EDI: Third Assigment

Federico Cignoli, Mat. 544952

July 9, 2024

Abstract

This report on web technologies will cover the main topics:

- Parallel Connections;
- Cache Mechanism;
- Benchmarking Tools;
- Performance Evaluation.

Web Technologies

The main objective of this assignment is to analyze and discuss the performance of different websites using various tools and configurations. The report is divided into four main sections.

In the first section, the performance of a website was evaluated and compared by changing the number of parallel connections set in the browser.

In the second section, the cache mechanism was applied and analyzed on different types of websites, which differ in terms of security and supported protocols.

In the last two sections, various tests were performed using three different benchmarking tools. Specifically, in the third section, an analysis was conducted on the outputs obtained by executing the `ab` command, specifying two very different websites: one known for commercial activities and one personal website used for educational purposes.

In the fourth section, two popular performance evaluation tools (`nghttp` and `h2load`) were used to evaluate the performance of corporate websites, with the corresponding results specified.

1. Analyze and discuss the role and the impacts of the number of parallel connections set inside the browser on the Page Load Times of commercial/institutional websites running HTTP/1.1, HTTP/2 or HTTP/3. Outline expected or unexpected behaviors in your experiments.

In this first experiment, the objective is to analyze and discuss the impact of parallel connections on the page load times of commercial and institutional websites. The Firefox browser, which defaults to six parallel connections, was used for this purpose. To change this number, you need to type `about:config` in the address bar and search for `network.http.max-persistent-connections-per-server`, as shown in Figure 1.

Figure 1

Figure 1 shows the maximum number of parallel connections for server, and also for proxy, setted in default. In the right side there is the pen icon where we can change this number.

The focus of this experiment is the institutional website:
`https://web.unipv.it/`.
Initially, the Page Load Time was analyzed with the default number of parallel connections set to 6. Subsequently, comparisons were conducted with alternative configurations, specifically setting the number of parallel connections to 20, and 50. Table 1 provides a summary of the different times recorded under each configuration.

Table 1

Parallel Connection	**Page Load Time (ms)**
6	0.755
20	0.984
50	1.60

Table 1 shows that: setting the number of parallel connections too high leads to a decrease in performance, as evidenced by the Page Load Time values in the last two cases. This occurs because such configuration can overload the server, causing unexpected behavior compared to the anticipated performance. Additionally, it's important to consider that some servers impose a maximum limit on the number of open connections per client/IP to prevent overloading and ensure fair access for all clients.

2. Analyze and discuss the impacts on the Page Load Times of the caching policies implemented by commercial/institutional websites running HTTP/1.1, HTTP/2 or HTTP/3 (possibly with unsecure and secure connections). Outline expected or unexpected behaviors in your experiments.

In this second section, an analysis of the effects of Page Load Time (PLT) caching policies on various websites has been conducted. The selected websites exhibit distinct characteristics:

Table 2: Website Protocol and Connection Type

Website	Protocol	Connection Type
http://www.inf.uniroma3.it/	HTTP/1	non-secure
https://vision.unipv.it/wmt/	HTTP/1	secure
https://www.amazon.it/	HTTP/2	secure
https://www.netflix.com/	HTTP/3	secure

`http://www.inf.uniroma3.it/` has been selected as a non-secure website that does not support HTTP/2. The lack of security is further evidenced in Figure 2: the "Domain" column displays a symbol indicating that the connection used to retrieve the specific resource is not secure.

Figure 2: www.inf.uniroma3.it

When this website is loaded without using the cache, the PLT is 826 ms, with a total of 12 requests. After the initial load, the page was reloaded with the cache mechanism enabled, resulting in a lower page load time (PLT = 584 ms). This is because some resources were retrieved from the cache without sending requests to the server.

