Varnish
para meros mortales

#DrupalCampES

Sobre mí

Martín González Robles

@mgzrobles

Responsable técnico de idealista/news

Recursos

- Varnish Cache Website

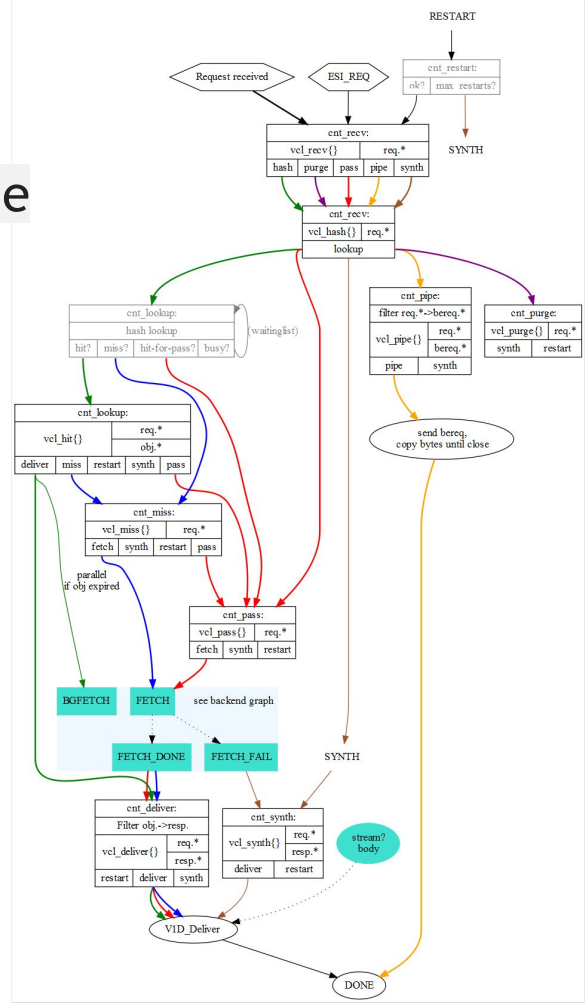
  https://www.varnish-cache.org/
- The Varnish Book for Varnish 4.0

  http://book.varnish-software.com/4.0/
- Sácale partido a Varnish - Rodrigo Alfaro - DCampES2013
  https://vimeo.com/81136713

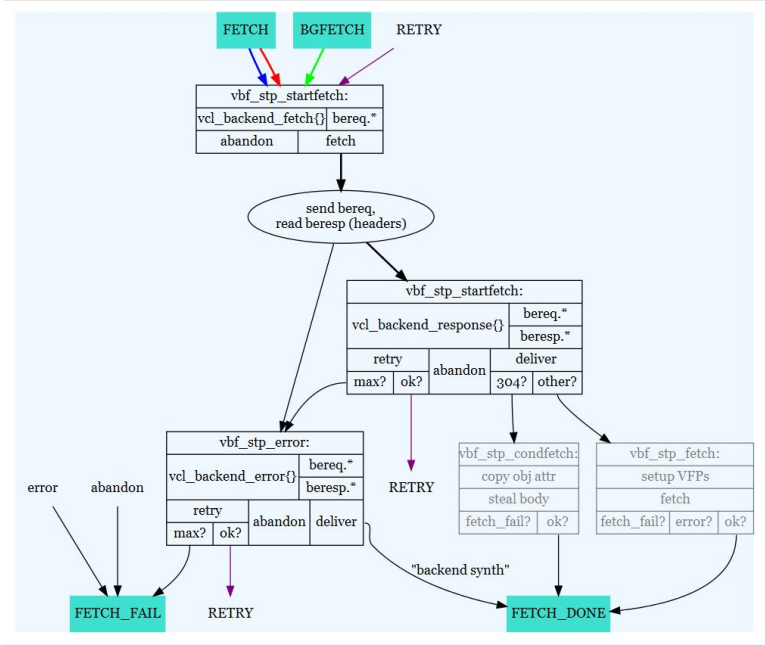
Índice

¿Qué es Varnish?

