Desafío 14:

Análisis de Performance:

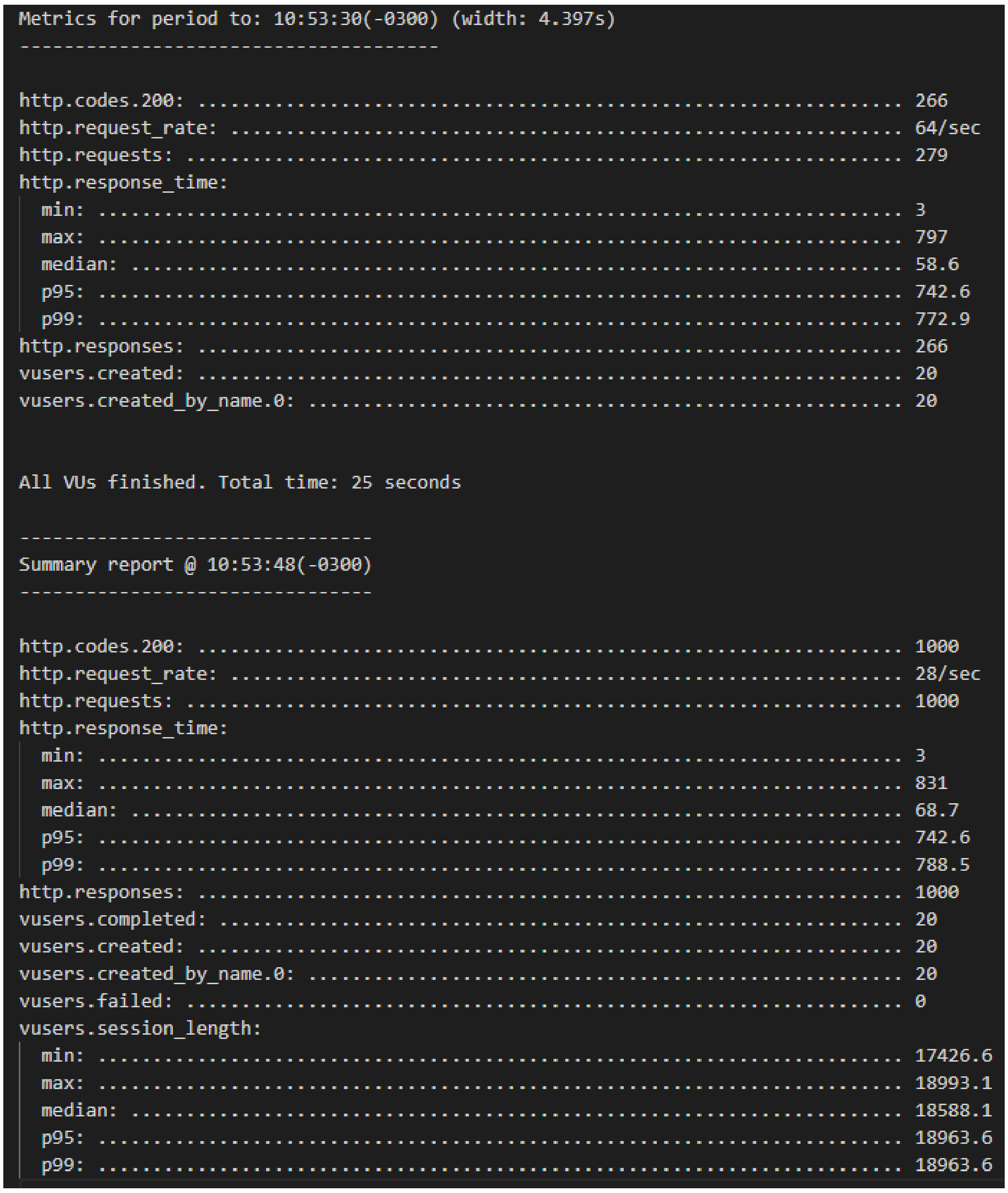
Punto 1:

Analisis con modo –Prof:

Prueba NO BLOQUEANTE con ruta /info

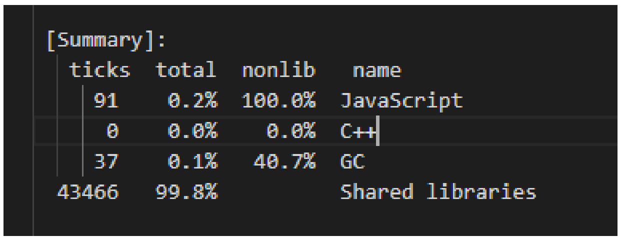
$ node - -prof src/server.js

$ artillery quick --count 20 -n 50 "http://localhost:8080/info" > result_nobloq.txt



```
Metrics for period to: 10:53:30(-0300) (width: 4.397s)
--------------------------------------

http.codes.200: ............................................................... 266
http.request_rate: ............................................................ 64/sec
http.requests: ................................................................ 279
http.response_time:
  min: ........................................................................ 3
  max: ........................................................................ 797
  median: ..................................................................... 58.6
  p95: ........................................................................ 742.6
  p99: ........................................................................ 772.9
http.responses: ............................................................... 266
vusers.created: ............................................................... 20
vusers.created_by_name.0: ..................................................... 20


All VUs finished. Total time: 25 seconds

--------------------------------
Summary report @ 10:53:48(-0300)
--------------------------------

http.codes.200: ............................................................... 1000
http.request_rate: ............................................................ 28/sec
http.requests: ................................................................ 1000
http.response_time:
  min: ........................................................................ 3
  max: ........................................................................ 831
  median: ..................................................................... 68.7
  p95: ........................................................................ 742.6
  p99: ........................................................................ 788.5
http.responses: ............................................................... 1000
vusers.completed: ............................................................. 20
vusers.created: ............................................................... 20
vusers.created_by_name.0: ..................................................... 20
vusers.failed: ................................................................ 0
vusers.session_length:
  min: ........................................................................ 17426.6
  max: ........................................................................ 18993.1
  median: ..................................................................... 18588.1
  p95: ........................................................................ 18963.6
  p99: ........................................................................ 18963.6
```

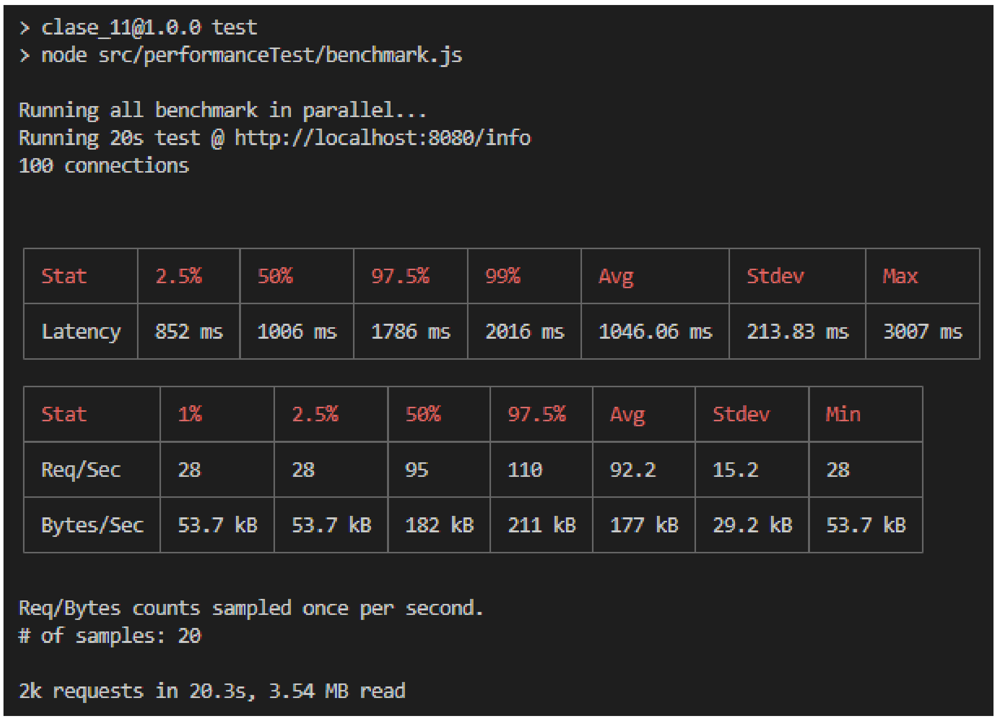
$ node.exe --prof-process nobloq-v8.log > resultProf_nobloq-v8.txt



```
 [Summary]:
   ticks  total  nonlib   name
     91    0.2%  100.0%  JavaScript
      0    0.0%    0.0%  C++
     37    0.1%   40.7%  GC
  43466   99.8%          Shared libraries
```

Autocannon

$ node src/performanceTest/bechmark.js



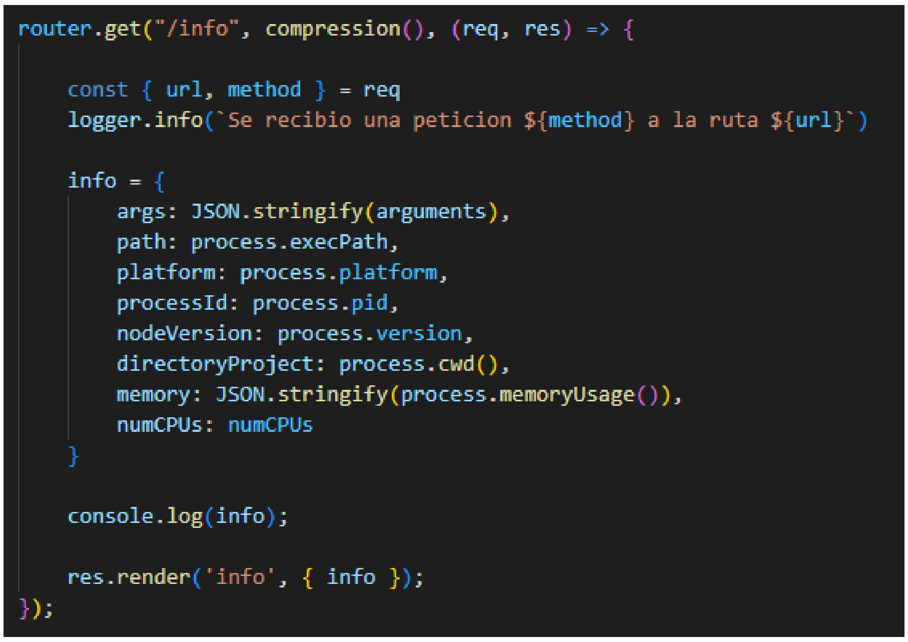
```
> clase_11@1.0.0 test
> node src/performanceTest/benchmark.js

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

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	852 ms	1006 ms	1786 ms	2016 ms	1046.06 ms	213.83 ms	3007 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	28	28	95	110	92.2	15.2	28
Bytes/Sec	53.7 kB	53.7 kB	182 kB	211 kB	177 kB	29.2 kB	53.7 kB

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

2k requests in 20.3s, 3.54 MB read
```

