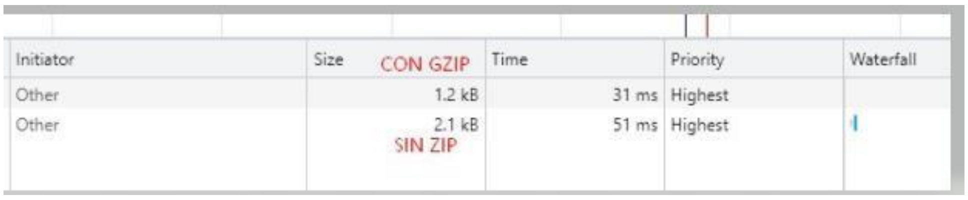
Ir a la ruta /info con y sin compresión, ver la diferencia de cantidad de bytes generados en un caso y o



Initiator	Size	Time	Priority	Waterfall
Other	1.2 kB	31 ms	Highest	
Other	2.1 kB	51 ms	Highest	

Test de performance sobre la ruta /info

Perfilamiento del servidor, realizando el test con --prof de node.js, analizando losresultados obtenidos luego de procesarlos con --prof-process. Request realizadasu**nArtillery**

Conconsolelog

[Summary]:

tickst o t a l n o n l i b name

158 1.0%

98.1%J a v a S c r i p t 0 0.0%

0.0%C + +

226 1.5%1 4 0 . 4 % G C

15349 99.0% Sharedlibraries

3 0.0% Unaccounted

Sinconsolelog

[Summary]:

