## node --inspect app.js

Performance del servidor y diferencias al incluir impresiones en consola.

## Profiler

Self Time		Total Time		Function
10266.7 ms		10266.7 ms		(idle)
184.1 ms	16.88 %	184.1 ms	16.88 %	▶ getCPUs
112.2 ms	10.29 %	112.2 ms	10.29 %	▶ writeUtf8String
42.4 ms	3.89 %	42.4 ms	3.89 %	(garbage collector)
32.5 ms	2.98 %	32.5 ms	2.98 %	(program)
25.2 ms	2.31 %	716.6 ms	65.69 %	▶ initialize
25.0 ms	2.29 %	25.0 ms	2.29 %	▶ writev
15.9 ms	1.46 %	470.3 ms	43.11 %	▶ compression
14.8 ms	1.36 %	24.6 ms	2.26 %	▶ writeHead
14.8 ms	1.36 %	7721.7 ms	707.86 %	▶ next
14.6 ms	1.34 %	25.4 ms	2.33 %	▶ nextTick
14.6 ms	1.34 %	814.0 ms	74.62 %	▶ session
14.2 ms	1.30 %	39.0 ms	3.58 %	►hash
13.0 ms	1.20 %	8621.6 ms	790.36 %	▶ handle

Figure 1. No console.log()

Self Time ▼		Total Time		Function
17103.2 ms		17103.2 ms		(idle)
1307.9 ms	36.26 %	1307.9 ms	36.26 %	▶ writeUtf8String
912.6 ms	25.30 %	2231.4 ms	61.87 %	▶ consoleCall
216.7 ms	6.01 %	216.7 ms	6.01 %	▶ getCPUs
63.2 ms	1.75 %	63.2 ms	1.75 %	(garbage collector)
38.0 ms	1.05 %	38.0 ms	1.05 %	▶ writev
36.9 ms	1.02 %	36.9 ms	1.02 %	(program)
27.3 ms	0.76 %	3130.1 ms	86.78 %	▶ initialize
21.0 ms	0.58 %	38.0 ms	1.05 %	▶ nextTick
18.0 ms	0.50 %	44.9 ms	1.25 %	▶ hash
16.9 ms	0.47 %	2856.5 ms	79.20 %	▶ compression
16.1 ms	0.45 %	16.1 ms	0.45 %	▶ Hash
15.5 ms	0.43 %	3240.3 ms	89.84 %	▶ session
15.2 ms	0.42 %	34270.8 ms	950.18 %	▶ next
15.1 ms	0.42 %	39940.6 ms	1107.38 %	▶ handle
14.8 ms	0.41 %	2834.9 ms	78.60 %	▶ (anonymous)
14.3 ms	0.40 %	24.5 ms	0.68 %	▶ writeHead

Figure 2. Console.log(data)

Se visualiza claramente como afecta la inclusión del *console.log(data)* en la ruta /info. En la segunda figura se ve un aumentco drástico en los tiempos de ejecución de las funciones consoleCall() y writeUtf8String().

## File ./routers/info.js

```
/* GET Info. */
      router.get('/info', compression(), function (_, res, next) {
          try {
0.1 ms
              const data = {
                  argv: config.argv,
0.1 ms
0.2 ms
                  directory: process.cwd(),
                  processID: process.pid,
                  nodeVersion: process.version,
                  processTitle: process.title,
1.0 ms
0.1 ms
                  memoryUsage: process.memoryUsage().rss,
                  so: process.platform,
0.1 ms
1.1 ms
                  CPUs: os.cpus().length
               // console.log(data);
6.0 ms
              res.status(201).json(data);
          catch (e) {
              logError(req, res, next);
              console.log(e.message);
      });
```

 $T_{TOTAL} = 8.52$ ms

Figura 1. No console.log(data)

```
/* GET Info. */
      router.get('/info', compression(), function (_, res, next)
          try {
0.3 ms
              const data = {
                  argv: config.argv,
0.4 ms
                  directory: process.cwd(),
0.2 ms
                  processID: process.pid,
                  nodeVersion: process.version,
                  processTitle: process.title,
                  memoryUsage: process.memoryUsage().rss,
0.1 ms
0.1 ms
                  so: process.platform,
                  CPUs: os.cpus().length
1.0 ms
              console.log(data);
8.5 ms
              res.status(201).json(data);
          catch (e) {
              logError(req, res, next);
              console.log(e.message);
          }
      });
```

 $T_{TOTAL} = 14.6$ ms

Figura 2. Console.log(data)

Se determina que el tiempo de imprimir en consola, mediante *console.log(data)* es de 2.3ms (Figura 2). Esto representa el 15.75% del tiempo total de ejecución del bloque **try{}**, el cual fue  $T_{TOTAL}$  = 14.6ms.

Se hace presente también la variabilidad en los tiempos de ejecución en las instrucciones. Donde en el segundo caso, las instrucciones tomaron más tiempo para ejecutarse que en el primer caso, a pesar de no existir modificaciones salvo por incluir o no la sentencia *console.log(data)*.

## Autocannon

Además de las siguientes gráficas, se encuentra a disposición en la carpeta performance/autocannon las gráficas generadas por la herramienta.

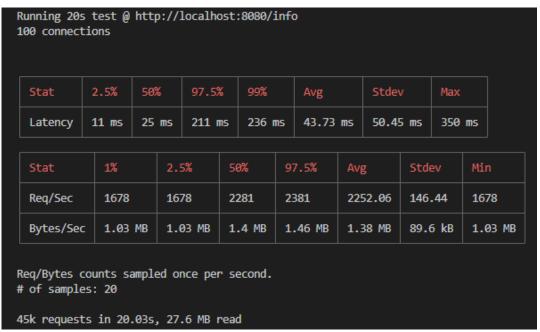


Figure 1. No console.log()

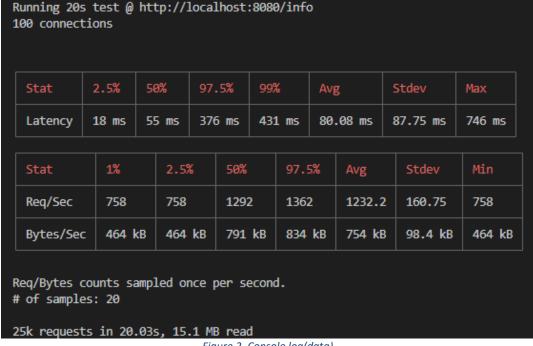


Figure 2. Console.log(data)