## Entrega 16 - Análisis Completo de Performance

-Vamos a trabajar sobre la ruta “/info” en modo FORK

1 – Perfilamiento de servidor (profiler)

**Información de ruta : “/info” NO BLOQUEANTE**

Para esto utilizamos el comando:

```
$ node --prof server.js
```

Utilizamos artillery, 20 request de 50 cuentas

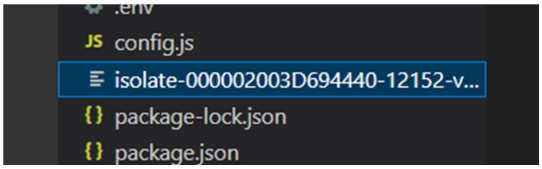
```
$ artillery quick --count 20 -n 50 "http://localhost:8080/info" > result_nobloq.txt
```

Podemos ver el resultado

result_nobloq.txt

```
  http.codes.200: ..................................................... 1000
  http.request_rate: ..................................................... 277/sec
  http.requests: ..................................................... 1000
  http.response_time:
    min: ..................................................... 2
    max: ..................................................... 53
    median: ..................................................... 16.9
    p95: ..................................................... 32.8
    p99: ..................................................... 46.1
  http.responses: ..................................................... 1000
  vusers.completed: ..................................................... 20
  vusers.created: ..................................................... 20
  vusers.created_by_name.0: ..................................................... 20
  vusers.failed: ..................................................... 0
  vusers.session_length:
    min: ..................................................... 510.4
    max: ..................................................... 1118.6
    median: ..................................................... 1022.7
    p95: ..................................................... 1107.9
    p99: ..................................................... 1107.9
```

Modificamos el archivo isolate que nos da el resultado del “—prof”. Lo renombraremos a “nobloq-v8.log”




Corremos el comando “- -proff-process” para procesar el archivo log a uno txt con formato resumido.

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

Resultado NO BLOQUEANTE

```
[Summary]:
  ticks  total  nonlib   name
      8    0.1%  100.0%  JavaScript
      0    0.0%    0.0%  C++
      9    0.1%  112.5%  GC
  15403   99.9%          Shared libraries
```

Utilizamos **autocannon**, creando un archivo nuevo “autocannon.js” con la importación y la función correspondiente

```
function run(url){
    const buf=[];
    const outputStream = new PassThrough();

    const inst = autocannon({
        url,
        connections:100,
        duration:20
    });

    autocannon.track(inst,{outputStream});

    outputStream.on('data',data => buf.push(data))
    inst.on('done', function(){
        process.stdout.write(Buffer.concat(buf));
    })
}

console.log('Running all benchmark in parallel...');

run('http://localhost:8080/info');
```

```
gabriel@gabriel-pc MINGW64 ~/Documents/CH/entrega/performance
$ node autocannon.js
```

Nos arroja por consola el siguiente resultado para “no bloqueante”

| Stat | 2.5% | 50% | 97.5% | 99% | Avg | Stdev | Max |
|---|---|---|---|---|---|---|---|
| Latency | 78 ms | 91 ms | 151 ms | 173 ms | 99.72 ms | 21.07 ms | 202 ms |

| Stat | 1% | 2.5% | 50% | 97.5% | Avg | Stdev | Min |
|---|---|---|---|---|---|---|---|
| Req/Sec | 688 | 688 | 1068 | 1140 | 997.15 | 144.87 | 688 |
| Bytes/Sec | 1.48 MB | 1.48 MB | 2.29 MB | 2.45 MB | 2.14 MB | 311 kB | 1.48 MB |

**Informacion de ruta : “/info” BLOQUEANTE**

Para esto agregamos un console log, con los datos de info, antes de realizar el envio

```
29      console.log(info);
30
31      res.render('info',{info});
```

Corremos el server *bloqueante* en modo “—prof”

```
$ node --prof server.js
```

Hacemos la misma carga con Artillery

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

Obtenemos los siguientes resultados

```
nance > ≡ result_bloq.txt
Phase completed: unnamed (index: 0, duration: 1s) 13:07:57(-0300)

All VUs finished. Total time: 10 seconds

--------------------------------
Summary report @ 13:08:04(-0300)
--------------------------------

http.codes.200: ..................................................................... 1000
http.request_rate: .................................................................. 146/sec
http.requests: ...................................................................... 1000
http.response_time:
  min: .............................................................................. 2
  max: .............................................................................. 147
  median: ........................................................................... 49.9
  p95: .............................................................................. 83.9
  p99: .............................................................................. 98.5
http.responses: ..................................................................... 1000
vusers.completed: ................................................................... 20
vusers.created: ..................................................................... 20
vusers.created_by_name.0: ........................................................... 20
vusers.failed: ...................................................................... 0
vusers.session_length:
  min: .............................................................................. 1995.1
  max: .............................................................................. 2801.2
  median: ........................................................................... 2671
  p95: .............................................................................. 2780
  p99: .............................................................................. 2780
```

