

## ANÁLISIS COMPLETO DE PERFORMANCE

**Formato:** link a un repositorio en Github con el proyecto cargado.

**Sugerencia:** no incluir los node\_modules

Desafío  
entregable



**>> Consigna:** Luego, realizar el análisis completo de performance del servidor con el que venimos trabajando.

Vamos a trabajar sobre la ruta '/info', en modo fork, agregando ó extrayendo un console.log de la información colectada antes de devolverla al cliente. Además desactivaremos el child\_process de la ruta '/randoms'

Para ambas condiciones (con o sin console.log) en la ruta '/info' OBTENER:

1) El perfilamiento del servidor, realizando el test con --prof de node.js. Analizar los resultados obtenidos luego de procesarlos con --prof-process.

Utilizaremos como test de carga Artillery en línea de comandos, emulando 50 conexiones concurrentes con 20 request por cada una. Extraer un reporte con los resultados en archivo de texto.

**CODER HOUSE**

```
$ npm run server-fork
```

```
$ npx artillery quick --count 50 -n 40 http://localhost:8081/info > result_fork.txt
```

```
$ npm run server-cluster
```

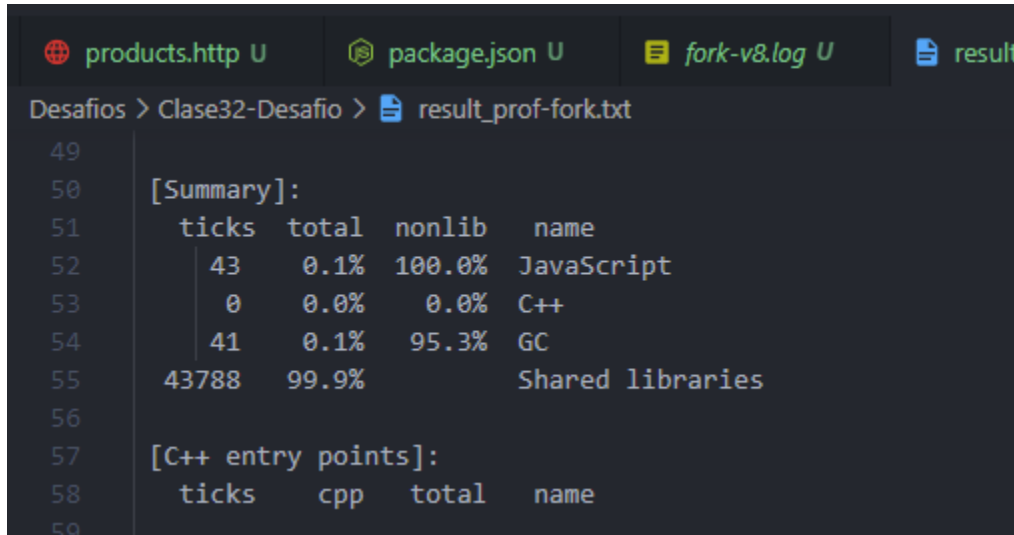
```
$ npx artillery quick --count 50 -n 40 http://localhost:8081/info > result_cluster.txt
```

```
index.js  products.http U  result_fork.txt U  ...  result_cluster.txt U X
Desafios > Clase32-Desafio > result_fork.txt
62 Report @ 06:47:21(-0500) 2022-02-03
63 Elapsed time: 45 seconds
64 Scenarios launched: 0
65 Scenarios completed: 50
66 Requests completed: 249
67 Mean response/sec: 42.91
68 Response time (msec):
69   min: 161
70   max: 932
71   median: 301
72   p95: 849.2
73   p99: 930
74 Codes:
75   200: 249
76
77 All virtual users finished
78 Summary report @ 06:47:21(-0500) 2022-02-03
79 Scenarios launched: 50
80 Scenarios completed: 50
81 Requests completed: 1000
82 Mean response/sec: 22.08
83 Response time (msec):
84   min: 4
85   max: 6273
86   median: 816
87   p95: 2790
88   p99: 4776.5
89 Scenario counts:
90   0: 50 (100%)
91 Codes:
92   200: 1000
93
Desafios > Clase32-Desafio > result_cluster.txt
62 Report @ 06:52:17(-0500) 2022-02-03
63 Elapsed time: 44 seconds
64 Scenarios launched: 0
65 Scenarios completed: 49
66 Requests completed: 160
67 Mean response/sec: 39.6
68 Response time (msec):
69   min: 166
70   max: 1028
71   median: 272.5
72   p95: 1010
73   p99: 1024.9
74 Codes:
75   200: 160
76
77 All virtual users finished
78 Summary report @ 06:52:17(-0500) 2022-02-03
79 Scenarios launched: 50
80 Scenarios completed: 50
81 Requests completed: 1000
82 Mean response/sec: 22.6
83 Response time (msec):
84   min: 4
85   max: 6163
86   median: 786
87   p95: 2395.5
88   p99: 4620
89 Scenario counts:
90   0: 50 (100%)
91 Codes:
92   200: 1000
93
```

```
$ npm run server:profiler:fork
```

```
$ npx artillery quick --count 50 -n 40 http://localhost:8081/info > result_prof_fork.txt
```

```
$ node --prof-process fork-v8.log > result_prof-fork.txt
```



```
49
50 [Summary]:
51   ticks  total  nonlib   name
52     43    0.1%  100.0% JavaScript
53      0    0.0%   0.0%    C++
54     41    0.1%  95.3%     GC
55  43788  99.9%           Shared libraries
56
57 [C++ entry points]:
58   ticks   cpp   total   name
59
```

