

# Desafío Loggers y Gzip

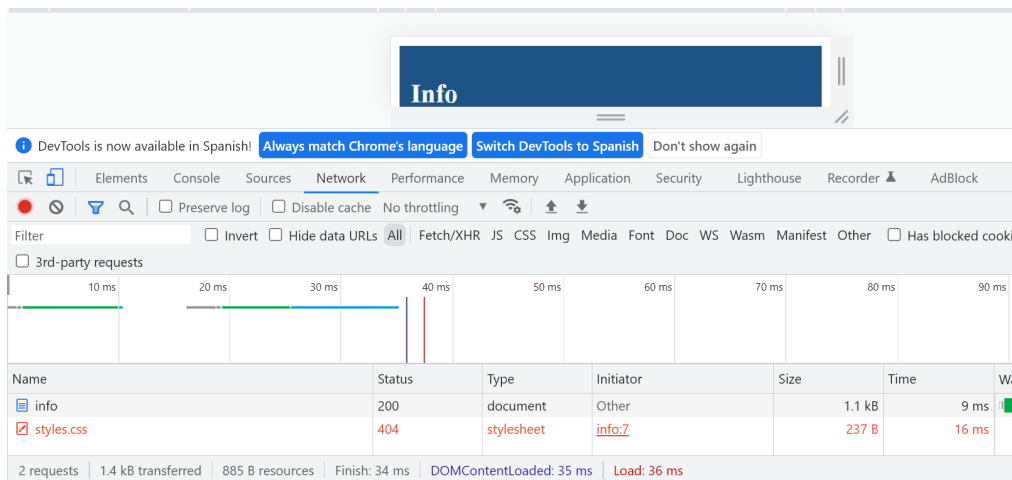
## Melina Señoráns Pérez

07/02/2023

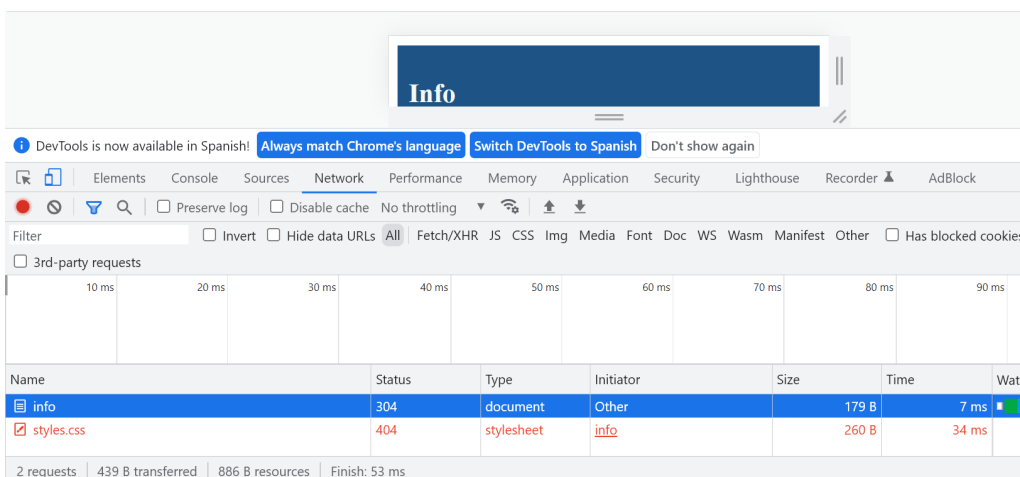
Verificar sobre la ruta /info con y sin compresión, la diferencia de cantidad de bytes devueltos en un caso y otro.

