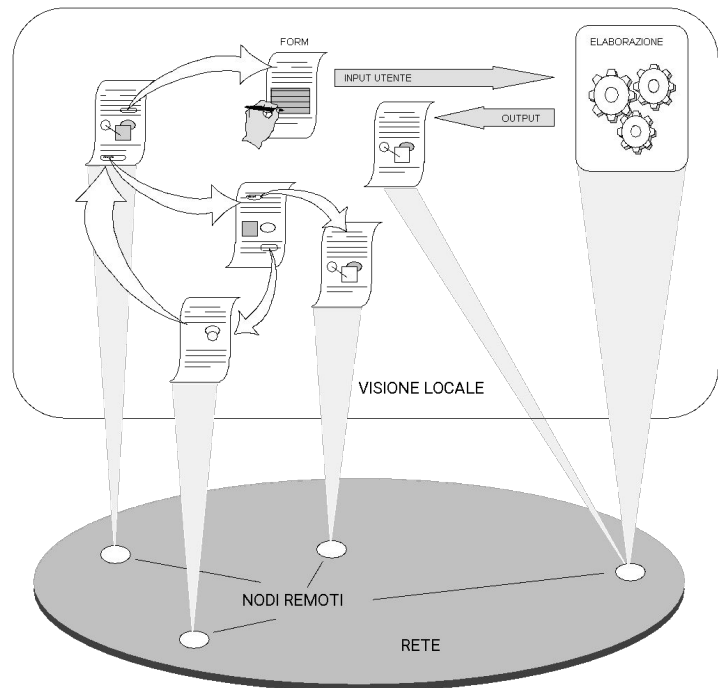


World-Wide Web (WWW)

Strutturazione in forma ipertestuale delle informazioni presenti su Internet
(**trasparenza** dalla allocazione delle informazioni)
(**semplicità** di utilizzo)

Iper testi: Vannevar Bush (1946)
WWW: Tim Berners-Lee (1989)



Il World Wide Web - 1

WWW

WWW è un insieme di protocolli e di standard usati per organizzare e accedere alle informazioni presenti sulla rete Internet.

Finalità

- Trasparenza accesso e allocazione
- Mezzo per reperire informazioni
- Supporto al lavoro collaborativo
- Nuovo mercato commerciale

Ragioni del suo successo

- Diffusione globale e capillare di Internet
- Uguale interfaccia utente indipendente dal tipo di Server, di Client o di macchina (=> facilità d'uso)
- Sistema modulare (=> scalabilità)
- Facilità di navigazione (uso di **browser** come Firefox, Chrome, Safari, ...)

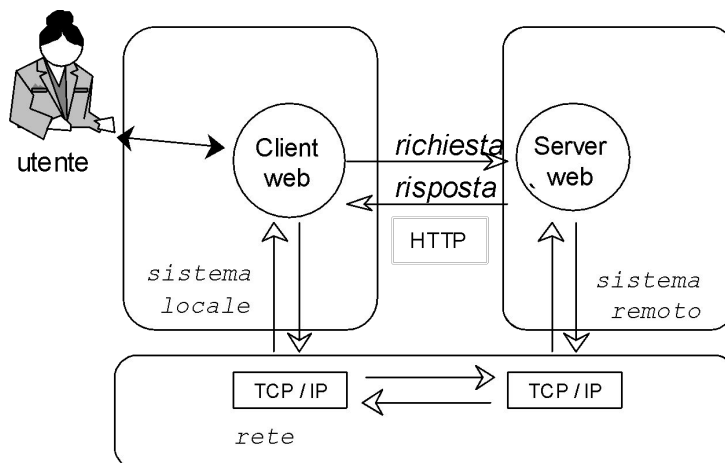
Il World Wide Web - 2

Principali standard del WWW

- Sistema di indirizzamento globale, **URL** (Universal Resource Locator).
- Protocollo applicativo, **HTTP** (HyperText Transfer Protocol).
- Linguaggio di strutturazione dei dati, **HTML** (HyperText Markup Language).
- Fogli di stile per formattazione dati, **CSS** (Cascading Style Sheets).
- Interfacce per applicazioni Web (**CGI** ed evoluzioni).
- Piattaforma di programmazione e run time environment per esecuzione programmi lato Client, **JavaScript**.
- Moltissimi altri standard, in continua evoluzione (XHR, WebSocket, WebRTC, HTTP/2, HTTP Live Streaming, Dynamic Adaptive Streaming over HTTP, etc.).

Il World Wide Web - 3

Architettura WWW



Modello fondamentale del Web è il **Client/Server**

Uso di TCP, porta standard **80**

Componenti

- Browser Web (Client per presentazione/gestione richieste)
- Server Web (recupero e invio informazioni)
- Applicazioni Web (esecuzione remota)
- Run time environment, application framework (sia Server che Client side), ...

Il World Wide Web - 4

URL - Uniform Resource Locator

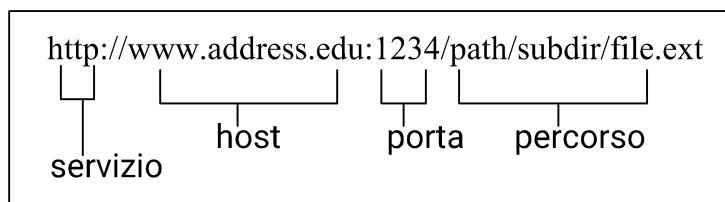
Nomi unici per le risorse del sistema, indicati da Client per recupero Server.

Uniform Resource Locator (URL):

- nodo contenente la risorsa (*documento o dati*)
- protocollo di accesso alla risorsa (*e.g. http, ftp*)
- numero di porta TCP (*porta di default del servizio*)
- localizzazione della risorsa nel Server.

`<protocollo>[://<host>] [:<porta>] [<percorso>]`

Sono riconosciuti i servizi Internet e relativi protocolli (http, ftp, telnet, nntp, mail...)



Il World Wide Web - 5

HTTP (HyperText Transfer Protocol) 1.0 [RFC 1945]

HTTP è il **protocollo applicativo** per il trasferimento delle informazioni tra Client e Server. Caratteristiche della versione 1.0:

- **request/response** (semplice richiesta e ricezione di dati)
- **connessione non permanente** (in HTTP 1.0 la connessione TCP viene mantenuta solo per il tempo necessario a trasmettere i dati).
- **stateless** (non mantiene nessuna informazione tra una richiesta e la successiva, *a differenza di FTP che mantiene una sessione durante la connessione e ricorda, per es., la dir in cui l'utente si trova*)

Motivazioni scelte progettuali:

Diversi link nella stessa pagina possono puntare a pagine di altri Server e si ritenne inutile stabilire delle connessioni più durevoli e costose.

Problemi:

- in HTTP 1.0, una pagina con molte figure viene trasferita con una separata connessione TCP per ognuna di esse
- la modalità di funzionamento stateless rappresenta un problema per l'implementazione di transazioni commerciali

Il World Wide Web - 6

HTTP request/response

Formato del messaggio di **richiesta**:

indirizzo del server	metodo di richiesta	campi opzionali	path+ nome file (in generale, i dati)
----------------------	---------------------	-----------------	---------------------------------------

Metodo di richiesta:

- GET richiesta di leggere una pagina Web
- HEAD richiesta di leggere l'header di una pagina Web
- PUT / PATCH richiesta di pubblicare / modificare una pagina Web
- POST append di dati (di solito HTML form)
- DELETE rimuove una pagina Web

HEAD per ottimizzare Web caching: i browser hanno cache con pagine visitate e ogni pagina è valida per la durata del suo "cache timeout" (specificato dal Server). Inoltre, prima di ri-caricare una pagina scaduta i browser verificano se è ancora valida usando HEAD (header contiene data ultima revisione documento).

Campi opzionali:

from: identità dell'utente richiedente (debole forma di autenticazione degli accessi)

referer: il documento contenente il link che ha portato al documento corrente

Il World Wide Web - 7

HTTP request

Esempio messaggio di **richiesta**:

```
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh;... )... Firefox/51.0
Accept: text/html,application/xhtml+xml,..., */*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=-12656974
Content-Length: 345

-12656974
(more data)
```

Request headers

General headers

Entity headers

Courtesy: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages>

Il World Wide Web - 8

HTTP – Codifica risposte

Come in FTP e SMTP, le **risposte** sono codificate con 3 cifre:

La prima cifra codifica le interazioni

- 1xx Informational responses (100 – 199)
- 2xx Successful responses (200 – 299)
- 3xx Redirection messages (300 – 399)
- 4xx Client error responses (400 – 499)
- 5xx Server error responses (500 – 599)

In HTTP, al codice di risposta seguono informazioni sull'oggetto (metadati come: data di modifica, tipo di oggetto. Ogni tipo di oggetto come immagini, HTML, audio, attiva gestione specifica da parte del Client) e l'oggetto stesso.

