**1.** Ruta `/info` con y sin compresión

**Con compresión**

Tamaño: 1.5 kB


**Sin compresión**

Tamaño: 3.4 kB

**2.** Ruta `/info` con y sin console.log

- Test con -prof de node
- Carga con Artillery

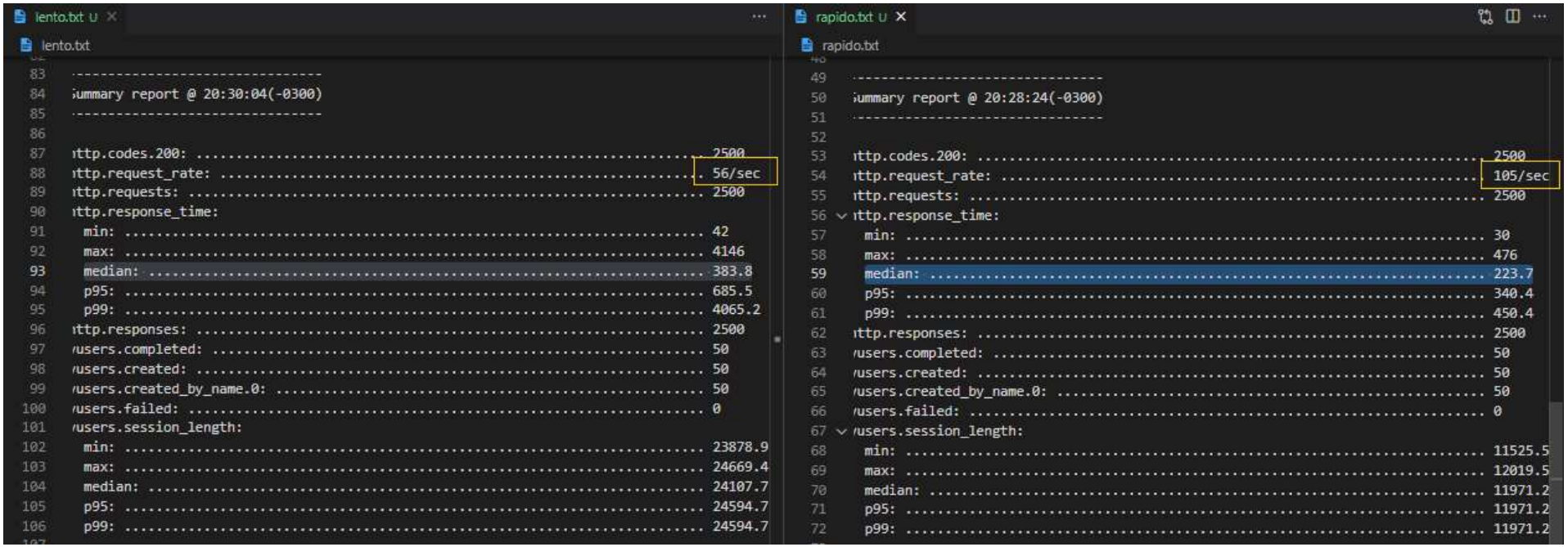
**-pof summary Con console.log (lento)**

```
[Summary]:
  ticks  total  nonlib   name
   160    3.5%   98.8%  JavaScript
     0    0.0%    0.0%  C++
    84    1.8%   51.9%  GC
  4440   96.5%          Shared libraries
     2    0.0%          Unaccounted
```

**-prof summary Sin console.log (rápido)**

```
[Summary]:
  ticks  total  nonlib   name
   148    1.0%   96.7%  JavaScript
     0    0.0%    0.0%  C++
   143    0.9%   93.5%  GC
 14976   99.0%          Shared libraries
     5    0.0%          Unaccounted
```

## Artillery. Lento vs Rápido



lento.txt U

```
Summary report @ 20:30:04(-0300)

http.codes.200: ..... 2500
http.request_rate: ..... 56/sec
http.requests: ..... 2500
http.response_time:
  min: ..... 42
  max: ..... 4146
  median: ..... 383.8
  p95: ..... 685.5
  p99: ..... 4065.2
http.responses: ..... 2500
vusers.completed: ..... 50
vusers.created: ..... 50
vusers.created_by_name.0: ..... 50
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 23878.9
  max: ..... 24669.4
  median: ..... 24107.7
  p95: ..... 24594.7
  p99: ..... 24594.7
```

rapido.txt U

```
Summary report @ 20:28:24(-0300)

http.codes.200: ..... 2500
http.request_rate: ..... 105/sec
http.requests: ..... 2500
http.response_time:
  min: ..... 30
  max: ..... 476
  median: ..... 223.7
  p95: ..... 340.4
  p99: ..... 450.4
http.responses: ..... 2500
vusers.completed: ..... 50
vusers.created: ..... 50
vusers.created_by_name.0: ..... 50
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 11525.5
  max: ..... 12019.5
  median: ..... 11971.2
  p95: ..... 11971.2
  p99: ..... 11971.2
```

