



EBook Gratuito

APPENDIMENTO

varnish

Free unaffiliated eBook created from
Stack Overflow contributors.

#varnish

Sommario

Di.....	1
Capitolo 1: Iniziare con la vernice	2
Osservazioni.....	2
Versioni.....	2
Examples.....	2
Installazione o configurazione.....	2
CentOS 7.....	2
Ubuntu.....	2
Debian.....	3
Vernice VCL.....	3
Capitolo 2: Costruire vmods	4
introduzione.....	4
Examples.....	4
Compilare e installare un vmod.....	4
Capitolo 3: Monitoraggio della vernice	5
introduzione.....	5
Examples.....	5
Metriche cliente: traffico in entrata.....	5
Prestazioni della cache.....	5
Monitoraggio degli oggetti memorizzati nella cache.....	6
Monitoraggio dei thread.....	6
Monitoraggio delle metriche di back-end.....	7
Capitolo 4: VCL incorporato	8
introduzione.....	8
Examples.....	8
Vernice 3.0.....	8
Vernice 4.0.....	10
Titoli di coda	14

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [varnish](#)

It is an unofficial and free varnish ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official varnish.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capitolo 1: Iniziare con la vernice

Osservazioni

Questa sezione fornisce una panoramica di che cos'è la vernice e perché uno sviluppatore potrebbe volerlo utilizzare.

Dovrebbe anche menzionare eventuali soggetti di grandi dimensioni all'interno della vernice e collegarsi agli argomenti correlati. Poiché la Documentazione per la vernice è nuova, potrebbe essere necessario creare versioni iniziali di tali argomenti correlati.

Versioni

Versione	Data di rilascio
5.1.2	2017/04/07
5.1.1	2017/03/16
5.0	2016/09/15
4.1.5	2016/02/09
4.0.4	2016/11/30
3.0.7	2015/03/23

Examples

Installazione o configurazione

Le seguenti sono istruzioni per l'installazione dell'ultima versione di Varnish su varie distribuzioni Linux.

CentOS 7

```
curl -s https://packagecloud.io/install/repositories/varnishcache/varnish5/script.rpm.sh |  
sudo bash
```

Ubuntu

```
apt-get install apt-transport-https  
curl https://repo.varnish-cache.org/GPG-key.txt | apt-key add -  
echo "deb https://repo.varnish-cache.org/ubuntu/ trusty varnish-4.1" \  

```

```
>> /etc/apt/sources.list.d/varnish-cache.list
apt-get update
apt-get install varnish
```

Debian

```
apt-get install apt-transport-https
curl https://repo.varnish-cache.org/GPG-key.txt | apt-key add -
echo "deb https://repo.varnish-cache.org/debian/ jessie varnish-4.1" \
  >> /etc/apt/sources.list.d/varnish-cache.list
apt-get update
apt-get install varnish
```

Vernice VCL

Varnish controlla e manipola le richieste HTTP usando Varnish Configuration Language (VCL). Il seguente frammento di VCL rimuove i cookie dalle richieste in arrivo alla sottodirectory / images:

```
sub vcl_recv {
    if (req.url ~ "^/images") {
        unset req.http.cookie;
    }
}
```

Leggi Iniziare con la vernice online: <https://riptutorial.com/it/varnish/topic/4705/iniziare-con-la-vernice>

Capitolo 2: Costruire vmods

introduzione

Non devi necessariamente compilare vmods se i binari per loro sono già disponibili per la tua piattaforma. Sia CentOS 6 che 7 possono utilizzare i build COPR di Ingvar per installare una raccolta di moduli aggiuntivi da Varnish Software:

<https://copr.fedorainfracloud.org/coprs/ingvar/varnish51/>

Examples

Compilare e installare un vmod

L'installazione di un vmod richiede una versione installata di Varnish Cache, inclusi i file di sviluppo. I requisiti possono essere trovati nella documentazione di Varnish.

