Configura todas las caché de WordPress

que no son pocas...

Si te vas a dedicar a Internet

has de conocer cómo funciona Internet (al menos un poco).

¡Hola!

Javier Casares

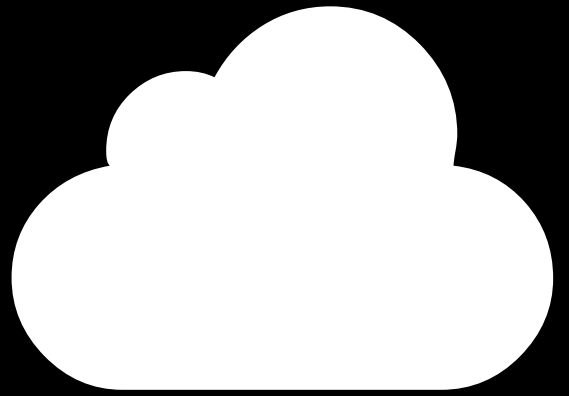
javiercasares.com

Desde 2005 usando WordPress

Director de Operaciones y SysAdmin en brutal.systems

brutal.

WordPress Hosting team



make.wordpress.org/hosting

We work to improve WordPress' end-user experience across hosting environments through industry collaboration and user education. Come join us!

The team meets in the #hosting-community Slack channel each week.

IMPORTANTE

Cada infraestructura es un mundo.

Debido a la combinatoria que existe, en algunos casos pondré ejemplos de una tecnología o de otra (por ejemplo, alguna vez será de Apache HTTPD, otras de nginx).

Mi objetivo es que esto sirva para dar una idea de qué se puede hacer y... StackOverflow ☺

Conceptos

WordPress es dinámico

WordPress por defecto es 100% dinámico, lo que significa que cada vez que alguien accede se ha de generar todo por completo, lo que tiene un cálculo muy elevado.

Algunos de estos elementos que pueden ralentizar los procesos son consultas a la base de datos, la propia ejecución de PHP, llamadas a API externas...

Hablemos de caché

La caché es un componente que almacena datos para que en solicitudes futuras se puedan devolver con mayor rapidez.

Los datos almacenados en un caché suelen ser el resultado de una petición previa.

En ocasiones es simplemente el duplicado de datos almacenados en otro lugar.

Caché == Reutilizar

Es por esto que se recomienda cachear.

Este proceso permite reutilizar resultados anteriormente calculados en varias ocasiones siguiendo una serie de reglas, lo que reduce el consumo de tareas repetitivas.

Caché en WordPress

Las 10 capas de caché

1. Caché del navegador
2. CDN (Content Delivery Network)
3. Caché de página, en servidor web o proxy
4. Caché de página, estática o por PHP
5. Opcode caché

Las 10 capas de caché

6. Caché de objetos
7. Caché de fragmentos
8. Transient API
9. Caché de base de datos
10. Caché de ficheros en disco

Qué caché usar

DEPENDE

Cada capa de caché tiene unos requerimientos y una utilidad.

Dependiendo de tu WordPress y el uso que le das, serán útiles ninguna, una, varias o todas las capas.

Cuantas más capas de caché uses, más recursos necesitas.

Caché del navegador

¿Qué es?

Los navegadores (Firefox, Chrome...) permiten almacenar ciertos elementos en el propio ordenador.

Se suele almacenar información que no cambia (suele ser caché de estáticos).

Caso y uso habitual

Cuando entramos en un sitio por primera vez cargamos muchos CSS, JavaScript, imágenes...

Muchos de estos elementos se van a volver a cargar en cada página vista que hagamos.

Objetivo: reducir la cantidad de peticiones de elementos iguales.

```
<IfModule mod_expires.c>
  ExpiresActive on
  #Varios
  ExpiresByType application/pdf A2592000
  ExpiresByType image/svg+xml A2592000
  #Imagenes
  ExpiresByType image/jpg A2592000
  ExpiresByType image/webp A2592000
  #Media
  ExpiresByType video/mp4 A2592000
  ExpiresByType video/webm A2592000
  #CSS/JS
  ExpiresByType text/css A2592000
  ExpiresByType text/javascript A2592000
  #Fuentes
  ExpiresByType application/font-woff2 A2592000
  ExpiresByType font/ttf A2592000
  ExpiresByType font/woff A2592000
  ExpiresByType font/woff2 A2592000
</IfModule>
```

Un poco de explicación

Esto es código para el .htaccess / Apache HTTPD.