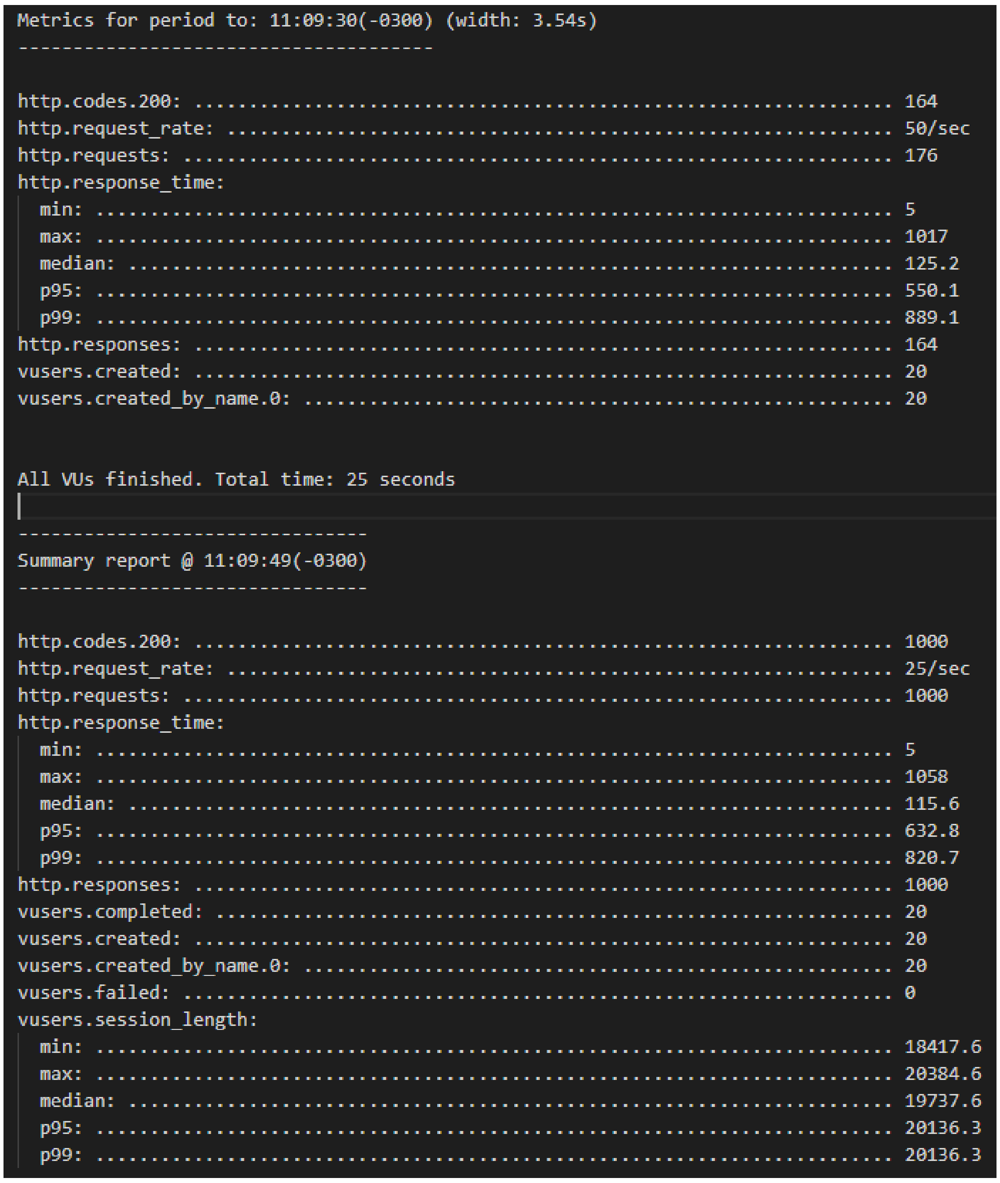
Prueba BLOQUEANTE con ruta /info   (Agregando console.log(info) en código de ruta)



```
router.get("/info", compression(), (req, res) => {

    const { url, method } = req
    logger.info(`Se recibio una peticion ${method} a la ruta ${url}`)

    info = {
        args: JSON.stringify(arguments),
        path: process.execPath,
        platform: process.platform,
        processId: process.pid,
        nodeVersion: process.version,
        directoryProject: process.cwd(),
        memory: JSON.stringify(process.memoryUsage()),
        numCPUs: numCPUs
    }

    console.log(info);

    res.render('info', { info });
});
```