Il codice sorgente è compilato con gli autotools:

```
sudo apt-get install libvarnishapi-dev || sudo yum install varnish-libs-devel
./bootstrap # If running from git.
./configure
make
make check # optional
sudo make install
```

Leggi Costruire vmods online: <https://riptutorial.com/it/varnish/topic/9669/costruire-vmods>

Capitolo 3: Monitoraggio della vernice

introduzione

Utilizzare `varnishstat` per monitorare le metriche numeriche di un'istanza di Varnish attualmente in esecuzione. La posizione sarà diversa in base all'installazione. L'esecuzione di `varnishstat -1` genererà tutte le metriche in un semplice formato `grep -able`.

Altre utilità sono disponibili per la visualizzazione dello stato e della registrazione della `varnishtop` : `varnishtop` , `varnishlog` , ecc.

Examples

Metriche cliente: traffico in entrata

Le metriche del cliente coprono il traffico tra il client e la cache di Varnish.

- `sess_conn` - Numero cumulativo di connessioni.
- `client_req` - Numero cumulativo di richieste client.
- `sess_dropped`: connessioni interrotte a causa di una coda completa.

Monitorare `sess_conn` e `client_req` per tenere traccia del volume del traffico, in aumento o in diminuzione, spiking, ecc. Modifiche improvvise potrebbero indicare problemi.

Monitora `sess_dropped` per vedere se la cache sta lasciando cadere qualsiasi sessione. In tal caso potrebbe essere necessario aumentare `thread_pool_max` .

```
varnishstat -1 | grep "sess_conn\\|client_req \\|sess_dropped"
MAIN.sess_conn          62449574          3.38 Sessions accepted
MAIN.client_req        184697229          9.99 Good client requests received
MAIN.sess_dropped          0          0.00 Sessions dropped for thread
```

Prestazioni della cache

Forse la metrica delle prestazioni più importante è l'hitrate.

Le rotte di vernice sono richieste in arrivo come questa:

- Hash, una richiesta memorizzabile nella cache. Questo potrebbe essere sia `hit` o `miss` a seconda dello stato della cache.
- Hitpass, una richiesta non memorizzabile nella cache.

Un hash con un `miss` e un `hitpass` saranno recuperati dal back-end del server e consegnati. Un hash con un `hit` verrà consegnato direttamente dalla cache.

Metriche da monitorare:

- `cache_hit`: numero di hash con un hit nella cache.
- `cache_miss`: numero di hash con errori nella cache.
- `cache_hitpass` - Numero di hitpasses come sopra.

```
varnishstat -1 | grep "cache_hit \|cache_miss \|cache_hitpass"
MAIN.cache_hit          99032838          5.36 Cache hits
MAIN.cache_hitpass      0                0.00 Cache hits for pass
MAIN.cache_miss         42484195          2.30 Cache misses
```

Calcola l'hitrate attuale in questo modo:

```
cache_hit / (cache_hit + cache_miss)
```

In questo esempio l'hitrate è 0,7 o 70%. Vuoi tenerlo il più alto possibile. Il 70% è un numero decente. Puoi migliorare l'hitrate aumentando la memoria e personalizzando il tuo vcl. Monitora anche i grandi cambiamenti nel tuo hitrate.

Monitoraggio degli oggetti memorizzati nella cache

Si monitorano gli oggetti memorizzati nella cache per vedere quanto spesso scadono e se sono "nuked".

- `n_expired` - Numero di oggetti scaduti.
- `n_lru_nuked` - Ultimi oggetti nuked usati di recente. Numero di oggetti messi a nudo (rimossi) dalla cache a causa della mancanza di spazio.

```
varnishstat -1 | grep "n_expired\|n_lru_nuked"
MAIN.n_expired          42220159          .      Number of expired objects
MAIN.n_lru_nuked       264005            .      Number of LRU nuked objects
```

Quello da guardare qui è `n_lru_nuked`, se la velocità è in aumento (la *velocità*, non solo il numero) la cache spinge fuori oggetti sempre più velocemente a causa della mancanza di spazio. È necessario aumentare la dimensione della cache.