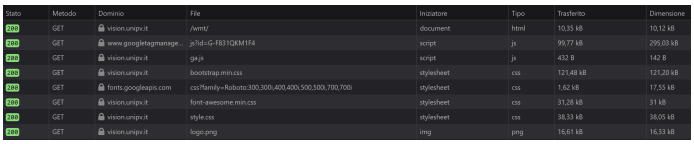

Figure 3: vision.unipv.it/wmt

`https://vision.unipv.it/wmt/` is an institutional website for the Web and Multimedia Technologies course (in Figure 3 we can see the lock symbol in the domain column), which supports HTTP/1. In the first case, without a caching mechanism, the PLT is 2.47s, higher than the previous case, which is due to the fact that the

number of requests is also larger (21). With the cache mechanism enabled, the PLT is significantly lower (434 ms), but the number of requests loaded is also reduced to 13.

`https://www.amazon.it/` is a commercial website that supports HTTP/2. It is designed to accommodate a large number of visitors compared to other websites, and the number of resources that need to be downloaded is also greater. Without caching, the PLT is 3.96s with 212 resources. With the cache mechanism active, the PLT is reduced to 765 ms with 55 resources downloaded.

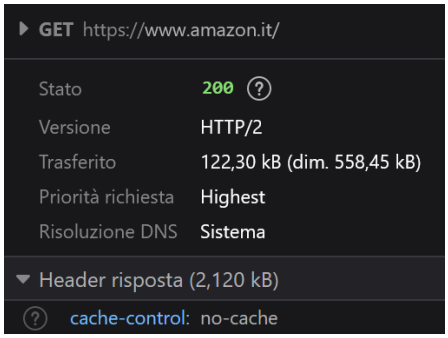

Figure 4: object that should not be stored

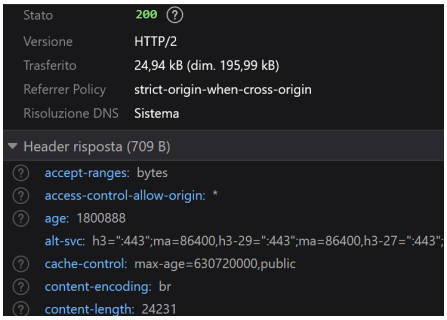

Figure 5: object that is suitable for storage

Figure 4 and Figure 5 illustrate different Cache-control values associated with a specific object: in one case, the object should not be stored, while in the other case, thanks to the `max-age` value (inside "Cache-control"), the response remains fresh for X seconds (max-age=630720000) after it was generated.

Moreover, even when the cache mechanism is enabled, a web server can include a specific header line in the HTTP response message called `Cache-Control` to define caching directives. These directives are:

- **public**: the response can be stored in any cache;
- **private**: the response is intended for a single user;
- **no-cache**: the response can be stored but not reused;
- **no-store**: the response must not be stored in any cache.

HTTP/3 represents the next generation of the HTTP protocol family, and major players like Google and Facebook are transitioning to HTTP/3. This is why the target website `https://www.netflix.com/` has been selected.

By default, browsers assume that servers do not support HTTP/3, so they initially send requests via HTTP/2 or HTTP/1. If the server does support HTTP/3, it responds with the `alt-svc` header to instruct the client that an alternative service can be used for future HTTP requests.

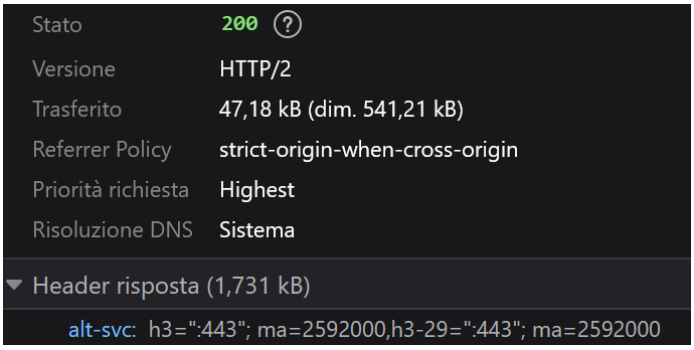

Figure 6

In the `alt-svg` line, Figure 6, indicates that the server has responded with the ‘alt-svc‘ header, which instructs the client on how to use an alternative version or mode of the service for future HTTP requests.