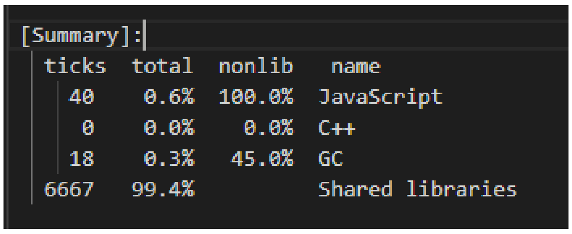
$ node - -prof src/server.js

$ artillery quick --count 20 -n 50 "http://localhost:8080/info" > result_bloq.txt



```
Metrics for period to: 11:09:30(-0300) (width: 3.54s)
--------------------------------------

http.codes.200: .............................................................. 164
http.request_rate: ........................................................... 50/sec
http.requests: ............................................................... 176
http.response_time:
  min: ....................................................................... 5
  max: ....................................................................... 1017
  median: .................................................................... 125.2
  p95: ....................................................................... 550.1
  p99: ....................................................................... 889.1
http.responses: .............................................................. 164
vusers.created: .............................................................. 20
vusers.created_by_name.0: .................................................... 20


All VUs finished. Total time: 25 seconds

--------------------------------
Summary report @ 11:09:49(-0300)
--------------------------------

http.codes.200: .............................................................. 1000
http.request_rate: ........................................................... 25/sec
http.requests: ............................................................... 1000
http.response_time:
  min: ....................................................................... 5
  max: ....................................................................... 1058
  median: .................................................................... 115.6
  p95: ....................................................................... 632.8
  p99: ....................................................................... 820.7
http.responses: .............................................................. 1000
vusers.completed: ............................................................ 20
vusers.created: .............................................................. 20
vusers.created_by_name.0: .................................................... 20
vusers.failed: ............................................................... 0
vusers.session_length:
  min: ....................................................................... 18417.6
  max: ....................................................................... 20384.6
  median: .................................................................... 19737.6
  p95: ....................................................................... 20136.3
  p99: ....................................................................... 20136.3
```

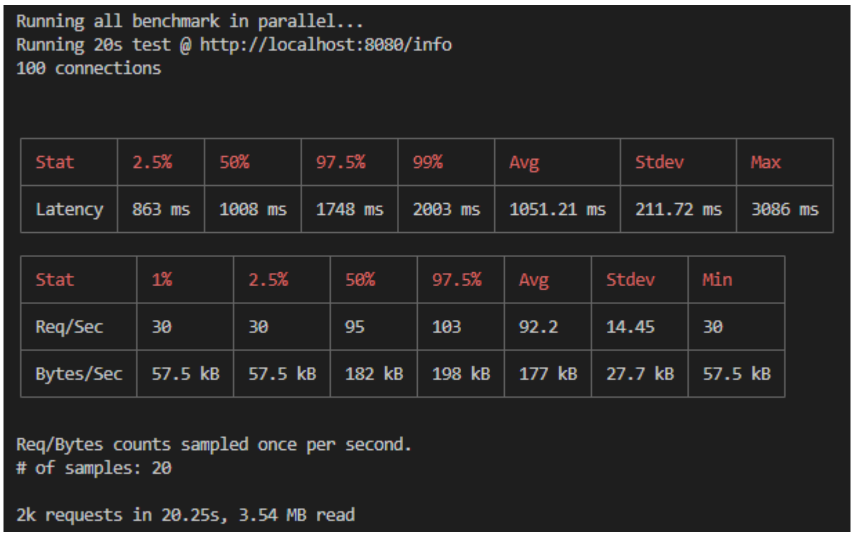
$ node.exe --prof-process bloq-v8.log > resultProf_bloq-v8.txt



```
[Summary]:
 ticks  total  nonlib   name
    40    0.6%  100.0%  JavaScript
     0    0.0%    0.0%  C++
    18    0.3%   45.0%  GC
  6667   99.4%          Shared libraries
```