La idea es listar los MIME TYPE que queremos cachear y aplicarles un tiempo.

A2592000 = 2.592.000 segundos = 30 días

Podemos cachear: PDF, SVG, imágenes, vídeo, audio, texto (TXT), CSS, JavaScript, fuentes...

Un poco de explicación

La primera petición de cada elemento devuelve un código HTTP 200 (OK).

A partir de las siguientes, devuelve un código HTTP 304 (Not Modified).

CDN

Content Delivery Network

¿Qué es?

Las redes de distribución de contenidos están pensadas para reducir la latencia, los tiempos de respuesta, a la hora de servir contenidos en distintas zonas geográficas.

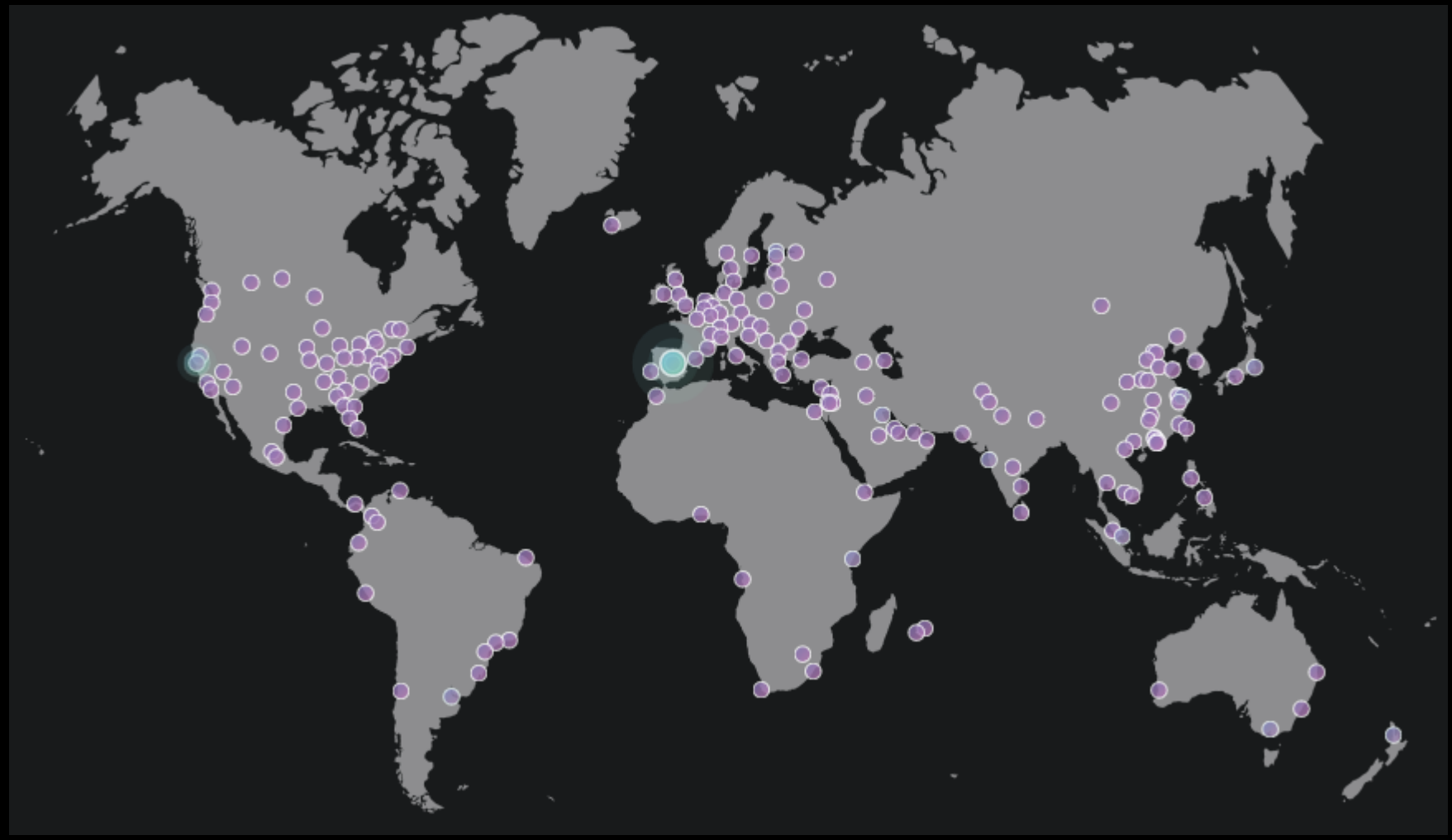
¿Qué es?