## ANÁLISIS COMPLETO DE PERFORMANCE

**Formato:** link a un repositorio en Github con el proyecto cargado.

**Sugerencia:** no incluir los node\_modules

Desafío  
entregable



### >> Consigna:

Luego utilizaremos Autocannon en línea de comandos, emulando 100 conexiones concurrentes realizadas en un tiempo de 20 segundos. Extraer un reporte con los resultados (puede ser un print screen de la consola)

2) El perfilamiento del servidor con el modo inspector de node.js --inspect. Revisar el tiempo de los procesos menos performantes sobre el archivo fuente de inspección.

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

Realizar un informe en formato pdf sobre las pruebas realizadas incluyendo los resultados de todos los test (texto e imágenes).

👉 Al final incluir la conclusión obtenida a partir del análisis de los datos.

**CODER HOUSE**

```
$ node --inspect app.js --port=3001
```

DevTools - Node.js

ConnectionProfilerConsoleSourcesMemory

Heavy (Bottom Up)

Profiles

CPU PROFILES

Profile 1Save

Self Time	Total Time	Function
0.9 ms 0.01 %	1.9 ms 0.02 %	▸ yy.locInfo base.js:20
0.1 ms 0.00 %	0.9 ms 0.01 %	▸ xor scram.ts:270
11.8 ms 0.14 %	106.2 ms 1.24 %	▸ writevGeneric node:internal/s...se commons:126
81.9 ms 0.96 %	81.9 ms 0.96 %	▸ writev
5.8 ms 0.07 %	118.4 ms 1.38 %	▸ writetop index.js:270
0.9 ms 0.01 %	88.1 ms 1.03 %	▸ writeend index.js:260
10.9 ms 0.13 %	37.9 ms 0.44 %	▸ write_ node: http_outgoing:730
625.3 ms 7.29 %	625.3 ms 7.29 %	▸ writeUtf8String
0.6 ms 0.01 %	1.2 ms 0.01 %	▸ writeU_Int32LE node:internal/buffer:690
0.0 ms 0.00 %	0.1 ms 0.00 %	▸ writeUInt32LE node:internal/buffer:704
8.5 ms 0.10 %	1298.6 ms 15.14 %	▸ writeOrBuffer node:internal/s...s/writable:365
1.2 ms 0.01 %	2.2 ms 0.03 %	▸ writeInt32LE node:internal/buffer:860
14.4 ms 0.17 %	26.1 ms 0.30 %	▸ writeHead node: http_server:269
7.6 ms 0.09 %	139.5 ms 1.63 %	▸ writeHead index.js:28
4.5 ms 0.05 %	800.0 ms 9.32 %	▸ writeGeneric node:internal/s...se commons:151
5.5 ms 0.06 %	307.9 ms 3.59 %	▸ writeCommand message_stream.ts:73
120.0 ms 1.40 %	123.8 ms 1.44 %	▸ writeBuffer
8.5 ms 0.10 %	317.6 ms 3.70 %	▸ write connection.ts:765
8.1 ms 0.09 %	24.0 ms 0.28 %	▸ write node:buffer:592
3.2 ms 0.04 %	31.1 ms 0.36 %	▸ write node: http_outgoing:701
3.0 ms 0.03 %	22.2 ms 0.26 %	▸ write node:buffer:1050
0.5 ms 0.01 %	31.6 ms 0.37 %	▸ write index.js:78
0.3 ms 0.00 %	1.8 ms 0.02 %	▸ write node:buffer:628
0.3 ms 0.00 %	0.3 ms 0.00 %	▸ write node:buffer:641
4.0 ms 0.05 %	20.7 ms 0.24 %	▸ wrapper wrappy.js:19
2.6 ms 0.03 %	2.6 ms 0.03 %	▸ wranmethods index.js:394

\$ npm start

\$ curl -X GET <http://localhost:3001/info>

\$ npm run test

```
> npm run test
fjhua@DESKTOP-I1RRBQ6 E: > Cursos > CoderHouse > ProgramadorBackend > ProyectoClases
```

```
> app@1.0.0 test
> node benchmark.js
```

```
Running all benchmarks in parallel ...
Running 20s test @ http://localhost:3001/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	40 ms	5317 ms	9467 ms	9769 ms	5162.67 ms	2442.1 ms	9836 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	0	0	11	45	13.95	10.66	1
Bytes/Sec	0 B	0 B	22.2 kB	90.7 kB	28.1 kB	21.5 kB	2.01 kB

```
Req/Bytes counts sampled once per second.
```

```
424 requests in 20.18s, 562 kB read
45 errors (45 timeouts)
```