Client Side



Backend Side



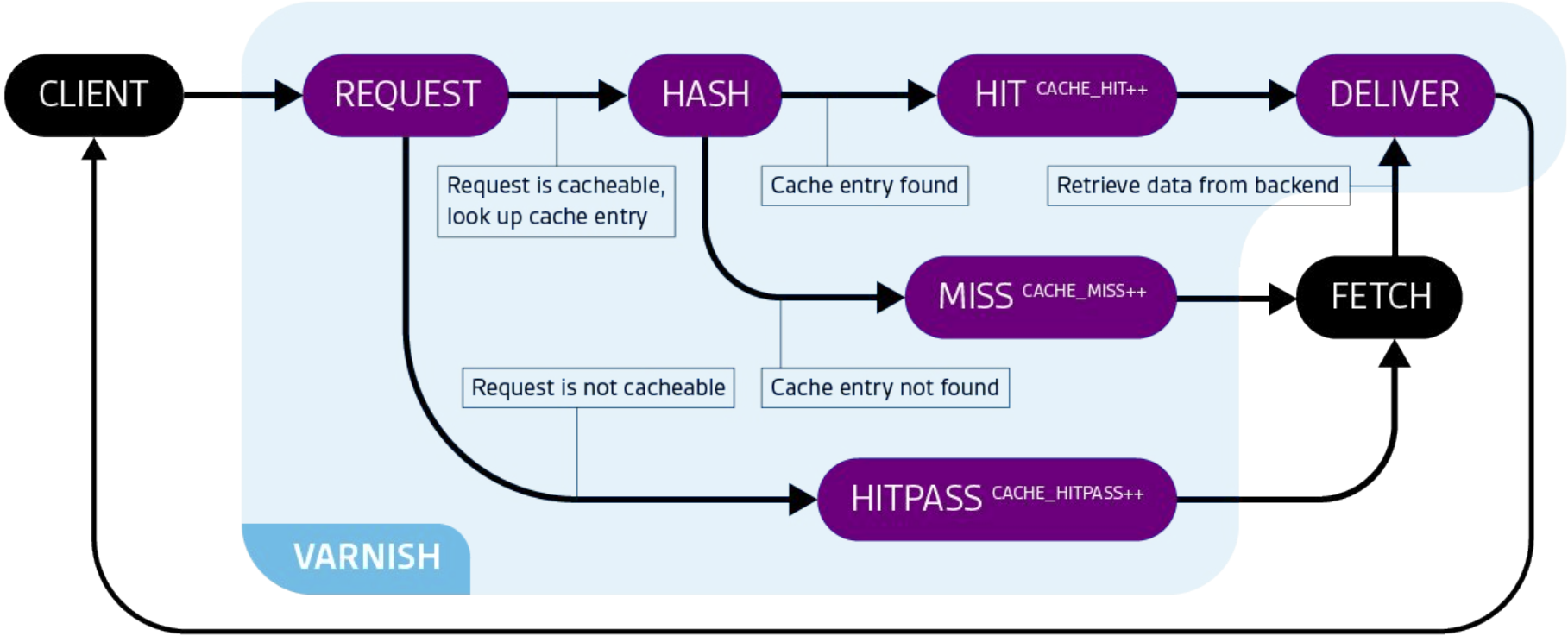
https://www.varnish-cache.org/docs/trunk/reference/states.html

Varnish Cache es un acelerador de aplicaciones web.

Se sitúa delante de nuestro servidor HTTP y almacena una copia de la petición solicitada.

Se suele usar como balanceador.

Flujo de una petición


CLIENT
REQUEST
HASH
HIT CACHE_HIT++
DELIVER
Request is cacheable, look up cache entry
Cache entry found
Retrieve data from backend
MISS CACHE_MISS++
FETCH
Request is not cacheable
Cache entry not found
HITPASS CACHE_HITPASS++
VARNISH

vcl_recv

Principio de la petición. Trabajaremos con el objeto “req”.