Si tu proyecto WordPress es internacional, sin duda es una buena opción para, al menos, contenidos estáticos como JavaScript o CSS además de imágenes y contenidos media.

Si tu proyecto es bastante local, lo mejor es disponer de una infraestructura muy bien conectada en ese país.

Caso y uso habitual

Cloudflare



Caso y uso habitual

¿Dónde están tus usuarios?

Focalízate en aquellos lugares / que se lleven el 80% de tu tráfico.

Si la mayor parte del tráfico es local (de un mismo país) puedes montarte “tu propia CDN”.

¿Vale la pena usar una CDN en España? Si tu alojamiento web está físicamente en España y tus usuarios en España, la respuesta es no (en general).

Invalidación

Si vas a usar una CDN recuerda utilizar plugins de invalidación.

Si cambias un contenido cacheable, dile al proveedor que lo limpie de su caché y que la próxima visita lo vuelva a pedir y almacenar.

Este es uno de los mayores errores de aquellos que usan CDN, que no invalidan.

Un poco de explicación

Cuando se configura una CDN todo el tráfico pasa por ese proveedor.

El sistema, en base a unas reglas, decide si se devuelve lo cacheado o se va al servidor a buscar al servidor principal.

En el caso de WordPress, en general, sólo se cachean los estáticos.

Extra

Un artículo sobre configuración de CDN, pero sólo para los estáticos.

Cómo configurar, bien, una CDN en WordPress
www.casares.blog/configurar-cdn-wordpress/

Caché de página

en servidor web o proxy

¿Qué es?

Cuando se hace una solicitud de una página, por ejemplo la principal de un blog, se ejecutan varios procesos en PHP, se hace el cálculo para la recuperación de los contenidos en la base de datos, se calculan configuraciones personalizadas y finalmente se pinta por pantalla.

¿Por qué no mostrar la información calculada en la primera de esas dos ocasiones?

¿Qué es?

La caché de página o en proxy almacena esa copia pre calculada de la página en un servidor intermedio.

Esta capa intermedia puede ser un *proxy*, habitualmente gestionado por herramientas como nginx (reverse proxy) o Varnish Cache que almacenan una copia de la solicitud y la sirven hasta que el sistema queda invalidado de forma manual o automáticamente cuando llega al tiempo de expiración.

