Matteo Enna
#WCTRN23

Velocizzare WordPress, **Possiamo farlo con Varnish Cache!**

Argomenti

Come funziona il nostro sito, il protocollo HTTP

Prestazioni

Varnish

Proxy e Reverse Proxy

Funzionamento

ESI e VCL

Matteo Enna

Lavoro come **Back End Developer** per Tannico.

Lavoro principalmente con il PHP per lavorare con CMS Open Source, ultimamente con WordPress e Magento.

Piccolo ripasso di reti

È NOIOSO MA SARÒ BREVE

Cosa succede quando entro in un sito?

COSA FACCIAMO

Proviamo a digitare sul browser:
https://torino.wordcamp.org/

COSA SUCCEDE

Il nostro browser fa questa richiesta
GET /schedule HTTP/1.1
Host: https://torino.wordcamp.org/
Accept: text/html
User-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36

QUESTO È IL PROTOCOLLO HTTP!

Ora che succede?

WEB SERVER

Apache o NGINX ricevono la nostra richiesta

CODICE

Partendo dalle rotte indicate verrà eseguito del codice PHP

DATABASE

In quasi tutti i siti moderni per ogni rotta vengono fatte diverse Query

e poi...

RENDER

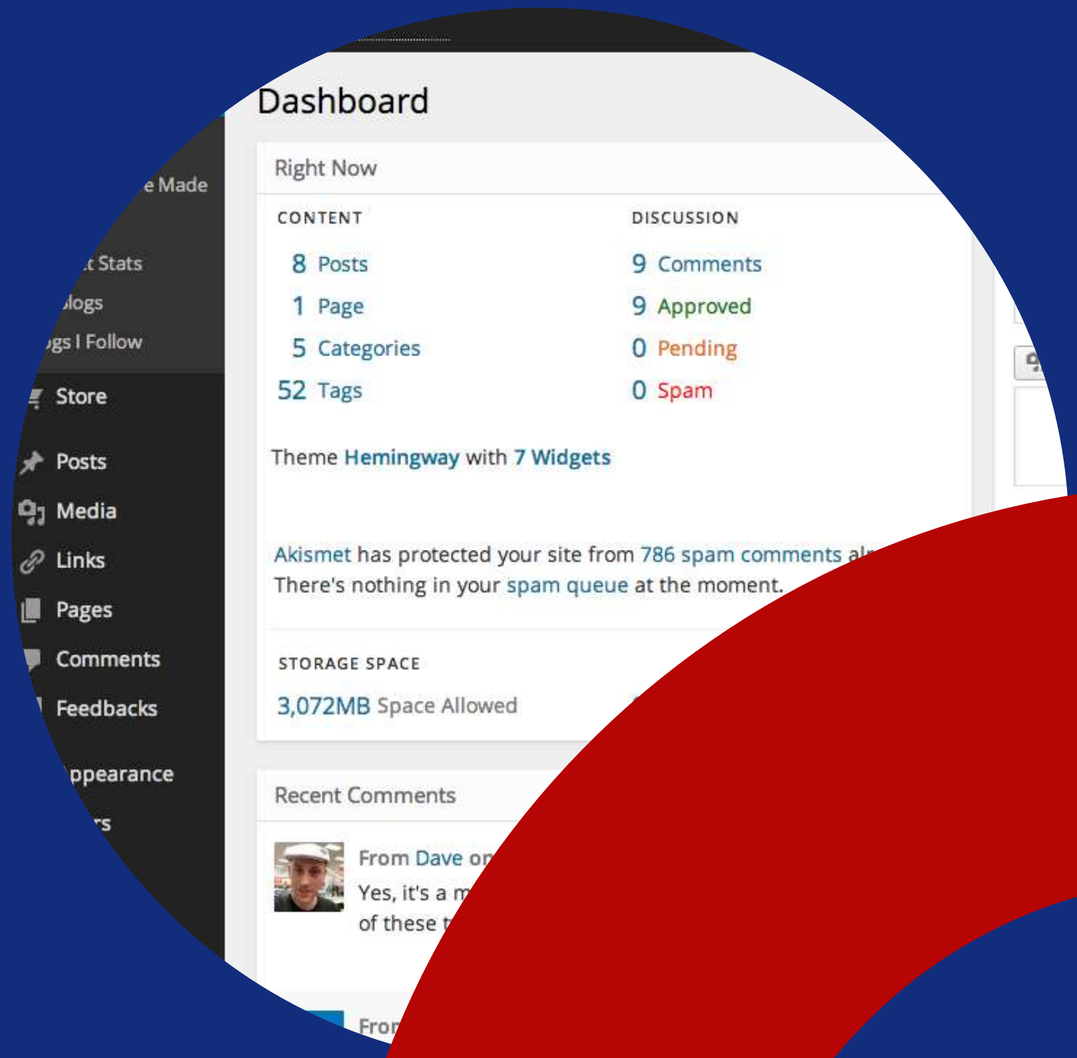
Viene generato l'HTML o il json che serve per il funzionamento della pagina.