tickst o t a l n o n l i b name

156 1.5% 99.4%JavaScript0

0.0% 0.0%C + +

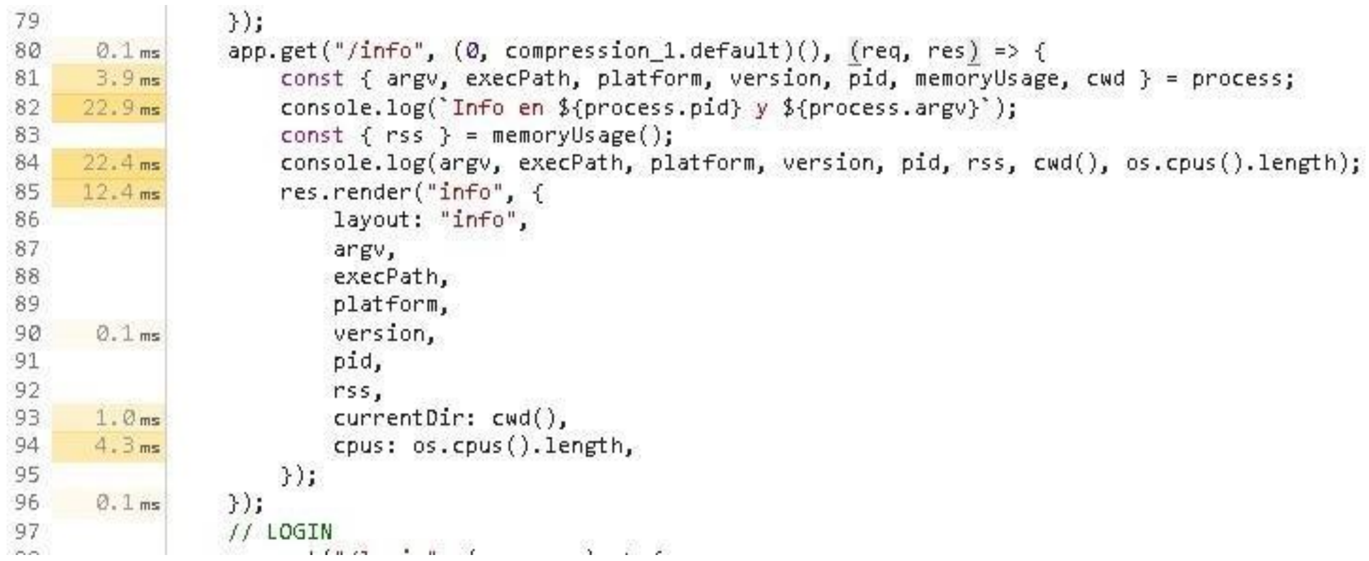
218 2.2%1 3 8 . 9 % G C

9948 98.4% Sharedlibraries

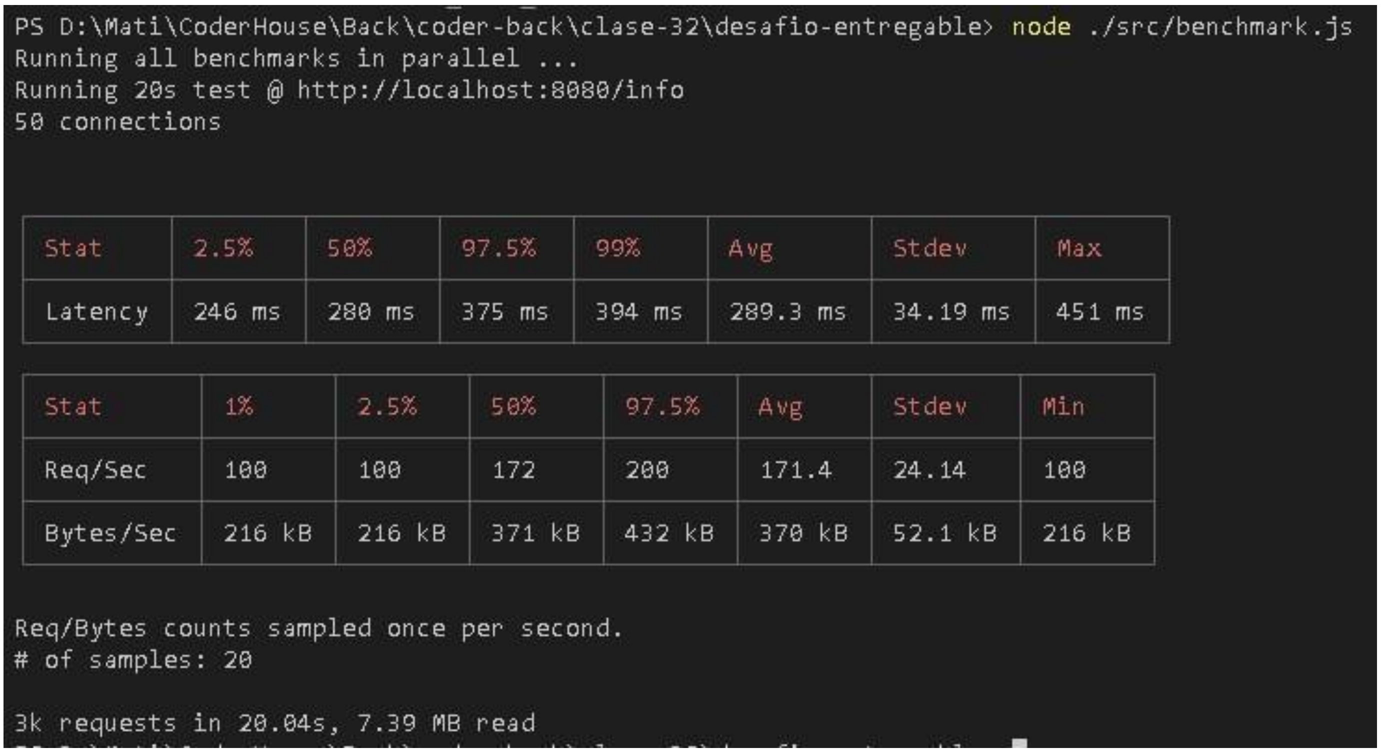
1 0.0% Unaccounted

Pruebas utilizando **Autocannon** en línea de comandos, emulando 100 conexiones concurrentes realizadas en un tiempo de 20segundos.

