

ANÁLISIS COMPLETO DE PERFORMANCE

Se trabajó sobre la ruta '/info' y ruta '/info-log' , en modo fork:

Además desactivé el child_process de la ruta '/randoms' para todas las opciones.

```
47 router.get('/api/randoms', (req, res) => {
48   logger.info(`Ruta: /api/randoms, Método: get.`)
49
50   let cant = req.query.cant
51   if (!cant) cant = 100000000
52
53   //proceso fork
54
55   //Desactivo el child process para Desafío LOGGERS, GZIP Y ANALISIS DE PERFORMANCE
56   //*****
57
58   /* const forked = fork(path.join(path.dirname(''), '/api/randoms.js'))
59   forked.on('message', msg => {
60     if (msg == 'listo') {
61       forked.send(cant)
62     } else {
63       res.send(msg)
64     }
65   }) */
66
67 })
```

1 – ruta info

En terminal 1: node server.js --port 8081 --modo FORK

En terminal 2: artillery quick --count 50 -n 10 "http://localhost:8081/info" > result_fork.txt

2 - ruta /info-log

En terminal 1: node server.js --port 8081 --modo FORK

En terminal 2: artillery quick --count 50 -n 10 "http://localhost:8081/info-log" > result_fork_consolelog.txt

3 -

En terminal 1: node --prof server.js --port 8081 --modo FORK

En directorio del desafío: curl -X GET "http://localhost:8081/info"

En terminal 2: artillery quick --count 50 -n 10 "http://localhost:8081/info" > result_info_prof.txt

Renombro archivo: isolate-0000018F57F86560-2548-v8.log

Como archivo: isolate-info-v8.log

En terminal 1: node --prof-process isolate-Fork-SinConsoleLog.log > result-info-log_prof.txt

En terminal 2: artillery quick --count 50 -n 10 "http://localhost:8081/info-log" > result_info-log_prof.txt

Renombro archivo: isolate-0000018F57F86560-2550-v8.log

Como archivo: isolate-info-log-v8.log

4 – Test autocannon ruta /info

Terminal 1: node server.js --port 8081 --modo FORK

Terminal 2: node benchmark.js

Terminal 3: artillery quick --count 50 -n 10 "http://localhost:8081/info" > result_autocannon-info.txt

```
PS C:\Users\PC\Desktop\coderhouse\Backend\clases\clase32-DesafioLoggersGzipAnalysisPerformance> node benchmark.js
Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8081/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	318 ms	550 ms	1170 ms	1263 ms	554.29 ms	194.69 ms	1418 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	46	46	189	300	179.75	64.45	46
Bytes/Sec	207 kB	207 kB	851 kB	1.35 MB	809 kB	290 kB	207 kB

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

4k requests in 20.21s, 16.2 MB read

Test autocannon ruta /info-log

Terminal 1: node server.js --port 8081 --modo FORK

Terminal 2: node benchmark.js

Terminal 3: artillery quick --count 50 -n 10 "http://localhost:8081/info-log" > result_autocannon-info-log.txt

```
PS C:\Users\PC\Desktop\coderhouse\Backend\clases\clase32-DesafioLoggersGzipAnalysisPerformance> node benchmark.js
Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8081/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	334 ms	429 ms	777 ms	958 ms	459.47 ms	122.29 ms	1192 ms

