Desafío LOGGERS, GZIP y ANÁLISIS DE PERFORMANCE.

Aplicando GZIP en ruta “/info”:

Sin Gzip: 1.9 kB de transferencia.

Con Gzip: 1.2 kB de transferencia.

Performance del servidor en ruta “/info”:

- PRENDEMOS EL SV EN MODO PROF:

  -node --prof server.js

- EJECUTAMOS TEST DE CARGA CON ARTILLERY:

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

- DECODIFICAMOS EL ARCHIVO LOG QUE SE CREO:

-node --prof-process isolate-0000018CEDDA9680-4584-v8.log > profiler.txt

Resultado del test de carga con Artillery:

-Con console.log()

```
All VUs finished. Total time: 5 seconds

--------------------------------
Summary report @ 20:16:40(-0300)
--------------------------------

http.codes.200: ..................................................................... 1000
http.request_rate: .................................................................. 301/sec
http.requests: ...................................................................... 1000
http.response_time:
  min: .............................................................................. 16
  max: .............................................................................. 167
  median: ........................................................................... 127.8
  p95: .............................................................................. 156
  p99: .............................................................................. 162.4
http.responses: ..................................................................... 1000
vusers.completed: ................................................................... 50
vusers.created: ..................................................................... 50
vusers.created_by_name.0: ........................................................... 50
vusers.failed: ...................................................................... 0
vusers.session_length:
  min: .............................................................................. 2365.4
  max: .............................................................................. 2725.1
  median: ........................................................................... 2618.1
  p95: .............................................................................. 2725
  p99: .............................................................................. 2725
```

-Sin Console.log():

```
All VUs finished. Total time: 3 seconds

--------------------------------
Summary report @ 21:52:41(-0300)
--------------------------------

http.codes.200: ..................................................................... 1000
http.request_rate: .................................................................. 272/sec
http.requests: ...................................................................... 1000
http.response_time:
  min: .............................................................................. 5
  max: .............................................................................. 102
  median: ........................................................................... 63.4
  p95: .............................................................................. 89.1
  p99: .............................................................................. 100.5
http.responses: ..................................................................... 1000
vusers.completed: ................................................................... 50
vusers.created: ..................................................................... 50
vusers.created_by_name.0: ........................................................... 50
vusers.failed: ...................................................................... 0
vusers.session_length:
  min: .............................................................................. 1077.3
  max: .............................................................................. 1464.9
  median: ........................................................................... 1326.4
  p95: .............................................................................. 1465.9
  p99: .............................................................................. 1465.9
```

-Como podemos observar, el tiempo de ejecución es menor en la ruta la cual carece del console.log() con la información.

-En cuanto al resultado del archivo “profiler.txt”:

En general, la mayor parte del tiempo de la aplicación se dedica a la biblioteca compartida "ntdll.dll" (con 13923 ticks, lo que representa el 97,7% del tiempo total de la aplicación), lo que sugiere que es posible que haya un problema de rendimiento con el sistema operativo o la biblioteca en sí. Además, la sección de JavaScript muestra varias funciones con una pequeña fracción del tiempo total (con 12 ticks, lo que representa solo el 0,1% del tiempo total de la aplicación), lo que sugiere que no hay un cuello de botella claro en el código JavaScript.

Autocannon – prueba de rendimiento:

-autocannon -c 100 -d 20 http://localhost:8080/info

Los resultados son los siguientes:

-Con console.log():

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

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	188 ms	223 ms	297 ms	355 ms	227.12 ms	25.63 ms	389 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	300	300	442	480	439.2	39.85	300
Bytes/Sec	531 kB	531 kB	782 kB	849 kB	777 kB	70.5 kB	531 kB

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

9k requests in 20.15s, 15.5 MB read
```

-Sin console.log:

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

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	97 ms	113 ms	147 ms	164 ms	116.48 ms	14.17 ms	248 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	600	600	883	926	856	72.46	600
Bytes/Sec	1.06 MB	1.06 MB	1.56 MB	1.64 MB	1.51 MB	128 kB	1.06 MB

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

17k requests in 20.09s, 30.3 MB read
```

El perfilamiento del servidor con el modo inspector de node.js --inspect:

-node --inspect server.js

Ejecutando una carga con Autocannon:

- autocannon -c 100 -d 20 http://localhost:8080/info

<u>Tiempos del código:</u>

-Con console.log():

```
         //INFO UTILIZANDO EL OBJETO PROCESS
  0.8 ms app.get("/info", gzipMiddleware, (req, res) => {
  5.0 ms   logger.info(
  2.2 ms     `Se ha recibido una petición ${req.method} en la ruta ${req.originalUrl}`
           );

  0.6 ms   const info = {
 11.9 ms     args: args._[0] || args["port"] || args["p"] || JSON.stringify(args),
  0.6 ms     platform: process.platform,
             version: process.version,
  0.8 ms     memory: process.memoryUsage().rss,
  0.1 ms     path: process.cwd(),
  0.1 ms     pid: process.pid,
  5.3 ms     folder: path.dirname(new URL(import.meta.url).pathname),
             numCPUs: numCPUs,
           };

  6.6 ms   console.log(info);

 29.6 ms   res.render("info", { info });
  0.2 ms });
```

-Sin console.log():

```
         //INFO UTILIZANDO EL OBJETO PROCESS
  1.4 ms app.get("/info", gzipMiddleware, (req, res) => {
  5.0 ms   logger.info(
  5.9 ms     `Se ha recibido una petición ${req.method} en la ruta ${req.originalUrl}`
           );

  0.8 ms   const info = {
 20.5 ms     args: args._[0] || args["port"] || args["p"] || JSON.stringify(args),
  0.3 ms     platform: process.platform,
  0.2 ms     version: process.version,
  0.4 ms     memory: process.memoryUsage().rss,
  0.1 ms     path: process.cwd(),
  0.1 ms     pid: process.pid,
  8.8 ms     folder: path.dirname(new URL(import.meta.url).pathname),
             numCPUs: numCPUs,
           };

 37.6 ms   res.render("info", { info });
  0.3 ms });
```

<u>Diagrama de flama con 0x:</u>

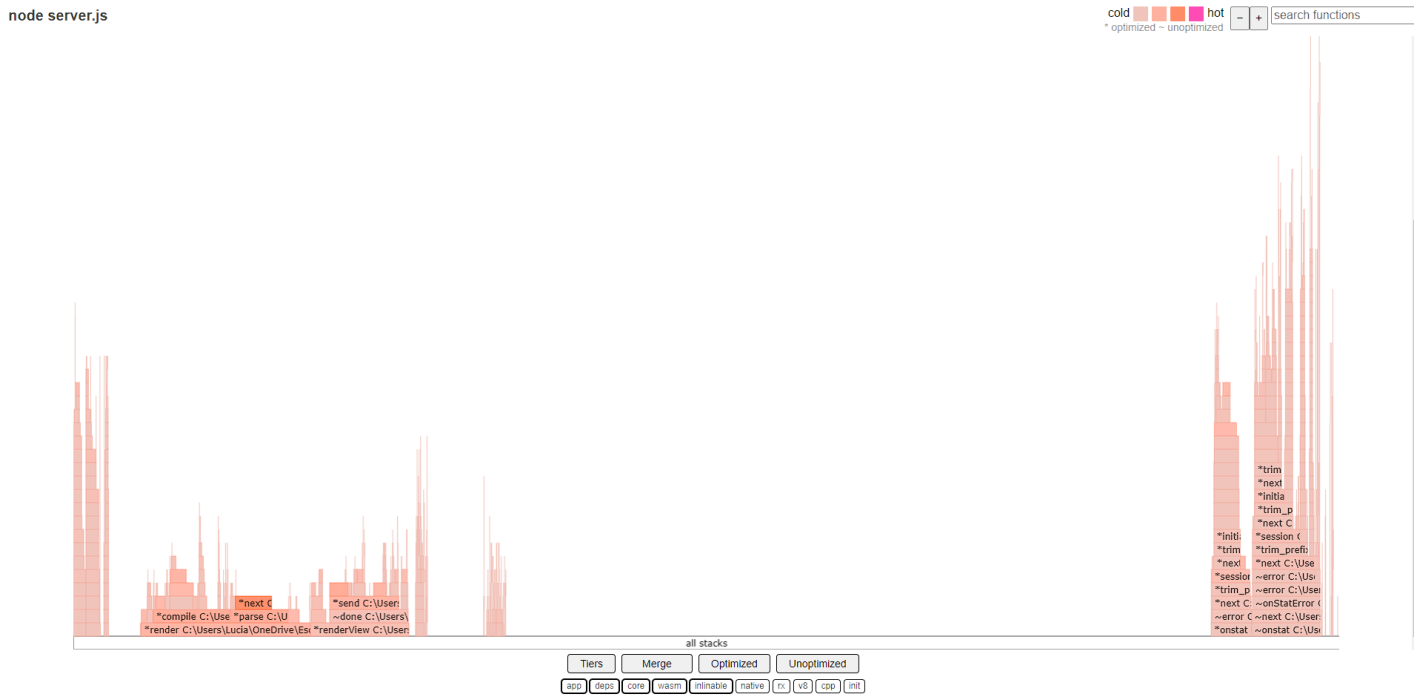
Con nuestro archivo de test “benchmark.js” configurado y los scripts necesarios en el package.json, ejecutamos:

-npm run profiling

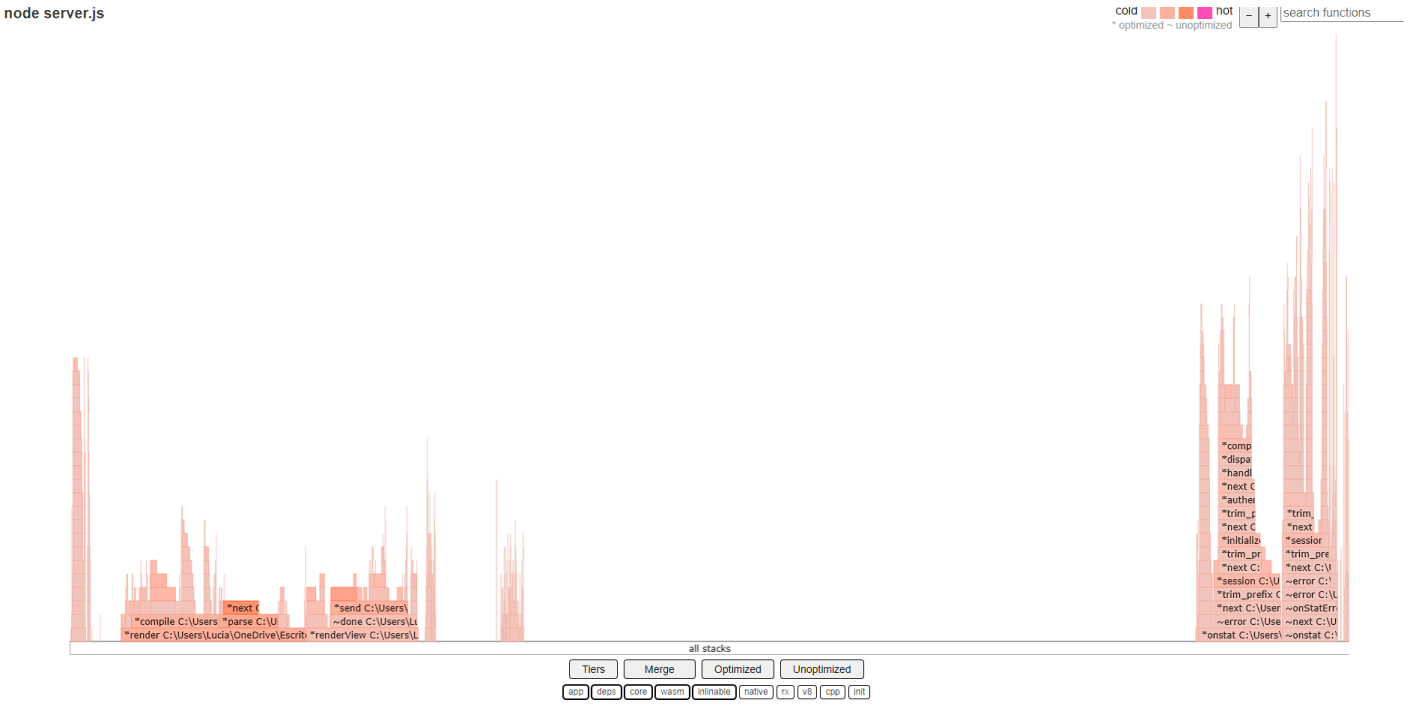
-npm run test

Al apagar el servidor se crea la carpeta con el gráfico solicitado:

-Con console.log()



-Sin console.log()



Conclusión:

Al examinar los datos recopilados, se puede notar que las pruebas realizadas sin imprimir en consola la información obtenida en la ruta "/info" tuvieron una duración más corta y transmitieron

menos datos. Sin embargo, debido a la escasa cantidad de información manejada, los resultados de las pruebas ofrecen poco para analizar.