie möglich vorne stehen. Also lieber `(where (and hit (contains (rxheader "polish"))) (live))` als `(where (and (contains (rxheader "polish")) hit) (live))`.

12 Vollständigkeit

Diese Darstellung ist unvollständig. VarnishQ kennt noch weitere Verarbeitungsformen und Transformationen. Aber dieses Cheat-Sheet sollte nicht mehr als 4 Seiten lang werden.

13 BNF

13.1 Tags

$$
\begin{aligned}
<mehrfach-tag> ::= & GZIP|OBJRESPONSE|OBJPROTOCOL|BACKEND && (1)\\
& |HIT|VCL\_ACL|OBJHEADER|TTL|TXHEADER && (2)\\
& |VCL\_RETURN|HASH|VCL\_CALL|RXHEADER && (3)\\
& && (4)\\
<einfach-tag> ::= & WORKTHREAD|TXURL|TXSTATUS|TXRESPONSE| && (5)\\
& |TXREQUEST|TXPROTOCOL|STATSESS && (6)\\
& |SESSIONOPEN|SESSIONCLOSE|RXURL && (7)\\
& |RXSTATUS|RXRESPONSE|RXREQUEST && (8)\\
& |RXPROTOCOL|REQSTART|REQEND|INTERRUPTED && (9)\\
& |FETCH\_BODY|FETCHERROR|EXPKILL|EXPBAN && (10)\\
& |DEBUG|CLI|BACKEND\_HEALT|BACKENDREUSE && (11)\\
& |BACKENDOPEN|BACKENDCLOSE && (12)\\
& && (13)\\
<tag> ::= & <einfach-tag> \mid <mehrfach-tag> && (14)\\
& && (15)\\
& && (16)
\end{aligned}
$$

13.2 Werte

$$< integer >::= < integer - tag > \tag{17}$$
$$|(nth - num < integer >< string >) \tag{18}$$
$$|(numb < regex >< string >) \tag{19}$$
$$|(epoch < integer >^{6} [< integer >]) \tag{20}$$
$$|integer - literal \tag{21}$$
$$\tag{22}$$
$$\tag{23}$$

$$< integer - vergleich >::= = | < | > | >= | <= |/ = \tag{24}$$
$$\tag{25}$$
$$\tag{26}$$

$$< string >::=\text{"}irgendeintext\text{"} \tag{27}$$
$$| < einfach - tag > \tag{28}$$
$$|(RXHEADER < string >) \tag{29}$$
$$|(TXHEADER < string >) \tag{30}$$
$$|(OBJHEADER < string >) \tag{31}$$
$$\tag{32}$$
$$< regex >::= < string > \tag{33}$$
$$\tag{34}$$
$$\tag{35}$$

13.3 Bedingungen

$$< string - vergleich >::=string = \tag{36}$$
$$|string < \tag{37}$$
$$|string > \tag{38}$$
$$|und - so - weiter \tag{39}$$
$$\tag{40}$$
$$\tag{41}$$

$$< condition >::= < tag > \tag{42}$$
$$|(< integer - vergleich >< integer >^{*}) \tag{43}$$
$$|(< string - vergleich >< string >^{*}) \tag{44}$$
$$|(MATCHES < regex >< string >) \tag{45}$$
$$|(AND < condition >^{*}) \tag{46}$$
$$|(OR < condition >^{*}) \tag{47}$$
$$|(NOT < condition >^{*}) \tag{48}$$
$$|(ANY < condition >^{*}) \tag{49}$$
$$\tag{50}$$
$$\tag{51}$$

$$< liste >::= < mehrfach - tag > \tag{52}$$
$$|'(< atom >^{*}) \tag{53}$$
$$\tag{54}$$
$$\tag{55}$$

13.4 Streams

$$
\begin{aligned}
&< stream >::=(LIVE) && (56)\\
&\quad |(FILE < string >) && (57)\\
&\quad |(VLOG < sttring >) && (58)\\
&\quad |(FILE - starting - at < string >< integer > && (59)\\
&\quad |(WHERE < condition >< stream >) && (60)\\
&\quad |(SELECT - TAG'tag < stream >) && (61)\\
&\quad |(SELECT - TAGS < list >< stream >) && (62)\\
&\quad |(SELECT < tcd - condition >< stream >) && (63)\\
& && (64)\\
& && (65)
\end{aligned}
$$

13.5 Queries

$$
\begin{aligned}
&< query >::=(APP < expr >< stream >) && (66)\\
&\quad |(DMP < stream >) && (67)\\
&\quad |(DUMP < stream >) && (68)\\
&\quad |(TAKE < integer >< stream >) && (69)\\
&\quad |(CHART < integer >< integer >< expr >< stream >) && (70)\\
& && (71)\\
& && (72)
\end{aligned}
$$