Con console log:

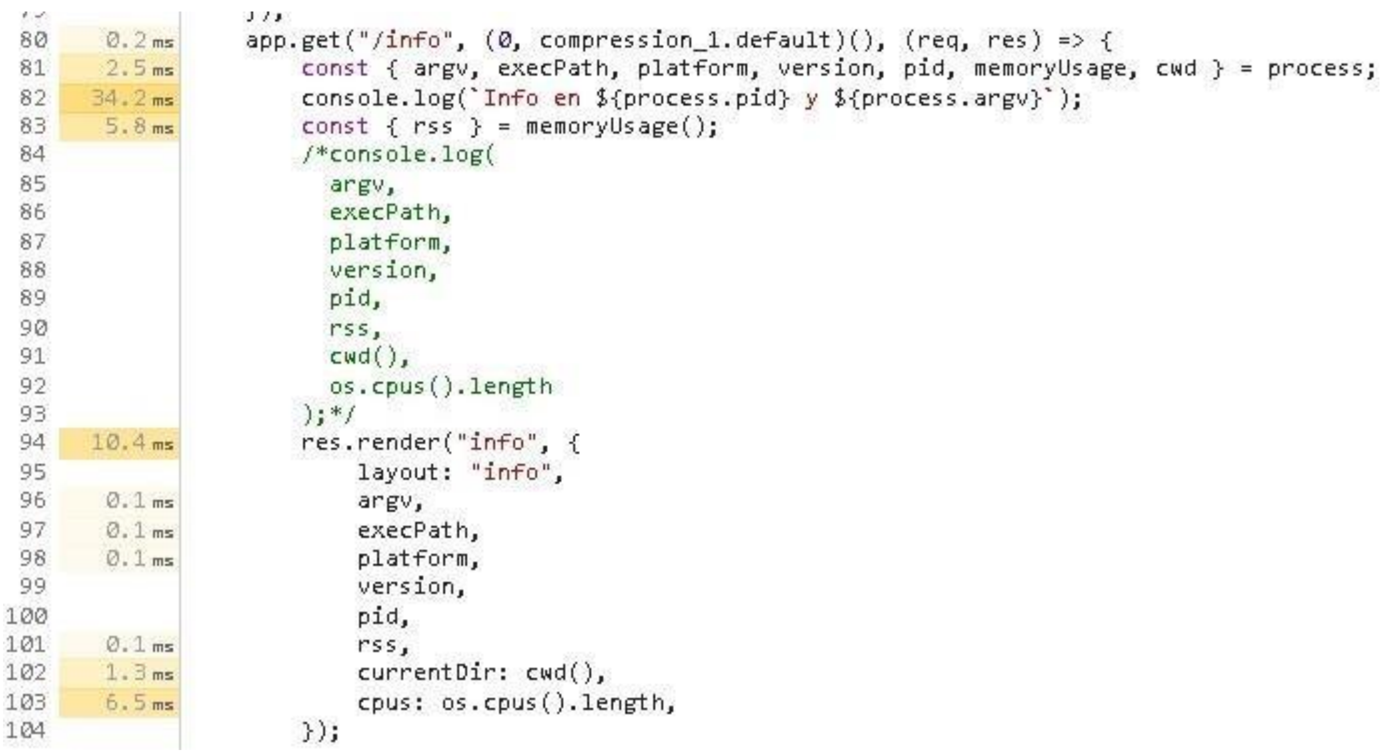


```
           });
  0.1 ms   app.get("/info", (0, compression_1.default)(), (req, res) => {
  3.9 ms       const { argv, execPath, platform, version, pid, memoryUsage, cwd } = process;
 22.9 ms       console.log(`Info en ${process.pid} y ${process.argv}`);
               const { rss } = memoryUsage();
 22.4 ms       console.log(argv, execPath, platform, version, pid, rss, cwd(), os.cpus().length);
 12.4 ms       res.render("info", {
                   layout: "info",
                   argv,
                   execPath,
                   platform,
  0.1 ms           version,
                   pid,
                   rss,
  1.0 ms           currentDir: cwd(),
  4.3 ms           cpus: os.cpus().length,
               });
  0.1 ms   });
           // LOGIN
```