HTTP response

Esempio messaggio di **risposta**:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Wed, 10 Aug 2016 13:17:18 GMT
Etag: "d9b3b803e9a0dc6f22e2f20a3e90f69c41f6b71b"
Keep-Alive: timeout=5, max=999
Last-Modified: Wed, 10 Aug 2016 05:38:31 GMT
Server: Apache
Set-Cookie: csrftoken=.....
Transfer-Encoding: chunked
Vary: Cookie, Accept-Encoding
X-Frame-Options: DENY
```

The diagram illustrates the structure of an HTTP response. It shows a list of headers and their corresponding categories:

- Response headers:** Access-Control-Allow-Origin, Connection, Date, Etag, Keep-Alive, Last-Modified, Server, Set-Cookie, X-Frame-Options.
- Entity headers:** Content-Encoding, Content-Type.
- General headers:** Transfer-Encoding, Vary.

(body)

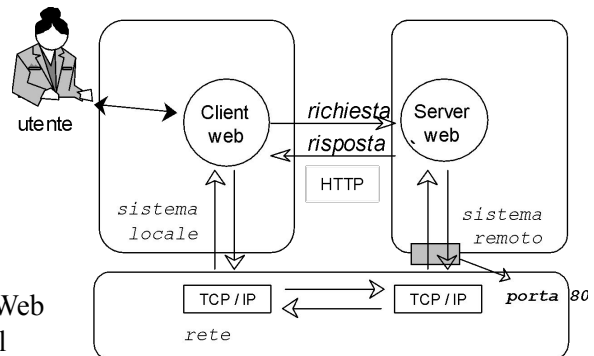
Courtesy: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages>

Interazione Client browser / Server Web

1. Il browser (HTTP Client) riceve un URL dall'utente
2. Il browser richiede al DNS di risolvere il nome della macchina specificata nell'URL
3. Il DNS risponde con l'indirizzo IP
4. Il browser stabilisce una connessione TCP sulla porta 80 dell'indirizzo IP del Server
5. Il browser manda una richiesta con metodo GET percorso_pagina_web
6. Il Server risponde con la pagina Web richiesta

Attenzione: in HTTP 1.0 la presenza in una pagina Web di altri oggetti (immagini, applets, ecc.) costringeva il Client ad aprire una **differente connessione** per ognuno degli oggetti necessari.

Il Client HTTP può eseguire richieste seguendo protocolli differenti (es. FTP) su porte differenti.



EVOLUZIONI DI HTTP

HTTP 1.1 è la release attuale di HTTP [RFC 7230-7237]. Ha introdotto il concetto di "connessioni persistenti", in cui **una singola connessione TCP è utilizzata per trasferimenti multipli** (molte GET).

HTTPS fornisce un trasferimento sicuro delle informazioni, garantendo la riservatezza delle informazioni scambiate in una sessione HTTP. Fa uso di un sottostante canale cifrato realizzato con TLS.

WAP (Wireless Access Protocol) è uno stack di protocolli per fornire l'accesso Web a dispositivi portatili, palmari, telefoni cellulari, etc.

Non è solo un protocollo, ma tutta una suite per affrontare diversi aspetti. Contiene un linguaggio di markup (WML), un protocollo di trasporto, un protocollo di sicurezza, etc. Dei WAP proxy sono incaricati di trasformare le esistenti pagine HTML in pagine WML per la visualizzazione sui dispositivi mobili. WAP è stato un fallimento, poiché partiva da assunzioni di scenario rivelatesi sbagliate.

HTTP/2 e HTTP/3 nuove versioni di HTTP, recentissime e ancora in fase di adozione iniziale. Protocolli binari, risp. su TCP e QUIC, con supporto a request multiplexing.

HTML (HyperText Markup Language)

HTML è un linguaggio di specifica delle informazioni che deriva da SGML (Standard Generalized Markup Language). È un **markup language** (come TeX, RTF, etc.).

I linguaggi markup definiscono delle aree di testo attraverso l'uso di **tag**. I diversi tag determinano come verrà trattato il testo incluso.

Caratteristica innovativa di **HTML** è la possibilità di creare e definire dei collegamenti (link) verso altri documenti, anche remoti.

Tag definiti **funzionalmente**, non visualmente.

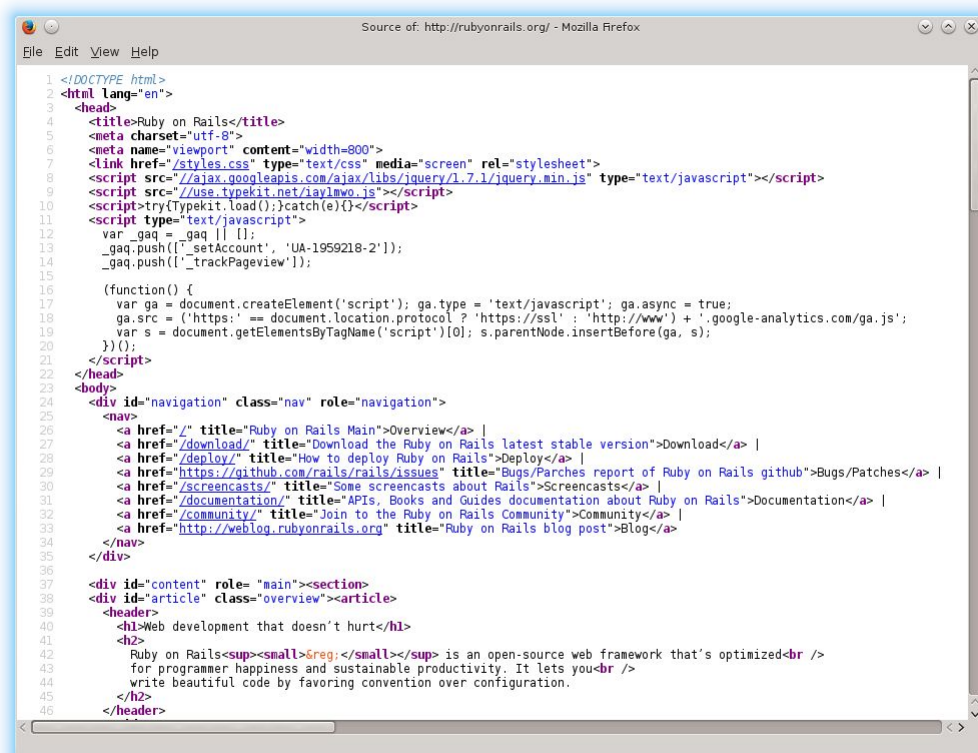
`<H1>testo</H1>` definisce **testo** come header 1

header 1 è visualizzato in modo specifico da ogni browser (per esempio con un font large e bold, oppure capitalized e centered).

Esistono editor per creare documenti **HTML** (possibile traduzione automatica di documenti di testo in formato HTML).

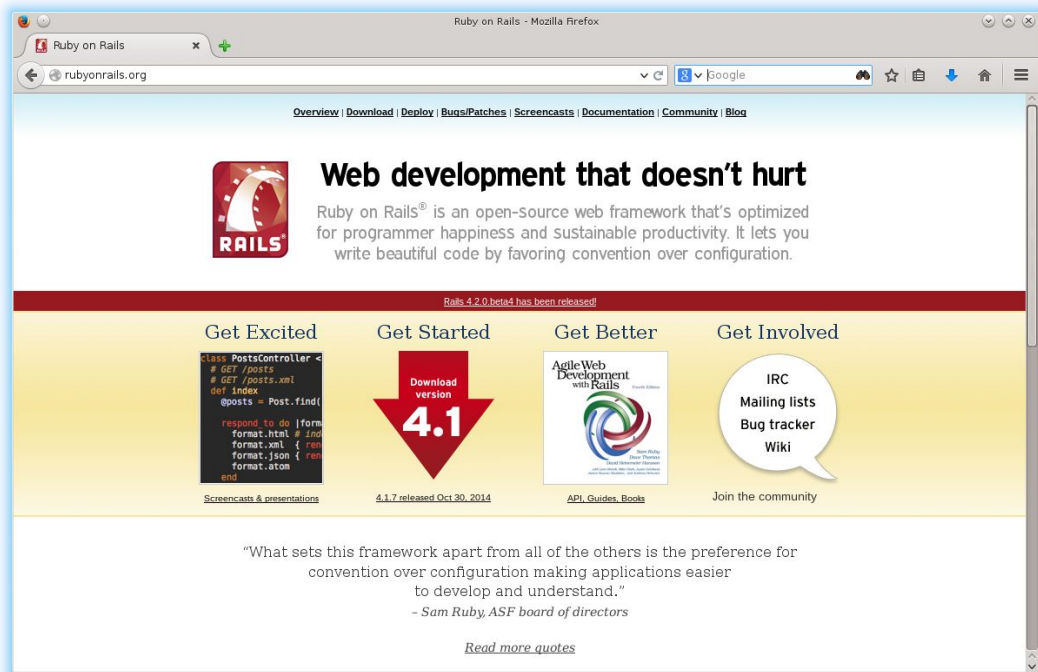
Altri linguaggi markup: **WML**, **XML**, etc.

Esempio di pagina HTML (codice)



```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <title>Ruby on Rails</title>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=900">
7     <link href="/styles.css" type="text/css" media="screen" rel="stylesheet">
8     <script src="/ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js" type="text/javascript"></script>
9     <script src="/use_typekit.net/ia1mwo.js"></script>
10    <script>try{Typekit.load();}catch(e){}</script>
11    <script type="text/javascript">
12      var _gaq = _gaq || [];
13      _gaq.push(['_setAccount', 'UA-1959218-2']);
14      _gaq.push(['_trackPageview']);
15
16      (function() {
17        var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async = true;
18        ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') + '.google-analytics.com/ga.js';
19        var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(ga, s);
20      })();
21    </script>
22  </head>
23  <body>
24    <div id="navigation" class="nav" role="navigation">
25      <nav>
26        <a href="/" title="Ruby on Rails Main">Overview</a> |
27        <a href="/download/" title="Download the Ruby on Rails latest stable version">Download</a> |
28        <a href="/deploy/" title="How to deploy Ruby on Rails">Deploy</a> |
29        <a href="https://github.com/rails/rails/issues" title="Bugs/Patches report of Ruby on Rails github">Bugs/Patches</a> |
30        <a href="/screencasts/" title="Some screencasts about Rails">Screencasts</a> |
31        <a href="/documentation/" title="APIs, Books and Guides documentation about Ruby on Rails">Documentation</a> |
32        <a href="/community/" title="Join to the Ruby on Rails Community">Community</a> |
33        <a href="http://weblog.rubyonrails.org" title="Ruby on Rails blog post">Blog</a>
34      </nav>
35    </div>
36
37    <div id="content" role="main"><section>
38      <div id="article" class="overview"><article>
39        <headers>
40          <h1>Web development that doesn't hurt</h1>
41          <h2>
42            Ruby on Rails<sup><small><strong></small></sup> is an open-source web framework that's optimized<br />
43            for programmer happiness and sustainable productivity. It lets you<br />
44            write beautiful code by favoring convention over configuration.
45          </h2>
46        </headers>
```

Esempio di pagina HTML (visualizzazione)



Il World Wide Web - 15

Cascading Style Sheets (CSS)

CSS è un linguaggio per la formattazione di documenti HTML.

Formattazione viene separata dalla presentazione per velocizzare lo sviluppo e facilitare il riuso del codice.

La presentazione è diventata così importante nel Web moderno che sono nati diversi framework per facilitare lo sviluppo del codice CSS (per es., Sass, post-CSS, ecc.) e la realizzazione di pagine Web “responsive”, ovverosia con una formattazione in grado di adattarsi naturalmente a schermi di diversa orientazione e dimensioni, tramite CSS (per es., Bootstrap, Foundation, ecc.).

Lo standard CSS è stato aggiornato molte volte nel corso degli anni. Attualmente siamo alla versione 3, che propone un’architettura modulare e una serie di funzioni avanzate (per es., transform) che non sono ancora supportate da tutti i maggiori browser Web.

Il World Wide Web - 16

Dynamic Web Technologies

Le prime pagine apparse sul Web erano scritte in HTML ed erano statiche e immutabili. Evoluzione del Web verso una **maggiore dinamicità**.

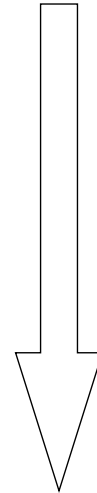
Tipi di pagine:

Pagine statiche. Sono documenti Web il cui contenuto è definito al momento della loro creazione e non cambia più. Qualunque richiesta di pagina statica fornisce sempre la stessa risposta.

Pagine dinamiche. Sono documenti Web che vengono creati dal Web Server solo quando un browser li richiede. In tale caso il Server Web attiva un'applicazione che crea il documento in modo dinamico e lo invia al browser. Il contenuto di un documento dinamico può variare di richiesta in richiesta.

Pagine attive. Una pagina attiva ha un contenuto che NON è completamente specificato dal Server. È un pezzo di programma che viene mandato dal Server al browser, dove viene eseguito localmente. Una pagina attiva può interagire con l'utente e con altre risorse ed entità, per cui il suo contenuto non è mai statico.

Web 1.0



Web 2.0

Il World Wide Web - 17

Dynamic Web: Vantaggi e Svantaggi

	Vantaggi	Svantaggi
Pagine Statiche	Semplicità Affidabilità Performance (e caching)	Poco flessibili (modificate a ogni cambiamento)
Pagine Dinamiche	Per informazioni con elevata frequenza di cambiamento (e-commerce, CMS, etc.) Per il browser sono come delle pagine statiche	Computazione sul Server (Server più potenti) Prestazioni più basse (tempo di elaborazione, no cache) Richiedono buone capacità di programmazione Interazione "clicca e aspetta"
Pagine Attive	Interazione continua con l'utente Minore carico sul Server Modelli di programmazione innovativi e interessanti (Web components, declarative, etc.)	Riversano costi sui Client (browser più sofisticati e macchine più potenti) Richiedono ottime capacità di programmazione Problemi di portabilità (coinvolgono tutti Client) Problemi di sicurezza (eseguono sui Client) Soluzione di elevatissima complessità

Il World Wide Web - 18

Dynamic Web: Implementazione

Il supporto alla dinamicità (pagine dinamiche e attive) richiede delle estensioni all'architettura Client/Server del Web.

Nuovi componenti sul lato Client e/o sul lato Server per supportare pagine dinamiche e attive.

Tecnologie di supporto per pagine dinamiche:

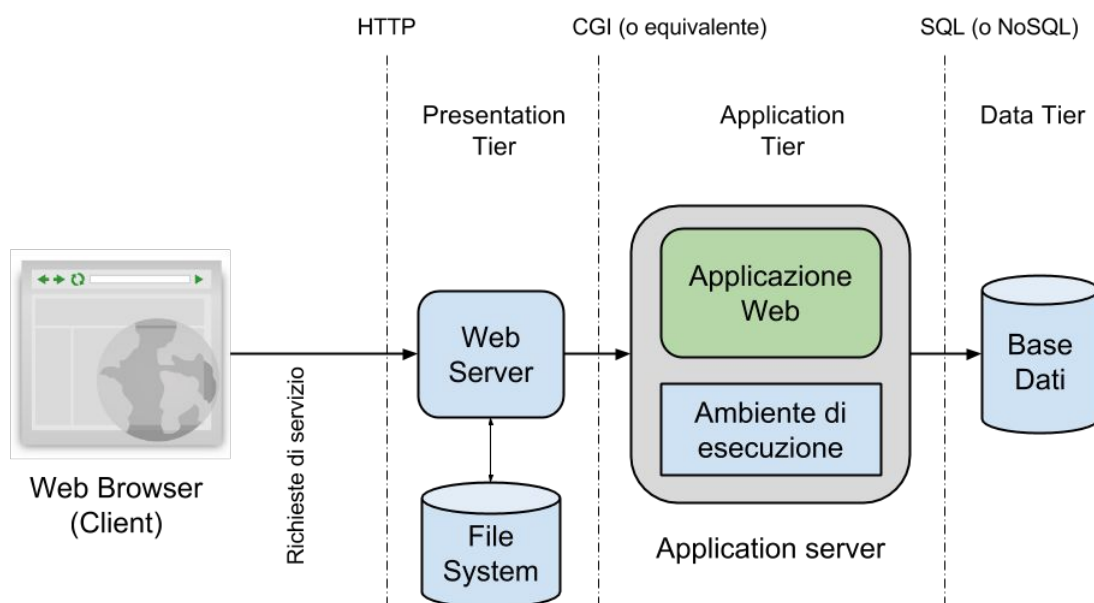
Applicazioni CGI

Piattaforme di sviluppo applicazioni Web (PHP, Ruby on Rails, JSP, ASP, ...)

Tecnologie di supporto per pagine attive:

JavaScript, XMLHttpRequest, ...

Architettura Estesa del Client/Server Web



I tre tier più in dettaglio

Presentation Tier

Il Web server effettua il parsing e la sanitizzazione delle richieste HTTP, serve i contenuti statici (es. pagine statiche, immagini, CSS, ecc.), e inoltra le richieste di pagine dinamiche all'Application Server. (Nonostante la maggior parte degli Application Server sia in grado di servire direttamente richieste HTTP, è sempre opportuno adottare un Web Server per ragioni di performance e sicurezza.)

Application Tier

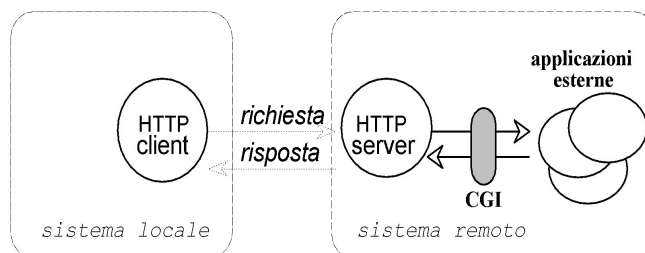
L'Application Server si occupa di eseguire la business logic dell'applicazione Web e di generare le pagine dinamiche.

Data Tier

Il DataBase fornisce le funzioni di gestione dei dati richieste dall'applicazione Web.

Tecnologie a supporto pagine dinamiche

Common Gateway Interface (CGI)



CGI è uno **standard per interfacciare** un Server Web con applicazioni esterne (residenti sulla macchina Server).

CGI fornisce all'utente la capacità di eseguire un'applicazione sulla macchina Server remota.

CGI furono prima tecnologia per pagine dinamiche. Importanza storica, molte altre tecnologie ora in uso.

Programmazione CGI

CGI permette a utenti di eseguire applicazione sul nodo dove risiede Server Web.

Applicazioni CGI possono essere scritte in qualunque linguaggio, come C/C++, Perl, TCL, qualunque shell Unix, Visual Basic, AppleScript, ...

Si attiva un programma Unix, modello filtro con variabili di ambiente predefinite.

CGI come **Interfaccia** tra Server Web e applicazione CGI, attenzione a:

- variabili di ambiente
- linea di comando
- standard input
- standard output

Programmazione CGI

Variabili d'ambiente, utilizzate dal Server per dare informazioni di servizio ad appli CGI:

SERVER_SOFTWARE, nome e versione del Server HTTP

SERVER_NAME, nome nodo Server o suo indirizzo

GATEWAY_INTERFACE, versione interfaccia CGI cui il Server aderisce

REQUEST_METHOD, metodo invocato nella richiesta

REMOTE_HOST, REMOTE_ADDR, AUTH_TYPE, REMOTE_USER,

CONTENT_TYPE, CONTENT_LENGTH,

Linea di comando

per richieste di tipo ISINDEX, per ricerche di testo nei documenti. Le parole da ricercare sono inserite dal Server sulla linea di comando dell'applicazione CGI. (compatibilità)

Standard input

il Server ridirige sull'ingresso della applicazione CGI i dati ricevuti dal Client (browser). Il numero di byte è nella variabile d'ambiente CONTENT_LENGTH, il tipo dei dati MIME nella CONTENT_TYPE.

Standard output

l'applicazione CGI manda il risultato dell'elaborazione sullo standard output verso il Server, che a sua volta prepara i dati e li spedisce al Client.

Client HTTP => Server HTTP => Applicazione CGI

Tipicamente, un utente inserisce dati via **form**. Attributi del **form tag**.