**3.** Autocannon

**Rápido**

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

| Stat | 2.5% | 50% | 97.5% | 99% | Avg | Stdev | Max |
|---|---|---|---|---|---|---|---|
| **Latency** | 550 ms | 1006 ms | 2333 ms | 2509 ms | 1054.45 ms | 417.43 ms | 2515 ms |

| Stat | 1% | 2.5% | 50% | 97.5% | Avg | Stdev | Min |
|---|---|---|---|---|---|---|---|
| **Req/Sec** | 0 | 0 | 99 | 174 | 93.85 | 41.8 | 29 |
| **Bytes/Sec** | 0 B | 0 B | 342 kB | 601 kB | 324 kB | 144 kB | 100 kB |

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

2k requests in 20.24s, 6.48 MB read
```

**Lento**

```
Running 20s test @ http://localhost:8080/info2
100 connections
```

| Stat | 2.5% | 50% | 97.5% | 99% | Avg | Stdev | Max |
|---|---|---|---|---|---|---|---|
| **Latency** | 695 ms | 925 ms | 1794 ms | 1869 ms | 1068 ms | 344.92 ms | 1947 ms |

| Stat | 1% | 2.5% | 50% | 97.5% | Avg | Stdev | Min |
|---|---|---|---|---|---|---|---|
| **Req/Sec** | 0 | 0 | 100 | 170 | 93.35 | 44.43 | 27 |
| **Bytes/Sec** | 0 B | 0 B | 345 kB | 587 kB | 322 kB | 153 kB | 93.3 kB |

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

2k requests in 20.37s, 6.45 MB read
```

**4.** –inspect

**Rápido**

```
19.6 ms  renderRouter.get("/info", (req, res) => {
106.3 ms     const info = getInfo();
         res.render("info", { info });
         });

         renderRouter.get("/info2", (req, res) => {
             const info = getInfo();
             console.log(info);
             res.render("info", { info });
         });

 2.1 ms  const getInfo = () => {
 2.6 ms      return {
16.0 ms          argumentos: process.argv.slice(2),
 1.7 ms          OS: process.platform,
 1.1 ms          nodeVersion: process.version,
 3.8 ms          RSS: process.memoryUsage().rss,
 3.5 ms          execPath: process.execPath,
 1.1 ms          processID: process.pid,
 1.2 ms          processPath: process.argv[1],
 8.0 ms          numCPUs: require("os").cpus().length,
 0.2 ms      };
         };
```

**Lento**

```
 3.7 ms  renderRouter.get("/info2", (req, res) => {
 3.1 ms      const info = getInfo();
85.6 ms      console.log(info);
113.8 ms     res.render("info", { info });
 1.9 ms  });

 0.6 ms  const getInfo = () => {
 1.3 ms      return {
19.9 ms          argumentos: process.argv.slice(2),
 0.8 ms          OS: process.platform,
 2.7 ms          nodeVersion: process.version,
 7.1 ms          RSS: process.memoryUsage().rss,
 2.3 ms          execPath: process.execPath,
 1.5 ms          processID: process.pid,
 1.5 ms          processPath: process.argv[1],
 2.7 ms          numCPUs: require("os").cpus().length,
         };
         };
```

**5.** Diagrama de flama con 0x, en la carpeta del proyecto: slow-0x y fast-0x

## Conclusiones

Artillery se ejecutó con 50 conexiones que ejecutaron 50 request, gracias a las 2500 peticiones, se puede observar

**LENTO VS RÁPIDO**

56 request/seg  VS 105 request/seg

383.8 ms VS 223.7 ms

Me llama la atención que con autocannon se realizaron 2000 peticiones (apenas 20% menos que con artillery) y la diferencia entre lento y rápido no es tan sustancial como se observa en artillery.

Por su parte, usando `–inspect` se puede ver al detalle las líneas de código que más tiempo conllevan.

`-pof` y el diagrama de flama, a mi, particularmente, no me aportaron información clara.