MULTIMEDIALI

Ricordiamoci i fine js, css e tutte le immagini.

SIAMO NEL 2023

Ricordiamoci tutte le chiamate Ajax che vengono eseguite.



Esempio

Immaginiamo la home page di un WooCommerce...

Esempio

Immaginiamo la home page di un WooCommerce.
Magari durante il black friday!

COSA ABBIAMO?

- Una lunga pagina con tutte le principali promozioni;
- Dei bellissimi e grandissimi banner;
- Tante liste di prodotti e le loro immagini in evidenza;
- I nostri CSS e JS;
- Qualche ciclo for che scorre tutte le principali categorie e propone i migliori prodotti e le loro caratteristiche principali;
- Tantissimi utenti!
- Tantissimi clienti!
- Tantissimi carrelli!
- Tantissimi ordini!

SIAMO PRONTI?

Esempio

Immaginiamo la home page di un WooCommerce.
Magari durante il black friday!

SIAMO PRONTI?

- Ogni immagine sul sito è ottimizzata e usiamo il lazy load;
- Css e Js minificati ed inseriti correttamente;
- Abbiamo un Hosting adeguato;
- Usiamo una CDN;
- Cache del browser;
- Cache a livello di backend per query ed oggetti;
- Quello che possiamo rendere statico, facciamolo!

Dimenticavamo qualcosa?

TANTISSIMI UTENTI, TANTISSIMI CLIENTI, TANTISSIMI CARRELLI E TANTISSIMI ORDINI!

Tanti, tanti tanti, tanti

VISITA, REGISTRAZIONE, CARRELLO E ORDINE

Abbiamo visto prima che una semplice visita significa diverse chiamate HTTP, quindi dobbiamo passare per il Web Server, il PHP e le dovute query.

QUINDI...

Le nostre soluzioni sono tutte utili ed efficaci, ma possiamo fare altro?

E ora parliamo di Varnish

Varnish Cache è un web accelerator o HTTP accelerator rilasciato con una licenza libera.

MA FORSE "CACHE" È IL TERMINE SBAGLIATO!

