EDI: Third Lab Report

Sepideh Hayati 515184

June 25, 2023

Abstract

This study aimed to enhance our understanding of the HTTP protocol, caching, and the impact of parallel connections on page load times for commercial and institutional websites. By examining these factors, we aimed to gain valuable insights into website performance and responsiveness.

1 Web Technologies

1.1 Methodology and experimental setup

In the lab, we aimed to understand the workings of the HTTP protocol and its types. We also explored the impact of caching on page load times and network traffic. To test web server performance, we simulated heavy loads using Apache benchmarking tools.

1.2 Experimental results

1.2.1 Analyze and discuss the impacts of the number of parallel connections set inside the browser on the Page Load Times of commercial/institutional websites.

To examine the impact of parallel connections on the load time of commercial/organizational web pages, I used the Firefox browser. I performed the following steps to obtain the output and compare the results:

- Initially, I disabled the browser cache in Firefox(F2). I achieved this by navigating to *"about:config"* (F1) in the address bar and changing the value of *"browser.cache.disk.enable"* to *"false"*. This allowed me to assess the effect of parallel connections on page load time without the influence of the cache.

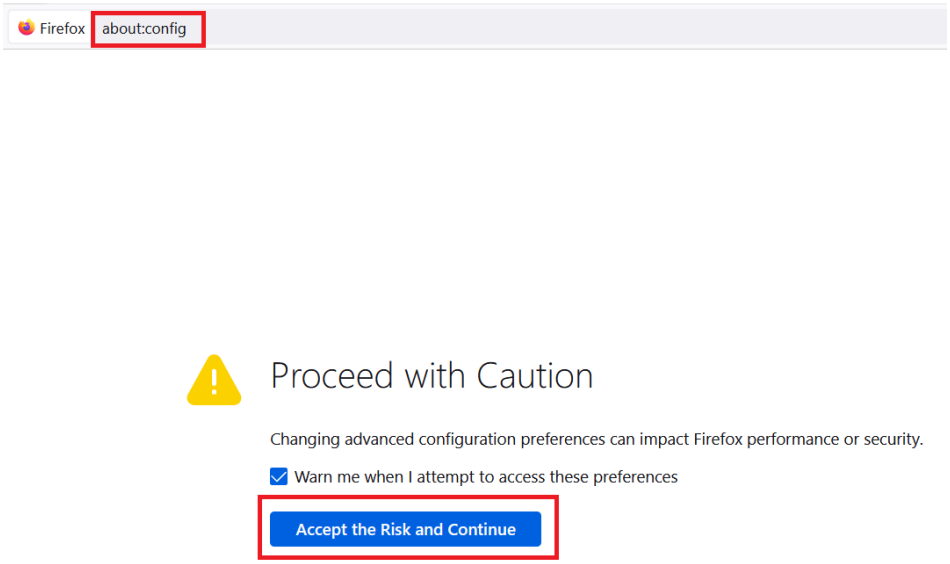

Figure 1: Mozila FireFox-about:config

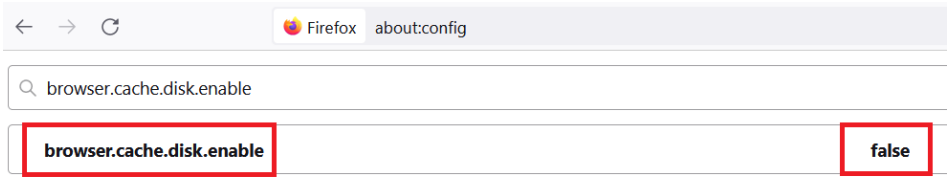

Figure 2: disabled the browser cache in Firefox

- To ensure the deactivation of the memory cache, I made the following changes:
  - I set *"browser.cache.memory.enable"* to *"false"*.(F3).
  - And also I Set *"browser.cache.memory.capacity"* to *"zero"*. (F4)