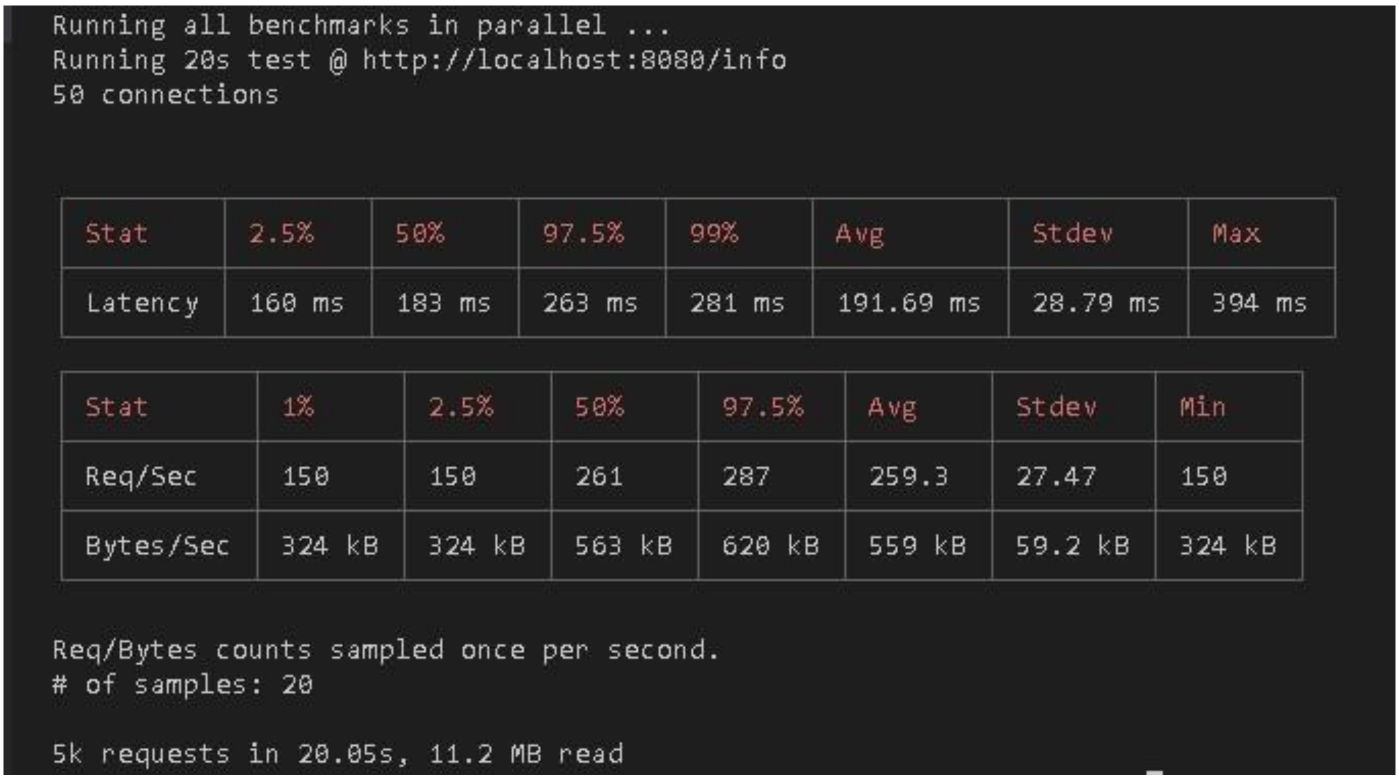


```
PS D:\Mati\CoderHouse\Back\coder-back\clase-32\desafio-entregable> node ./src/benchmark.js
Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8080/info
50 connections


┌─────────┬────────┬────────┬────────┬────────┬──────────┬──────────┬────────┐
│ Stat    │ 2.5%   │ 50%    │ 97.5%  │ 99%    │ Avg      │ Stdev    │ Max    │
├─────────┼────────┼────────┼────────┼────────┼──────────┼──────────┼────────┤
│ Latency │ 246 ms │ 280 ms │ 375 ms │ 394 ms │ 289.3 ms │ 34.19 ms │ 451 ms │
└─────────┴────────┴────────┴────────┴────────┴──────────┴──────────┴────────┘
┌───────────┬────────┬────────┬────────┬────────┬────────┬─────────┬────────┐
│ Stat      │ 1%     │ 2.5%   │ 50%    │ 97.5%  │ Avg    │ Stdev   │ Min    │
├───────────┼────────┼────────┼────────┼────────┼────────┼─────────┼────────┤
│ Req/Sec   │ 100    │ 100    │ 172    │ 200    │ 171.4  │ 24.14   │ 100    │
├───────────┼────────┼────────┼────────┼────────┼────────┼─────────┼────────┤
│ Bytes/Sec │ 216 kB │ 216 kB │ 371 kB │ 432 kB │ 370 kB │ 52.1 kB │ 216 kB │
└───────────┴────────┴────────┴────────┴────────┴────────┴─────────┴────────┘

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

3k requests in 20.04s, 7.39 MB read
```