```
<TITLE>Esempio di Form </TITLE>
<H1>Esempio di Form </H1>
<FORM METHOD="POST"
ACTION="http://www.ing.unife.it/cgi-bin/post-query">
Inserisci del testo: <INPUT NAME="entry"> e premi per invio: <INPUT
TYPE="submit" VALUE="Invio"> </FORM>
```

Dove:

ACTION URL di chi processa la query

METHOD metodo usato per sottomettere il form:

POST il form con i dati è spedito come data body

GET il form con i dati è spedito attaccato all'URL action?name=value&name=value

caso GET `http://www.ing.unife.it/cgi-bin/get-query?entry=testo`

caso POST `http://www.ing.unife.it/cgi-bin/post-query`
e come data body: `entry=testo`

Applicazione CGI => Server HTTP

Applicazione CGI usa **standard output** per mandare al Server i dati. I dati sono identificati da un header.

Esempio:

invio di un full document di risposta con il corrispondente MIME type (text/html, text/plain per testo ASCII, etc.), come:

```
Content-type: text/html
<HTML><HEAD>
<TITLE>output di HTML da script CGI</TITLE>
</HEAD><BODY>
<H1>titolo</H1>
semplice <STRONG>prova</STRONG>
</BODY></HTML>
```

Esempio di Applicazione CGI in Linguaggio C

```
#include <stdio.h>
...

int main(int argc, char *argv[]) {
    int cl;
    /* generazione del full document di risposta */
    printf("Content-type: text/html");
    cl = atoi(getenv("CONTENT_LENGTH"));
    for (x=0; cl && (!feof(stdin)); x++) {
        ...
        /* elaborazione dell'input (stdin) */
    }
    printf("<H1>Query Results</H1>");
    printf("You submitted ...");
    for(x=0; x <= m; x++)
        printf(".....", ... , ....);
}
```

Il World Wide Web - 27

Esempio di Applicazione CGI in Ruby

```
#!/usr/bin/env ruby

require 'json'
require 'cgi'

cgi = CGI.new
# CGI tries to parse the request body as form parameters so
# a blob of JSON awkwardly ends up as the one and only
# parameter key.
request = JSON.parse(cgi.params.keys.first)

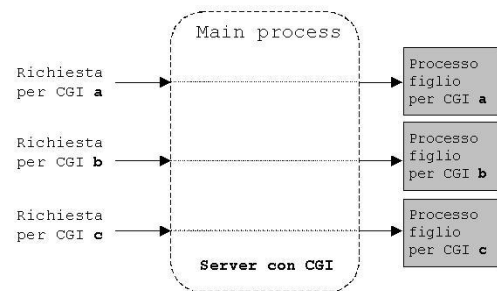
serve(request) # magic happens right here

cgi.out("status" => "OK", "type" => "text/plain",
        "connection" => "close") do
    "Success"
end
```

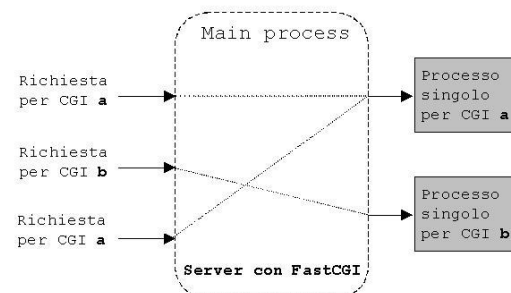
Il World Wide Web - 28

CGI: problemi, limiti e costi

A ogni richiesta, viene attivato un **processo** per gestione CGI (overhead di generazione)



Evoluzioni: FAST CGI prevedono un processo già attivo per ogni servizio CGI specificato



Il World Wide Web - 29

Oltre CGI: Interfacce Moderne tra Tier 1 e 2

Con lo sviluppo delle tecnologie Web, sono nate interfacce molto più moderne, sofisticate e performanti di CGI per la connessione tra Web Server e Application Server:

- Varianti avanzate di CGI (es. FastCGI Process Manager di PHP)
- HTTP stesso, con Application Server che opera in modalità reverse proxy (es. Rails con Unicorn o Puma, Node.js, ecc.)
- Moduli di estensione dei Web server specifici per una particolare piattaforma di sviluppo (es. mod_php, mod_python, mod_uwsgi, Phusion Passenger, ecc.)

All'interno dell'Application Server, le interfacce tra l'ambiente di esecuzione e le applicazioni vere e proprie sono tipicamente platform-specific (es. Rack per Ruby, WSGI per Python, ecc.)

Il World Wide Web - 30

Esempio di configurazione per stack nginx + Rails/Puma

```
upstream myapp { # configurazione connessione a Puma
    # Puma accetta richieste HTTP su socket Unix
    server unix:///var/www/myapp/tmp/puma.sock;
}

server { # configurazione server nginx
    listen 80;
    server_name myapp.com; # URL applicazione
    keepalive_timeout 5;