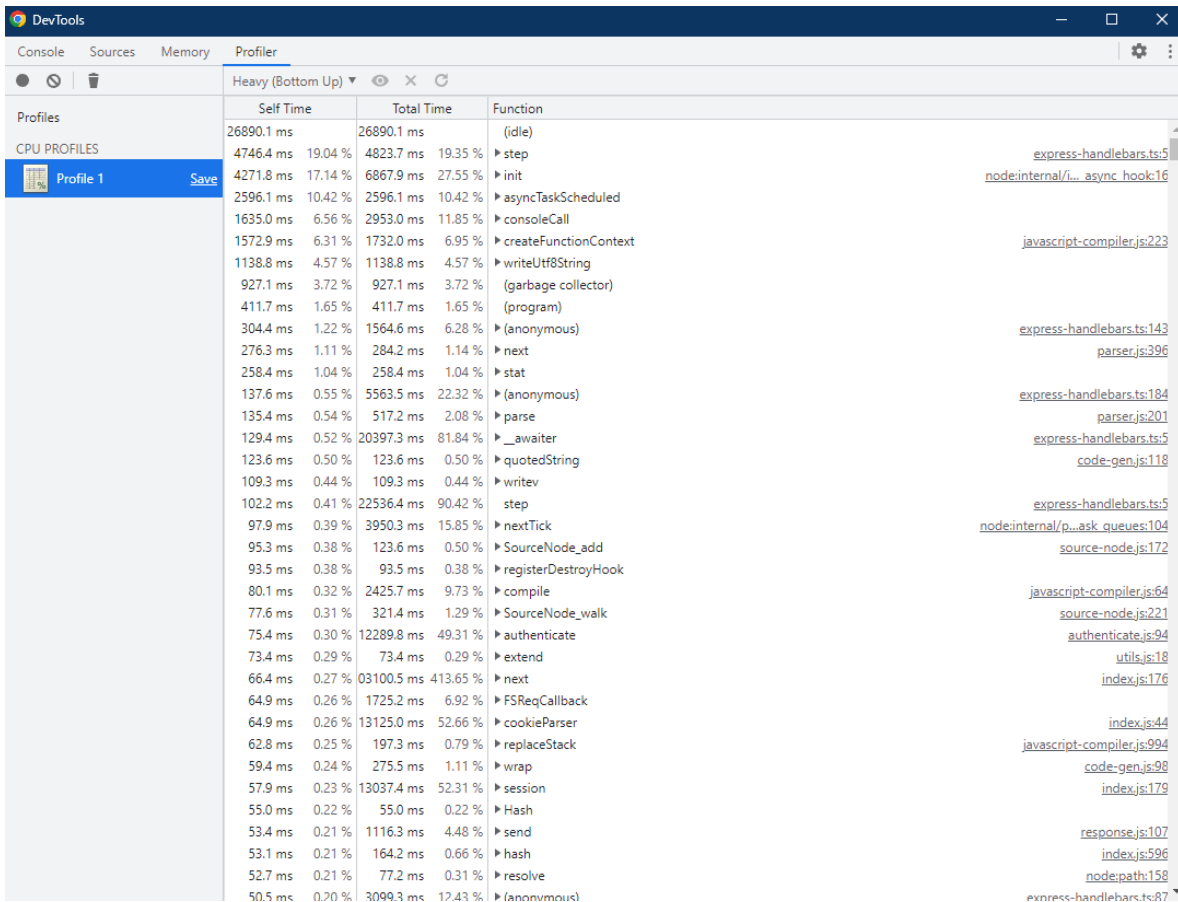
Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	97	97	217	298	214.5	55.49	97
Bytes/Sec	437 kB	437 kB	977 kB	1.34 MB	965 kB	250 kB	437 kB

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

4k requests in 20.25s, 19.3 MB read

6 - Perfilamiento del servidor con modo inspector de node.js --inspect

- Terminal 1: `node --inspect server.js`
- Ingreso a inspect de chrome (`chrome://inspect/`) y grabo test
- Terminal 2: `artillery quick --count 50 -n 10 "http://localhost:8080/info" > result_inspect-info.txt`
- Y luego `artillery quick --count 50 -n 10 "http://localhost:8080/info-log" > result_inspect-info-log.txt`