Sin console log:

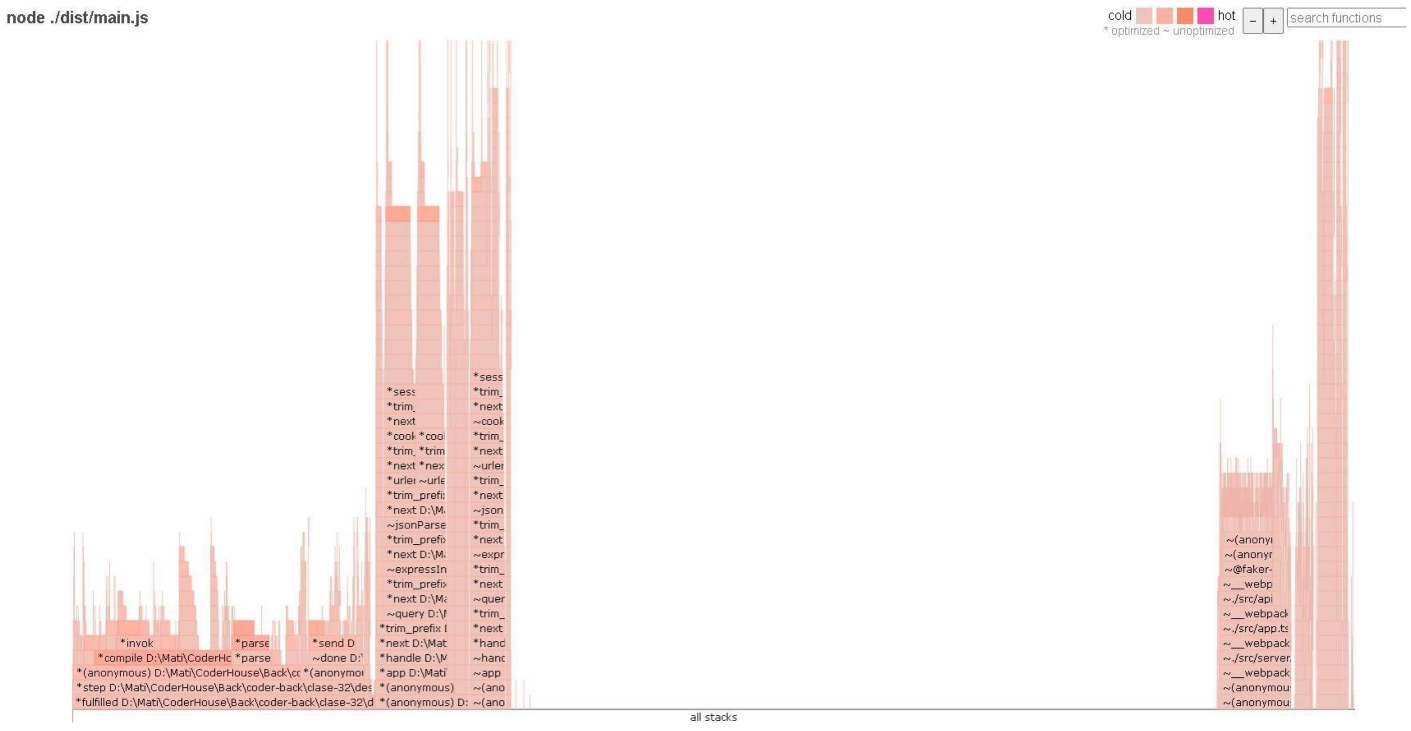


```
});
0.2 ms    app.get("/info", (0, compression_1.default)(), (req, res) => {
2.5 ms        const { argv, execPath, platform, version, pid, memoryUsage, cwd } = process;
34.2 ms       console.log(`Info en ${process.pid} y ${process.argv}`);
5.8 ms        const { rss } = memoryUsage();
              /*console.log(
                argv,
                execPath,
                platform,
                version,
                pid,
                rss,
                cwd(),
                os.cpus().length
              );*/
10.4 ms       res.render("info", {
                  layout: "info",
0.1 ms            argv,
0.1 ms            execPath,
0.1 ms            platform,
                  version,
                  pid,
0.1 ms            rss,
1.3 ms            currentDir: cwd(),
6.5 ms            cpus: os.cpus().length,
              });
```