Autocannon

$ node src/performanceTest/bechmark.js



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


┌─────────┬────────┬─────────┬─────────┬─────────┬────────────┬───────────┬─────────┐
│ Stat    │ 2.5%   │ 50%     │ 97.5%   │ 99%     │ Avg        │ Stdev     │ Max     │
├─────────┼────────┼─────────┼─────────┼─────────┼────────────┼───────────┼─────────┤
│ Latency │ 863 ms │ 1008 ms │ 1748 ms │ 2003 ms │ 1051.21 ms │ 211.72 ms │ 3086 ms │
└─────────┴────────┴─────────┴─────────┴─────────┴────────────┴───────────┴─────────┘
┌───────────┬─────────┬─────────┬────────┬────────┬────────┬─────────┬─────────┐
│ Stat      │ 1%      │ 2.5%    │ 50%    │ 97.5%  │ Avg    │ Stdev   │ Min     │
├───────────┼─────────┼─────────┼────────┼────────┼────────┼─────────┼─────────┤
│ Req/Sec   │ 30      │ 30      │ 95     │ 103    │ 92.2   │ 14.45   │ 30      │
├───────────┼─────────┼─────────┼────────┼────────┼────────┼─────────┼─────────┤
│ Bytes/Sec │ 57.5 kB │ 57.5 kB │ 182 kB │ 198 kB │ 177 kB │ 27.7 kB │ 57.5 kB │
└───────────┴─────────┴─────────┴────────┴────────┴────────┴─────────┴─────────┘

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

2k requests in 20.25s, 3.54 MB read
```

Punto 2:

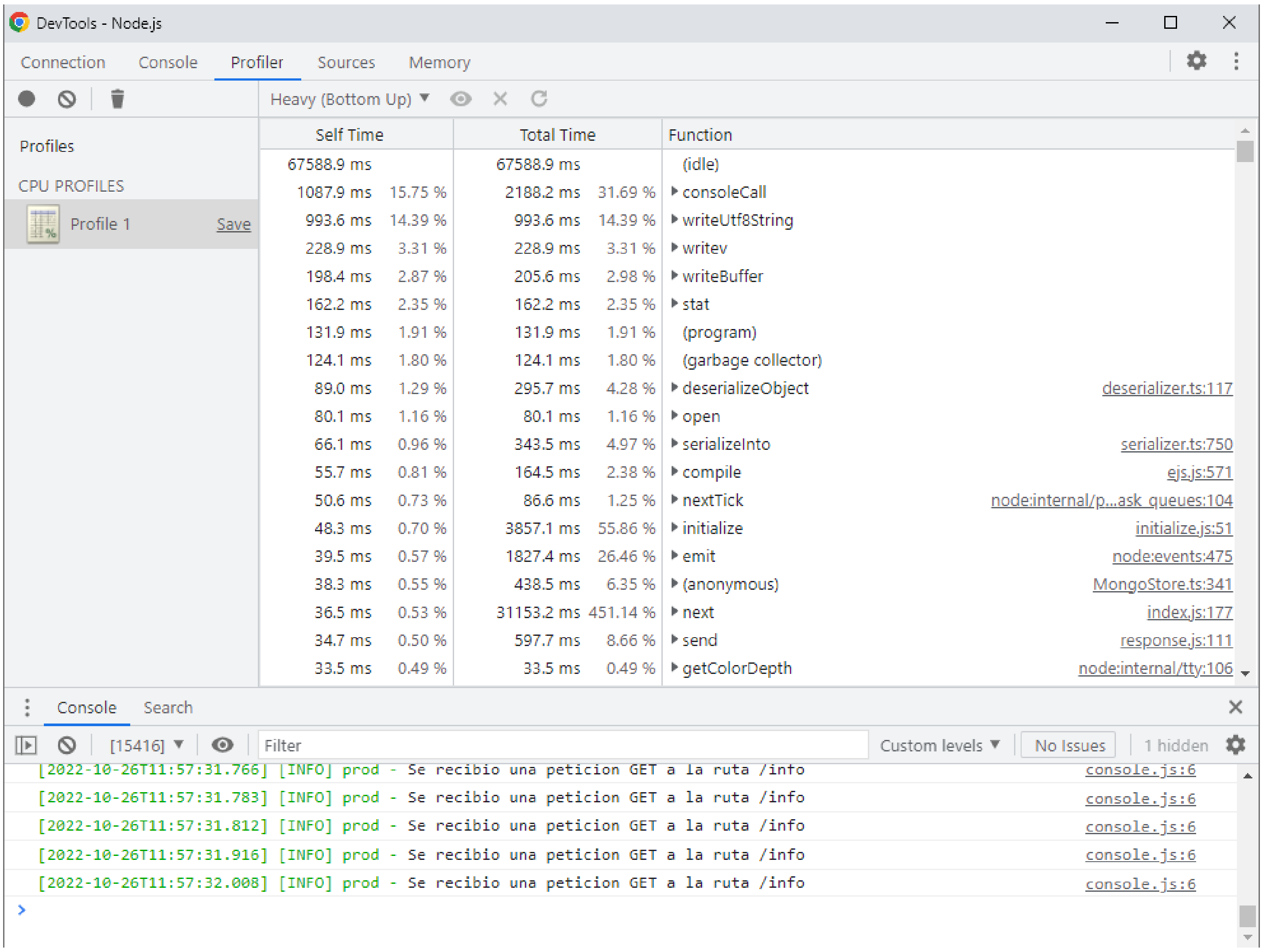
Analisis con modo Inspect

NO BLOQUEANTE:

$ node - -inspect src/server.js

$ artillery quick --count 20 -n 50 “http://localhost:8080/info”

chrome://inspect



Self Time		Total Time		Function	
67588.9 ms		67588.9 ms		(idle)	
1087.9 ms	15.75 %	2188.2 ms	31.69 %	consoleCall	
993.6 ms	14.39 %	993.6 ms	14.39 %	writeUtf8String	
228.9 ms	3.31 %	228.9 ms	3.31 %	writev	
198.4 ms	2.87 %	205.6 ms	2.98 %	writeBuffer	
162.2 ms	2.35 %	162.2 ms	2.35 %	stat	
131.9 ms	1.91 %	131.9 ms	1.91 %	(program)	
124.1 ms	1.80 %	124.1 ms	1.80 %	(garbage collector)	
89.0 ms	1.29 %	295.7 ms	4.28 %	deserializeObject	deserializer.ts:117
80.1 ms	1.16 %	80.1 ms	1.16 %	open	
66.1 ms	0.96 %	343.5 ms	4.97 %	serializeInto	serializer.ts:750
55.7 ms	0.81 %	164.5 ms	2.38 %	compile	ejs.js:571
50.6 ms	0.73 %	86.6 ms	1.25 %	nextTick	node:internal/p...ask_queues:104
48.3 ms	0.70 %	3857.1 ms	55.86 %	initialize	initialize.js:51
39.5 ms	0.57 %	1827.4 ms	26.46 %	emit	node:events:475
38.3 ms	0.55 %	438.5 ms	6.35 %	(anonymous)	MongoStore.ts:341
36.5 ms	0.53 %	31153.2 ms	451.14 %	next	index.js:177
34.7 ms	0.50 %	597.7 ms	8.66 %	send	response.js:111
33.5 ms	0.49 %	33.5 ms	0.49 %	getColorDepth	node:internal/tty:106