Il `n_expired` è più `n_expired` alla tua applicazione. Un tempo più lungo per vivere diminuirà questo numero ma d'altra parte non rinnoverà gli oggetti più spesso. Anche la cache potrebbe richiedere più dimensioni.

Monitoraggio dei thread

Devi tenere traccia di alcune metriche dei thread per guardare la tua Varnish Cache. Sta finendo le risorse del sistema operativo o funziona bene.

- `thread` - Numero di thread in tutti i pool.
- `threads_created`: numero di thread creati.
- `threads_failed`: numero di volte in cui Varnish non è riuscito a creare un thread.
- `threads_limited` - Numero di volte in cui Varnish è stato costretto a non creare un thread da quando è stato raggiunto il limite massimo.

- `thread_queue_len` - Lunghezza della coda corrente. Numero di richieste in attesa di un thread.
- `sess_queued`: numero di volte in cui non sono stati disponibili thread in modo che una richiesta debba essere accodata.

```
varnishstat -1 | grep "threads\|thread_queue_len\|sess_queued"
MAIN.threads          100      .    Total number of threads
MAIN.threads_limited   1        0.00 Threads hit max
MAIN.threads_created   3715     0.00 Threads created
MAIN.threads_destroyed 3615     0.00 Threads destroyed
MAIN.threads_failed    0        0.00 Thread creation failed
MAIN.thread_queue_len  0        .    Length of session queue
MAIN.sess_queued       2505     0.00 Sessions queued for thread
```

Se `thread_queue_len` non è 0 significa che Varnish ha esaurito le risorse e ha iniziato a mettere in coda le richieste. Ciò ridurrà le prestazioni di tali richieste. Devi indagare sul perché.

Guarda anche `threads_failed`. Se aumenta, significa che il tuo server ha esaurito le risorse in qualche modo. Aumentare i numeri in `threads_limited` significa che potrebbe essere necessario aumentare i server `thread_pool_max`.

Monitoraggio delle metriche di back-end

Ci sono un certo numero di parametri che descrivono la comunicazione tra Varnish e i suoi backend.

Le metriche più importanti qui potrebbero essere queste:

- `backend_busy` - Numero di stati http 5xx ricevuti da un back-end. Con VCL puoi configurare Varnish per provare un altro backend se questo accade.
- `backend_fail` - Numero di volte che la Varnish non è riuscita a connettersi al back-end. Questo può avere un numero di cause (nessuna connessione TCP, molto tempo per il primo byte, molto tempo tra i byte). Se ciò accade, il tuo back-end non è in salute.
- `backend_unhealthy` - Numero di volte in cui Varnish non ha potuto "eseguire il ping" del back-end (non ha risposto con una risposta HTTP 200).

```
varnishstat -1 | grep "backend_"
MAIN.backend_conn      86913481  4.70 Backend conn. success
MAIN.backend_unhealthy 0         0.00 Backend conn. not attempted
MAIN.backend_busy      0         0.00 Backend conn. too many
MAIN.backend_fail      7         0.00 Backend conn. failures
MAIN.backend_reuse     0         0.00 Backend conn. reuses
MAIN.backend_toolate   0         0.00 Backend conn. was closed
MAIN.backend_recycle   0         0.00 Backend conn. recycles
MAIN.backend_retry     0         0.00 Backend conn. retry
MAIN.backend_req       86961073  4.70 Backend requests made
```

Leggi Monitoraggio della vernice online: <https://riptutorial.com/it/varnish/topic/9072/monitoraggio-della-vernice>

Capitolo 4: VCL incorporato

introduzione

Il VCL integrato contiene procedure incluse ed eseguite per *ultime* da Varnish.

Possono integrare VCL definito dall'utente fornendo una logica appropriata per la maggior parte dei siti. Ad esempio, ignora la cache per le richieste POST e / o in presenza di cookie o intestazioni di autorizzazione.