Varnish è un reverse Proxy

...altro ripasso

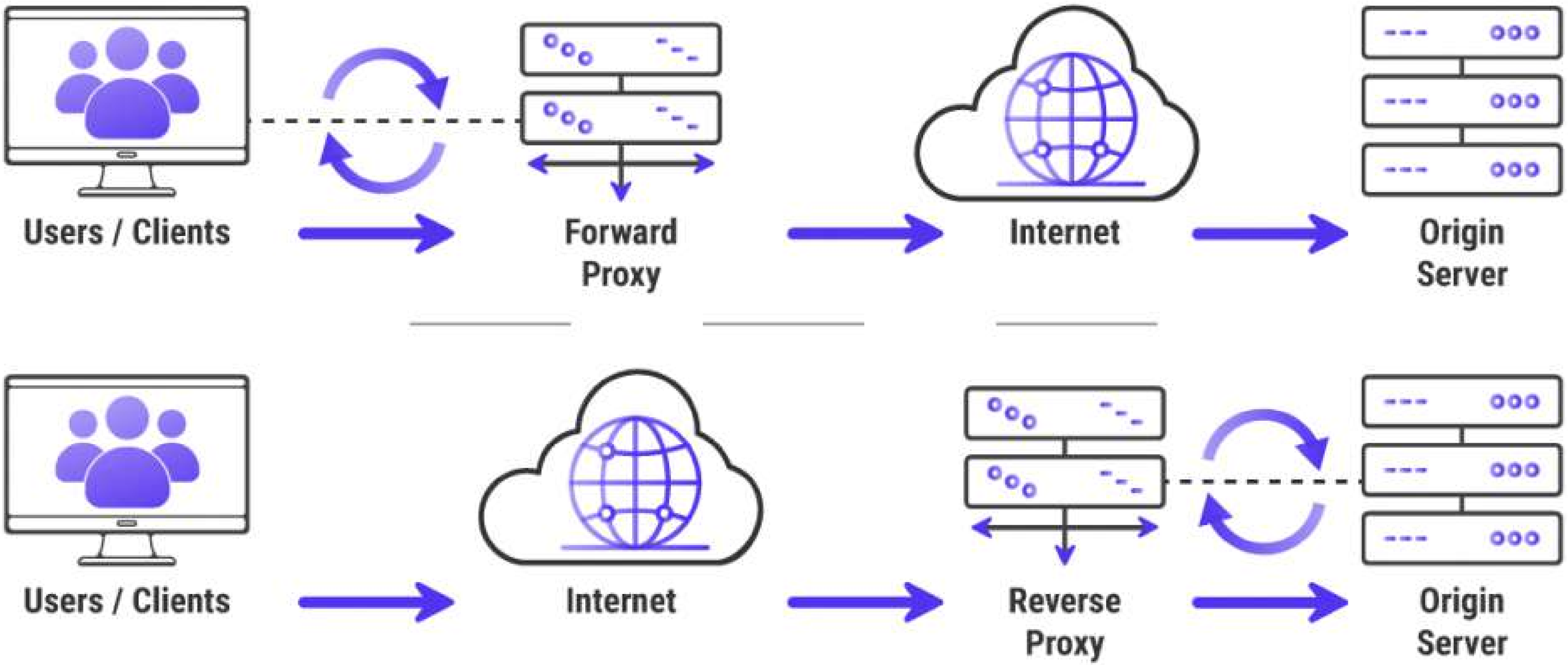
PROXY (FORWARD)

Un tipo di server che funge da **intermediario** per le richieste da parte degli utenti alla ricerca di risorse su altri server.

REVERSE PROXY

Come il Proxy Forward, si trova tra l' utente finale ed il server. Con la differenza che con il Forward lavoriamo tra il client ed internet, invece con il **Reverse** lavoriamo tra internet ed il server finale.

Forward Proxy vs Reverse Proxy



Credits: Vito Lavecchia - https://vitolavecchia.altervista.org/

Reverse Proxy

BILANCIO DEL CARICO

Se si dispone di più server web, il reverse proxy può instradare le richieste in base al carico di ciascun computer.

PRIVACY MIGLIORATA

L' uso di un reverse proxy nasconde le informazioni del server finale. Se qualcuno sta curiosando, vedrà solo fino al reverse proxy.

PIÙ CERTIFICATI SSL

Il reverse proxy può gestire le richieste HTTP in entrata e ottenere i dati non crittografati richiesti dal server finale.

CACHE

Un reverse proxy può memorizzare nella cache i contenuti statici e dinamici del tuo sito web, riducendo in tal modo il carico sul tuo server.

Come funziona

STEP 0

La richiesta raggiunge prima Varnish che controlla se la pagina che vogliamo visitare è già presente in memoria.

STEP 1

Se la pagina è in memoria, Varnish restituisce subito la pagina altrimenti passerà la richiesta al Web Server, solitamente Apache o NGINX

STEP 2

Il web server, processerà la richiesta e partiranno le varie funzioni del linguaggio di backend e varie query DB per poi preparare la pagina html

STEP 3

La pagina html così generata verrà inviata direttamente a Varnish che ne salverà il contenuto e lo invierà al client

Quindi

NESSUNA ELABORAZIONE

Se prima, il tempo era diviso tra invio ed elaborazione, una pagina già memorizzata su Varnish impiega solo il tempo di invio

PAGINE WEB MA ANCHE TANTO ALTRO

Abbiamo fatto un esempio con una pagina html, ma esistono tanti servizi che usano HTTP.

ABBIAMO DETTO PROTOCOLLO HTTP

Protocollo HTTP, significa anche tutte le architetture basa su di esse. Pensiamo per esempio cosa si può fare con SOAP e REST

CHE COSA CI STIAMO DIMENTICANDO?

Siamo nel 2022, le pagine web sono dinamiche. Se Varnish ci restituisce la stessa pagina dov'è la dinamicità?

