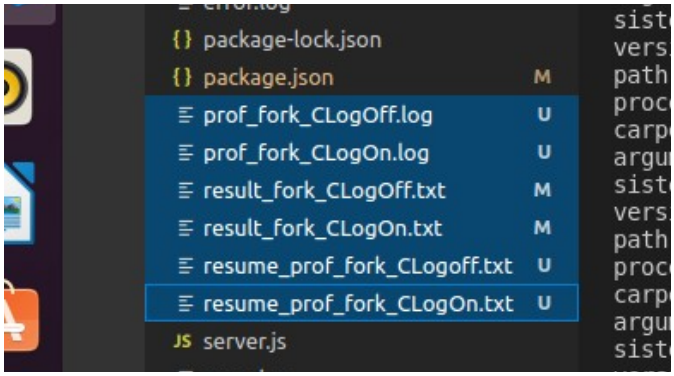
Informe de análisis de Performance:

1) Corriendo el servidor con los script : dev-F-CL (con console.log) y dev-F (sin console.log) y utilizando artillery

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "dev-F-CL": "node --prof server.js 8081 FORK C-LOG-ON",
  "dev-F": "node --prof server.js 8082 FORK C-LOG-OFF",
```

```
PROBLEMS   TERMINAL   OUTPUT   DEBUG CONSOLE                    2: bash
ruben@ruben-Lenovo-V330-15IKB:~/works/backend/Entrega 32$ artillery quick --count 50 -n 20 http://localhost:8081/info >
result_fork_CLogOn.txt
ruben@ruben-Lenovo-V330-15IKB:~/works/backend/Entrega 32$ artillery quick --count 50 -n 20 http://localhost:8082/info >
result_fork_CLogOff.txt
```

Resultados en los siguientes archivos:



Resumen_prof_fork_Clogoff.txt (sin console.log)

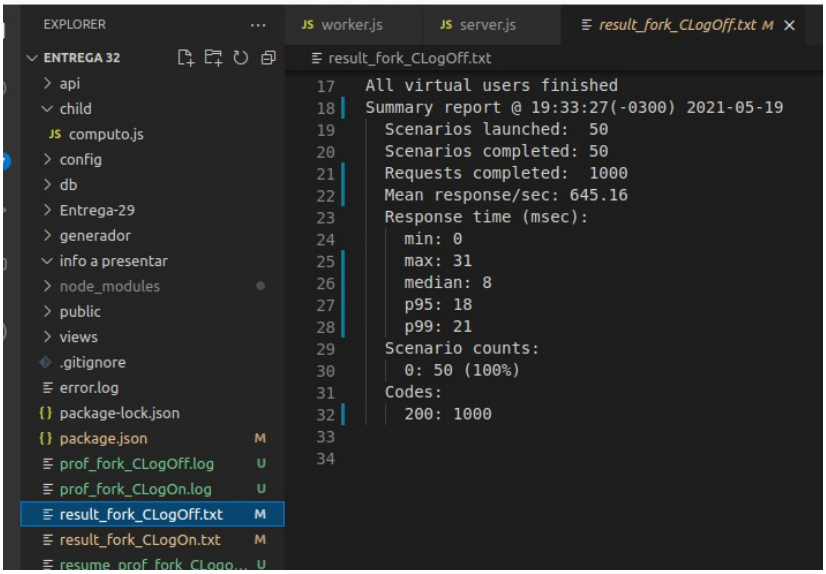
```
worker.js   server.js   resume_prof_fork_CLogoff.txt U
resume_prof_fork_CLogoff.txt
Statistical profiling result from prof_fork_CLogOff.log, (2109 ticks, 1 unaccounted, 0 excluded).

 [Shared libraries]:
   ticks  total  nonlib   name
   1308   62.0%          /usr/lib/x86_64-linux-gnu/libnode.so.72
```

Resumen_prof_fork_Clogoff.txt (con console.log)

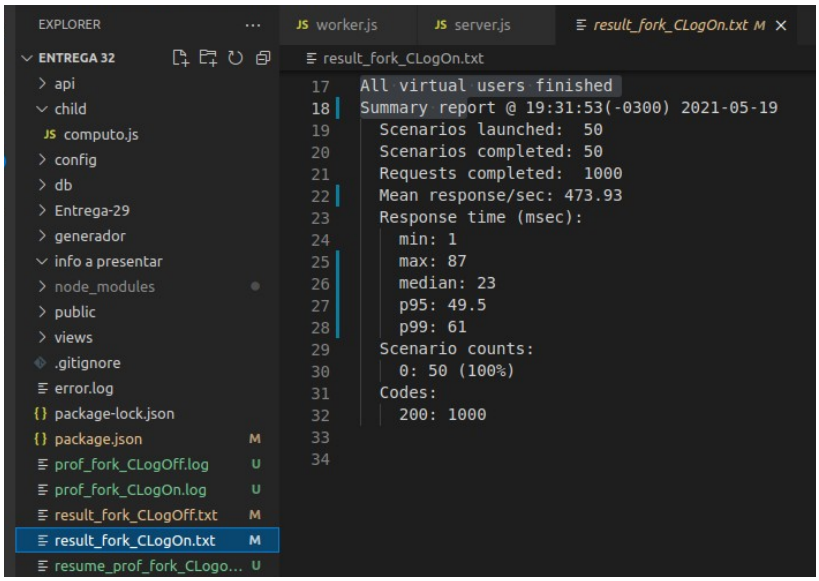
```
worker.js   server.js   resume_prof_fork_CLogOn.txt U
resume_prof_fork_CLogOn.txt
Statistical profiling result from prof_fork_CLogOn.log, (2740 ticks, 5 unaccounted, 0 excluded).