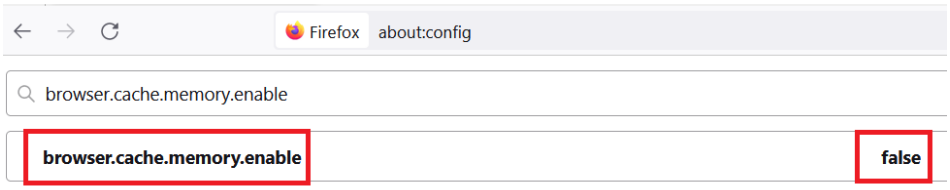

Figure 3: browser.cache.memory.enable

Figure 4: browser.cache.memory.capacity

- Next, I modified the number of parallel connections in the same path. I navigated to *"network.http.max-persistent-connections-per-server"* and selected my desired number. In the first experiment, I chose 6, and in the subsequent stage, I selected 10. (F5)

Figure 5: network.http.max-persistent-connections-per-server

- Then, I opened the *Developer Tools* and accessed the *Network tab*. I loaded commercial/institutional websites within the same window.
  - commercial www.youtube.com and www.amazon.com

Based on the results, having more parallel connections can make web pages load faster because the browser can load things at the same time. (F6), (F7) Using domain sharding, which means spreading resources across different domains, can make loading even faster by having multiple connections. However, during testing with different internet connections, I observed that the speed of Wi-Fi and mobile internet could impact the results, and increasing the number of parallel connections does not necessarily result in a significant increase in loading speed.