    # percorso file statici della mia applicazione
    root /var/www/myapp/public;

    location / {
        # redirigo richieste HTTP a upstream myapp
        proxy_pass http://myapp;
        break;
    }
    ...
}
```

Per un esempio più completo si veda: <https://github.com/puma/puma/blob/master/docs/nginx.md>

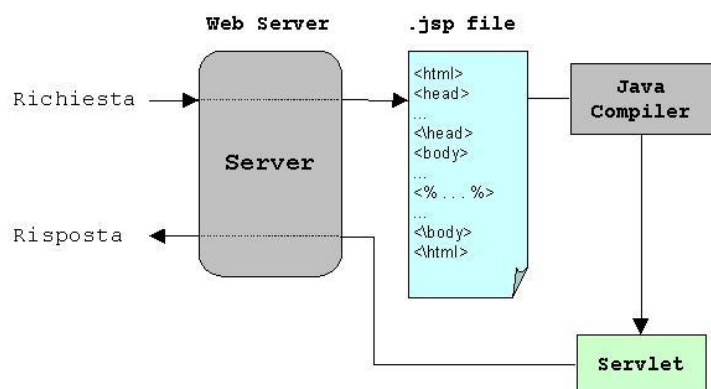
Il World Wide Web - 31

Java Server Pages (JSP)

JSP è stata una delle prime tecnologie per la realizzazione di pagine dinamiche.

Parte della pagina HTML viene specificata utilizzando codice Java. Queste parti (o meglio componenti) sono passate alla macchina virtuale integrata nel Server, permettendo lo sviluppo di pagine HTML dinamiche (usando servlet).

1. Pagine JSP sono estensioni di HTML che contengono parti dinamiche specificate in **Java** tra tag `<% %>`
2. Il codice Java passato alla macchina virtuale integrata nel Server per produrre una **servlet**
3. Si **compila** «on the fly» il codice Java e si **attiva** la servlet
4. Si produce la **pagina HTML risultato**



Il World Wide Web - 32

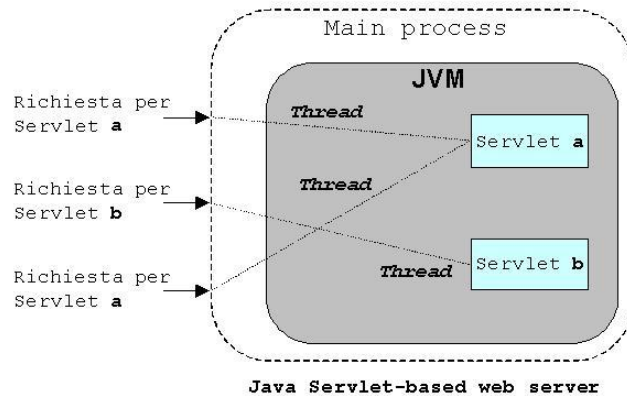
Java Servlet

Le **servlet** sono estensioni di attività in esecuzione sul Server e integrabili facilmente con il Server Web (via JVM)

Le **servlet** sono componenti di codice Java residenti sul Web Server
se **invoke** producono attività nella JVM eseguendo come processi leggeri

+ Costi di attivazione molto limitati
+ Non usciamo dall'ambiente del Server
+ Possiamo gestire facilmente mutua esclusione o parallelismo

- Architettura **eccessivamente complicata** (almeno per le limitate funzioni che offre)
- Portabilità più teorica che pratica



Java Server Pages

come collante per unire:

- codice HTML
- componenti riusabili (*Enterprise JavaBeans*)
- applicazioni remote (servlet)
- codice Java
- script basati su linguaggio Java

Le **JSP** sono portabili in quanto non assumono una specifica architettura di Web Server (come le ASP), ma solo la presenza di una JVM.

Il mercato si è evoluto **enormemente** negli ultimi 10 anni e ha prodotto soluzioni significativamente più semplici ed eleganti rispetto a JSP (tecnologia fine anni '90).

Un esempio più recente: Roda - 1

Ecco un'esempio di applicazione Web minimale a la *studiare.unife.it* realizzata con il framework Roda (<https://github.com/jeremyevans/roda>) basato sul linguaggio Ruby:

```
require 'roda'; require 'sequel'
DB = Sequel.postgres('my_db', user: 'user', password: 'password',
                    host: 'localhost')

class MyESSE3 < Roda
  plugin :render
  plugin :rodauth { enable :login, :logout, :change_password }
  route do |r|
    r.rodauth; rodauth.require_authentication

    r.get "/voti" do # GET /voti
      @voti = DB["SELECT * FROM esami WHERE matricola = ?",
                rodauth.account_id ]
      view("voti") # carica template views/voti.erb
    end
  end
end

run MyESSE3.freeze.app
```

Il World Wide Web - 35

Un esempio più recente: Roda - 2

Ed ecco il template views/voti.erb:

```
<!doctype html>
<html lang="it">
<head>
  <title>I tuoi voti</title>
  <!-- ... eventuali tag meta e link a file .css e .js -->
</head>
<body>
  <h1>Ecco i tuoi voti</h1>
  <table>
    <tr><th>Insegnamento</th><th>Risultato</th></tr>
    <% @voti.each do |voto| %>
    <tr>
      <td><%= voto.nome_insegnamento %></td>
      <td><%= voto.risultato %></td>
    </tr>
    <% end %>
  </table>
</body>
</html>
```

Il World Wide Web - 36

Altre piattaforme

PHP

Tecnologia per generare pagine dinamiche, molto diffusa nel mondo Linux. (Tecnologia su cui è basato Facebook.)

Active Server Pages (ASP)

Definite da Microsoft per integrare HTML e componenti script (VB, C#). Supportate da Microsoft **IIS** Internet Information Server (non portabili).

Ruby on Rails

Framework che rappresenta un punto di riferimento tecnologico del mercato. Basato sul linguaggio Ruby, ha avuto il merito di introdurre innumerevoli innovazioni (MVC, ReST, ORM, DB migration, TDD/BDD, CSS preprocessing, asset pipeline, ecc.) che sono state successivamente recepite anche dai competitor.

Node.js

Piattaforma di nuova generazione, basata sull'uso del linguaggio JavaScript (lato Server) e di modelli di programmazione a eventi.

Tecnologie a Supporto di Pagine Attive – Java applet

Il linguaggio HTML è interpretato ... allo stesso modo, possiamo pensare a **linguaggi di scripting** intrinsecamente portabili e interpretati sul **Client**

Java applet

Le applet Java sono state la prima tecnologia che permetteva l'esecuzione di codice lato browser. Esse rappresentano però un enorme problema di sicurezza e sono fortunatamente state deprecate.