Specifically, ‘alt-svc‘ specifies alternatives to the main service, in this case indicating that the server supports HTTP/3 (specified with ‘h3‘) on the same standard HTTPS port (‘:443‘). The ‘ma=2592000‘ instructions indicate the maximum duration of validity for the specified alternative, expressed in seconds.

Additionally, ‘h3-29‘ indicates another variant of HTTP/3, with support for the previous version, and it also has a specified maximum duration of validity.

Finally, there is a significant difference between using the cache mechanism and not using it: without the cache mechanism, the PLT is 2.07 s, while when activated, it is only 922 ms.

3. Test the performance of commercial/institutional websites using the ab – Apache HTTP server benchmarking tool with different options. Analyze and discuss the results of your experiments and outline any expected or unexpected behavior.

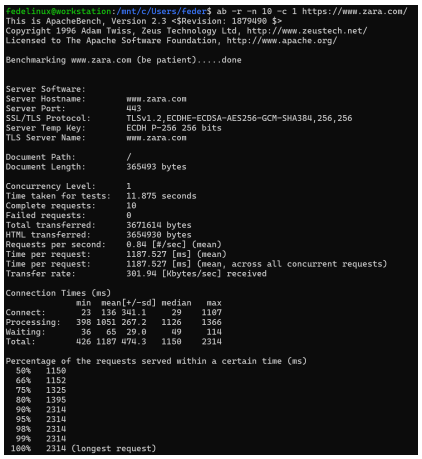

Figure 7: https://www.zara.com/

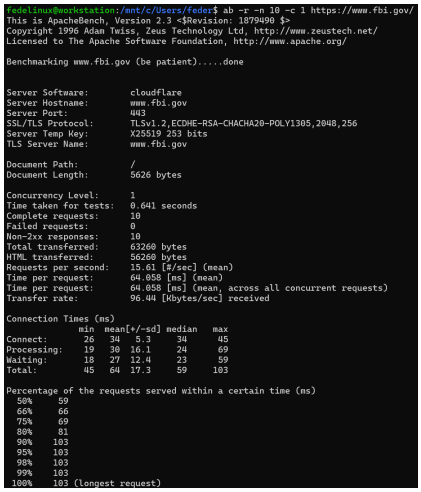

Figure 8: https://www.fbi.gov/

In the Figure 7, we can observe that: the server uses the TLSv1.2 protocol with ECDHE-ECDSA-AES256-GCM-SHA384 encryption and a temporary ECDH P-256 256-bit key. During the test, which lasted 11.875 seconds, 10 requests were completed without any failures.

The concurrency level was 1, meaning that one request was executed at a time. The server handled an average of 0.84 requests per second, with an average request time of 1187.527 milliseconds. The transfer rate was 301.94 Kbytes/sec.

The connection times showed that 50% of the requests were served within 1150 milliseconds, while the maximum time for a request was 2314 milliseconds. This indicates good server stability under the test load, with no failed or errored requests.

In the Figure 8, we can observe that: during the test, the Cloudflare server handled 10 requests in a total of 0.641 seconds without failures. The concurrency level was 1, with an average of 15.61 requests per second and an average time of 64.058 milliseconds per request. www.fbi.gov shows much higher performance than www.zara.com in terms of time per request and requests per second. Moreover, www.fbi.gov has much lower connection and processing times than www.zara.com. In addition, requests to www.fbi.gov are served much more consistently (all within 103 ms), while www.zara.com has much greater variability (up to 2314 ms).

The line `Non-2xx responses: 10` indicates that all 10 completed requests did not receive a response with an HTTP status code of 2xx (which normally indicates success). In other words, none of the requests received a successful response (such as a 200 OK code).

4. Test the performance of commercial/institutional websites using the nghttp and h2load tools with different options. Analyze and discuss the results of your experiments and outline any expected or unexpected behavior.