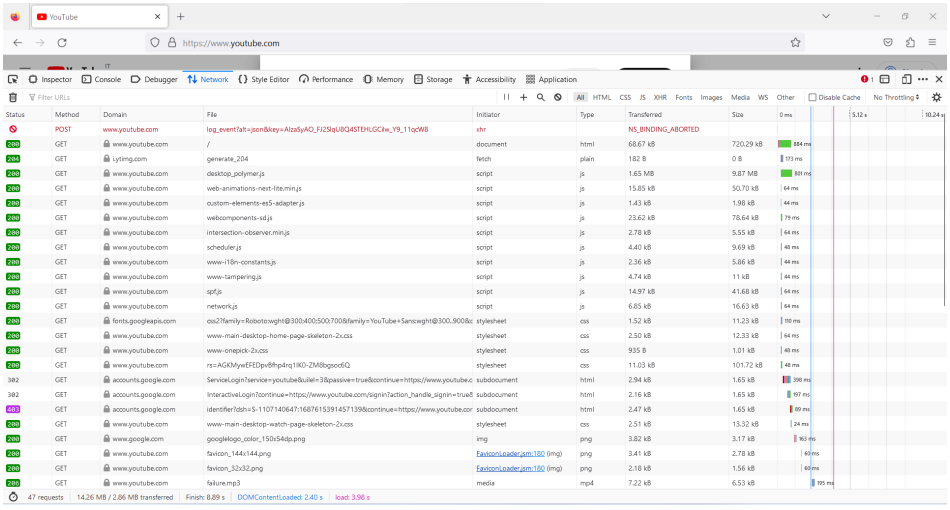

Figure 6: network.http.max-persistent-connections-per-server:6

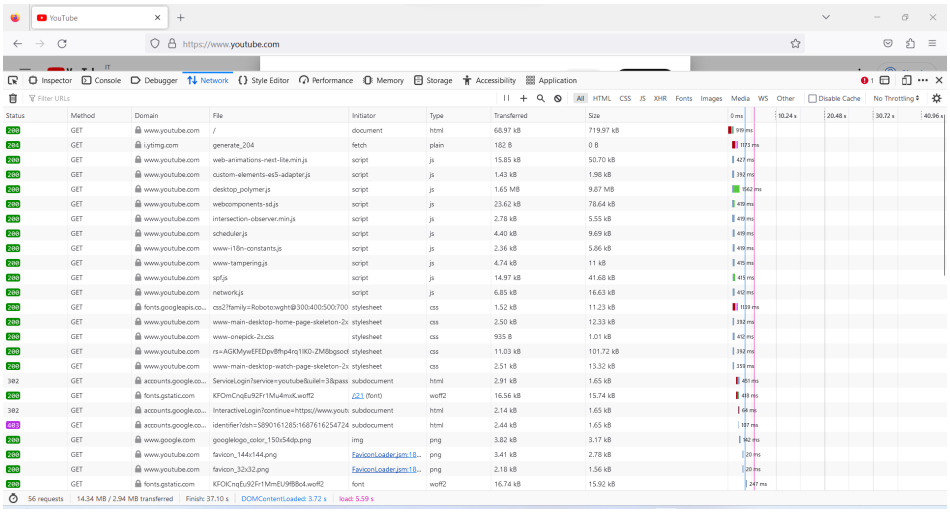

Figure 7: network.http.max-persistent-connections-per-server:10

1.2.2 Analyze and discuss the impacts of caching policies implemented by different commercial/institutional websites on the Page Load Times. Consider websites that support HTTP/1.1, HTTP/2, and HTTP/3 (possibly with insecure and secure connections). Did you notice any expected or unexpected behavior?

To examine the impact of caching policies on the load time of commercial/organizational web pages, I used the Firefox browser. I performed the following steps to obtain the output and compare the results, I disabled the browser cache in Firefox. (F8) I achieved this by navigating to *"about:config"* in the address bar and changing the value of *"browser.cache.disk.enable"* to *"false"*. This allowed me to assess the effect of cache enabled/disabled on page load time.

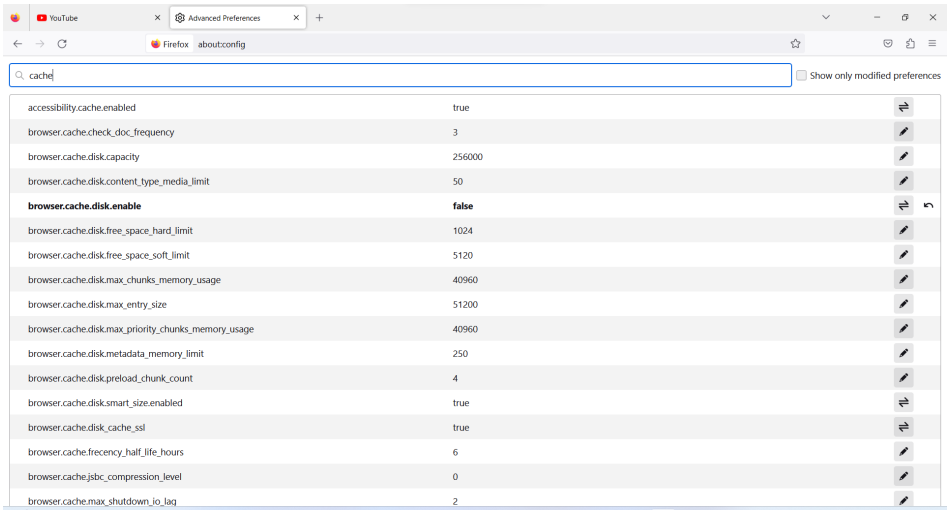

Figure 8: browser.cache.disk.enabled/disabled

The experiment shows that the caching strategies and tools influence the time it takes to load web pages (PLT). When the browser cache is disabled, every new search or page update requires all items to be reloaded, causing the site to become larger and increasing the time it takes to load. because resources cannot be directly accessed from the computer cache and need to be downloaded again, resulting in longer page load times (PLT).

1.2.3 Analyze and discuss the performance of different commercial/institutional websites obtained under different conditions using the *ab* Apache HTTP server benchmarking tool.

The ab command (ApacheBench) is used for benchmarking web servers. by using *ab* command I can send a specified number of requests to a web server and measures various performance metrics, such as the number of requests per second, the average response time, and the throughput. My example include *-n* for specifying the number of requests to be sent, *-c* for setting the concurrency level (number of requests to be sent concurrently).