```
<applet code=Animator.class  
  codebase="../example"  
  width=55 height=68>  
  <param name=endimage value=10>  
  <param name=pauses value="2500|100">  
</applet>
```



Tecnologie a Supporto di Pagine Attive – JavaScript

È un linguaggio di scripting interpretato dal browser

Browser moderni offrono ambienti di programmazione JavaScript **molto sofisticati e performanti** (V8, SpiderMonkey, asm.js, etc.).

Il mercato si sta orientando verso la realizzazione di applicazioni JavaScript thick client, denominate “Single page Web application” (React, AngularJS, ecc.).

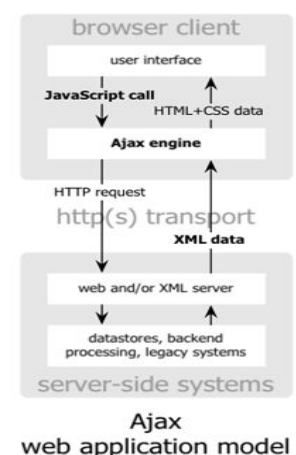
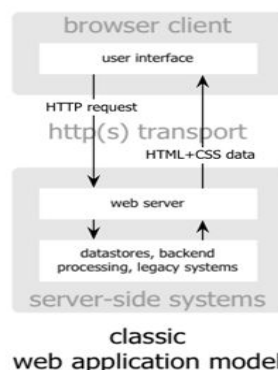


Il World Wide Web - 39

AJAX

AJAX è un mix di tecnologie per la creazione di pagine Web “attive”

- XMLHttpRequest: componente del browser che permette di inviare richieste (e ricevere risposte) HTTP in modo asincrono
- XML e JSON: linguaggi indipendenti dalla piattaforma per la descrizione della struttura delle informazioni (solo contenuti, non presentazione)
- Web Services: insieme di standard per favorire interoperabilità tra applicazioni (ReST)



In seguito a interazioni con l’utente (es. click, sottomissione form) o a risposte dal Server, lato Client **vengono invocate delle funzioni JavaScript pre-registrate come callback per gestire il corrispondente evento.**

Modello di programmazione a eventi, molto complicato. Per qualche semplice esempio, si veda: <https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener>

Il World Wide Web - 40

HTML 5

HTML 5 è la nuova versione di HTML, che introduce nuove funzioni per la realizzazione di applicazioni ricche e interattive aperte in un browser:

- riorganizzazione semantica tag contenuti (es. tag <article>, <section>, ecc.);
- nuovi tag per la navigazione;
- geolocalizzazione;
- canvas per il disegno 2D;
- nuovi tag multimediali <audio> e <video>;
- local storage;
- Web workers.

Molte delle nuove funzioni sono basate su o fortemente integrate con il supporto JavaScript del browser.

Per ulteriori informazioni: <http://diveintohtml5.info>

Il Web 2.0

La continua evoluzione delle tecnologie Web sta portando a un cambiamento di paradigma, dal Web come contenitore di informazioni al Web come piattaforma applicativa.

Le tecnologie usate nel Web 2.0 non sono innovative, ma sono evoluzione delle precedenti. Sono innovative le applicazioni sviluppate con tali tecnologie.

Il termine Web 2.0 non indica una nuova tecnologia, ma piuttosto vuole marcare un punto preciso di un'evoluzione che continua. Alcune differenze tra Web 1.0 e Web 2.0:

Web 1.0	Web 2.0
Architettura client/server	Architettura orientata ai servizi
Siti	Servizi
Navigazione di pagine	Uso di applicazioni
Clicca e aspetta	Asincrono
Unidirezionale	Bidirezionale
Directory (taxonomy)	Tagging (folksonomy)
Pubblicazione contenuti	Distribuzione contenuti

Lo stato nel Web

Un problema fondamentale per molte applicazioni Web è quello dello stato dell'interazione tra il Client (tipicamente il browser) e il Server. Si ricordi che il protocollo HTTP è stateless.

Per poter implementare funzioni come lo “shopping cart” in un'applicazione Web, è necessario introdurre il concetto di “sessione di navigazione” e gli strumenti per modificarne lo stato e conservarlo tra una richiesta HTTP e quella successiva.

Varie soluzioni al problema dello stato, realizzate a livello Client oppure Server.

Soluzioni Client-side (ogni Client provvede a memorizzare i dati di sessione necessari)
I browser memorizzano localmente una serie di attributi che forniscono automaticamente ai Server da cui li hanno ottenuti **per simulare lo stato della interazione**
Cookies, Hidden Form Field

Soluzioni Server-side (in cui il Server tiene traccia di tutte le sessioni aperte)
Il Server memorizza tutti i dati o gli oggetti che rappresentano le sessioni aperte, questo può diventare un collo di bottiglia per le prestazioni.
Uso di “session object” (JSP, Ruby on Rails, etc.)

Cookie

Lo stato sul Web è basato sul meccanismo dei Cookie, ovverosia un semplice DB (formato nome = valore) che viene continuamente scambiato tra Client e Server negli header HTTP dei messaggi.

Il cookie è inizializzato dal Server che lo invia al browser in una set-Cookie header line della HTTP response.

Nel successivo dialogo con il Server, il Client includerà sempre una Cookie header line nelle HTTP request inviate.

Client memorizza (con scadenza) gli attributi (**stato**) da usare per successive interazioni.
Cookie anche per specificare preferenze utente.

Spesso uno stesso Server ha molti cookie per pagina che vengono ripresentati solo al Cliente corretto.

Mantenuti dal Cliente, per esempio su memoria di massa. Cookies con scadenza e anche cifrati.

Soluzioni Client-side

Tutto lo stato nei Cookie

Posso pensare di salvare tutto lo stato della sessione nei Cookie.

Soluzione inefficiente e insicura (anche in caso di cifratura dei cookie, sto comunque immettendo informazioni sensibili in rete).

Hidden Form Field

In alternativa ai Cookie, posso utilizzare i campi HIDDEN previsti dal linguaggio HTML. In questi campi può essere memorizzata una coppia nome=valore che può essere palleggiata tra il Client e il Server per identificare una sessione di dialogo.

Soluzione limitata. YMMV.

Soluzioni Server-side

Session object

La maggior parte degli ambienti di programmazione per applicazioni Web fornisce un meccanismo (session object) per mantenere le informazioni di stato specifiche di una particolare sessione con un Client HTTP.

Le informazioni di ciascuna sessione di navigazione sono composte da:

- **session ID** univoco, memorizzato in un apposito Cookie, che viene salvato dal Client e incluso in ogni richiesta HTTP;
- **session object**, in cui sono memorizzate le informazioni di stato, che viene salvato lato server.