Corremos el comando “- -proff-process” para procesar el archivo log a uno txt con formato resumido. Ahora con el archivo bloq-v8.log

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

Obtenemos el resultado del profiler

```
[Summary]:
  ticks  total  nonlib   name
      9    0.2%  100.0%  JavaScript
      0    0.0%    0.0%  C++
      8    0.2%   88.9%  GC
   3705   99.8%          Shared libraries
```

Corremos el archivo autocannon.js nuevamente pero en modo bloqueante

| Stat | 2.5% | 50% | 97.5% | 99% | Avg | Stdev | Max |
|---|---|---|---|---|---|---|---|
| Latency | 235 ms | 282 ms | 408 ms | 445 ms | 291.51 ms | 44.8 ms | 497 ms |

| Stat | 1% | 2.5% | 50% | 97.5% | Avg | Stdev | Min |
|---|---|---|---|---|---|---|---|
| Req/Sec | 238 | 238 | 349 | 393 | 341.55 | 36.71 | 238 |
| Bytes/Sec | 511 kB | 511 kB | 749 kB | 844 kB | 733 kB | 78.8 kB | 511 kB |

**2 – Perfilamiento del Servidor en modo "—inspect"**

Hacemos la primera prueba con "/info" *no bloqueante*

```
$ node --inspect server.js
```

Testeamos con artillery con los mismo paremetros

```
gabriel@gabriel-pc MINGW64 ~/Documents/CH/entrega/performance
$ artillery quick --count 20 -n 50 "http://localhost:8080/info"
Running scenarios...
Phase started: unnamed (index: 0, duration: 1s) 13:29:52(-0300)
```

Inspeccionamos los datos con el "profiler" que viene integrado en DevTools y obtenemos los siguientes resultados

```
13
14          router.get("/info",compression(), (req, res) => {
15
16    5.0 ms    logger.log('info', `ROUTE: ${req.path} - METHOD: ${req.method}`);
17
18    0.5 ms    info = {
19    7.3 ms        args : JSON.stringify(argumento),
20    0.1 ms        plataform: process.platform,
21    0.1 ms        version : process.version,
22    4.3 ms        memory: JSON.stringify(process.memoryUsage()),
23    0.3 ms        path : process.execPath,
24    0.2 ms        pid : process.pid,
25    0.5 ms        dir : process.cwd(),
26    0.1 ms        cpus : numCPUs
27              }
28
29    5.9 ms    res.render('info',{info});
30          });
31
32          /* --------- random con fork -------- */
```

Mismo proceso pero ahora de manera *bloqueante*

```
13
14          router.get("/info",compression(), (req, res) => {
15
16    4.9 ms    logger.log('info', `ROUTE: ${req.path} - METHOD: ${req.method}`);
17
18    1.5 ms    info = {
19   12.9 ms        args : JSON.stringify(argumento),
20    0.2 ms        plataform: process.platform,
21    0.3 ms        version : process.version,
22    5.0 ms        memory: JSON.stringify(process.memoryUsage()),
23    0.4 ms        path : process.execPath,
24    0.4 ms        pid : process.pid,
25    1.1 ms        dir : process.cwd(),
26                  cpus : numCPUs
27              }
28
29    5.6 ms    console.log(info);
30
31   14.9 ms    res.render('info',{info});
32          });
33
```

### 3 – Diagrama de Flama usando 0x

Para esta actividad utilizamos la misma ruta “/info” pero usando de aplicación de carga a “autocannon”

Iniciamos nuestro servidor en modo “0x”

```
$ 0x server.js
```

Ejecutamos el Test de “autocannon” y nos genera los archivos de 0x

Grafico de Flama para “ NO Bloqueante”