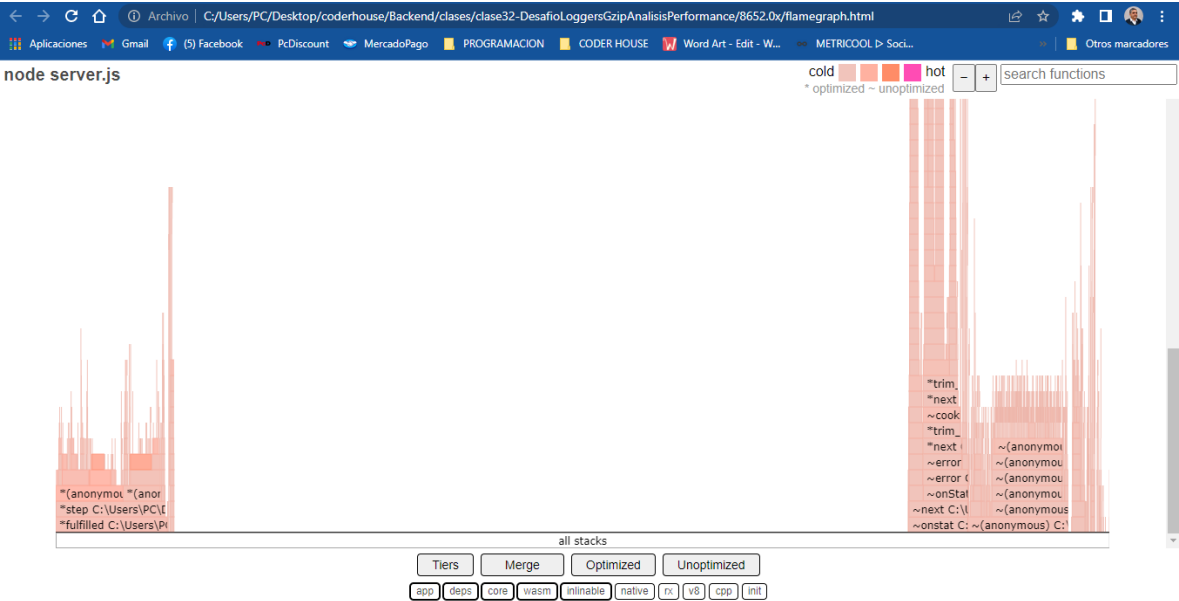


Profiles	Self Time	Total Time	Function
	26890.1 ms	26890.1 ms	(idle)
CPU PROFILES	4746.4 ms 19.04 %	4823.7 ms 19.35 %	▶ step express-handlebars.ts:5
Profile 1	4271.8 ms 17.14 %	6867.9 ms 27.55 %	▶ init node:internal/... async hook:16
	2596.1 ms 10.42 %	2596.1 ms 10.42 %	▶ asyncTaskScheduled
	1635.0 ms 6.56 %	2953.0 ms 11.85 %	▶ consoleCall
	1572.9 ms 6.31 %	1732.0 ms 6.95 %	▶ createFunctionContext javascript-compiler.js:223
	1138.8 ms 4.57 %	1138.8 ms 4.57 %	▶ writeUtf8String
	927.1 ms 3.72 %	927.1 ms 3.72 %	(garbage collector)
	411.7 ms 1.65 %	411.7 ms 1.65 %	(program)
	304.4 ms 1.22 %	1564.6 ms 6.28 %	▶ (anonymous) express-handlebars.ts:143
	276.3 ms 1.11 %	284.2 ms 1.14 %	▶ next parser.js:396
	258.4 ms 1.04 %	258.4 ms 1.04 %	▶ stat
	137.6 ms 0.55 %	5563.5 ms 22.32 %	▶ (anonymous) express-handlebars.ts:184
	135.4 ms 0.54 %	517.2 ms 2.08 %	▶ parse parser.js:201
	129.4 ms 0.52 %	20397.3 ms 81.84 %	▶ __awaiter express-handlebars.ts:5
	123.6 ms 0.50 %	123.6 ms 0.50 %	▶ quotedString code-gen.js:118
	109.3 ms 0.44 %	109.3 ms 0.44 %	▶ writev
	102.2 ms 0.41 %	22536.4 ms 90.42 %	▶ step express-handlebars.ts:5
	97.9 ms 0.39 %	3950.3 ms 15.85 %	▶ nextTick node:internal/p...ask queues:104
	95.3 ms 0.38 %	123.6 ms 0.50 %	▶ SourceNode_add source-node.js:172
	93.5 ms 0.38 %	93.5 ms 0.38 %	▶ registerDestroyHook
	80.1 ms 0.32 %	2425.7 ms 9.73 %	▶ compile javascript-compiler.js:64
	77.6 ms 0.31 %	321.4 ms 1.29 %	▶ SourceNode_walk source-node.js:221
	75.4 ms 0.30 %	12289.8 ms 49.31 %	▶ authenticate authenticate.js:94
	73.4 ms 0.29 %	73.4 ms 0.29 %	▶ extend utils.js:18
	66.4 ms 0.27 %	03100.5 ms 413.65 %	▶ next index.js:176
	64.9 ms 0.26 %	1725.2 ms 6.92 %	▶ FSReqCallback
	64.9 ms 0.26 %	13125.0 ms 52.66 %	▶ cookieParser index.js:44
	62.8 ms 0.25 %	197.3 ms 0.79 %	▶ replaceStack javascript-compiler.js:994
	59.4 ms 0.24 %	275.5 ms 1.11 %	▶ wrap code-gen.js:98
	57.9 ms 0.23 %	13037.4 ms 52.31 %	▶ session index.js:179
	55.0 ms 0.22 %	55.0 ms 0.22 %	▶ Hash
	53.4 ms 0.21 %	1116.3 ms 4.48 %	▶ send response.js:107
	53.1 ms 0.21 %	164.2 ms 0.66 %	▶ hash index.js:596
	52.7 ms 0.21 %	77.2 ms 0.31 %	▶ resolve node:path:158
	50.5 ms 0.20 %	3099.3 ms 12.43 %	▶ (anonymous) express-handlebars.ts:87