I session object sono salvati sul Server in uno specifico **session store**, ovverosia un semplice DB che contiene oggetti referenziabili tramite il rispettivo session ID.

Varie possibilità di definire il tempo di vita di ogni session object.

Tempo di caricamento

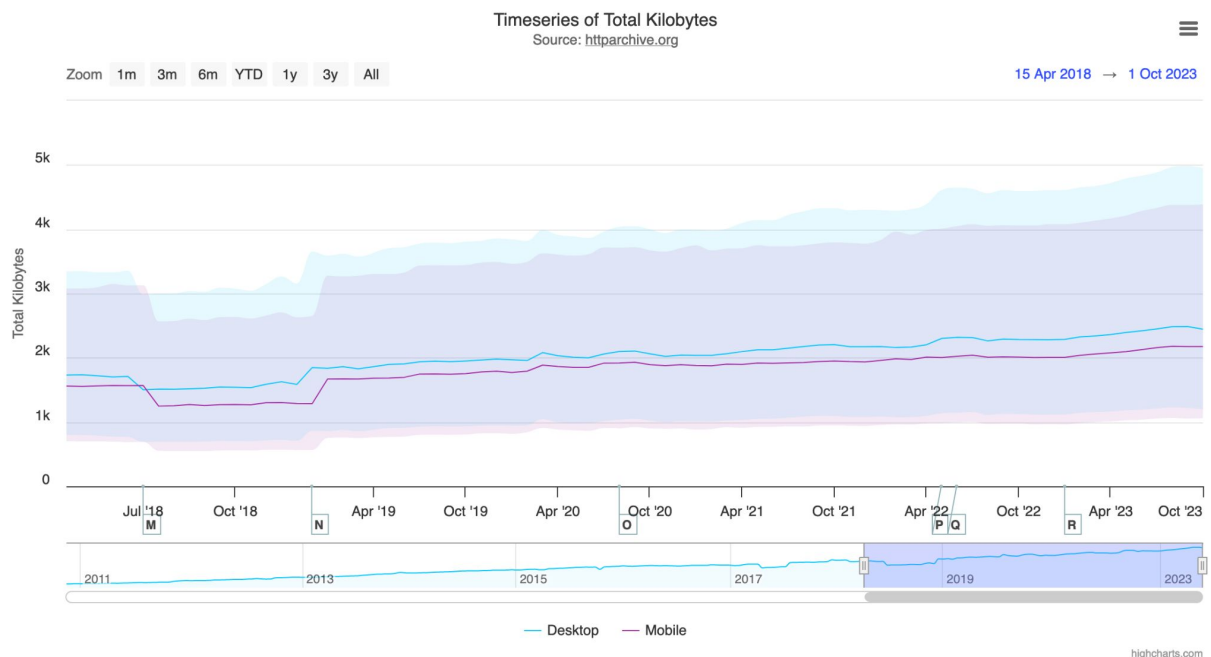
Nelle moderne applicazioni Web, i tempi di caricamento percepiti dagli utenti sono importantissimi. Per mantenere un utente “attivo” su un sito Web, è necessario servirgli pagine Web in circa 250 msec, come mostra la tabella seguente.

Gli ingegneri di Amazon fanno i salti mortali per ridurre al minimo i tempi di caricamento delle pagine su HTTP/1.1. Quelli di Google stanno lavorando a piattaforme molto più performanti di quelle attuali per il Web del futuro (HTTP/2 e soprattutto HTTP/3).

Delay	User perception
0–100 msec	Instant
100–300 msec	Small perceptible delay
300–1000 msec	Machine is working
1+ sec	Likely mental context switch
10+ sec	Task is abandoned

Il World Wide Web - 47

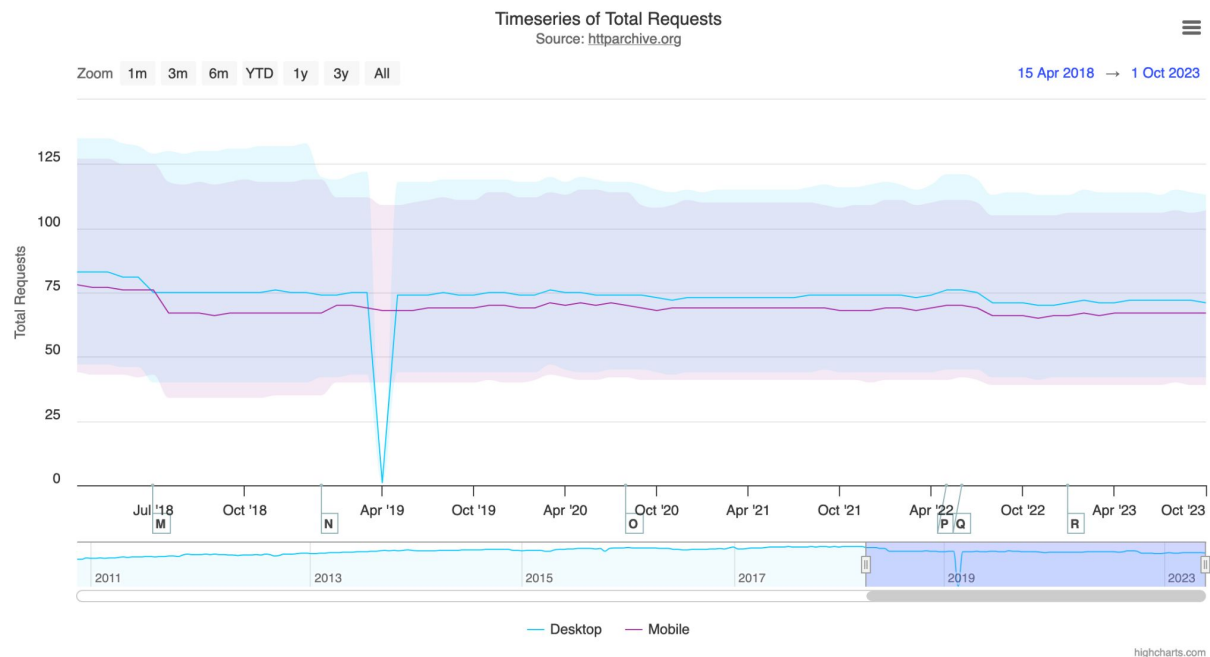
Dimensione pagine Web



<https://httparchive.org/reports/state-of-the-web> (dato del 21/11/2023)

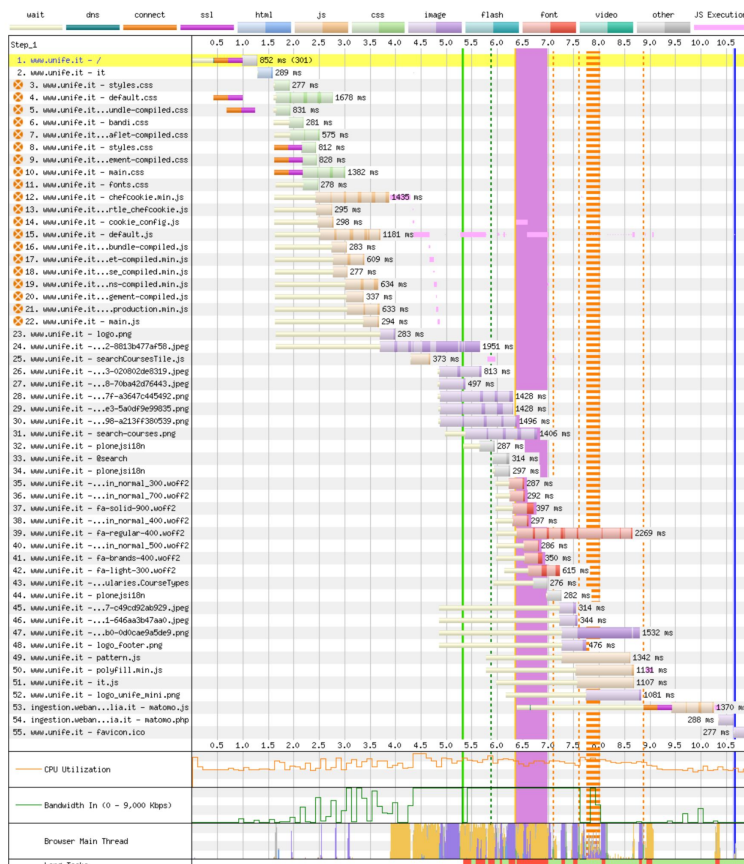
Il World Wide Web - 48

N. risorse necessarie per visualizzare una pagina Web



<https://httparchive.org/reports/state-of-the-web> (dato del 21/11/2023)

Il World Wide Web - 49



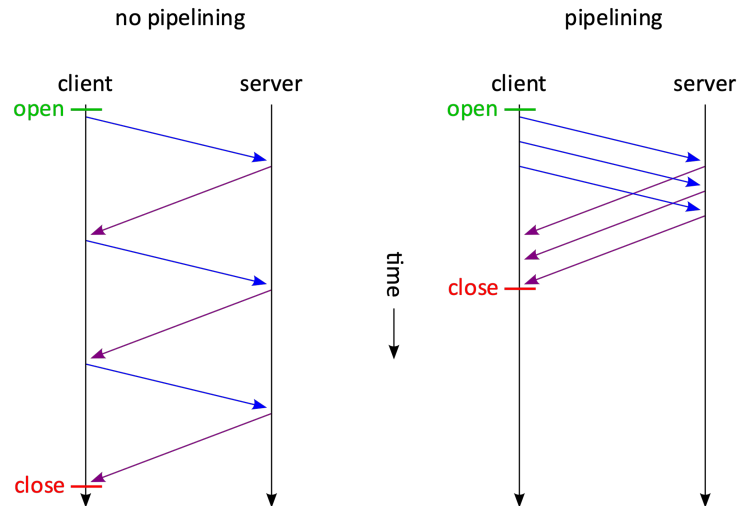
Esempio: www.unife.it

Questa immagine è il risultato ottenuto eseguendo <https://www.webpagetest.org> per il sito www.unife.it (dato del 21/11/2023)

Il World Wide Web - 50

HTTP 1.1 Pipelining

Nota: HTTP/1.1 è nato per supportare il request pipelining. Questa feature però **non è mai stata effettivamente implementata e adottata** per motivi di **sicurezza e compatibilità con proxy**.

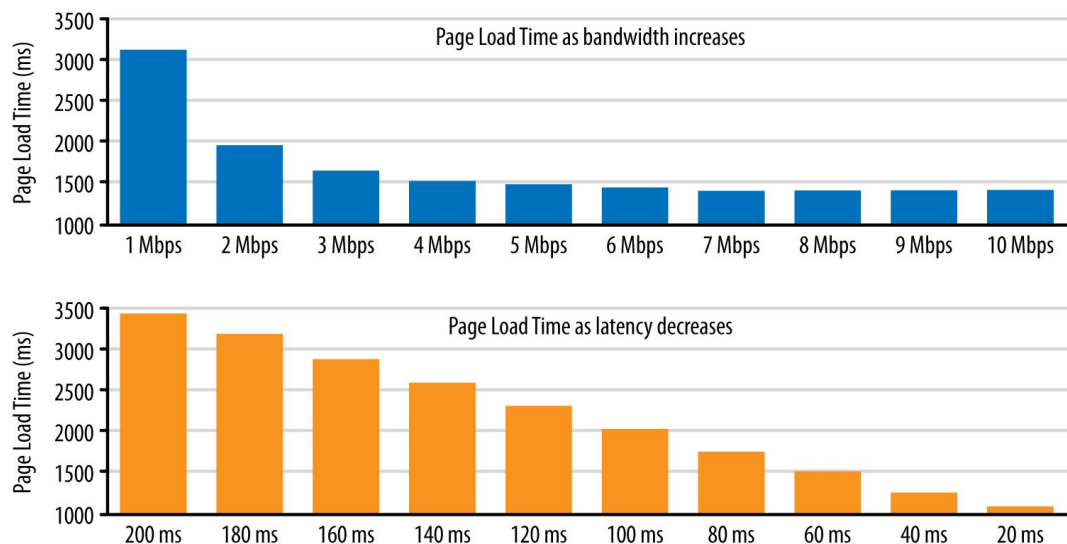


Courtesy: https://it.wikipedia.org/wiki/HTTP_pipelining

Ma la banda larga aiuta?

Purtroppo, **assolutamente no**. Andare oltre i 3 Mbps non comporta guadagni significativi in termini di tempi di caricamento. Per ottimizzare i tempi di caricamento occorre invece diminuire la latenza di comunicazione tra Client e Server.

Problema: la velocità della luce è un limite insormontabile.



Web Caching

Nella progettazione di applicazioni Web moderne, l'uso intelligente del caching è sempre più importante.

Diversi tipi di cache:

- Browser cache
- Transparent proxy, condivisa a livello di LAN, per **contenuti (pagine e asset come figure e file CSS e JavaScript) pubblici**
- Reverse proxy, a livello di applicazione Web, per evitare re-rendering (potenzialmente molto onerosi) di **interesse pagine dinamiche pubbliche**
- Cache server, a livello di applicazione Web, per evitare re-rendering di **porzioni di pagine dinamiche private** («user-specific») frequentemente richiesti e particolarmente onerosi da generare

Sul mercato sono disponibili componenti software molto sofisticati, come Squid, Varnish e memcached, per implementare il caching nelle applicazioni Web.

(Per qualche interessante dettaglio sull'architettura di Varnish si veda:

<http://varnish-cache.org/docs/trunk/phk/notes.html>)

Il World Wide Web - 53

Web Caching - HTTP Header dedicati

Caching è basato su diversi header HTTP dedicati:

Last-Modified: definito dal Server, specifica ultima data di modifica della risorsa (le risorse Web, come pagine, immagini, ecc., possono cambiare nel tempo)

If-Modified-Since: definito dal Client, dice al Server di mandare la risorsa solo se più nuova di quella già in possesso

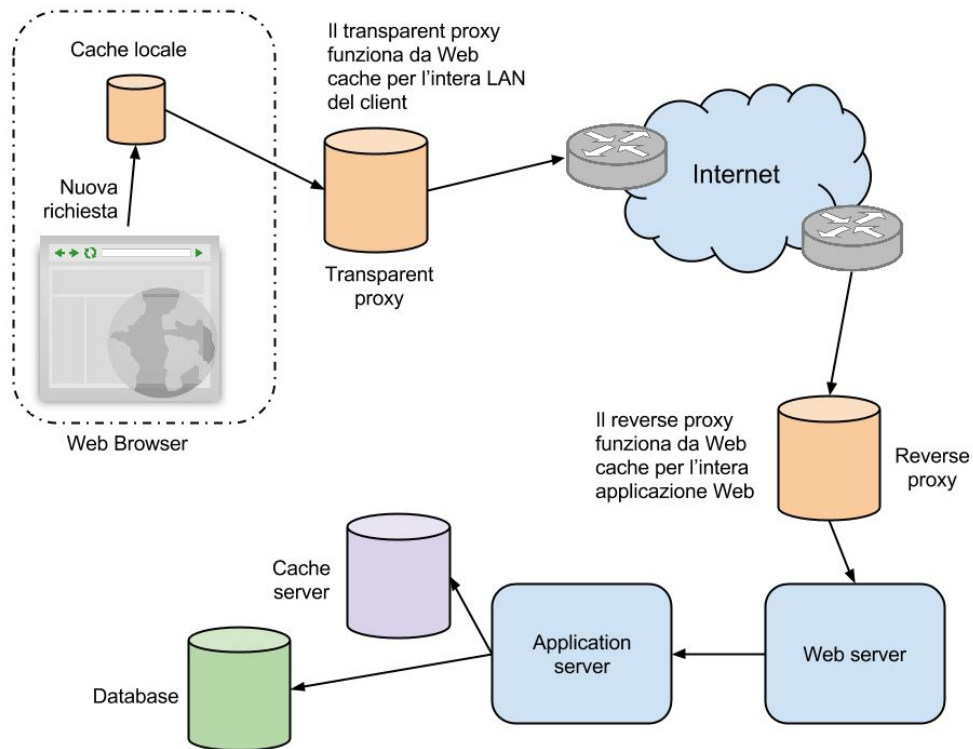
ETag: definito dal Server, specifica identificativo univoco per versione attuale della risorsa

If-None-Match: definito dal Client, dice al Server di mandare la risorsa solo se l'identificativo è diverso rispetto a quello della versione già in possesso

Cache-Control: definito dal Server, specifica se è possibile salvare la pagina in una cache condivisa, solo in una cache privata, o in nessuna cache, e il massimo tempo di vita (age) per cui la pagina può essere salvata in cache

Il World Wide Web - 54

Architettura Web Caching



Il World Wide Web - 55

Content Delivery Network (CDN)

Uso di Content Delivery Network (CDN) per ottimizzare la distribuzione di contenuti statici richiesti molto frequentemente (es. libreria jQuery, framework Bootstrap, ecc.) o di grandi dimensioni (es. video, file ZIP, ecc.).

CDN sono sistemi distribuiti che automaticamente:

- gestiscono la replica di contenuti statici in più *location* all'interno della rete;
- servono all'utente finale la copia di un contenuto statico prelevata dalla location più vicina a lui/lei.

Esempi:

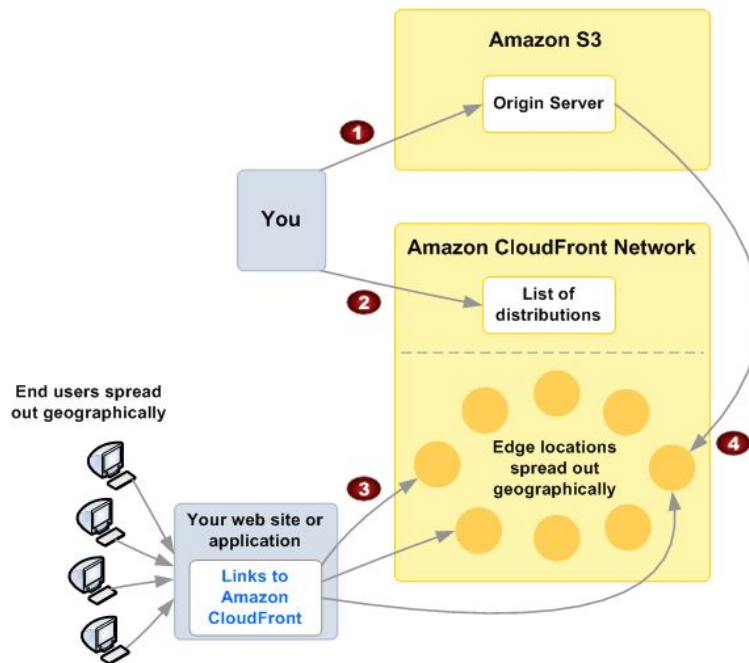
Akamai

Amazon Cloudfront

Google Hosted Libraries <https://developers.google.com/speed/libraries/>

Il World Wide Web - 56

Esempio CDN: Amazon CloudFront



Il World Wide Web - 57

HTTP/2

La versione 2 di HTTP presenta interessanti potenzialità dal punto di vista della performance:

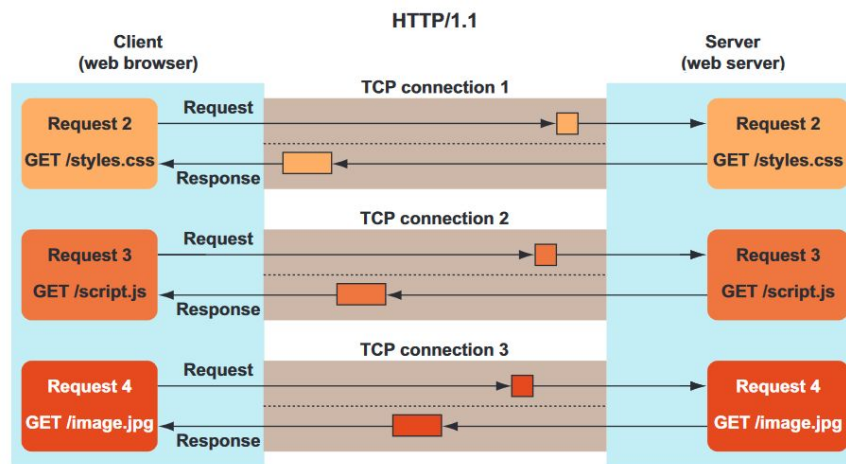
- **protocollo binario**
- compressione header
- stream multiplexing
- server push

<https://web.dev/performance-http2/>

Il World Wide Web - 58

HTTP/2 - Stream multiplexing

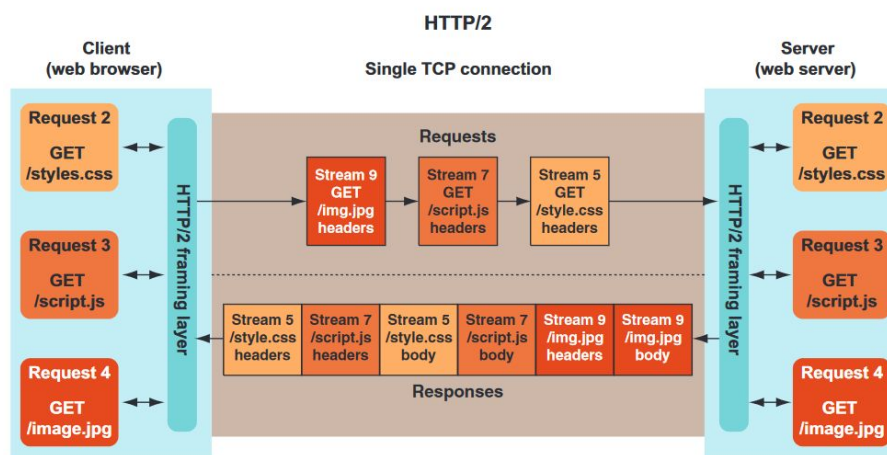
Una delle funzioni principali di HTTP/2 è lo stream multiplexing, che permette di sopperire al mancato supporto (de facto) al pipelining di HTTP/1.1



Courtesy: B. Pollard, "HTTP/2 in Action", Manning, 2019.

HTTP/2 - Stream multiplexing

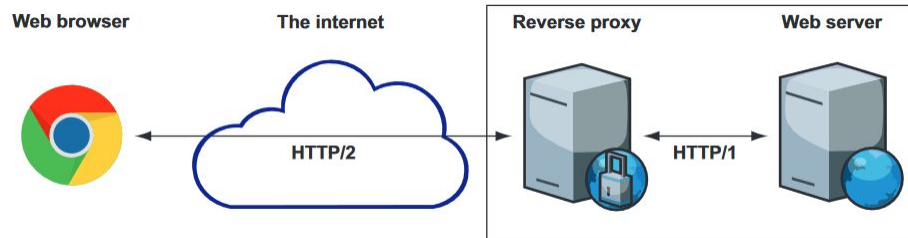
Potendo utilizzare più stream (virtuali, mappati su una singola connessione TCP) siamo in grado di utilizzare una connessione sola tra Client e Server:



Courtesy: B. Pollard, "HTTP/2 in Action", Manning, 2019.

HTTP/2 - Deployment con reverse proxy

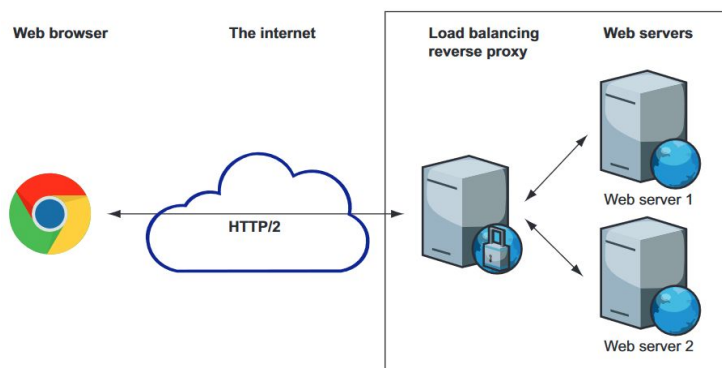
Deployment di applicazioni esistenti (ovverosia che “parlano” HTTP/1.1) su HTTP/2 tramite l’uso di adattatori dedicati in modalità reverse proxy (es h2o: <https://h2o.example.net>)