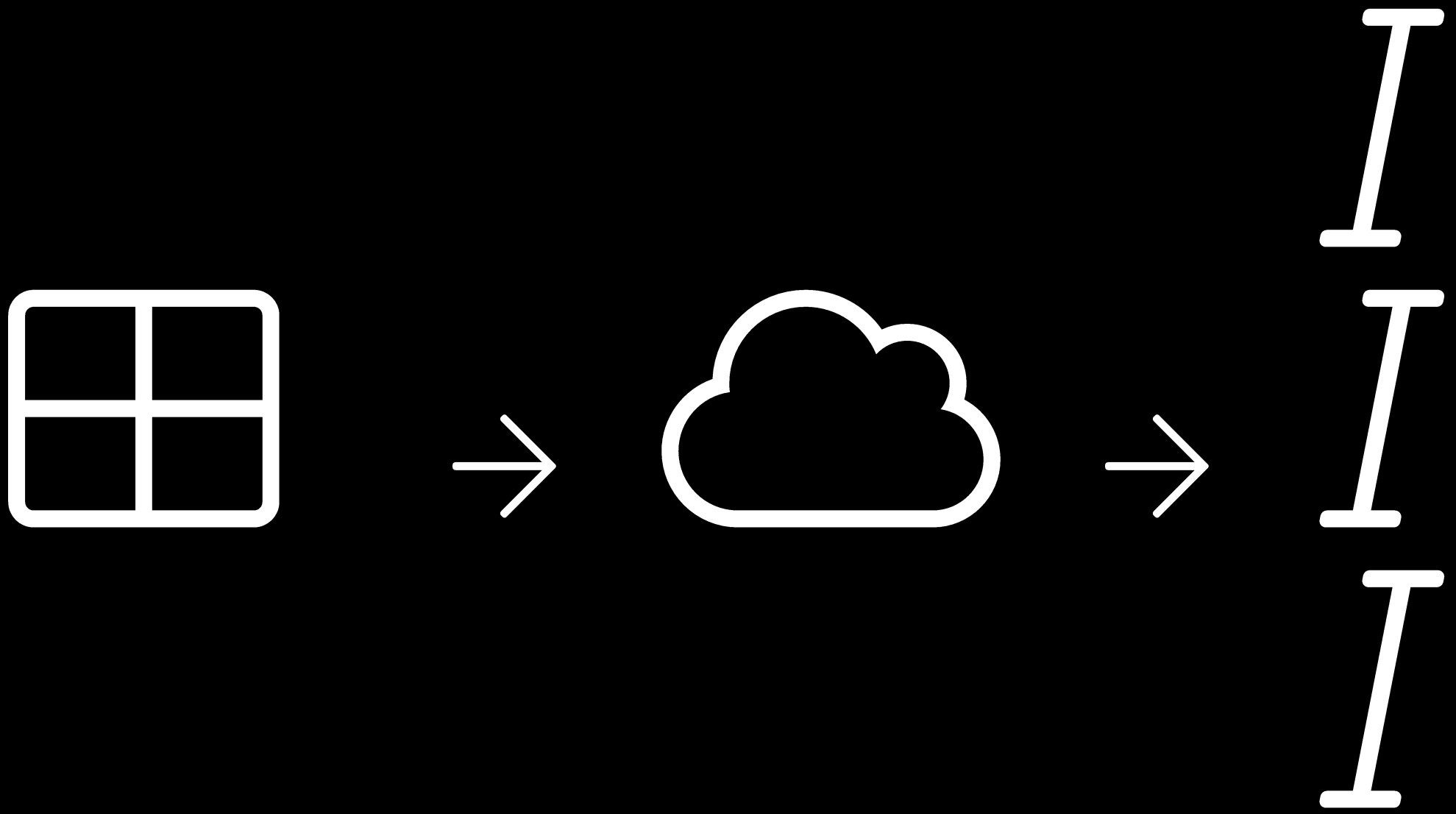
Caso y uso habitual

Si tienes picos de tráfico esta es una de las opciones principales de caché.

No recomendable para páginas que cambian “muchas veces por minuto” (¿conoces muchas de ellas?)

Imagínate que sale tu web por la tele o que tienes un efecto Menéame, con este sistema puedes servir 1.000 req/s sin despeinarte.

Requiere de un sistema de invalidación (manual o automática).



Un poco de explicación

La idea es “interceptar” las peticiones que realizan los usuarios.

La primera petición va al servidor y se calcula, yendo lento. Se almacena en la caché.

Las siguientes peticiones se sirven pre calculadas hasta que se cumple la regla de invalidación.

Un poco de explicación

En general es necesario el uso de un servidor externo (Varnish, nginx...).

Se puede configurar en la propia máquina o en una externa.

Por rendimiento, a veces es mejor tener un servidor de 1 CPU, 1 GB RAM y 10 GB SSD junto al servidor web.

Caché de página

estática o por PHP

¿Qué es?

Es la caché de la que habitualmente se habla cuando se habla de caché de WordPress.

permite la posibilidad de realizar la copia en el propio servidor web; no es la opción más rápida pero siempre será mejor que tener que calcular todo.

La mayoría de plugins de caché que hay para WordPress utilizan este sistema.

Caso y uso habitual

Úsalo, siempre. Confíguralo bien.

El sistema cuando se llama la primera vez crea un fichero HTML en el sistema de ficheros (normalmente en el /wp-content/cache/).

Cuando se hace una petición se busca ese fichero, sino, se llama al sistema normal.

Algunos plugins interesantes para gestionar la caché pueden ser: WP Super Cache, WP Fastest Cache, W3 Total Cache, Simple Cache.