DevTools				
Console Sources Memory Profiler				
Tree (Top Down)				
Profiles				
CPU PROFILES				
Profile 1	Save			
	Self Time	Total Time	Function	
26890.1 ms		26890.1 ms	(idle)	
927.1 ms 3.72 %		927.1 ms 3.72 %	(garbage collector)	
411.7 ms 1.65 %		411.7 ms 1.65 %	(program)	
33.5 ms 0.13 %	14005.9 ms 56.19 %		▶ callbackTrampoline	
30.7 ms 0.12 %	1055.7 ms 4.24 %		▶ processTicksAndRejections	
22.5 ms 0.09 %	158.6 ms 0.64 %		▶ parserOnMessageComplete	
16.2 ms 0.07 %	7233.9 ms 29.02 %		▶ fulfilled	
15.0 ms 0.06 %	939.4 ms 3.77 %		▶ parserOnHeadersComplete	
10.7 ms 0.04 %	70.9 ms 0.28 %		▶ emitHook	
10.4 ms 0.04 %	24.1 ms 0.10 %		▶ promiseAfterHook	
9.8 ms 0.04 %	20.6 ms 0.08 %		▶ promiseBeforeHook	
4.7 ms 0.02 %	16.4 ms 0.07 %		▶ (anonymous)	
3.8 ms 0.02 %	3.8 ms 0.02 %		after	
3.6 ms 0.01 %	3.6 ms 0.01 %		step	
3.5 ms 0.01 %	3.5 ms 0.01 %		before	
1.6 ms 0.01 %	1.6 ms 0.01 %		processPromiseRejections	
1.6 ms 0.01 %	38.4 ms 0.15 %		▶ processTimers	
0.8 ms 0.00 %	0.8 ms 0.00 %		emitDestroyScript	
0.6 ms 0.00 %	0.6 ms 0.00 %		destroy	
0.5 ms 0.00 %	0.5 ms 0.00 %		(anonymous)	
0.5 ms 0.00 %	0.5 ms 0.00 %		emitBeforeScript	
0.3 ms 0.00 %	0.3 ms 0.00 %		hasHooks	
0.2 ms 0.00 %	0.2 ms 0.00 %		setRequestTimeout	
0.2 ms 0.00 %	0.2 ms 0.00 %		resume_	
0.2 ms 0.00 %	0.2 ms 0.00 %		needFinish	
0.2 ms 0.00 %	0.2 ms 0.00 %		trackPromise	
0.2 ms 0.00 %	0.2 ms 0.00 %		ExpressHandlebars_compileTemplate	
0.2 ms 0.00 %	0.2 ms 0.00 %		readFileAfterClose	
0.0 ms 0.00 %	5.7 ms 0.02 %		▶ emitInitNative	

Diagrama de flama usando 0x y autocannon



```
PS C:\Users\PC\Desktop\coderhouse\Backend\clases\clase32-DesafioLoggersGzipAnalisisPerformance> npm test

> socket@1.0.0 test
> node benchmark.js

Runnig all benchmarks in parallel ...
Running 20s test @ http://localhost:8080/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	686 ms	852 ms	1405 ms	1428 ms	879.51 ms	160.56 ms	1568 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	7	7	112	156	111.2	29.84	7
Bytes/Sec	31.3 kB	31.3 kB	502 kB	698 kB	498 kB	134 kB	31.3 kB

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

2k requests in 20.17s, 9.95 MB read
Running 20s test @ http://localhost:8080/info-log
100 connections

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	701 ms	863 ms	1322 ms	1347 ms	886.49 ms	158.69 ms	1462 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	0	0	101	196	110	38.24	86
Bytes/Sec	0 B	0 B	452 kB	878 kB	492 kB	171 kB	385 kB

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

2k requests in 20.23s, 9.85 MB read