1. **nghttp:** This tool allows interaction with web servers that support the HTTP/2 protocol and provides performance evaluation. HTTP/2 is based on HTTP request and response messages, which are mapped into frames.

2. **h2load:** This is a benchmarking tool for HTTP/2 and HTTP/1.1, providing various options that can help perform different tests. The flags used in this experiment are `-n`, `-c`, `-D`, and `--warm-up-time`.

nghttp

```
fedelinux@workstation:/mnt/c/Users/feder$ nghttp -vasn
    ↪ https://www.zara.com/it/ |more
[ 0.023] Connected
The negotiated protocol: h2
[ 0.050] send SETTINGS frame <length=12, flags=0x00,
    ↪ stream_id=0>
          (niv=2)
          [SETTINGS_MAX_CONCURRENT_STREAMS(0x03):100]
          [SETTINGS_INITIAL_WINDOW_SIZE(0x04):65535]
[ 0.050] send PRIORITY frame <length=5, flags=0x00,
    ↪ stream_id=3>
          (dep_stream_id=0, weight=201, exclusive=0)
[ 0.050] send PRIORITY frame <length=5, flags=0x00,
    ↪ stream_id=5>
          (dep_stream_id=0, weight=101, exclusive=0)
[ 0.050] send PRIORITY frame <length=5, flags=0x00,
    ↪ stream_id=7>
          (dep_stream_id=0, weight=1, exclusive=0)
[ 0.050] send PRIORITY frame <length=5, flags=0x00,
    ↪ stream_id=9>
          (dep_stream_id=7, weight=1, exclusive=0)
[ 0.050] send PRIORITY frame <length=5, flags=0x00,
    ↪ stream_id=11>
          (dep_stream_id=3, weight=1, exclusive=0)
[ 0.050] send HEADERS frame <length=41, flags=0x25,
    ↪ stream_id=13>
          ; END_STREAM | END_HEADERS | PRIORITY
          (padlen=0, dep_stream_id=11, weight=16, exclusive
             ↪ =0)
          ; Open new stream
          :method: GET
```

```
        :path: /it/
        :scheme: https
        :authority: www.zara.com
        accept: */*
        accept-encoding: gzip, deflate
        user-agent: nghttp2/1.43.0
[ 0.083] recv SETTINGS frame <length=30, flags=0x00,
    ↪ stream_id=0>
        (niv=5)
        [SETTINGS_HEADER_TABLE_SIZE(0x01):4096]
        [SETTINGS_MAX_CONCURRENT_STREAMS(0x03):100]
        [SETTINGS_INITIAL_WINDOW_SIZE(0x04):65535]
        [SETTINGS_MAX_FRAME_SIZE(0x05):16384]
        [SETTINGS_MAX_HEADER_LIST_SIZE(0x06):32768]
[ 0.083] recv SETTINGS frame <length=0, flags=0x01,
    ↪ stream_id=0>
        ; ACK
        (niv=0)
[ 0.083] send SETTINGS frame <length=0, flags=0x01,
    ↪ stream_id=0>
        ; ACK
        (niv=0)
[ 0.177] recv (stream_id=13, sensitive) :status: 200
[ 0.178] recv (stream_id=13, sensitive) mime-version: 1.0
[ 0.178] recv (stream_id=13, sensitive) content-type:
    ↪ text/html
[ 0.178] recv (stream_id=13, sensitive) cache-control: no
    ↪ -cache, no-store, must-revalidate