Se parte della logica incorporata non è necessaria, un utente può aggiungere una chiamata `return()` da una procedura in cui la logica VCL incorporata non è desiderabile.

Examples

Vernice 3.0

```
/*-
 *
 * The default VCL code.
 *
 * NB! You do NOT need to copy & paste all of these functions into your
 * own vcl code, if you do not provide a definition of one of these
 * functions, the compiler will automatically fall back to the default
 * code from this file.
 *
 */

sub vcl_recv {
    if (req.restarts == 0) {
        if (req.http.x-forwarded-for) {
            set req.http.X-Forwarded-For =
                req.http.X-Forwarded-For + ", " + client.ip;
        } else {
            set req.http.X-Forwarded-For = client.ip;
        }
    }
    if (req.request != "GET" &&
        req.request != "HEAD" &&
        req.request != "PUT" &&
        req.request != "POST" &&
        req.request != "TRACE" &&
        req.request != "OPTIONS" &&
        req.request != "DELETE") {
        /* Non-RFC2616 or CONNECT which is weird. */
        return (pipe);
    }
    if (req.request != "GET" && req.request != "HEAD") {
        /* We only deal with GET and HEAD by default */
        return (pass);
    }
    if (req.http.Authorization || req.http.Cookie) {
        /* Not cacheable by default */
    }
}
```

```

        return (pass);
    }
    return (lookup);
}

sub vcl_pipe {
    # Note that only the first request to the backend will have
    # X-Forwarded-For set.  If you use X-Forwarded-For and want to
    # have it set for all requests, make sure to have:
    # set bereq.http.connection = "close";
    # here.  It is not set by default as it might break some broken web
    # applications, like IIS with NTLM authentication.
    return (pipe);
}

sub vcl_pass {
    return (pass);
}

sub vcl_hash {
    hash_data(req.url);
    if (req.http.host) {
        hash_data(req.http.host);
    } else {
        hash_data(server.ip);
    }
    return (hash);
}

sub vcl_hit {
    return (deliver);
}

sub vcl_miss {
    return (fetch);
}

sub vcl_fetch {
    if (beresp.ttl <= 0s ||
        beresp.http.Set-Cookie ||
        beresp.http.Vary == "*") {
        /*
         * Mark as "Hit-For-Pass" for the next 2 minutes
         */
        set beresp.ttl = 120 s;
        return (hit_for_pass);
    }
    return (deliver);
}

sub vcl_deliver {
    return (deliver);
}

sub vcl_error {
    set obj.http.Content-Type = "text/html; charset=utf-8";
    set obj.http.Retry-After = "5";
    synthetic {
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

```

```

<html>
  <head>
    <title>"} + obj.status + " " + obj.response + {"</title>
  </head>
  <body>
    <h1>Error "} + obj.status + " " + obj.response + {"</h1>
    <p>"} + obj.response + {"</p>
    <h3>Guru Meditation:</h3>
    <p>XID: "} + req.xid + {"</p>
    <hr>
    <p>Varnish cache server</p>
  </body>
</html>
"};
  return (deliver);
}

sub vcl_init {
  return (ok);
}

sub vcl_fini {
  return (ok);
}

```

Vernice 4.0

```

/*
 * The built-in (previously called default) VCL code.
 *
 * NB! You do NOT need to copy & paste all of these functions into your
 * own vcl code, if you do not provide a definition of one of these
 * functions, the compiler will automatically fall back to the default
 * code from this file.
 *
 * This code will be prefixed with a backend declaration built from the
 * -b argument.
 */

vcl 4.0;

#####
# Client side

sub vcl_recv {
  if (req.method == "PRI") {
    /* We do not support SPDY or HTTP/2.0 */
    return (synth(405));
  }
  if (req.method != "GET" &&
      req.method != "HEAD" &&
      req.method != "PUT" &&
      req.method != "POST" &&
      req.method != "TRACE" &&
      req.method != "OPTIONS" &&
      req.method != "DELETE") {
    /* Non-RFC2616 or CONNECT which is weird. */
    return (pipe);
  }
}