### Cantidad de bytes devueltos según uso de gzip:

- Sin compresión: 1.1 kB



- Con compresión: 179 B



## ANÁLISIS COMPLETO DE PERFORMANCE

### Comparación de ruta “/info” con y sin console.log()

Iniciando servidor en modo fork:

```
pm2 start server.js --name="pruebaCluster" --watch -i max -- -- 8080
```

```
pm2 start server.js --name="pruebaFork" --watch -- -- 8080
```

#### 1) Comparación utilizando artillery y el profiling de Node:

##### a) Sin console.log:

###### ★ Resultados de archivo “artillery-info-sin-console.txt”:

All VUs finished. Total time: 15 seconds.

```
http.codes.200: ..... 1000
http.request_rate: ..... 54/sec
http.requests: ..... 1000
http.response_time:
  min: ..... 104
  max: ..... 1416
  median: ..... 497.8
  p95: ..... 1022.7
  p99: ..... 1326.4
http.responses: ..... 1000
```

comando utilizado:

```
artillery quick --count 50 -n 20 http://localhost:8080/info > artillery-info-sin-console.txt
```

###### ❖ Resultado de profiling de Node (archivo “result-prof-sin-console.txt”):

Ticks totales: 3610

[Summary]:

ticks	total	nonlib	name
87	2.4%	97.8%	JavaScript
0	0.0%	0.0%	C++
50	1.4%	56.2%	GC
3521	97.5%		Shared libraries
2	0.1%		Unaccounted

##### b) Con console.log

###### ★ Resultados de archivo “artillery-info-con-console.txt”:

All VUs finished. Total time: 23 seconds

```
http.codes.200: ..... 1000
http.request_rate: ..... 53/sec
http.requests: ..... 1000
http.response_time:
  min: ..... 126
  max: ..... 2618
  median: ..... 944
  p95: ..... 1978.7
  p99: ..... 2515.5
```

http.responses: ..... 1000

comando utilizado:

artillery quick --count 50 -n 20 http://localhost:8080/info > artillery-info-con-console.txt

❖ **Resultados del profiling de Node, archivo “result-prof-con-console.txt”:**

Ticks totales: 4324

[Summary]:

ticks	total	nonlib	name
109	2.5%	100.0%	JavaScript
0	0.0%	0.0%	C++
59	1.4%	54.1%	GC
4215	97.5%		Shared libraries

**Conclusiones:**

- **Artillery:** Al agregar el console.log(), aumenta el tiempo de respuesta en 8 segundos (53%), y la mediana del tiempo de respuesta de http aumenta 447 ms (95%).
- **Profiling de Node:** Al aplicar el console.log(), el aumento de ticks totales es del 19,8% (714 ticks). La proporción del aumento fue variable entre categorías: en Javascript, los ticks aumentaron un 25,3%; en Shared Libraries los ticks aumentaron un 19,71%, y en GC un 18%.

→ **Test de carga con Autocannon: comparación (con y sin console.log) haciendo 100 conexiones concurrentes en 20 segundos).**

**Reporte: print screen de consola.**

→ **Resultado con console.log():**

The screenshot shows a Visual Studio Code terminal window with the following content:

```

NginxBNode > public > .js routelInfojs > ...
10 const carpetaProyecto = process.cwd();

> node benchmark.js

Running 20s test @ http://localhost:8080/info
100 connections

Stat      2.5%    50%    97.5%   99%    Avg      Stdev    Max
Latency   110 ms  125 ms  181 ms  230 ms  130.59 ms  21.21 ms  302 ms

Stat      1%      2.5%   50%    97.5%   Avg      Stdev    Min
Req/Sec   400     400    780    836     761.3    94.86    400
Bytes/Sec 456 kB  456 kB  889 kB  952 kB  867 kB   108 kB   456 kB

Req/Bytes counts sampled once per second.
# of samples: 20

15k requests in 20.05s, 17.3 MB read
PS D:\Melina\programacion\backend\desafios\14-logger-gzip\NginxBNode\public>
  