Simple Cache + nginx

Existe un plugin muy simple de cache llamado Simple Cache. Este plugin gestiona la cache y sus cambios y crea una estructura de URL directamente sobre el sistema de ficheros.

```
/wp-content/cache/simple-cache/example.com/hello-world/index.html
```

Simple Cache + nginx

Por defecto nginx hace llamadas así:

```
location / {
  try_files $uri $uri/ /index.php?$args;
}
```

Y tendríamos que hacer llamadas así:

```
location / {
 try_files try_files "/wp-content/cache/simple-
cache/${http_host}${request_uri}index.gzip.html"
$uri $uri/ /index.php?$args;
}
```

Un poco de explicación

En los casos en los que se generen ficheros físicos en el servidor, hay que configurar el servidor web para que intente hacer primero la llamada al fichero físico (sin pasar por PHP) y, si no lo encuentra, ir al sistema normal (index.php).

Los plugins suelen llevar explicaciones sobre cómo aplicar estos cambios.

Opcode caché

Caché de PHP

¿Qué es?

Cada vez que llega una petición a tu servidor web, PHP ha de ejecutarse y calcular todo, pero PHP permite un sistema interno de caché de operaciones, lo que significa que se almacena una copia de cada ejecución en memoria o en disco.

Si está activo y se ejecuta de nuevo la misma operación, el sistema aprovechará este sistema para devolver el cálculo mucho más rápido, ya que no ha de calcular todo por completo.

Caso y uso habitual

Úsalo si tu proveedor lo tiene.

Si no lo tiene y dispones de acceso al servidor, configúralo.

Algunos plugins que pueden ser útiles: OPcache Reset, WP OPcache, OPCache Scripts.

Cómo instalarlo

Ubuntu:

```
$ apt -y install php7.3-opcache
```

CentOS:

```
$ yum -y install php73-php-opcache
```

NOTA: puede variar depende del repo desde el que hayas hecho tu instalación.

Parada técnica

Vamos a ponernos en situación

10 capas de caché

- Navegador
- CDN
- Página (proxy y servidor)
- PHP (Opcache)

¿Qué queda?

- Objetos / Transient API / Base de Datos
- Fragmentos
- Ficheros

Caché de objetos

¿Qué es?

Hace muchas versiones de WordPress que tenemos la posibilidad de almacenar algunos elementos en la llamada caché de objetos.

Estos objetos principalmente suelen ser elementos pre calculados asociados a consultas.

Habitualmente lo que se suele intentar cachear son los resultados de consultas a la base de datos.

Caso y uso habitual

Gracias a este sistema y con la ayuda de servidores de almacenamiento externos como memcached o Redis podemos almacenar los datos calculados en unos sistemas pensados para una lectura muy rápida sin tener que almacenarlos en la base de datos.

De esta forma, el sistema no tendrá que volver a ejecutar los cálculos sino que serán leídos directamente de un almacenamiento.

Caso y uso habitual

Lo interesante de este sistema es que no cachea toda la página calculada, sino partes de ella, lo que hace que en páginas que no se pueden cachear por completo, sí que lo hagan ciertas partes que no tienen porqué cambiar.

Algunos plugins que pueden ser útiles: Redis Object Cache, WP Redis, WP Memcached Manager.

Redis

En caso de tener un VPS o dedicado, ten un servidor Redis (aunque sea pequeño).

Primero lo instalamos, por ejemplo en CentOS:

```
$ yum -y install http://rpms.remirepo.net/enterprise/remi-release-7.rpm
$ yum -y upgrade
$ yum -y --enablerepo=remi install redis
```

Redis

Posteriormente lo configuramos.

```
$ vim /etc/redis.conf

maxmemory 512mb
maxmemory-policy allkeys-lru

$ systemctl restart redis.service
```

Redis

El problema (o no). ¿Tienes varios WordPress?
Que no se machaquen entre ellos.

```
define( 'WP_CACHE_KEY_SALT', ‘random12ab34cd:');
```

O aplica algo genérico para todos:

```
define( 'WP_CACHE_KEY_SALT', $_SERVER['HTTP_HOST'].':');
```

memcached

¡Atención!

Recuerda que si instalas memcached, por defecto viene con su acceso externo abierto y sin contraseña, por lo que cualquiera puede *floodear* tu sistema.

```
$ vim /etc/memcached.conf
-l 127.0.0.1
-U 0
$ service memcached restart
```

Caché de fragmentos

¿Qué es?

Este tipo de caché permite disponer de caché por partes dentro del conjunto de una misma página.