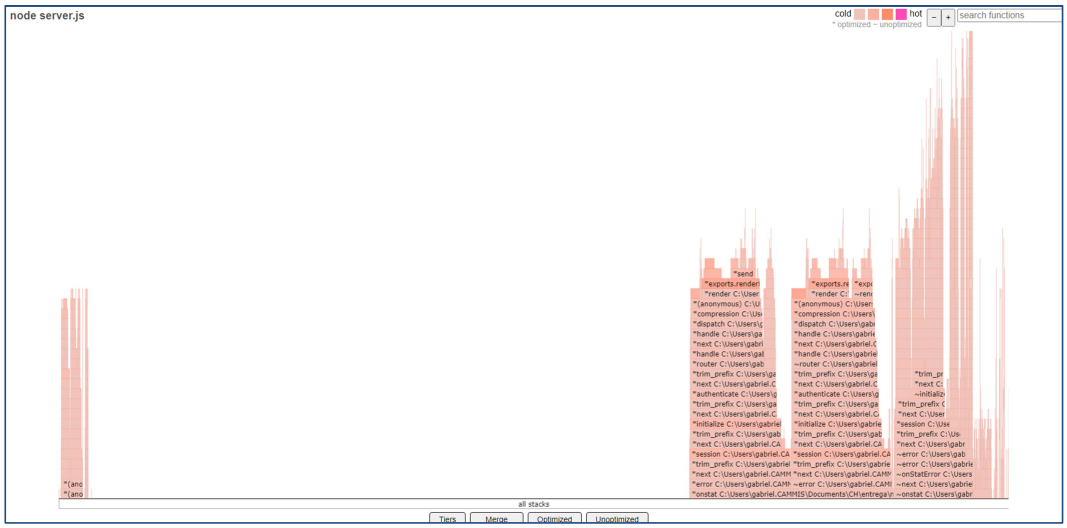



Gráfico de Flama para “Bloqueante”

## Análisis de Resultados

### Profiler Bult-In

**NO BLOQUEANTE**

```
[Summary]:
   ticks  total  nonlib   name
       8   0.1%  100.0%  JavaScript
       0   0.0%    0.0%  C++
       9   0.1%  112.5%  GC
   15403  99.9%          Shared libraries
```

**BLOQUEANTE**

```
[Summary]:
   ticks  total  nonlib   name
       9   0.2%  100.0%  JavaScript
       0   0.0%    0.0%  C++
       8   0.2%   88.9%  GC
    3705  99.8%          Shared libraries
```

Podemos notar que el servidor no bloqueante cuenta con 5 veces la cantidad de ticks sobre el bloqueante

### Artillery

NO BLOQUEANTE

```
http.codes.200: ..... 1000
http.request_rate: ..... 277/sec
http.requests: ..... 1000
http.response_time:
  min: ..... 2
  max: ..... 53
  median: ..... 16.9
  p95: ..... 32.8
  p99: ..... 46.1
http.responses: ..... 1000
vusers.completed: ..... 20
vusers.created: ..... 20
vusers.created_by_name.0: ..... 20
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 510.4
  max: ..... 1118.6
  median: ..... 1022.7
  p95: ..... 1107.9
  p99: ..... 1107.9
```

BLOQUEANTE

```
http.codes.200: ..... 1000
http.request_rate: ..... 146/sec
http.requests: ..... 1000
http.response_time:
  min: ..... 2
  max: ..... 147
  median: ..... 49.9
  p95: ..... 83.9
  p99: ..... 98.5
http.responses: ..... 1000
vusers.completed: ..... 20
vusers.created: ..... 20
vusers.created_by_name.0: ..... 20
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 1995.1
  max: ..... 2801.2
  median: ..... 2671
  p95: ..... 2780
  p99: ..... 2780
```

Podemos ver que en el archivo *no bloqueante* se registra, un *velocidad de respuesta* de 277/sec y en la bloqueante una respuesta de 146/sec, la mitad que el no bloqueante, generando una media de respuesta final de casi el triple de lentitud.

Autocannon

NO BLOQUEANTE

| Stat | 2.5% | 50% | 97.5% | 99% | Avg | Stdev | Max |
|---|---|---|---|---|---|---|---|
| Latency | 78 ms | 91 ms | 151 ms | 173 ms | 99.72 ms | 21.07 ms | 202 ms |

| Stat | 1% | 2.5% | 50% | 97.5% | Avg | Stdev | Min |
|---|---|---|---|---|---|---|---|
| Req/Sec | 688 | 688 | 1068 | 1140 | 997.15 | 144.87 | 688 |
| Bytes/Sec | 1.48 MB | 1.48 MB | 2.29 MB | 2.45 MB | 2.14 MB | 311 kB | 1.48 MB |

BLOQUEANTE

| Stat | 2.5% | 50% | 97.5% | 99% | Avg | Stdev | Max |
|---|---|---|---|---|---|---|---|
| Latency | 235 ms | 282 ms | 408 ms | 445 ms | 291.51 ms | 44.8 ms | 497 ms |

| Stat | 1% | 2.5% | 50% | 97.5% | Avg | Stdev | Min |
|---|---|---|---|---|---|---|---|
| Req/Sec | 238 | 238 | 349 | 393 | 341.55 | 36.71 | 238 |
| Bytes/Sec | 511 kB | 511 kB | 749 kB | 844 kB | 733 kB | 78.8 kB | 511 kB |

En esta comparación, podemos ver fácilmente, como el servidor no bloqueante, es fácilmente 3 veces mas rápido, con menor latencia y con mayor respuestas por segundo

0x

NO BLOQUEANTE

BLOQUEANTE

Podemos ver como en el proceso no bloqueante, el grafico es mucho más angosto, con picos más altos y en el bloqueante, el grafico es 1/3 más ancho que el primero.

## Conclucion

A partir de los diversos analisis obtenidos en el modulo, podemos tener muy enclaro que cualqueir proceso sincronico, como lo es una pequeña funcion  de “console.log”, puede afectar altamente al rendimiento de nuestra aplicación de manera negativa.