Courtesy: B. Pollard, “HTTP/2 in Action”, Manning, 2019.

HTTP/2 - Deployment con load balancer HTTP/2-ready

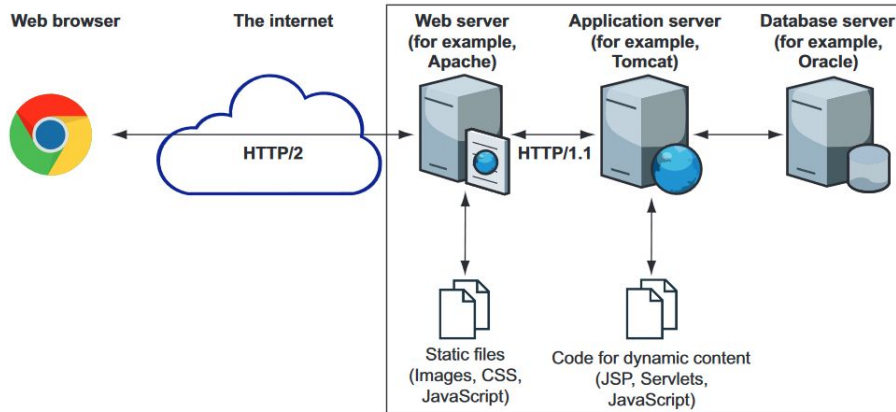
Deployment di applicazioni esistenti (ovverosia che “parlano” HTTP/1.1) su HTTP/2 tramite l’uso di load balancer HTTP/2-ready che funzionano in modalità reverse proxy (es. Varnish, HAProxy)



Courtesy: B. Pollard, “HTTP/2 in Action”, Manning, 2019.

HTTP/2 - Deployment con Web server HTTP/2-ready

Deployment di applicazioni esistenti (ovverosia che “parlano” HTTP/1.1) su HTTP/2 tramite Web server HTTP/2-ready (es. Apache, NGINX)



Courtesy: B. Pollard, “HTTP/2 in Action”, Manning, 2019.

HTTP/2

HTTP/2 comporta anche qualche problema:

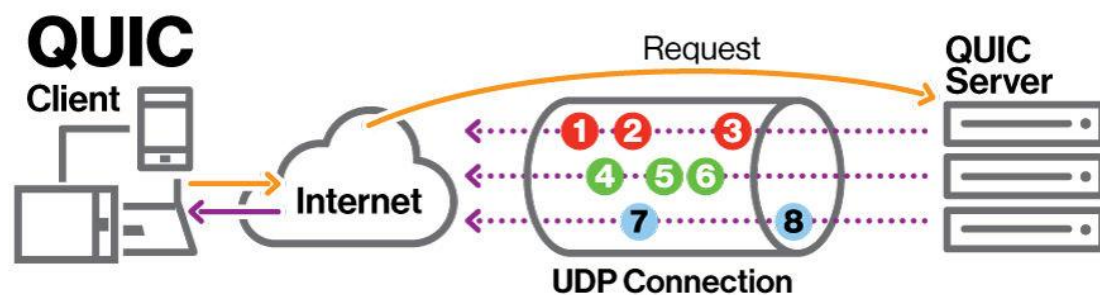
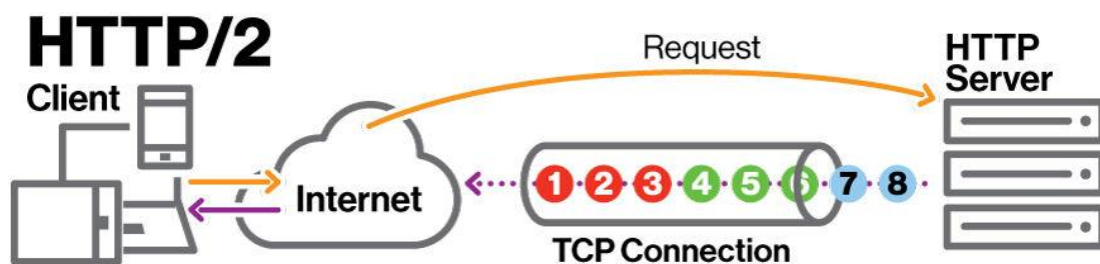
- **Server push richiede modello di sviluppo diverso delle applicazioni**
 - Per un'implementazione efficiente del server push di una pagina, l'applicazione dovrebbe passare al Web server la lista di URL richieste per la visualizzazione della pagina e il Web server a sua volta dovrebbe ricordarsi quale di queste URL ha già servito al client e mandargli solo quelle che mancano (*rottura del modello stateless!!!*)
 - Recentemente (11/11/2020) deprecato dagli sviluppatori di Chrome:
«Intent to Remove: HTTP/2 and gQUIC server push»
<https://groups.google.com/a/chromium.org/g/blink-dev/c/K3rYLVmQUBY/m/vOWBKZGoAQAJ?pli=1>
- **Necessari componenti dedicati per servire applicazioni esistenti attraverso HTTP/2**
- **Spesso nei deployment reali si ha un incremento di performance minore del previsto**

HTTP/3

Più in generale, HTTP/2 è stata pensata come una versione “ponte”, un nuovo protocollo binario che implementa le semantiche di HTTP/1.1 su comunicazioni cifrate (via TLS) su TCP. (Implementare un protocollo multistreaming su TCP, che offre semantica di comunicazione a singolo flusso, ha veramente poco senso.)

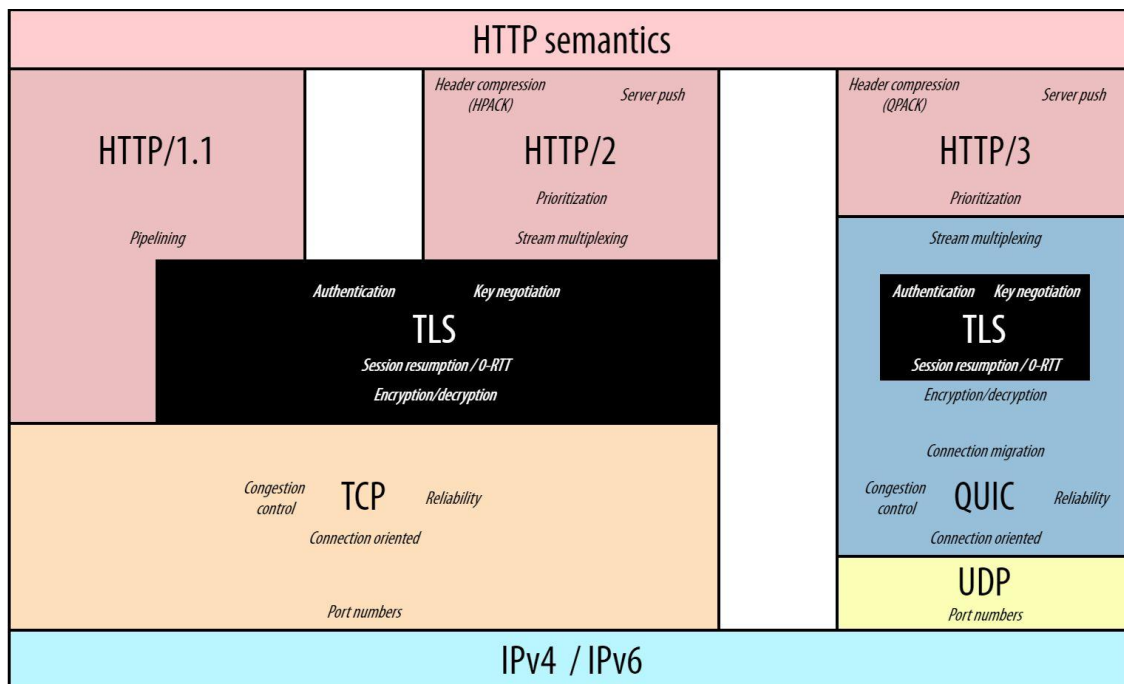
HTTP/3 rappresenta il passaggio da TCP a QUIC, un protocollo di trasporto di nuova generazione con supporto al multistreaming e particolarmente adatto per applicazioni request-response à la HTTP (ma anche à la RPC).

QUIC Multiplexing



Courtesy: <https://www.debugbear.com/blog/http3-quic-protocol-guide>

HTTP/1.1, HTTP/2 e HTTP/3

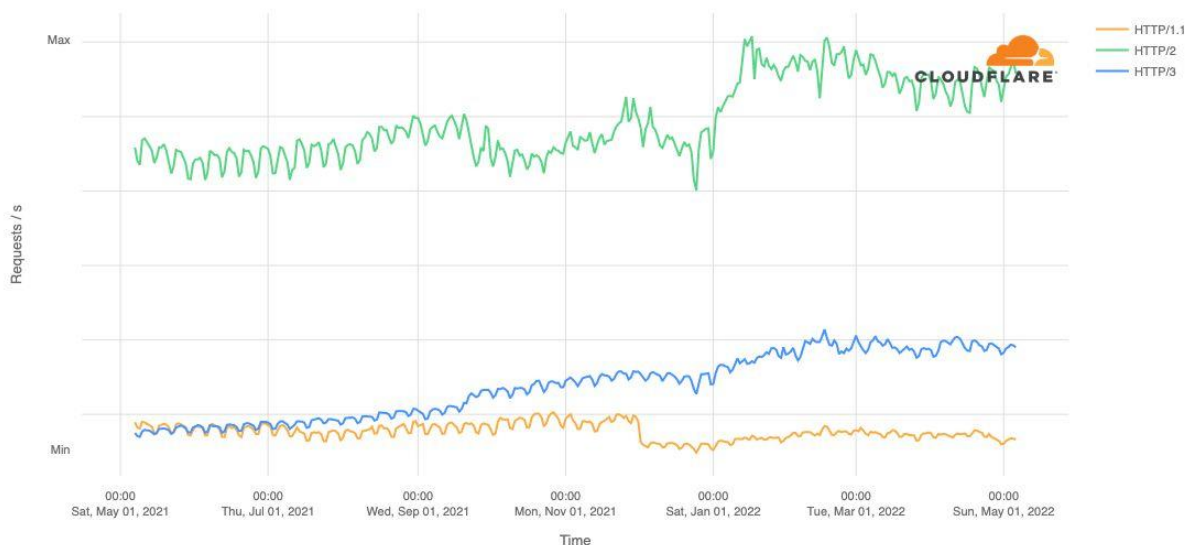


Courtesy: <https://www.debugbear.com/blog/http3-quic-protocol-guide>

Il World Wide Web - 67

HTTP/1.1, HTTP/2 e HTTP/3: diffusione attuale

Number of requests by HTTP version (Worldwide)



Dati forniti da Cloudflare e aggiornati a luglio 2022

Courtesy: <https://www.debugbear.com/blog/http3-quic-protocol-guide>

Il World Wide Web - 68

HTTP/2 & HTTP/3: Are they worth it?

Esistono comunque dubbi sull'effettiva performance di HTTP/2 e HTTP/3 per quanto riguarda applicazioni real-life:

- <https://info.varnish-software.com/blog/how-its-made-varnish-5.0-and-http/2>
- <https://events.drupal.org/dublin2016/sessions/http2-what-no-one-telling-you>

Poul-Henning Kamp, l'architetto principale di Varnish, ha espresso critiche molto forti su HTTP/2 in numerose sedi:

- <https://varnish-cache.org/docs/trunk/phk/http20.html>
- «HTTP/2.0 - The IETF is Phoning It In», Communications of the ACM, <http://queue.acm.org/detail.cfm?id=2716278> (lettura fortemente consigliata)

Lo stesso Robin Marx, ricercatore ed evangelist di QUIC e HTTP/3, ha ammesso che feature come 0-RTT e no-HoL-blocking sono di difficile adozione:

- <https://twitter.com/programmingart/status/1334551536844812296>
- <https://twitter.com/programmingart/status/1399443692214181892>
- <https://www.smashingmagazine.com/2021/09/http3-practical-deployment-options-part3/>

Il World Wide Web - 69

What about QUIC?

Vi sono anche perplessità sull'effettiva performance di QUIC:

“QUICKer or not? - an Empirical Analysis of QUIC vs TCP for Video Streaming QoE Provisioning”
https://irp-cdn.multiscreensite.com/d3de1972/files/uploaded/Conference_60_Michael_Seufert.pdf

... e una grande varietà di diverse implementazioni:

“Same Standards, Different Decisions: A Study of QUIC and HTTP/3 Implementation Diversity”
https://qlog.edm.uhasselt.be/epiq/files/QUICImplementationDiversity_Marx_final_11jun2020.pdf

Peraltro, non esiste una API standard per QUIC, ma una moltitudine di implementazioni: <https://github.com/quicwg/base-drafts/wiki/Implementations>, tutte basate su un modello di programmazione event-driven, che è *****SIGNIFICATIVAMENTE***** più complesso di quello delle socket stream.

Quindi, l'adozione di QUIC comporta al momento la scrittura di codice vendor-specific e di elevata complessità.

Il World Wide Web - 70

Ottimizzazioni di performance o aberrazioni?

La lentezza di caricamento delle pagine Web ha spinto colossi come Google e Facebook a sperimentare tecnologie che, oltre a ottimizzare le performance, permettono alle suddette aziende un controllo molto maggiore dei contenuti forniti da terze parti (Google AMP, Facebook Instant Articles).

Allo stesso tempo, le stesse aziende stanno spingendo fortemente il mercato verso soluzioni Fat Client (basate ad esempio su Angular.js, React.js, GraphQL) che propongono un modello di programmazione *****ENORMEMENTE***** più complesso, con una superficie d'attacco da far tremare le vene ai polsi, non sempre necessario per (o addirittura compatibile con uno sviluppo sostenibile di) applicazioni Web di medio/piccole dimensioni.

Fate sempre attenzione a non cadere vittima del *cargo culting* e/o del *coolness factor* quando realizzate applicazioni Web!!!

Il World Wide Web - 71

Web: Principali problemi

WWW ha determinato un ampliamento (esponenziale) della fascia di utenza e deve affrontare problemi di crescita:

- **traffico** (*banda di trasmissione limitata*). Infrastrutture in fibra ottica, collegamenti satellitari, etc.