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


┌─────────┬────────┬────────┬────────┬────────┬───────────┬──────────┬────────┐
│ Stat    │ 2.5%   │ 50%    │ 97.5%  │ 99%    │ Avg       │ Stdev    │ Max    │
├─────────┼────────┼────────┼────────┼────────┼───────────┼──────────┼────────┤
│ Latency │ 160 ms │ 183 ms │ 263 ms │ 281 ms │ 191.69 ms │ 28.79 ms │ 394 ms │
└─────────┴────────┴────────┴────────┴────────┴───────────┴──────────┴────────┘
┌───────────┬────────┬────────┬────────┬────────┬────────┬─────────┬────────┐
│ Stat      │ 1%     │ 2.5%   │ 50%    │ 97.5%  │ Avg    │ Stdev   │ Min    │
├───────────┼────────┼────────┼────────┼────────┼────────┼─────────┼────────┤
│ Req/Sec   │ 150    │ 150    │ 261    │ 287    │ 259.3  │ 27.47   │ 150    │
├───────────┼────────┼────────┼────────┼────────┼────────┼─────────┼────────┤
│ Bytes/Sec │ 324 kB │ 324 kB │ 563 kB │ 620 kB │ 559 kB │ 59.2 kB │ 324 kB │
└───────────┴────────┴────────┴────────┴────────┴────────┴─────────┴────────┘

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

5k requests in 20.05s, 11.2 MB read
```

Diagrama de flama con 0x, emulando la carga con Autocannon:


node ./dist/main.js
cold
hot
* optimized ~ unoptimized
search functions
*invok
*compile D:\Mati\CoderHc
*(anonymous) D:\Mati\CoderHouse\Back\cc
*step D:\Mati\CoderHouse\Back\coder-back\clase-32\des
*fulfilled D:\Mati\CoderHouse\Back\coder-back\clase-32\d
*parse
*parse
*send D
~done D:\
*(anonymoı
*sess
*trim_
*next
*cook
*trim_
*next
*urlei
*trim_prefi
*next D:\M
~jsonParse
*trim_prefix
*next D:\M
~expressIn
*trim_prefix
*next D:\Ma
~query D:\M
*trim_prefix I
*next D:\Mat
*handle D:\M
*app D:\Mati
*(anonymous)
*(anonymous) D:
~(anonyı
~(anonyr
~@faker-
~__webp
~./src/api
~__webpack
~./src/app.ts
~__webpack
~./src/server
~__webpack
~(anonymous
~(anonymous
all stacks