```
[2022-10-26T11:57:31.766] [INFO] prod - Se recibio una peticion GET a la ruta /info    console.js:6
[2022-10-26T11:57:31.783] [INFO] prod - Se recibio una peticion GET a la ruta /info    console.js:6
[2022-10-26T11:57:31.812] [INFO] prod - Se recibio una peticion GET a la ruta /info    console.js:6
[2022-10-26T11:57:31.916] [INFO] prod - Se recibio una peticion GET a la ruta /info    console.js:6
[2022-10-26T11:57:32.008] [INFO] prod - Se recibio una peticion GET a la ruta /info    console.js:6
>
```

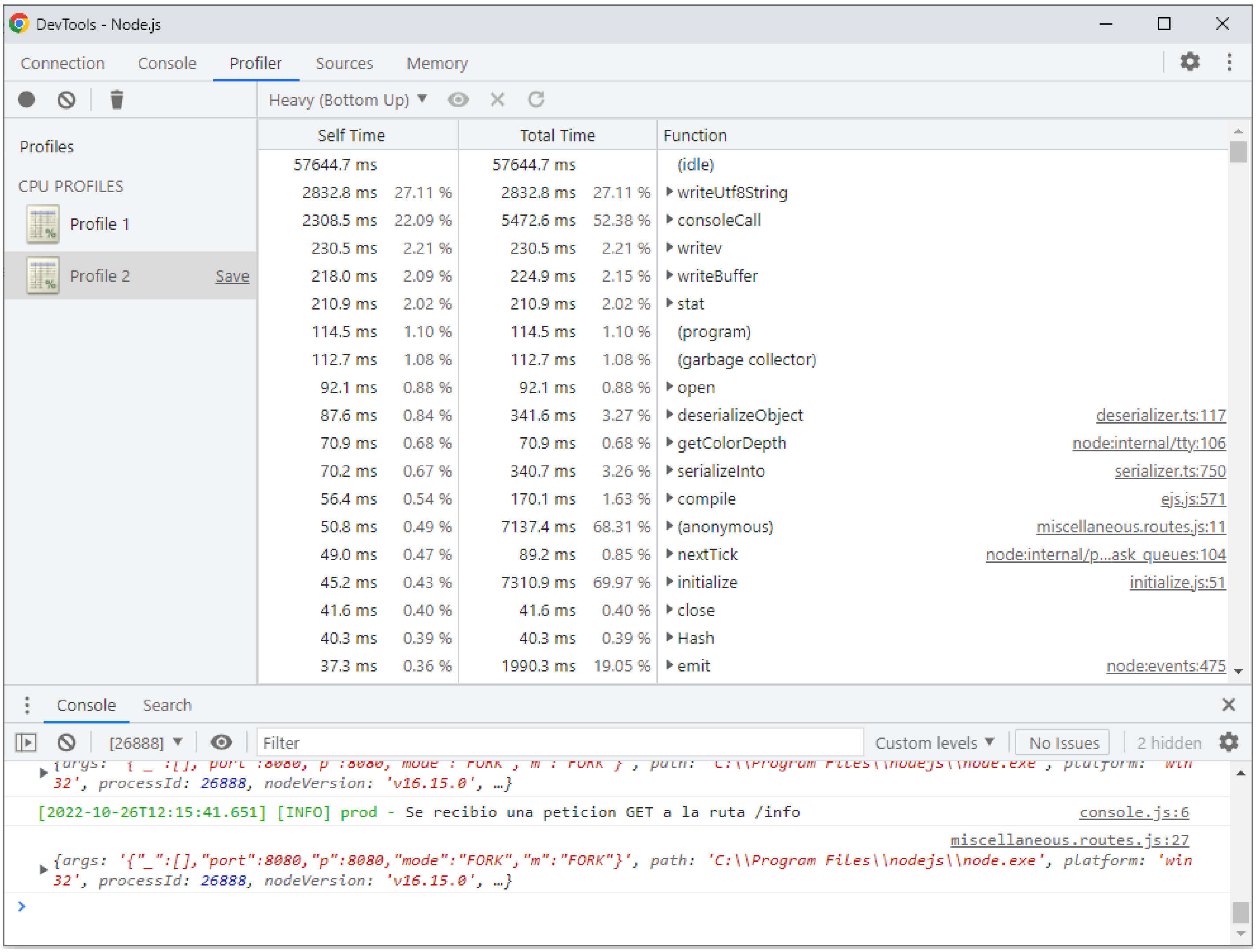
Análisis con modo Inspect

BLOQUEANTE:

$ node - -inspect src/server.js

$ artillery quick --count 20 -n 50 “http://localhost:8080/info”

chrome://inspect



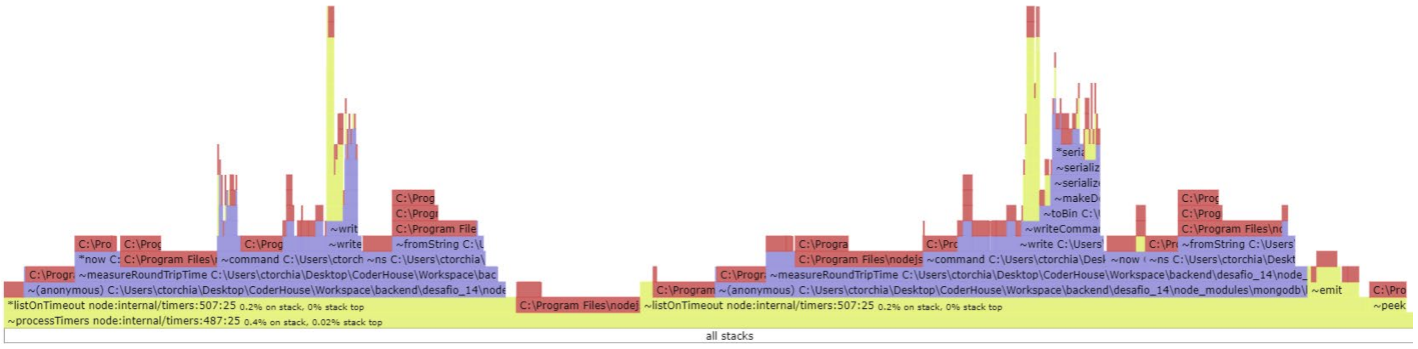
```
DevTools - Node.js
Connection   Console   Profiler   Sources   Memory