The results for websites www.Enel.it:

- The benchmarking results for *ab -n 100 -c 10 https://www.enel.it/it/offerte* showed a total of 54 failed requests out of 100, resulting in non-2xx responses, with an average time per request of 177.662 ms and a transfer rate of 68.94 Kbytes/sec. (F9)

```
sepideh@ubuntu:~$ ab -n 100 -c 10 https://www.enel.it/it/offerte
This is ApacheBench, Version 2.3 <$Revision: 1901567 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking www.enel.it (be patient).....done


Server Software:
Server Hostname:        www.enel.it
Server Port:            443
SSL/TLS Protocol:       TLSv1.3,TLS_AES_128_GCM_SHA256,2048,128
Server Temp Key:        X25519 253 bits
TLS Server Name:        www.enel.it

Document Path:          /it/offerte
Document Length:        752 bytes

Concurrency Level:      10
Time taken for tests:   1.777 seconds
Complete requests:      100
Failed requests:        54
   (Connect: 0, Receive: 0, Length: 54, Exceptions: 0)
Non-2xx responses:      100
Total transferred:      125428 bytes
HTML transferred:       75261 bytes
Requests per second:    56.29 [#/sec] (mean)
Time per request:       177.662 [ms] (mean)
Time per request:       17.766 [ms] (mean, across all concurrent requests)
Transfer rate:          68.94 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       77  110  17.4    106     156
Processing:    33   47   5.4     47      60
Waiting:       33   47   5.4     46      59
Total:        118  157  19.0    154     207

Percentage of the requests served within a certain time (ms)
  50%    154
  66%    165
  75%    170
  80%    174
  90%    185
  95%    196
  98%    206
  99%    207
 100%    207 (longest request)
```

Figure 9: ab -n 100 -c 10 https://www.enel.it/it/offerte

The results for websites www.italiarail.com:

- The benchmarking results for *ab -n 100 -c 10 https://www.italiarail.com/rail-passes* showed all 100 requests completed successfully, with an average time per request of 501.204 ms and a transfer rate of 1,783.33 Kbytes/sec.

The first website *https://www.enel.it/it/offerte* had 54 failed requests, achieved a higher request rate of 56.29 requests per second, and had a faster average time per request of 177.662 milliseconds compared to the second website *https://www.italiarail.com/rail-passes*, which had no failed requests, a lower request rate of 19.95 requests per second, and a slower average time per request of 501.204 milliseconds. (F10)