```

→ **Resultado sin console.log():**

The screenshot shows a Visual Studio Code terminal window with the following content:

```

NginxBNode > public > .js package.json > {} scripts > test
1 {

Running 20s test @ http://localhost:8080/info
100 connections

Stat      2.5%    50%    97.5%   99%    Avg      Stdev    Max
Latency   100 ms  113 ms  159 ms  197 ms  117.97 ms  19.1 ms  292 ms

Stat      1%      2.5%   50%    97.5%   Avg      Stdev    Min
Req/Sec   498     498    878    907     842.4    95.15    498
Bytes/Sec 566 kB  566 kB  998 kB  1.03 MB  958 kB   108 kB   566 kB

Req/Bytes counts sampled once per second.
# of samples: 20

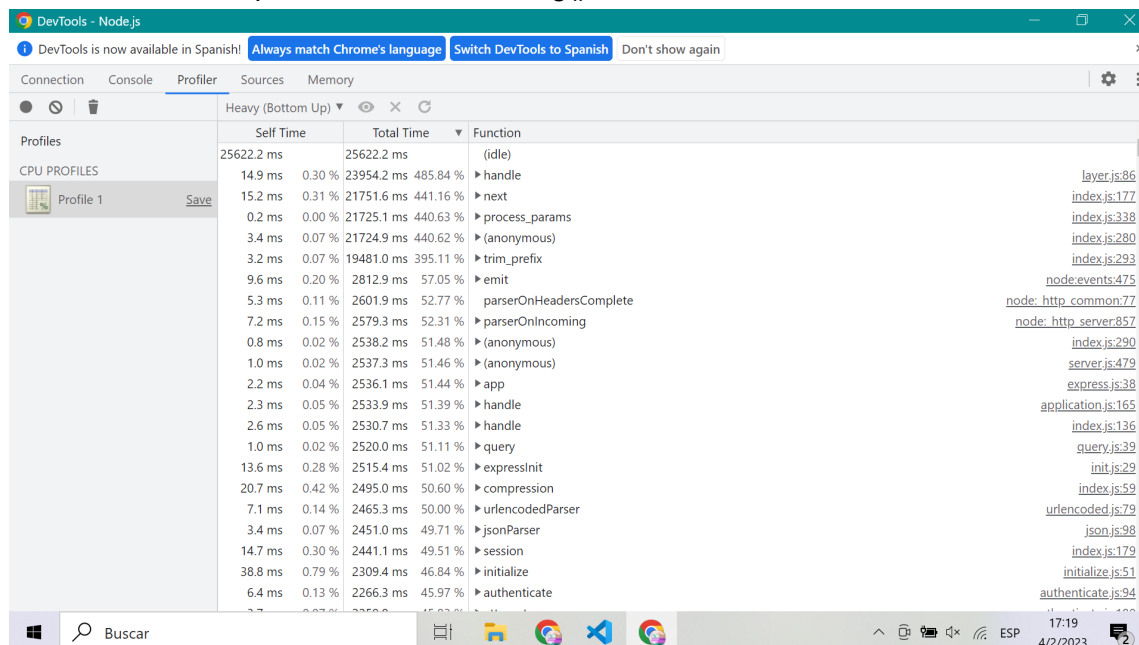
17k requests in 20.05s, 19.2 MB read
PS D:\Melina\programacion\backend\desafios\14-logger-gzip\NginxBNode\public>
  