```

```

if (req.method != "GET" && req.method != "HEAD") {
    /* We only deal with GET and HEAD by default */
    return (pass);
}
if (req.http.Authorization || req.http.Cookie) {
    /* Not cacheable by default */
    return (pass);
}
return (hash);
}

sub vcl_pipe {
    # By default Connection: close is set on all piped requests, to stop
    # connection reuse from sending future requests directly to the
    # (potentially) wrong backend. If you do want this to happen, you can undo
    # it here.
    # unset bereq.http.connection;
    return (pipe);
}

sub vcl_pass {
    return (fetch);
}

sub vcl_hash {
    hash_data(req.url);
    if (req.http.host) {
        hash_data(req.http.host);
    } else {
        hash_data(server.ip);
    }
    return (lookup);
}

sub vcl_purge {
    return (synth(200, "Purged"));
}

sub vcl_hit {
    if (obj.ttl >= 0s) {
        // A pure unadulterated hit, deliver it
        return (deliver);
    }
    if (obj.ttl + obj.grace > 0s) {
        // Object is in grace, deliver it
        // Automatically triggers a background fetch
        return (deliver);
    }
    // fetch & deliver once we get the result
    return (fetch);
}

sub vcl_miss {
    return (fetch);
}

sub vcl_deliver {
    return (deliver);
}

```

```

/*
 * We can come here "invisibly" with the following errors: 413, 417 & 503
 */
sub vcl_synth {
    set resp.http.Content-Type = "text/html; charset=utf-8";
    set resp.http.Retry-After = "5";
    synthetic( {"<!DOCTYPE html>
<html>
  <head>
    <title>} + resp.status + " " + resp.reason + {"</title>
  </head>
  <body>
    <h1>Error "} + resp.status + " " + resp.reason + {"</h1>
    <p>} + resp.reason + {"</p>
    <h3>Guru Meditation:</h3>
    <p>XID: "} + req.xid + {"</p>
    <hr>
    <p>Varnish cache server</p>
  </body>
</html>
"} );
    return (deliver);
}

#####
# Backend Fetch

sub vcl_backend_fetch {
    return (fetch);
}

sub vcl_backend_response {
    if (beresp.ttl <= 0s ||
        beresp.http.Set-Cookie ||
        beresp.http.Surrogate-control ~ "no-store" ||
        (!beresp.http.Surrogate-Control &&
            beresp.http.Cache-Control ~ "no-cache|no-store|private") ||
        beresp.http.Vary == "*") {
        /*
         * Mark as "Hit-For-Pass" for the next 2 minutes
         */
        set beresp.ttl = 120s;
        set beresp.uncacheable = true;
    }
    return (deliver);
}

sub vcl_backend_error {
    set beresp.http.Content-Type = "text/html; charset=utf-8";
    set beresp.http.Retry-After = "5";
    synthetic( {"<!DOCTYPE html>
<html>
  <head>
    <title>} + beresp.status + " " + beresp.reason + {"</title>
  </head>
  <body>
    <h1>Error "} + beresp.status + " " + beresp.reason + {"</h1>
    <p>} + beresp.reason + {"</p>
    <h3>Guru Meditation:</h3>
    <p>XID: "} + bereq.xid + {"</p>
    <hr>

```

```
<p>Varnish cache server</p>
</body>
</html>
"} );
    return (deliver);
}

#####
# Housekeeping

sub vcl_init {
    return (ok);
}

sub vcl_fini {
    return (ok);
}
```

Leggi VCL incorporato online: <https://riptutorial.com/it/varnish/topic/5001/vcl-incorporato>

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con la vernice	alejdg , Community , Daniel V.
2	Costruire vmods	Daniel V.
3	Monitoraggio della vernice	Jensd
4	VCL incorporato	Daniel V. , fgsch , Redithion