[ 0.178] recv (stream_id=13, sensitive) expires: 0
[ 0.178] recv (stream_id=13, sensitive) vary: Accept-
    ↪ Encoding
[ 0.178] recv (stream_id=13, sensitive) content-encoding:
    ↪  gzip
[ 0.178] recv (stream_id=13, sensitive) date: Tue, 28 May
    ↪  2024 20:31:29 GMT
[ 0.178] recv (stream_id=13, sensitive) content-length:
    ↪ 1085
[ 0.178] recv (stream_id=13, sensitive) set-cookie:
    ↪ ITXSESSIONID=316cc689e0e0e1d387fce91ea7d23230;
    ↪ expires=Tue, 28-May-
2024 21:01:29 GMT; path=/; domain=.zara.com; secure
[ 0.178] recv (stream_id=13, sensitive) set-cookie:
    ↪ ITXDEVICEID=e63c627aeff6972fdda452a326a7042b;
    ↪ expires=Wed, 28-May-2
025 20:31:29 GMT; path=/; domain=.zara.com; secure
```

```
[ 0.178] recv (stream_id=13, sensitive) cache-control: no
   ↪ -store, private
[ 0.178] recv (stream_id=13, sensitive) strict-transport-
   ↪ security: max-age=31536000 ; includeSubDomains
[ 0.178] recv (stream_id=13, sensitive) set-cookie:
   ↪ ak_bmsc=AD96D335492DADFECFD51DDD622F5A0C
   ↪ ~000000000000000000000000
0000~
   ↪ YAAQkOgWAp1L6aSPAQAACpzmwBcFzmFn2KjO8DPadmOHLOR5yTKi
   ↪
nrzuqBKo5fkzTRan5om3RMDlaWIpoAYWuyEPQ03DLADHqO7X7SU2VxgutF
   ↪
C7L3bSxvB2h+
   ↪ zCNWknBY60LoO3UbLTcDeXzmAxq5CuiKV1ITz1CTAqxvQkt2sD6c
   ↪ +dWJ9r/
   ↪ vAyyETe8dMHepCr5A52Tu57bMvUl7ahUFuJgp3vlOq6tRDpW5
2CWuXsuEWdke0zH9/i4h3rl8r4Xie/
   ↪ aJugvmG5qBNFEz8qJCzZg2OBbZ1UZrPbGX+1
   ↪ cZJs6DbN3JlO1zEgKfWkusZbDGGGA==; Domain=.zara.com;
   ↪ Pat
h=/; Expires=Tue, 28 May 2024 22:31:29 GMT; Max-Age=7200
[ 0.178] recv HEADERS frame <length=818, flags=0x04,
   ↪ stream_id=13>
       ; END_HEADERS
       (padlen=0)
       ; First response header
[ 0.210] recv DATA frame <length=1085, flags=0x01,
   ↪ stream_id=13>
       ; END_STREAM
[ 0.213] send GOAWAY frame <length=8, flags=0x00,
   ↪ stream_id=0>
       (last_stream_id=0, error_code=NO_ERROR(0x00),
          ↪ opaque_data(0)=[])
***** Statistics *****

Request timing:
  responseEnd: the time when last byte of response was
     ↪ received
          relative to connectEnd
 requestStart: the time just before first byte of request
     ↪  was sent
          relative to connectEnd. If '*' is shown, this
             ↪ was
          pushed by server.
     process: responseEnd - requestStart
       code: HTTP status code
```

```
       size: number of bytes received as response body
          ↪ without
            inflation.
        URI: request URI

see http://www.w3.org/TR/resource-timing/#processing-
   ↪ model
sorted by 'complete'
id responseEnd requestStart process code size request
   ↪ path
 13 +159.87ms +75us 159.79ms 200 1K /it/
```