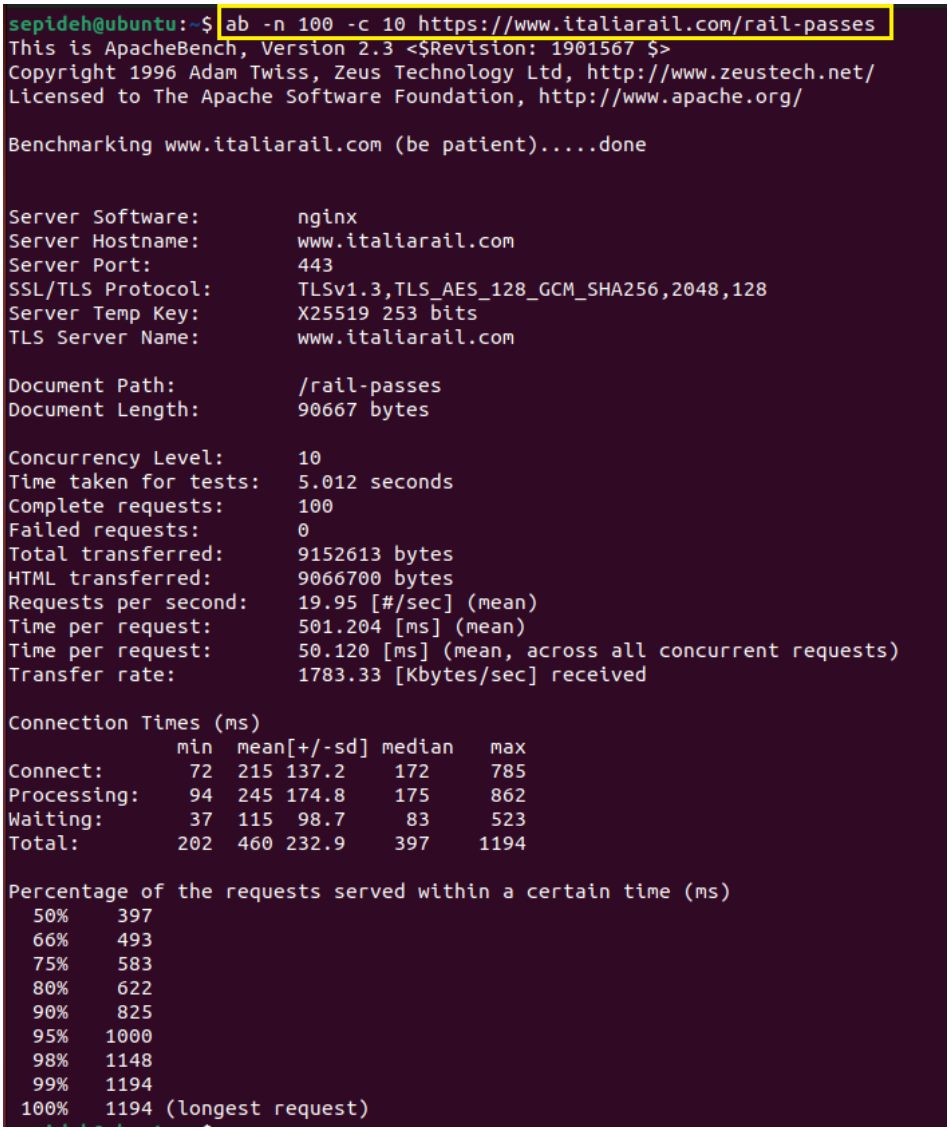

Figure 10: ab -n 100 -c 10 https://www.italiarail.com/rail-passes

1.2.4 Analyze and discuss the performance of different commercial/institutional websites obtained under different conditions using the *nghttp* and *h2load* tools. In the experiments with h2load analyze the role of the warm-up time.

Both *nghttp* and *h2load* are part of the nghttp2 project, which provides a set of tools used for performance testing and benchmarking of HTTP/2 and HTTP/3 protocols.

nghttp

I used the command *nghttp -vasn https://www.enel.it* to analyze the network communication of the website *https://www.enel.it* using the *nghttp* tool. By running this command, I could gain insights into the network communication of the *https://www.enel.it* website, including HTTP/2 streams and associated data frames. (F11)

- -v (verbose) Displays detailed output
- -a (all) Includes all available information in the output, like headers, trailers, and data frames.

- -s (streams): Prints information about individual streams, including IDs and states.
- -n (no-dep): Disables dependency tree printing.

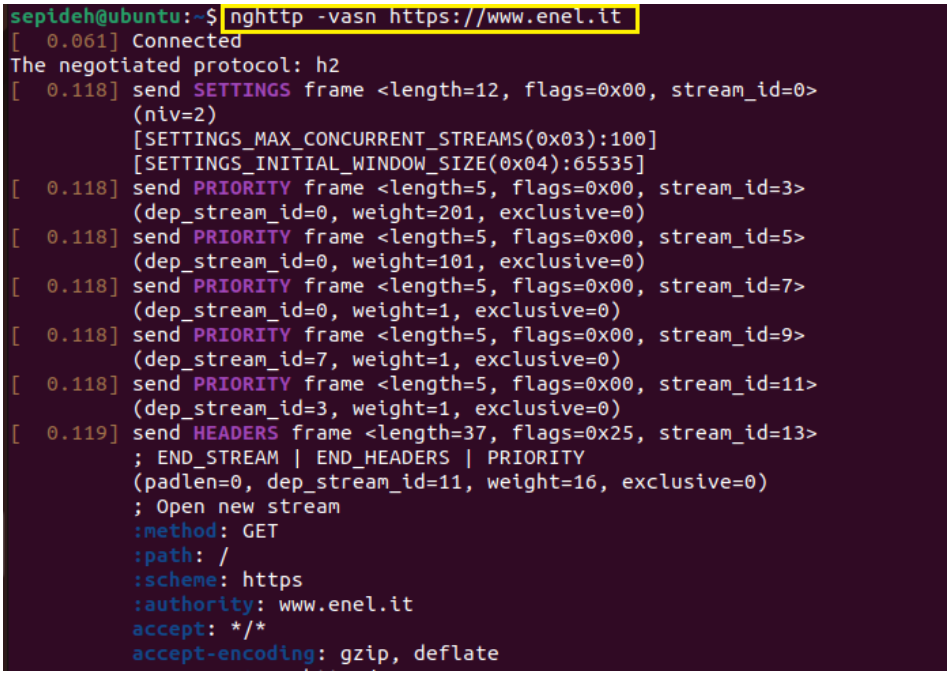

Figure 11: nghttp -vasn https://www.enel.it

Results

The *nghttp* command is used to analyze the network communication of the *enel.it* website. The output shows data frames being received and sent with different lengths, flags, and stream IDs. Window update frames were also observed to adjust the flow control window size. The communication involved streams identified by various stream IDs, including 91, 101, 105, and 113.