El sistema más conocido es el ESI (Edge Side Includes) y que, con sistemas externos que lo soporta, permitirían disponer de varias partes cacheadas y fusionar antes de ofrecer al cliente final.

Caso y uso habitual

¿Ninguno?

Uno de los casos más habituales es “el carrito de un comercio electrónico”.

En general, cuando estás en una tienda lo único que cambia entre usuarios es el “numerito” de cuántos productos tienes metidos en la cesta. El resto de la web es igual para todos.

Ese bloque de la cesta es lo que se podría cachear.

WordPress y ESI

WordPress por defecto no incorpora este sistema, aunque sí que se puede gestionar de forma parcial con sistemas como LiteSpeed Cache o Varnish Cache.

El resultado

Pruebas y comparativas

Infraestructura inicial:

-1 CPU / 2 GB RAM / 10 GB SSD

-Ubuntu 18.04.2 LTS

-Nginx 1.16.0

-PHP (FPM) 7.3.7

-MariaDB 10.4.6

-Redis 4.0.9

Pruebas y comparativas

Para hacerlo más real, vamos a cargar “elementos de test”.

http://wptest.io/

Así tendremos imágenes, y algunos contenidos que hagan que parezca más real.

Configuraremos, para hacerlo más pesado, 100 contenidos en la página principal.

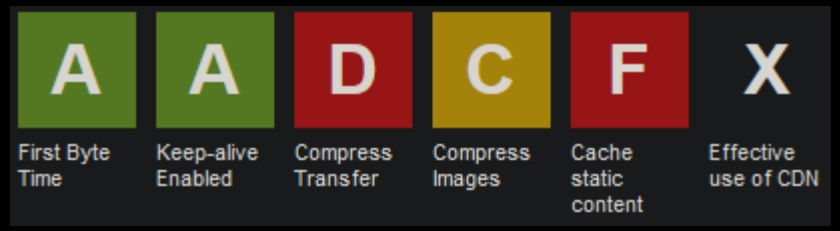
Las pruebas de análisis serán con webpagetest.org desde España.

Prueba a página principal

No es óptimo porque hay muchas llamadas a Twitter, Facebook, etc...

Prueba 1: con Opcache

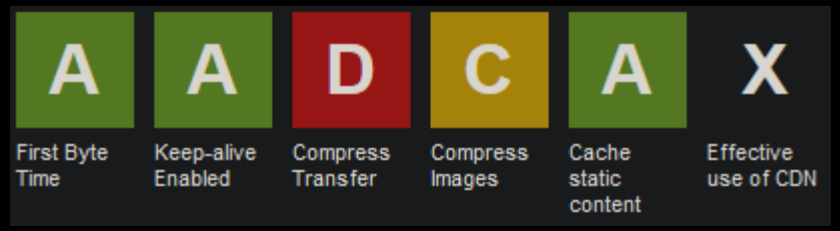
Resultado del análisis.



							Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Speed Index	Last Painted Hero	First Interactive (beta)	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View	2.461s	0.351s	1.500s	1.680s	1.500s	> 1.902s	2.461s	70	1,444 KB	3.013s	74	1,447 KB	$$$--
Repeat View	2.450s	0.192s	1.500s	1.620s	1.500s	> 1.642s	2.450s	8	40 KB	2.484s	9	41 KB	

Prueba 2: activamos caché navegador

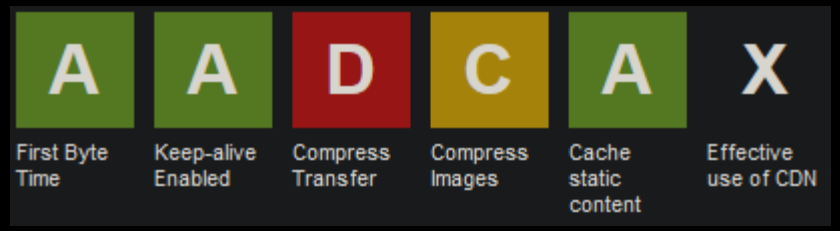
Resultado del análisis.



							Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Speed Index	Last Painted Hero	First Interactive (beta)	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View	2.517s	0.189s	1.500s	1.692s	1.500s	> 2.358s	2.517s	71	1,446 KB	2.831s	74	1,447 KB	$$$--
Repeat View	2.239s	0.190s	1.100s	1.283s	1.100s	> 1.380s	2.239s	8	40 KB	2.279s	9	41 KB	

Prueba 3: activamos Redis

Resultado del análisis.



							Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Speed Index	Last Painted Hero	First Interactive (beta)	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View	2.263s	0.208s	1.300s	1.544s	1.300s	> 1.657s	2.263s	71	1,446 KB	2.612s	74	1,447 KB	$$$--
Repeat View	2.259s	0.224s	1.200s	1.448s	1.200s	> 1.461s	2.259s	8	40 KB	2.294s	9	41 KB	

Prueba 4: activamos caché de página

Resultado del análisis.

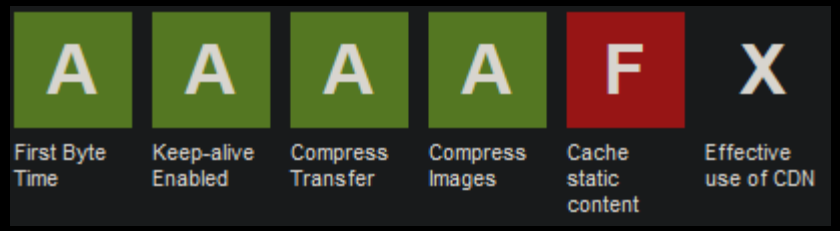
							Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Speed Index	Last Painted Hero	First Interactive (beta)	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View	2.134s	0.123s	1.100s	1.288s	1.100s	> 1.709s	2.134s	70	1,444 KB	2.637s	74	1,447 KB	$$$--
Repeat View	2.210s	0.113s	1.100s	1.222s	1.100s	> 1.595s	2.210s	8	40 KB	2.244s	9	41 KB	

Prueba a post galería

Una entrada con 5 imágenes

Prueba 1: con Opcache

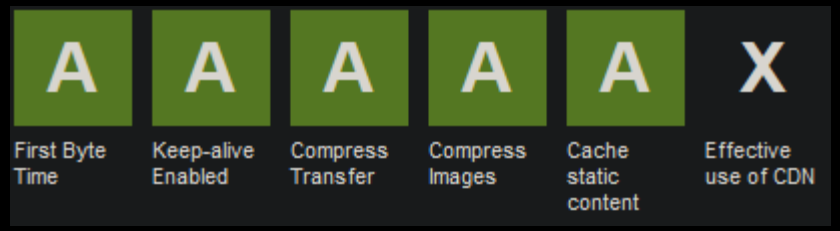
Resultado del análisis.



						Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Speed Index	Last Painted Hero	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View	0.632s	0.121s	0.600s	0.600s	0.600s	0.632s	14	90 KB	0.662s	15	91 KB	$----
Repeat View	0.582s	0.124s	0.600s	0.600s	0.600s	0.582s	1	6 KB	0.650s	2	7 KB	

Prueba 2: activamos caché navegador

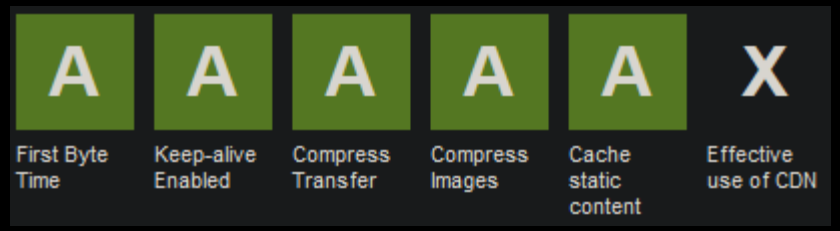
Resultado del análisis.



						Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Speed Index	Last Painted Hero	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View	0.693s	0.122s	0.600s	0.609s	0.700s	0.693s	14	90 KB	0.720s	15	91 KB	$----
Repeat View	0.595s	0.127s	0.600s	0.600s	0.600s	0.595s	1	6 KB	0.628s	2	7 KB	

Prueba 3: activamos Redis

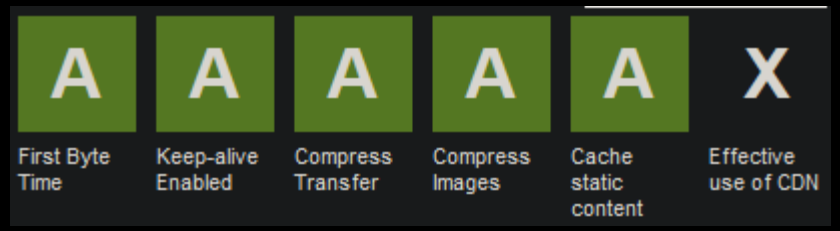
Resultado del análisis.



						Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Speed Index	Last Painted Hero	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View	0.659s	0.140s	0.600s	0.645s	0.700s	0.659s	14	90 KB	0.691s	15	91 KB	$---
Repeat View	0.741s	0.128s	0.700s	0.709s	0.800s	0.741s	1	6 KB	0.770s	2	7 KB	

Prueba 4: activamos caché de página

Resultado del análisis.



						Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Speed Index	Last Painted Hero	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View	0.571s	0.100s	0.600s	0.600s	0.600s	0.571s	14	90 KB	0.604s	15	91 KB	$----
Repeat View	0.544s	-	0.500s	0.509s	0.600s	0.544s	0	0 KB	0.581s	1	1 KB	

¡No es para tanto!

¿Seguro?

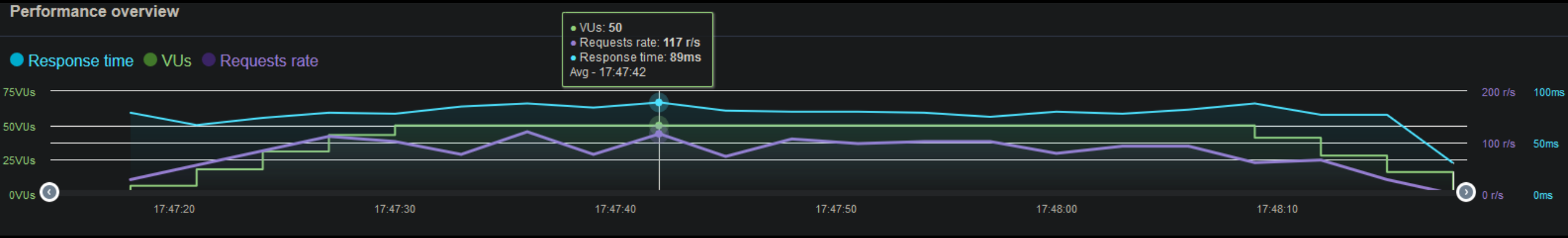
Y si hacemos... 50 peticiones/minuto

Con una petición es posible que no se note al fin y al cabo, está la máquina libre… pero y si hacemos muchas peticiones concurrentes?

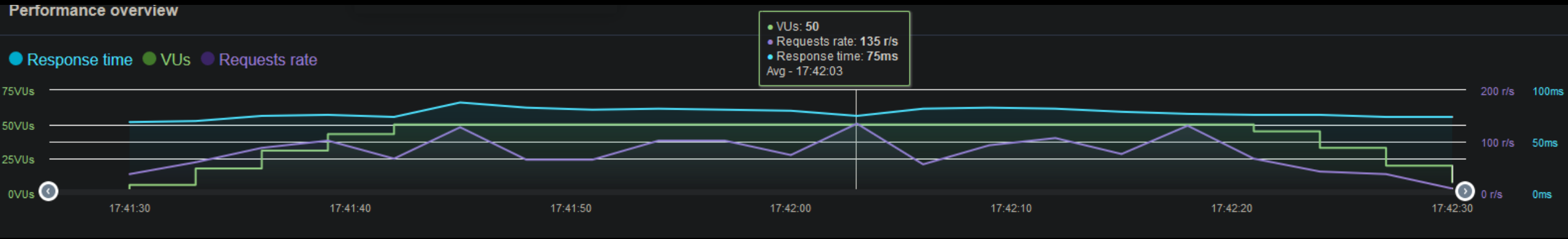
Para ello usaremos LoadImpact.

Sin caché vs. Con caché

Sin caché:



Con caché:



Conclusión

Activando simplemente 4 capas de caché sin necesidad de montar ninguna cosa muy grande, se pueden conseguir mejoras de rendimiento importantes.

Recuerda activar, al menos:

-Caché de navegador

-Caché de PHP (Opcache)

-Caché de objetos (Redis)

-Caché de página

Gracias

javiercasares.com

javier@casares.org

¿Necesitas ayuda con tus sistemas? brutal.systems

WordPress para SysAdmins wpalojamiento.com