Tareas:

- Decidir si hacer una redirección
- Seleccionar backend - Balanceo
- Comprobar accesos
- Marcar como “static” o “anonimizar” una request
- Normalizar URLs
- No cachear una petición

vcl_recv

```
## REDIRECCIONES ##

if (req.url ~ "^/foo.*") {

 set req.url = regsub(req.url,  "^/foo(.*)", "/bar\1");

 return (synth(301, "http://" + req.http.host + req.url));

}
```

```
## SELECCION BACKEND ##

# Directors
set req.backend_hint = bar.backend();
# Backend específico por algún motivo
if (req.url ~ "^/url-muy-lenta$") {
  set req.backend = my_backend_longwait;
}
```

https://www.varnish-cache.org/docs/4.0/users-guide/vcl-backends.html

vcl_recv

```
## ACCESSO ##

if (req.url ~ "/(?:cron|install|update)\.php") {

 if (client.ip ~ localnetwork) {

   return (pipe);

 } else {

   return (synth(404, "Page not found."));

}
```

```
## QUITAR COOKIES ##

if ( req.url  ~ "^/sites/files/"

  || req.url  ~ "^/(.*)[?|&]amp(.*)" ) {

  unset req.http.Cookie;

}
```

vcl_recv

```
## NORMALIZAR URL ##

if (req.http.host ~ "(?i)^(www.)?dominio.com") {
  set req.http.host = "dominio.com";
}

# Normalize the query arguments

set req.url = std.querysort(req.url);
```

```
# NO CACHEAR SI TIENE COOKIE DE SESIÓN

if ( req.http.Cookie ~ "SESS" ) {

  return (pass);

}
```

vcl_hash

Aquí se generará la key para buscar y/o guardar el objeto de la petición.

Lo más habitual será guardar una versión por url.

También podemos guardar una versión basándonos en geolocalización (necesitaremos un vmod), o por cookie, etc.

Echémosle imaginación.

vcl_hash

```
sub vcl_hash {

	# Drupal ddos vulnerability

	hash_data(regsub(req.url, "\?(.*)itok=.+[^&amp;]", "\1"));

	if (req.http.host) {

		hash_data(req.http.host);

	} else {

		hash_data(server.ip);

	}

	if ( req.http.Cookie ) {

		hash_data(req.http.Cookie);

	}

	return (lookup);

}
```

https://www.varnish-cache.org/docs/4.0/users-guide/vcl-hashing.html
http://drupalblog.torchbox.com/post/45261810748/varnish-and-drupal-72021-dos-vulnerability

Otras subrutinas

- vcl_miss
  - No se ha encontrado en caché
- vcl_hit
  - Encontrado en caché
- vcl_pass
  - Enviar la petición a backend y no cachear
- vcl_backend_fetch
  - Justo antes de llamar a backend
- vcl_backend_error
  - Tratamos posibles errores desde backend y podemos reintentar la petición.

vcl_backend_response

Tratamos la petición devuelta desde backend.

Podemos modificar cabeceras, el tiempo de caché, habilitar ESI

https://www.varnish-cache.org/docs/4.0/users-guide/esi.html

vcl_backend_response

```
sub vcl_backend_response {

	// Hacia afuera dejamos 0 segundos de max-age para que esté en los navegadores.

	set beresp.http.cache-control = "max-age=0";

	if (beresp.http.Content-Type ~ "(image|audio|video|flash|css|javascript)") {

		set beresp.http.cache-control = "max-age=1209600";

		set beresp.ttl = 2w;

		if (beresp.status >= 500 && beresp.status < 600) {
		  set beresp.http.Cache-Control = "no-cache, max-age=0, must-revalidate";
		  set beresp.ttl = 0s;
		  set beresp.uncacheable = true;
		  return(deliver);
		}

	}
```

vcl_deliver

Tenemos todo para mandar una respuesta al cliente.

Terminaremos de definir las cabeceras finales porque en este punto ya no se cachearían.

También podremos terminar de componer el contenido de la respuesta, por ejemplo para 404 o 500.

vcl_deliver

```
sub vcl_deliver {

  if (std.ip(req.http.True-Client-IP, client.ip) ~
locales) {

    if (resp.http.x-varnish ~ " ") {

      set resp.http.X-cache = "HIT";

      set resp.http.X-Cache-Hits = obj.hits;

    } else {

      set resp.http.X-cache = "MISS";

    }
```

```
if ( resp.status >= 500 ) {

      return (synth(500, "Internal Server Error."));

}

else if (resp.status >= 400 && resp.status < 500) {

       return (synth(404, "Page not found."));

}
```

vcl_synth

Se le llama para devolver un contenido generado por el propio vcl, no por backend. Aquí podremos **personalizar las páginas 404/500.**

```
sub vcl_synth {

  if (resp.status >= 500 && resp.status < 600) {

    set resp.http.Content-Type = "text/html; charset=utf-8";

    synthetic(std.fileread("/var/www/50x.html"));

    return(deliver);

  }
```

```
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>

"} );
```

Tips and tricks

Uso de funciones y “variables”

Podemos programar nuestras propias subroutines y llamarlas de forma sencilla.