h2load

h2load is a benchmarking tool that is part of the nghttp2 project. I used the command *h2load -n 1000 -c 10 -m 100 –warm-up-time=5s* for two websites and also I changed the warm-up time to see changes.

- -n 1000: the number of requests that are sent.
- With -c 10, there will be 10 clients simultaneously sending requests.
- -m 100: Each client can have up to 100 concurrent streams.
- –warm-up-time=5s: Sets the warm-up time before the actual benchmarking starts. In this case, it is set to 5 seconds. (I changed it to 10S)

Overall, the *https://www.italiarail.com* server, sending 1000 requests with 10 concurrent clients, and a warm-up time of 5/10 seconds. The tool measures various performance metrics such as request rate, response time, and status codes to evaluate the server's performance under load.

Results h2load ”ENEL.it”

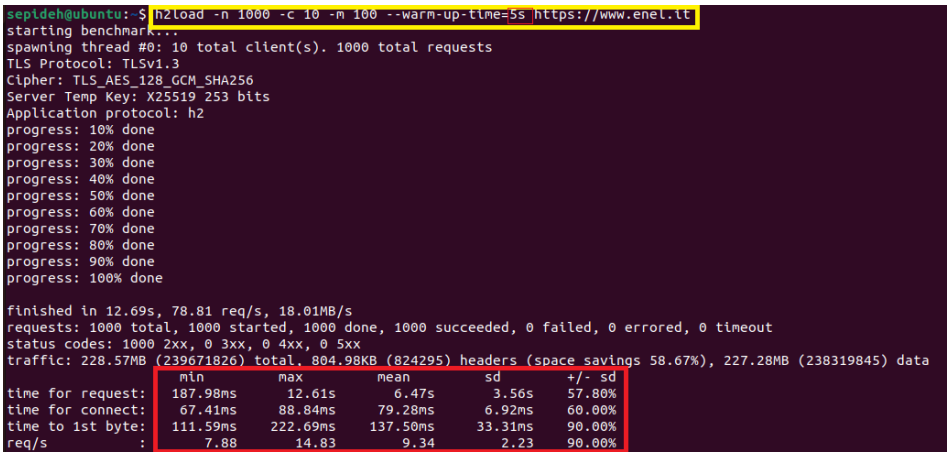

Figure 12: h2load -n 1000 -c 10 -m 100 –warm-up-time=5s https://www.enel.it

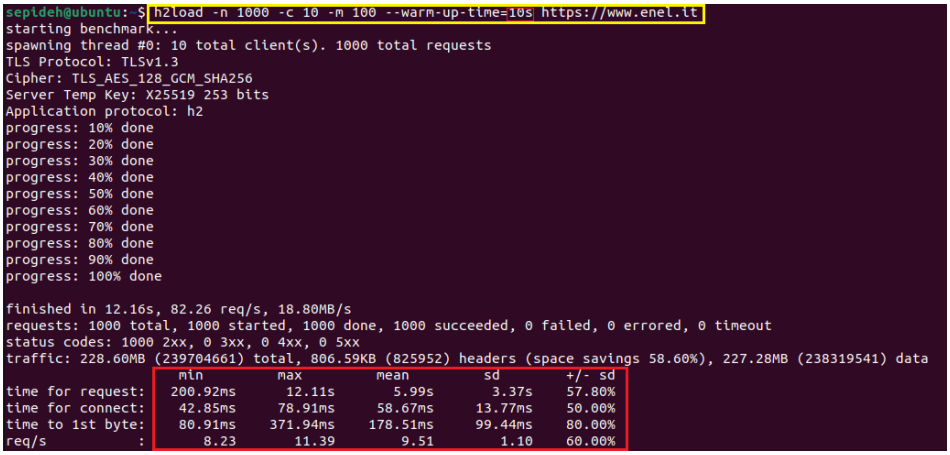

Figure 13: h2load -n 1000 -c 10 -m 100 –warm-up-time=10s https://www.enel.it