Alcune situazioni

Banner o elementi che gli editor cambiano spesso

Contenuti creati dagli utenti

Elementi come il carrelo

Ok, possiamo usare/abusare di Ajax ma forse ci sono soluzioni più adatte!

ESI

COS'È

Edge Side Include è un linguaggio per includere frammenti di pagine Web in altre pagine Web. Come un 'istruzione include HTML che funziona su HTTP.

E NON SOLO

Nella maggior parte dei siti web molti contenuti sono condivisi tra le pagine. Rigenerare questo contenuto per ogni visualizzazione di pagina è uno spreco ed ESI cerca di risolverlo permettendoti di decidere la politica della cache per ogni frammento individualmente

Dichiarazione ESI

Una volta che Varnish prenderà dalla memoria un file, non elaborerà le istruzioni ESI nei commenti HTML come:
<!-- ESI -->

Che può contenere:

- esi:include
- esi:remove

UN ESEMPIO BASH + HTML

BASH

```
#!/bin/sh
echo 'Content-type: text/html'
echo ''
date '+%Y-%m-%d %H:%M'
```

HTML

```
<HTML>
   <BODY>
      The time is: at this very moment.
   </BODY>
</HTML>
```

UN ESEMPIO BASH + HTML

```
sub vcl_fetch {
    if (req.url == "/test.html") {
        set beresp.do_esi = true; /* Do ESI processing */
        set beresp.ttl = 24 h; /* Sets the TTL on the HTML above */
    } elseif (req.url == "/cgi-bin/date.cgi") {
        set beresp.ttl = 1m; /* Sets a one minute TTL on */
        /* the included object */
    }
}
```

VCL

VARNISH CONFIGURATION LANGUAGE

Uno dei motivi che hanno reso popolare Varnish Cache è VCL, un linguaggio di configurazione molto flessibile che ricorda il C.

UN ESEMPIO VCL

```
sub vcl_recv {
    if(req.method == "PURGE" || req.method == "BAN") {
        return(purge);
    }
    if(req.method != "GET" && req.method != "HEAD") {
        return(pass);
    }
    if(req.url ~ "^/products/[0-9]+/"){
        set req.http.x-type = " product" ;
    }
}
```

VLC

HOOK E SUBROUTINE

Il VCL non funziona come un normale linguaggio, in cui si può prendere un progetto nuovo ed iniziare a creare le regole. Con VCL si lavora su Subroutine, intervenendo su hook per modificare semplicemente il suo normale flusso.

ESEMPI

vcl_recv: Eseguito all'inizio di ogni richiesta. **vcl_pipe**: Passa la richiesta direttamente al backend senza preoccuparti della memorizzazione nella cache.
vcl_pass: Passa la richiesta direttamente al backend. Il risultato non è memorizzato nella cache.
vcl_hit: Chiamato quando una ricerca nella cache ha esito positivo.

LOAD BALANCING

COME CI COMPORTIAMO SE IL NOSTRO SITO È DISTRIBUITO SU PIÙ SERVER?

LOAD BALANCING

Varnish Cache ha nativamente un sistema di bilanciamento delle chiamate mantenendo le funzionalità di salvataggio ed invio delle pagine memorizzate. La distribuzione dei carichi utilizza gli standard più famosi:

- Round robin;
- Weighted round robin
- Dynamic round robin

FreeBSD (2 clausole)

La ridistribuzione e l' uso in forma di codice sorgente e binaria, con o senza modifiche, sono permesse purché siano rispettate le seguenti condizioni:

1. Le ridistribuzioni di codice originale devono conservare la nota di copyright sopra riportato, questa lista di condizioni e il seguente disconoscimento.
2. Le ridistribuzioni in formule binarie devono riprodurre la nota di copyright sopra riportata, questa lista di condizioni e il seguente disconoscimento nella documentazione e/o altri materiali forniti con la distribuzione.

APPROFONDIMENTI

LINK UTILI

https://varnish-cache.org/docs/3.0/index.html
https://docs.varnish-software.com/
https://github.com/varnish
https://en.wikipedia.org/wiki/Varnish_(software)
https://stackoverflow.com/questions/tagged/varnish

Contatti:

- matteo.enna89@gmail.com
- matteoenna.it
- In tutti i social
- **Qui in giro :)**

Grazie