No admiten argumentos, pero podemos “jugar” con las cabeceras.

```
sub vcl_recv {
...
 call redirects;
 call init_custom_vars;
...
```

```
sub init_custom_vars {
 if (req.http.host ~ "^localhost") {
  set req.http.x-is-environment = "dev";
 }
 else {
   set req.http.x-is-environment = "prod";
 }
```

```
if (bereq.http.x-is-environment == "dev")
{
  set beresp.uncacheable = true;
}
```

“Debug” Logar eventos

Podemos escribir al syslog desde varnish.

```
import std;

sub vcl_recv {

...

  std.syslog(180, "RECV: " + req.http.host + req.url);

...

}
```

```
~$ sudo tailf /var/log/syslog
```

http://martingonzalez.es/blog/logar-eventos-al-syslog-en-varnish
https://es.wikipedia.org/wiki/Syslog

Comandos útiles de varnish

Top peticiones a backend

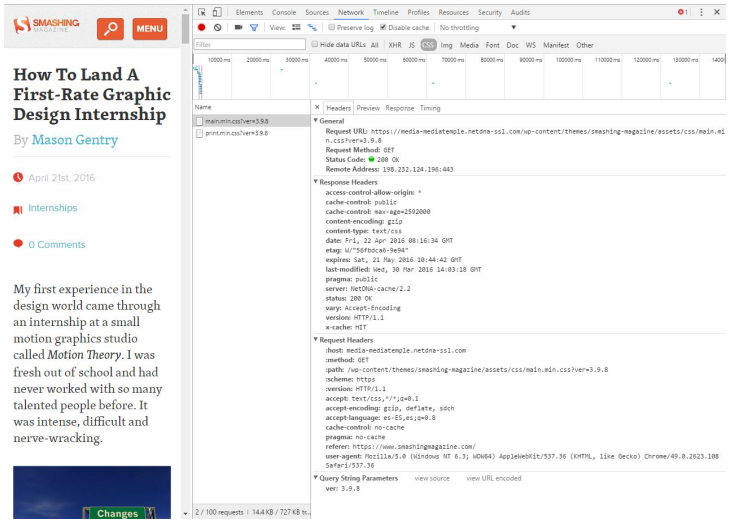
```
varnishtop -i BereqURL
```

Filtrar peticiones usando Query Language

```
varnishlog -q 'ReqUrl ~ "^/foo" and RespStatus > 200'
```

http://book.varnish-software.com/4.0/chapters/Examining_Varnish_Server_s_Output.html#query-language

Chrome DevTools - Network



× Headers Preview Response Timing

▼ General

Request URL: https://media-mediatemple.netdna-ssl.com/wp-content/themes/smashing-magazine/assets/css/main.min.css?ver=3.9.8

Request Method: GET

Status Code: 200 OK

Remote Address: 198.232.124.196:443

▼ Response Headers

access-control-allow-origin: *

cache-control: public

cache-control: max-age=2592000

content-encoding: gzip

content-type: text/css

date: Fri, 22 Apr 2016 08:16:34 GMT

etag: W/"56fbdca6-9e94"

expires: Sat, 21 May 2016 10:44:42 GMT

last-modified: Wed, 30 Mar 2016 14:03:18 GMT

pragma: public

server: NetDNA-cache/2.2

status: 200 OK

vary: Accept-Encoding

version: HTTP/1.1

x-cache: HIT

▼ Request Headers

:host: media-mediatemple.netdna-ssl.com

:method: GET

:path: /wp-content/themes/smashing-magazine/assets/css/main.min.css?ver=3.9.8

:scheme: https

:version: HTTP/1.1

accept: text/css,*/*;q=0.1

accept-encoding: gzip, deflate, sdch

accept-language: es-ES,es;q=0.8

cache-control: no-cache

pragma: no-cache

referer: https://www.smashingmagazine.com/

user-agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Safari/537.36

Médico de familia

Médico de familia

¿Qué pasa si nuestro backend está en mal estado?