Heavy (Bottom Up)

Profiles
CPU PROFILES
  Profile 1
  Profile 2   Save

   Self Time            Total Time           Function
 57644.7 ms           57644.7 ms             (idle)
  2832.8 ms  27.11 %   2832.8 ms  27.11 %  ▸ writeUtf8String
  2308.5 ms  22.09 %   5472.6 ms  52.38 %  ▸ consoleCall
   230.5 ms   2.21 %    230.5 ms   2.21 %  ▸ writev
   218.0 ms   2.09 %    224.9 ms   2.15 %  ▸ writeBuffer
   210.9 ms   2.02 %    210.9 ms   2.02 %  ▸ stat
   114.5 ms   1.10 %    114.5 ms   1.10 %    (program)
   112.7 ms   1.08 %    112.7 ms   1.08 %    (garbage collector)
    92.1 ms   0.88 %     92.1 ms   0.88 %  ▸ open
    87.6 ms   0.84 %    341.6 ms   3.27 %  ▸ deserializeObject          deserializer.ts:117
    70.9 ms   0.68 %     70.9 ms   0.68 %  ▸ getColorDepth              node:internal/tty:106
    70.2 ms   0.67 %    340.7 ms   3.26 %  ▸ serializeInto              serializer.ts:750
    56.4 ms   0.54 %    170.1 ms   1.63 %  ▸ compile                    ejs.js:571
    50.8 ms   0.49 %   7137.4 ms  68.31 %  ▸ (anonymous)                miscellaneous.routes.js:11
    49.0 ms   0.47 %     89.2 ms   0.85 %  ▸ nextTick                   node:internal/p...ask_queues:104
    45.2 ms   0.43 %   7310.9 ms  69.97 %  ▸ initialize                 initialize.js:51
    41.6 ms   0.40 %     41.6 ms   0.40 %  ▸ close
    40.3 ms   0.39 %     40.3 ms   0.39 %  ▸ Hash
    37.3 ms   0.36 %   1990.3 ms  19.05 %  ▸ emit                       node:events:475