In the output of the terminal, that we can observe before, shows to us some important things:

- **Connected:** Indicates that the connection to the server has been successfully established.
- **The negotiated protocol:** 'h2': This indicates that the negotiated protocol between the client and the server is HTTP/2.
- **send SETTINGS frame:** The client sends a settings frame to the server to configure the parameters of the HTTP/2 connection.
- **send PRIORITY frame:** The client sends priority frames to manage the transmission order of frames within the HTTP/2 connection.
- **send HEADERS frame:** The client sends a headers frame containing information about the HTTP/2 request, such as the request method, resource path, request headers, etc.
- **recv SETTINGS frame:** The client receives a settings frame from the server, containing configuration parameters supported by the server.
- **recv HEADERS frame:** The client receives a headers frame from the server's response, containing information about the HTTP/2 response headers, such as the response status, content type, response headers, etc.
- **recv DATA frame:** The client receives a data frame containing the body of the HTTP/2 response.
- **recv GOAWAY frame:** The client receives a GOAWAY frame, indicating that the server is closing the connection and specifies the last stream handled by the server.
- **Statistics:** This section provides detailed statistics on the interaction between the client and the server, such as response times, HTTP status code, response body size, etc.

h2load

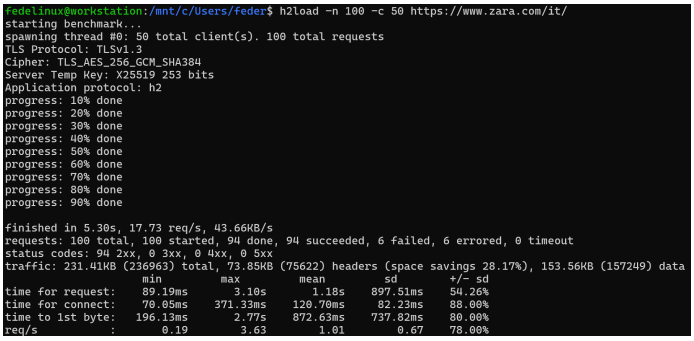

Figure 9

Instead here, the output in Figure 9 is the result of using the h2load command on https://www.zara.com/it/. Here's what the most impoerant data means:

- **100 total requests:** Indicates that 50 clients have been created to send a total of 100 requests to the server.
- **TLS Protocol:** 'TLSv1.3': shows the TLS protocol used for secure communication.
- **Cipher:** 'TLS_AES_256_GCM_SHA384': specifies the encryption algorithm used to ensure communication security.
- **Server Temp Key:** 'X25519 253 bits': provides information about the temporary key used by the server for key exchange.
- **Application protocol:** 'h2': indicates that the application protocol used is HTTP/2.
- **requests:** Provides details on the total number of requests sent, how many were started, how many were successfully completed, how many failed or generated errors, and how many timed out.
- **status codes:** shows the number of responses for each HTTP status code.

Now, I am attempting to conduct a `timing-based load-testing` test, which means that I am specifying the duration of the test instead of a predefined number of requests.number of requests

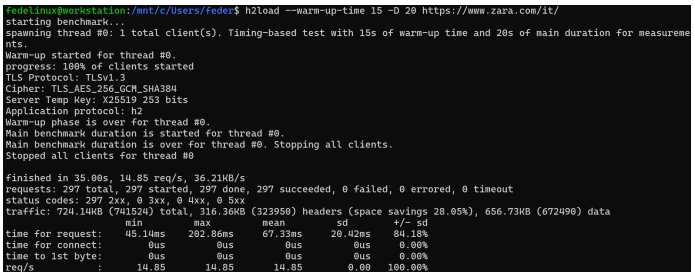

Figure 10

Using the `-D`, in Figure 10, flag indicates the duration requested by the tool for benchmarking the target, while with the `--warm-up-time` flag, we can specify the warming-up period. In this experiment, we required the testing duration to be 20 seconds, and the tests must start 15 seconds after the command was launched. The `--warm-up-time` can be useful when trying to evaluate performance on a website. In fact, it is possible to launch the same command multiple times, spacing out the start of each one by an amount of time specified in the `--warm-up-time` flag. Alternatively, it can be used for preloading objects and seeing different results compared to the previous cases.

Conclusion

This report has helped me better understand concepts related to web technologies that I had only encountered in theory. Concerning parallel connections, I gained a deeper understanding of the consequences of establishing a large number of parallel connections to a server, and why significantly increasing this number does not guarantee improved performance. The caching mechanism was also analyzed in detail in practice using different websites with varying characteristics. An analysis of website performance was made possible with specific tools, which need to be thoroughly examined to better understand the output they provide. Overall, it has been very important for me to receive practical feedback on what I have studied in theory.