Esto puede ser debido a muchas causas:

- Podemos tener un pico de tráfico
- Que la base de datos no responda
- Quizás falta de conexión con nuestro memcache
- El sistema de ficheros no es accesible
- O el motivo X según el caso

Médico de familia

Varnish tiene el **“grace mode”**, que permite devolver una versión almacenada en caché cuando no hay ningún backend disponible.

Médico de familia

```
probe healthcheck {
  .request =
    "GET /var/www/status.php HTTP/1.1"
      "Host: www.host.com"
      "Connection: close";
  .interval = 10s;
  .timeout = 2s;
  .window = 8;
  .threshold = 3;
}
```

```
backend web1 {
    .host = "127.0.0.1";
    .port = "8080";
    .connect_timeout = 100s;
    .first_byte_timeout = 60s;
    .between_bytes_timeout = 60s;
    .max_connections = 100;
    .probe = healthcheck;
}
```

https://www.varnish-cache.org/docs/4.0/reference/vcl.html#probes
https://www.varnish-cache.org/docs/4.0/reference/vcl.html#backend-definition

Médico de familia

```
<?php
register_shutdown_function('status_shutdown');
function status_shutdown() {
 exit();
}
define('DRUPAL_ROOT', getcwd());
try {
 require_once DRUPAL_ROOT . '/includes/bootstrap.inc';
 drupal_bootstrap(DRUPAL_BOOTSTRAP_DATABASE);
} catch (PDOException $e) {
 $errors[] = $prefix . "Database not responding: " . $e-
>getMessage();
}
```

```
if ($errors) {
  header('HTTP/1.1 500 Internal Server Error');
  error_log(implode(PHP_EOL, $errors), 0);
}
else {
  header("HTTP/1.1 200 OK");
}

// Exit immediately, note the shutdown function
registered at the top of the file.
exit();
```

Médico de familia

---

vcl_recv

```
set req.http.grace = "none";

if (!std.healthy(req.backend_hint)) {
  std.syslog(180, "CAIDOS: " + req.url);
  unset req.http.Cookie;
  return (lookup);
}
```

vcl_hit

```
sub vcl_hit {
 ###
 if (!std.healthy(req.backend_hint)) {
   if (obj.ttl + obj.grace > 0s) {
     set req.http.grace = "full";
     return (deliver);
   } else {
     return (fetch);
   }
 }
```

https://info.varnish-software.com/blog/grace-varnish-4-stale-while-revalidate-semantics-varnish

Médico de familia

```
sub vcl_backend_response {

    set beresp.ttl = 10s;

    set beresp.grace = 6h;

}




sub vcl_deliver {

    set resp.http.grace = req.http.grace;

}
```

Drupal & Varnish

Invalidación de caché

Expire + Varnish
https://www.drupal.org/project/expire + https://www.drupal.org/project/varnish

```
function MY_MODULE_expire_urls_alter(&$urls, $object_type, $object) {

  if (!empty($object->type)) {

    if ($object->type === 'article' && node_is in_front($object)) {

      $urls['front'] = url('<front>', array('absolute' => TRUE, 'alias' => TRUE));

    }

}
```

Invalidación de caché

```
$view = views_get_view('front');

$view->set_display('page');

$cache = $view->display_handler->get_plugin('cache');
```

```
$cache->cache_flush();
```

Invalidación de caché

Drupal

```
$req_url = ‘/foo’;

$host = _varnish_get_host();

varnish_purge($host, '^' .  $req_url . '$');
```

Shell cURL

```
~$ CURL -X PURGE "http://varnish-localhost/foo"
```

PHP - cURL

```
$curl = curl_init("http://varnish-localhost/foo");

curl_setopt($curl, CURLOPT_CUSTOMREQUEST, "PURGE");

curl_exec($curl);
```

varnishadm

```
~$ sudo varnishadm

varnish> ban req.url ~ “^/foo”
```

¿Preguntas?

¡Gracias!