- **sicurezza dei sistemi e delle informazioni**. Design for security
- **standardizzazione di protocolli e strumenti**. Apertura vs chiusura
- **facilitare recupero informazioni**. Favorire la ricerca delle informazioni con motori di ricerca, etc.
- **accessibilità**. Le informazioni e i servizi devono essere accessibili in maniera dipendente dalle preferenze (e abilità) degli utenti, dai dispositivi e dalle risorse a disposizione. Particolare attenzione alle diverse abilità (disabilità) degli utenti.
- **accesso mobile al Web**. Le moderne tecnologie wireless permettono collegamento a Internet ubiquo e pervasivo.
- **adattamento ed evoluzione**. per garantire una apertura alle nuove esigenze che si presentano nel Web (es. connessioni permanenti, **stato**, sicurezza, interattività).

Il World Wide Web - 72

La ricerca di informazioni sul WEB

Il Web è un ipertesto (un grafo) con milioni di nodi → problema di reperire le informazioni

Per facilitare la ricerca di informazioni su Internet, nei primi anni del Web si usavano **indici del Web**, organizzati con varie modalità:

- alfabetica
- per argomento (gerarchici)
- per area geografica (gerarchici)

Oggi si usano esclusivamente dei **motori di ricerca** che cercano parole chiave.

Esempi più importanti:

Google, Bing, DuckDuckGo, ...

Le dimensioni del problema (e di Google)

Novembre 2004, Google indicizzava più di:

- 4 miliardi di pagine Web
- 845 milioni di immagini
- 880 milioni di messaggi usenet

Settembre 2005, indicizzava:

- 8,168,684,336 pagine Web

Luglio 2008, indicizzava:

- più di un trilardo di pagine Web (di URL analizzati)

Statistiche accesso (2003): media 200 milioni ricerche/giorno

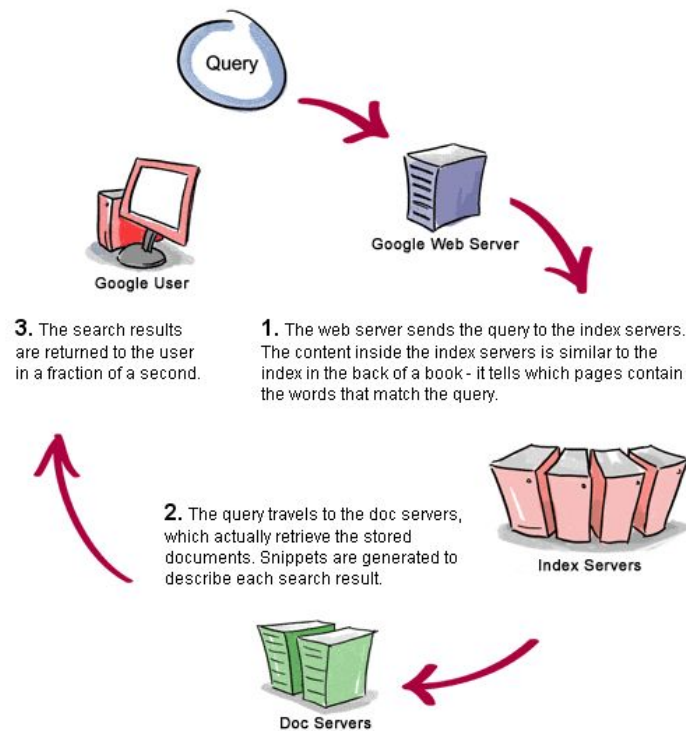
Architettura del sistema è un cluster di macchine Linux in rapida crescita:

2003 – circa 10.000 Server

2006 – circa 12 datacenter con circa 450.000 Server (Wikipedia)

2008 – circa 36 datacenter con circa 2.000.000 Server (Economist)

Come viene processata una query



Il World Wide Web - 75

La realizzazione di un indice del Web

La costruzione di un indice del Web prevede 2 fasi:

- **ricerca delle pagine Web** esplorando tutto il Web (o una sua parte)
- **indicizzazione** delle informazioni recuperate (creazione di una sintesi di ognuna delle pagine, selezione di parole chiave per ogni pagina)

Le informazioni raccolte sono rese disponibili sulla rete stessa.

Il World Wide Web - 76

Realizzazione di un indice (fase di ricerca)

Nella fase di ricerca si usano programmi che esplorano tutte le pagine Web dei siti presenti in rete. Programmi più diffusi: search engine, spider, crawler, worm, knowbots (knowledge robots).

La **ricerca** parte da una serie iniziale di pagine Web; per ogni pagina si prelevano i link presenti e si memorizzano localmente; algoritmi tipo breadth-first o depth-first; alla fine si ottiene un elenco di tutti gli URL della rete e si possono caricare in locale le varie pagine, per passare alla fase di indicizzazione, per produrre una breve sintesi di ogni pagina.

Problemi ricerca

La fase di ricerca delle informazioni presenta alcune difficoltà:

- dimensioni del grafo Web
- punto di partenza della ricerca
- tipo di ricerca: depth-first → stack overflow
breadth-first → dimensioni heap
- come trattare i link presenti nelle applicazioni Web (map CGI, JavaScript, etc.)
- URL obsoleti
- Macchine non raggiungibili rallentano la ricerca

Realizzazione di un indice (fase di indicizzazione)

La procedura di indexing estrae le parole chiave da ogni pagina Web memorizzata in locale (in area heap) nella fase di ricerca.

Per trovare le parole chiave:

- si scartano le parole poco significative (articoli, preposizioni, etc.)
- si scelgono che compaiono in titoli e sottotitoli
- si scelgono le parole che nella pagina hanno la frequenza maggiore
- ...

Per ogni parola chiave si memorizza in tabella la parola e l'URL che la contiene.

Alla fine della indicizzazione si ordina la tabella sulle parole chiave e si salva su file che verrà consultato per le ricerche da parte degli utenti

problemi:

- titoli pagine spesso poco significativi
- analisi intere pagine costosa
- pagine solo video o audio, oppure active map

Google (PageRank)

Preparazione indice di Google analoga ai concorrenti.

Google è **innovativo** nel modo di **ordinare i risultati** per evidenziare la loro (presunta) attinenza con i termini di ricerca. Le pagine più significative vengono mostrate per prime.

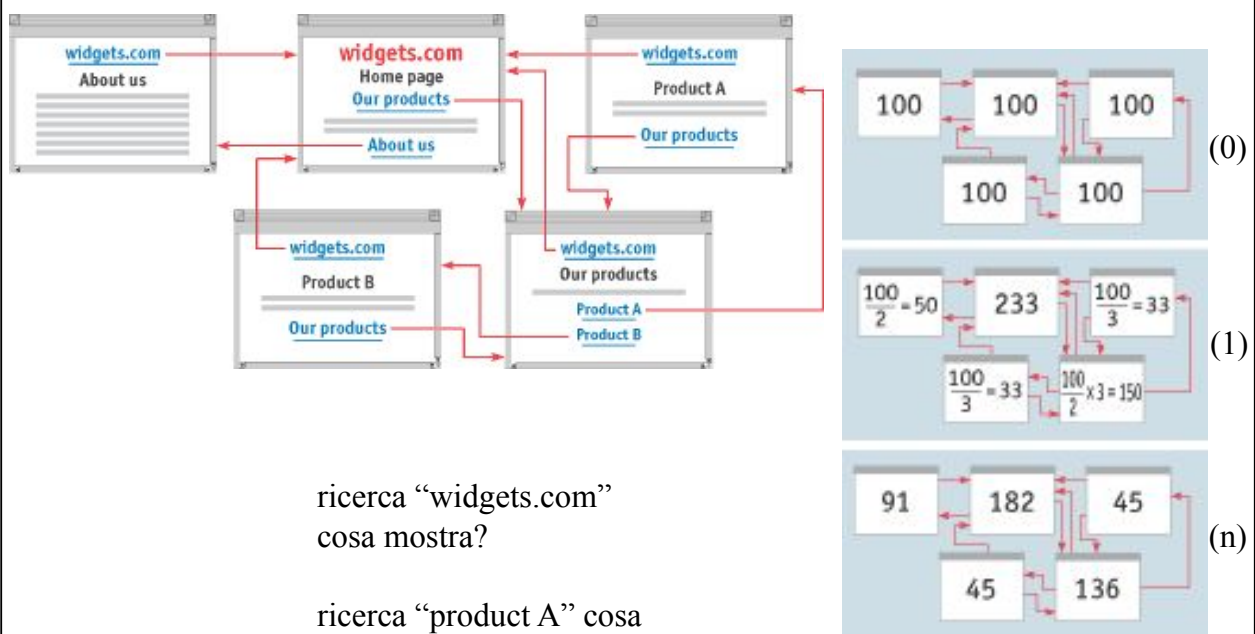
Algoritmo innovativo (**PageRank**), basato su idea che pagine più importanti sono più linkate, e se una pagina è linkata da un sito importante, sarà a sua volta una pagina importante.

Per una pagina Web u si ottiene il corrispondente rank $PR(u)$:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

dove B_u è l'insieme di pagine Web che contengono un link alla pagina u e $L(v)$ è il numero di link contenuti nella pagina v .

Google (PageRank)



Non solo motori di ricerca...

Google, Bing & co. stanno evolvendo rapidamente. Non sono più solo motori di ricerca, ma piattaforme per fornire servizi Web, in diretta contrapposizione al modello tradizionale di vendita del software (es. Microsoft Windows e Office).

Nuovi modelli contrapposti:

- “Software as a Service” (Google)
- “Software plus Services” (Microsoft)

Emerge **Cloud Computing** (la “nuvola” indica un’astrazione di una macchina virtuale remota di prestazioni scalabili a richiesta, su cui fare eseguire le applicazioni). Quali vantaggi? Quali criticità?

Primo esempio di servizio commerciale, **Amazon Web Services (AWS)**:

- Amazon Elastic Compute Cloud (EC2), macchine virtuali con sistemi operativi Linux e Windows, tariffe pay-per-use su ore CPU virtuali consumate
- Amazon Simple Storage Service (S3), storage on-line accessibile via Web Services, tariffe pay-per-use su area di storage usata e traffico generato