As you can see, when the warm-up time was set to 5 seconds (F12), the ”time to 1st byte” had a mean value of 137.50ms with an sd of 33.31ms. However, when the warm-up time was increased to 10 seconds (F13), the ”time to 1st byte” had a mean value of 178.51ms with a sd of 99.44ms. So I think with a longer warm-up time, the server take more time on average to send the initial response, and the response time became more variable.

Results h2load "Italiarail.com"

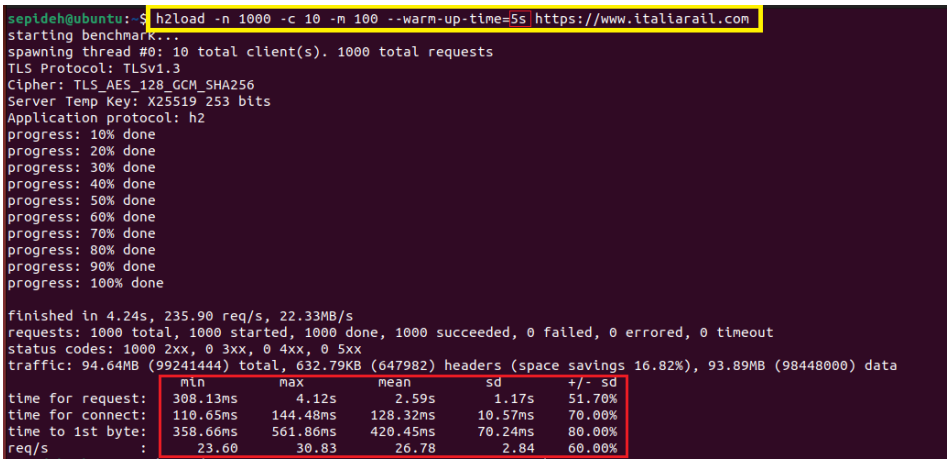

Figure 14: h2load -n 1000 -c 10 -m 100 –warm-up-time=5s https://www.Italiarail.com

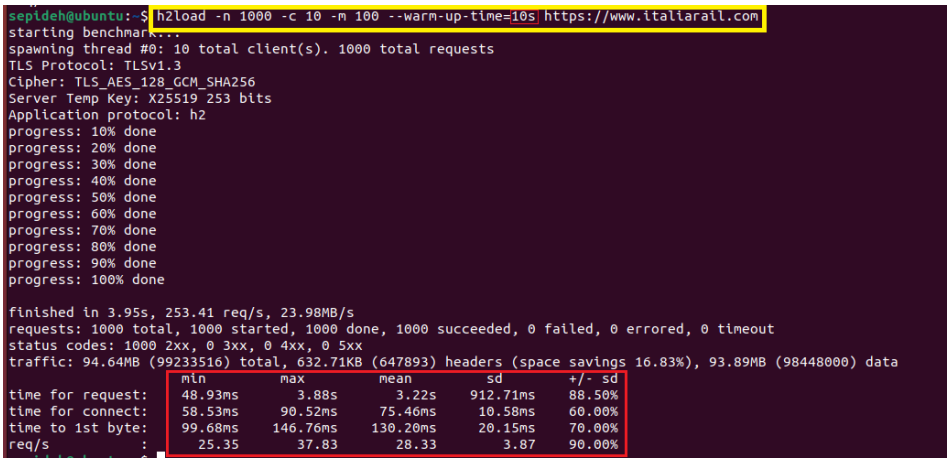

Figure 15: h2load -n 1000 -c 10 -m 100 –warm-up-time=10s https://www.Italiarail.com

2 Conclusion

In this experiment, we observed the significance of parallel connections and their configuration and examined their impact on the speed of page loading. By increasing the number of parallel connections, we observed faster load times for websites. This suggests that optimizing parallel connections can significantly improve the speed and responsiveness of web pages.