```

## 2) Perfilamiento del servidor con el modo inspector de node.js --inspect (en chrome).

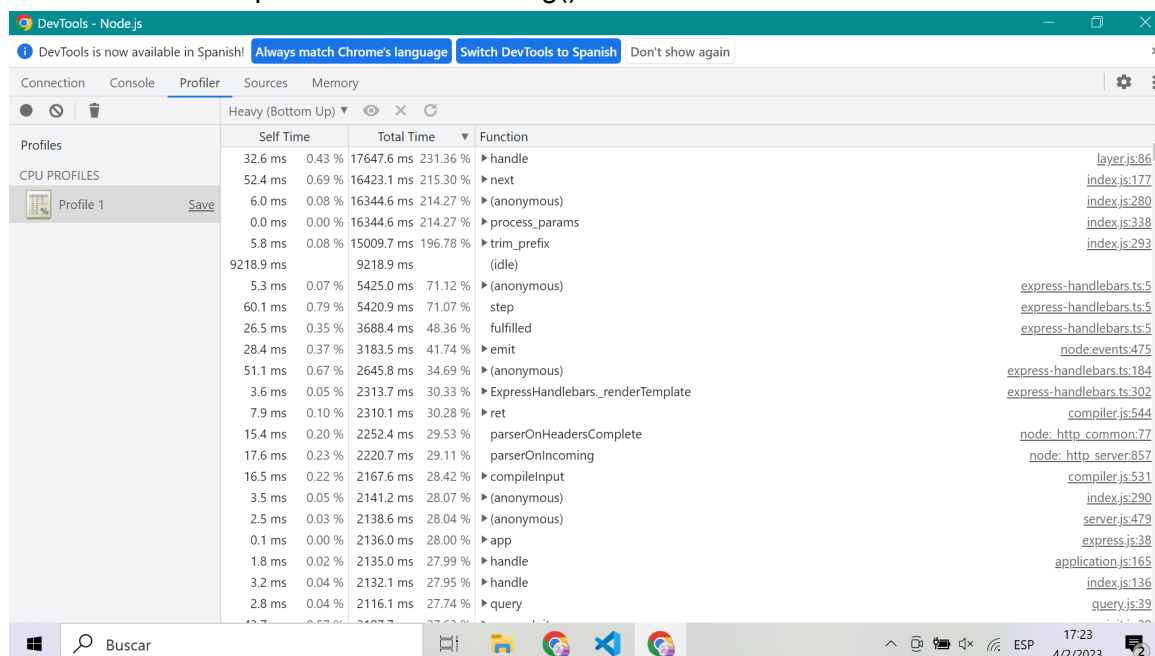
Servidor iniciado con `node --inspect server.js`

Comando utilizado para test de carga: `artillery quick --count 50 -n 20 http://localhost:8080/info`

Resultado de la inspección con `console.log()`:

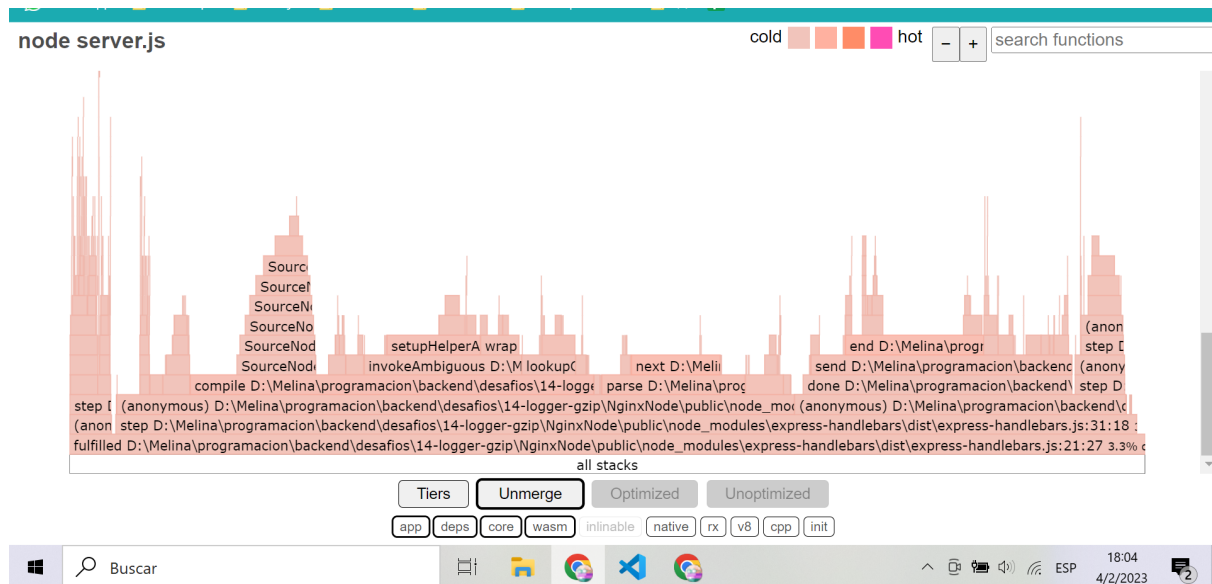


Resultado de la inspección sin `console.log()`:



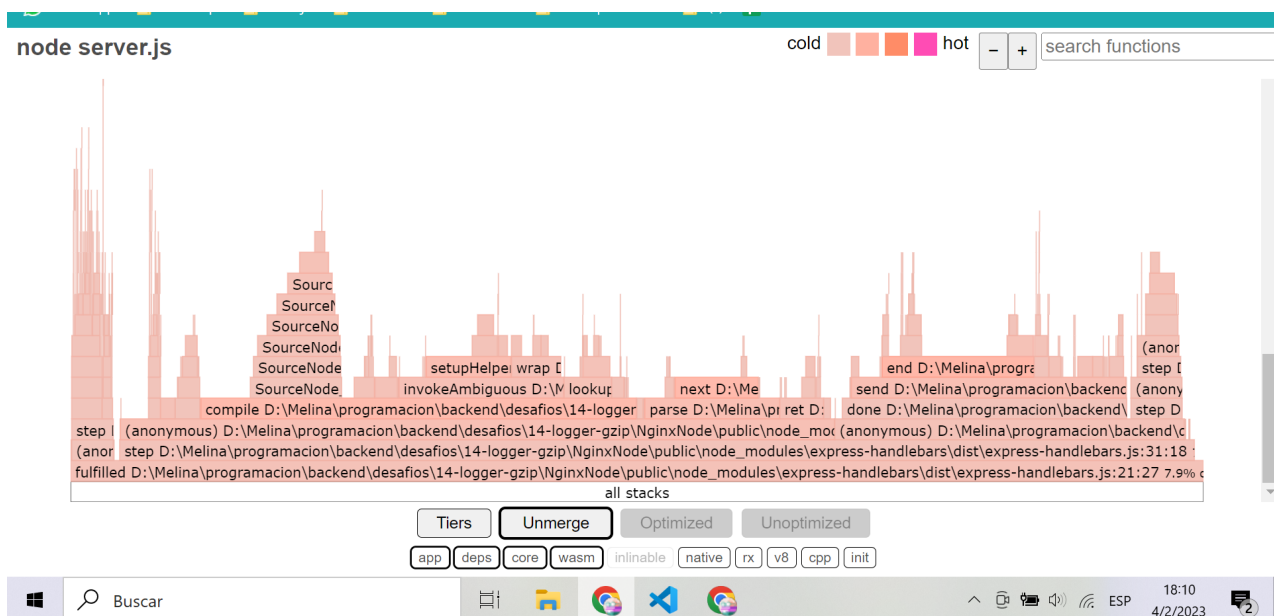
3) El diagrama de flama con 0x, emulando la carga con Autocannon con los mismos parámetros anteriores.

- **Resultado sin console.log():**



Flamegraph guardado en [NginxNode/public/952.0x/flamegraph-sin-console.html](http://NginxNode/public/952.0x/flamegraph-sin-console.html)

- **Resultado con console.log():**



Flamegraph guardado en [NginxNode/public/19612.0x/flamegraph-sin-console.html](http://NginxNode/public/19612.0x/flamegraph-sin-console.html)

No se ve una diferencia relevante en cuanto al bloqueo de los procesos entre el uso del console.log() o su ausencia al realizar una petición a la ruta "/info", ya que no se ve una extensión horizontal (en el tiempo) de los procesos.