Console   Search
[26888]   Filter   Custom levels   No Issues   2 hidden

▸ {args: '{"_":[],"port":8080,"p":8080,"mode":"FORK","m":"FORK"}', path: 'C:\\Program Files\\nodejs\\node.exe', platform: 'win
  32', processId: 26888, nodeVersion: 'v16.15.0', …}
[2022-10-26T12:15:41.651] [INFO] prod - Se recibio una peticion GET a la ruta /info                              console.js:6
                                                                                               miscellaneous.routes.js:27
▸ {args: '{"_":[],"port":8080,"p":8080,"mode":"FORK","m":"FORK"}', path: 'C:\\Program Files\\nodejs\\node.exe', platform: 'win
  32', processId: 26888, nodeVersion: 'v16.15.0', …}
>
```

Punto 3:

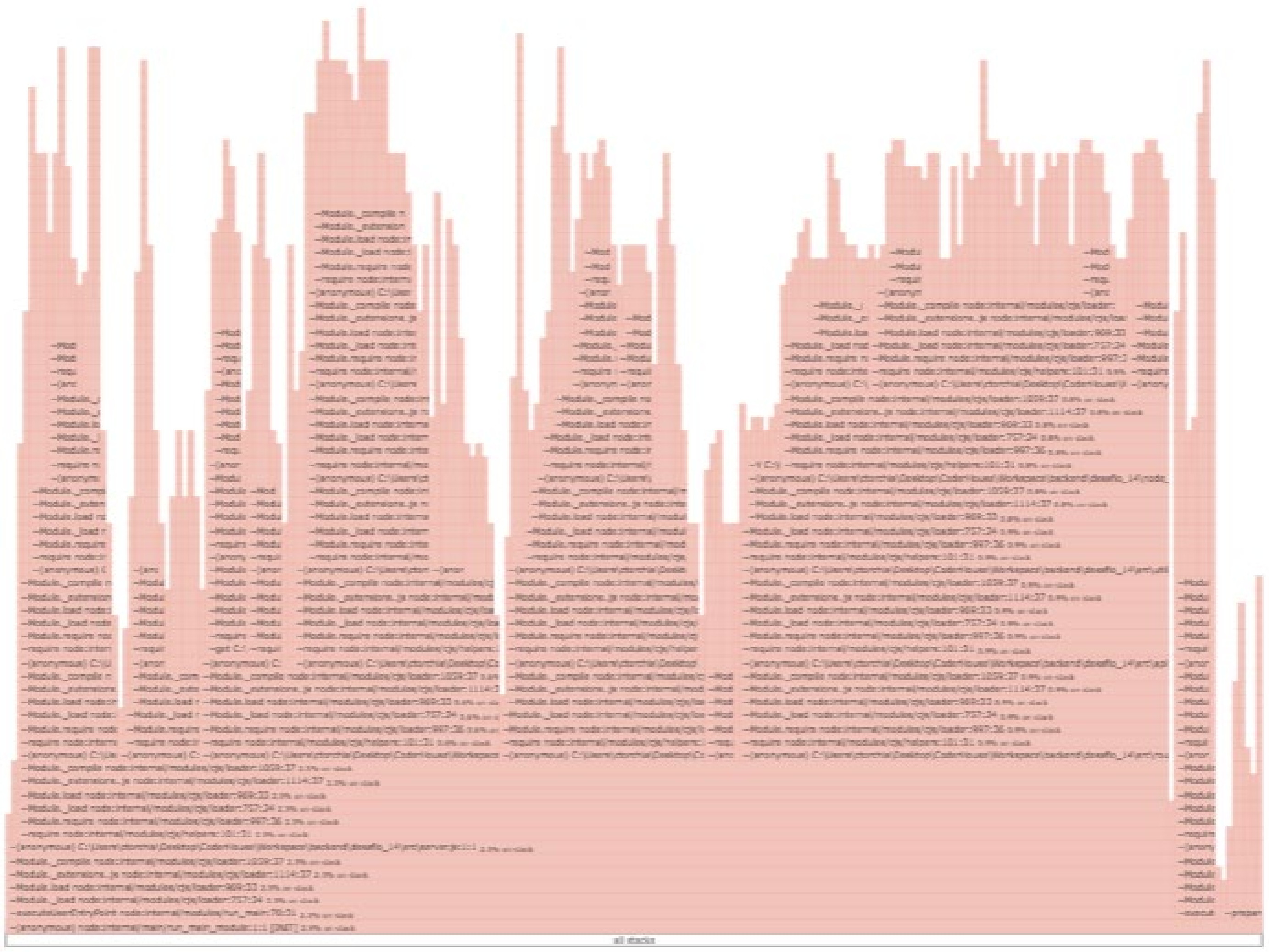
Analisis con diagrama de Flama

NO BLOQUEANTE:

```
$ 0x src/server.js
```



BLOQUEANTE:



Conclusión final:

Las ejecuciones de los procesos del servidor se realizan mucho mas rápido cuando no se registra código bloqueante en el desarrollo del mismo.

En esta prueba se agrega simplemente un console.log y ya se puede ver la gran diferencia en el rendimiento.