 [Shared libraries]:
   ticks  total  nonlib   name
   1794   65.5%          /usr/lib/x86_64-linux-gnu/libnode.so.72
```

Conclusión: utilizando los console.log consume mas ticks 2740 > 2109

Result_fork_ClogOff.txt (informe de artillery sin Console.log)



Result_fork_ClogOn.txt (informe de artillery con Console.log)



Conclusión: utilizando los console.log genera menos respuestas por segundos (473.93) que sin (645.16).

Utilizando -isnpect (con console.log)

Google Chrome — may 22 13:00

DevTools - Node.js

Connection | Profiler | Console | Sources | Memory

worker.js ×

```
126                 let nombre = req.user.displayName
127                 req.logout()
128                 res.render("logout", { nombre })
129             })
130             /* ------------------------------------------------------- */
131             /* ------------------------------------------------------- */
132             /* ------------------------------------------------------- */
133
134     0.7 ms  app.get('/info', (req,res) => {
135     0.2 ms      let argv=[]
136     5.6 ms      process.argv.forEach((val,index)=>{
137     0.2 ms          let newObj = {};
138    31.1 ms          newObj[index]=val
139     1.4 ms          argv.push(newObj)
140                 })
141
142     0.2 ms      let info=[{'argumento de entrada':argv},
143     0.3 ms      {'sistema operativo': process.platform},
144     0.5 ms      {'version de node': process.version},
145     2.0 ms      {'memoria utilizado MB': process.memoryUsage()},
146     0.8 ms      {'path de ejecucion': process.execPath},
147     0.5 ms      {'process id: ': process.pid},
148     1.0 ms      {'carpeta corriente':  process.cwd()}
149             ]
150
151    48.9 ms      console.log('argumento de entrada'+argv)
152    12.6 ms      console.log('sistema operativo'+ process.platform)
153    17.5 ms      console.log('version de node'+ process.version)
154    13.6 ms      console.log('path de ejecucion'+ process.execPath)
155    18.2 ms      console.log('process id: '+ process.pid)
156    19.0 ms      console.log('carpeta corriente'+ process.cwd())
157
158
159   223.2 ms     res.end(`${JSON.stringify(info)}`)
160             })
161
162
163             app.get('/randoms/:cant', (req,res) => {
164                 const computo = fork('./child/computo.js')
165
166                 let { cant } = req.params
```

```
^C
ruben@ruben-Lenovo-V330-15IKB:~/works/backend/Entrega 32$ node benchmark.js
Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8082/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	270 ms	327 ms	365 ms	391 ms	323.29 ms	29.58 ms	420 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	264	264	301	353	306.7	25.58	264
Bytes/Sec	160 kB	160 kB	183 kB	214 kB	186 kB	15.5 kB	160 kB

```
Req/Bytes counts sampled once per second.

6k requests in 20.05s, 3.72 MB read
ruben@ruben-Lenovo-V330-15IKB:~/works/backend/Entrega 32$
```

Ln 27,

Utilizando -isnpect (sin Console.log)

Google Chrome may 22 13:03

DevTools - Node.js

Connection Profiler Console Sources Memory

worker.js ×

```
131          /* ------------------------------------------------------------ */
132          /* ------------------------------------------------------------ */
133
134   5.0 ms app.get('/info', (req,res) => {
135   0.5 ms     let argv=[]
136  10.6 ms     process.argv.forEach((val,index)=>{
137   0.5 ms         let newObj = {};
138 150.1 ms         newObj[index]=val
139   3.8 ms         argv.push(newObj)
140              })
141
142   1.0 ms     let info=[{'argumento de entrada':argv},
143   1.6 ms     {'sistema operativo': process.platform},
144   1.1 ms     {'version de node': process.version},
145   2.7 ms     {'memoria utilizado MB': process.memoryUsage()},
146   1.8 ms     {'path de ejecucion': process.execPath},
147   0.8 ms     {'process id: ': process.pid},
148   1.9 ms     {'carpeta corriente':  process.cwd()}
149              ]
150
151              // console.log('argumento de entrada'+argv)
152   0.3 ms     // console.log('sistema operativo'+ process.platform)
153   1.1 ms     // console.log('version de node'+ process.version)
154              // console.log('path de ejecucion'+ process.execPath)
155              // console.log('process id: '+ process.pid)
156              // console.log('carpeta corriente'+ process.cwd())
157
158   0.2 ms
159 666.7 ms    res.end(`${JSON.stringify(info)}`)
160   1.9 ms }))
161
162
163          app.get('/randoms/:cant', (req,res) => {
```

```
6k requests in 20.05s, 3.72 MB read
ruben@ruben-Lenovo-V330-15IKB:~/works/backend/Entrega 32$ node benchmark.js
Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8082/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	28 ms	32 ms	57 ms	65 ms	33.97 ms	7.81 ms	143 ms

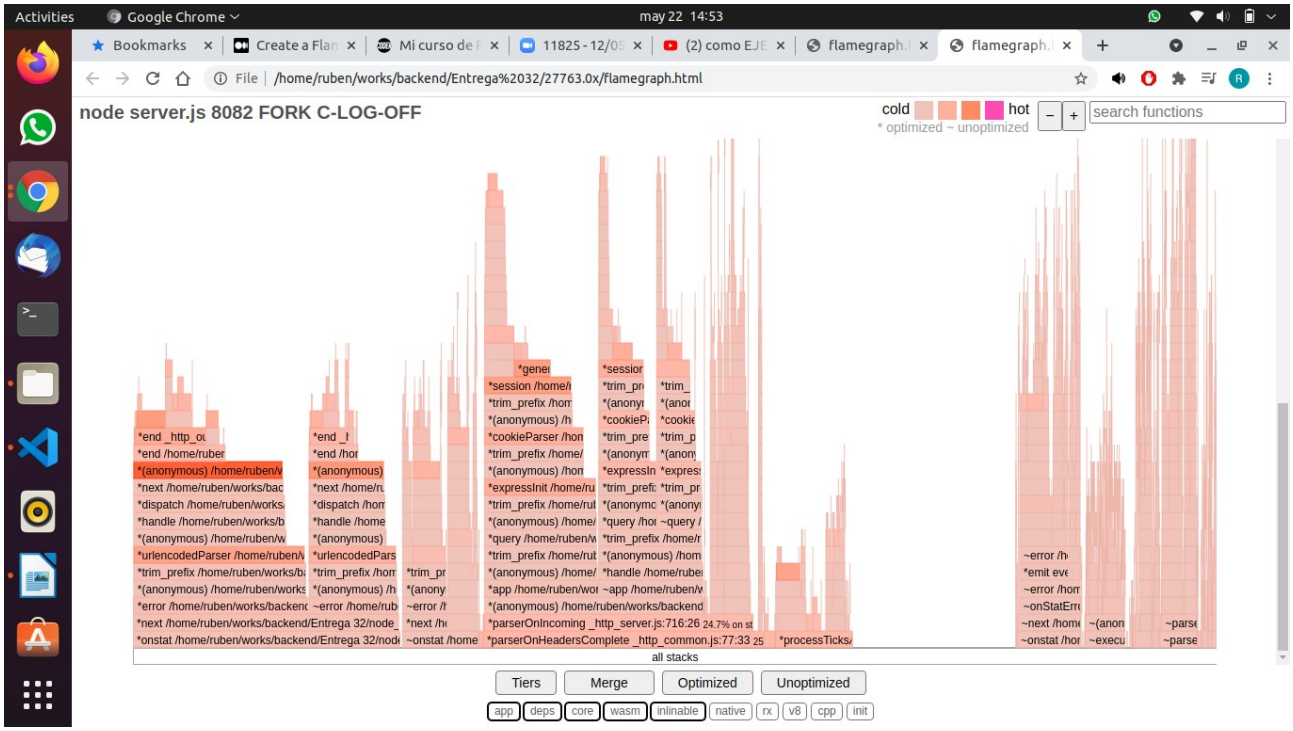
Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	1500	1500	3013	3177	2900.55	399.24	1500
Bytes/Sec	910 kB	910 kB	1.83 MB	1.93 MB	1.76 MB	243 kB	910 kB

```
Req/Bytes counts sampled once per second.

58k requests in 20.04s, 35.2 MB read
ruben@ruben-Lenovo-V330-15IKB:~/works/backend/Entrega 32$
```

Ln 154

Flame graph (sin consol.log) - se observa una zona blanca

(servidor desocupado)



Flame graph (con consol.log)