

Desafío 32 - Programación Backend

Camilo Gálvez Vidal

Se realizan las siguientes pruebas al servidor con las siguientes herramientas:

- **Node Prof**

- Resultados sin *console.log*

[Summary]:

| ticks | total | nonlib | name |
|-------|-------|--------|------------------|
| 0 | 0.0% | 0.0% | JavaScript |
| 87 | 87.9% | 100.0% | C++ |
| 25 | 25.3% | 28.7% | GC |
| 12 | 12.1% | | Shared libraries |

- Resultados con *console.log*

[Summary]:

| ticks | total | nonlib | name |
|-------|-------|--------|------------------|
| 213 | 3.8% | 4.0% | JavaScript |
| 5033 | 89.5% | 95.5% | C++ |
| 280 | 5.0% | 5.3% | GC |
| 354 | 6.3% | | Shared libraries |
| 22 | 0.4% | | Unaccounted |

- **Artillery**

- Resultados sin *console.log*

Started phase 0, duration: 1s @ 21:37:50(-0400) 2021-08-26

Report @ 21:37:53(-0400) 2021-08-26

Elapsed time: 2 seconds

Scenarios launched: 20

Scenarios completed: 20

Requests completed: 1000

Mean response/sec: 404.86

Response time (msec):

min: 3

max: 41

median: 26

p95: 35

p99: 38

Codes:

200: 1000

All virtual users finished

Summary report @ 21:37:53(-0400) 2021-08-26

Scenarios launched: 20

Scenarios completed: 20

Requests completed: 1000

Mean response/sec: 403.23

Response time (msec):

min: 3

max: 41

median: 26

p95: 35

p99: 38

```
Scenario counts:
  0: 20 (100%)
Codes:
  200: 1000
```

- **Resultados con *console.log***

```
Started phase 0, duration: 1s @ 21:37:10(-0400) 2021-08-26
Report @ 21:37:12(-0400) 2021-08-26
```

```
Elapsed time: 2 seconds
  Scenarios launched: 20
  Scenarios completed: 20
  Requests completed: 1000
  Mean response/sec: 403.23
  Response time (msec):
    min: 2
    max: 54
    median: 32
    p95: 46
    p99: 50.5
  Codes:
    200: 1000
```

```
All virtual users finished
```

```
Summary report @ 21:37:12(-0400) 2021-08-26
```

```
  Scenarios launched: 20
  Scenarios completed: 20
  Requests completed: 1000
  Mean response/sec: 401.61
  Response time (msec):
    min: 2
    max: 54
    median: 32
    p95: 46
    p99: 50.5
  Scenario counts:
    0: 20 (100%)
  Codes:
    200: 1000
```

- Chrome dev tools inspect
 - Resultados sin `console.log`

| | | | | | |
|---------|--------|----------|---------|--------------|-------------|
| 21.3 ms | 0.55 % | 466.6 ms | 12.08 % | ▼(anonymous) | server.js:1 |
| 21.3 ms | 0.55 % | 466.6 ms | 12.08 % | ► handle | layer.js:1 |

| | | |
|-----|--------|--|
| 171 | | |
| 172 | 0.4 ms | app.get('/info', (req,res) => { |
| 173 | | // Argumentos de entrada |
| 174 | 5.4 ms | const args = JSON.stringify(process.argv); |
| 175 | | // Nombre de plataforma (Sistema operativo) |
| 176 | | const so = process.platform; |
| 177 | | // Versión de NodeJS |
| 178 | 0.1 ms | const vNode = process.version; |
| 179 | | // Uso de memoria |
| 180 | 8.7 ms | const memory = JSON.stringify(process.memoryUsage(), null, 2); |
| 181 | | // Path de ejecución |
| 182 | 0.1 ms | const pathExec = process.execPath; |
| 183 | | // Process ID |
| 184 | | const processID = process.pid; |
| 185 | | // Carpeta corriente |
| 186 | 0.3 ms | const cwd = process.cwd(); |
| 187 | 0.8 ms | info = { args, so, vNode, memory, pathExec, processID, cwd, numCPUs }; |
| 188 | | // Agregar o remover según el test |
| 189 | | // console.log(info) |
| 190 | | //..... |
| 191 | 2.8 ms | res.render("info", info) |
| 192 | | }) |
| 193 | | |

- Resultados con `console.log`

| | | | | | |
|---------|--------|-----------|---------|--------------|-------------|
| 41.2 ms | 0.66 % | 2716.4 ms | 43.23 % | ▼(anonymous) | server.js:1 |
| 41.2 ms | 0.66 % | 2716.4 ms | 43.23 % | ► handle | layer.js:1 |

| | | |
|-----|---------|--|
| 171 | | |
| 172 | 0.1 ms | app.get('/info', [(req,res)] => { |
| 173 | | // Argumentos de entrada |
| 174 | 7.9 ms | const args = JSON.stringify(process.argv); |
| 175 | | // Nombre de plataforma (Sistema operativo) |
| 176 | 0.1 ms | const so = process.platform; |
| 177 | | // Versión de NodeJS |
| 178 | | const vNode = process.version; |
| 179 | | // Uso de memoria |
| 180 | 11.3 ms | const memory = JSON.stringify(process.memoryUsage(), null, 2); |
| 181 | | // Path de ejecución |
| 182 | 0.1 ms | const pathExec = process.execPath; |
| 183 | | // Process ID |
| 184 | 0.1 ms | const processID = process.pid; |
| 185 | | // Carpeta corriente |
| 186 | 1.1 ms | const cwd = process.cwd(); |
| 187 | 3.4 ms | info = { args, so, vNode, memory, pathExec, processID, cwd, numCPUs }; |
| 188 | | // Agregar o remover según el test |
| 189 | 14.6 ms | console.log(info) |
| 190 | | //..... |
| 191 | 1.8 ms | res.render("info", info) |
| 192 | | }) |
| 193 | | |

- 0x y Autocannon
 - Resultados sin *console.log*

```
> npm run test

> coderhouse_backend@0.0.0 test
> node benchmark.js

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

| Stat | 2.5% | 50% | 97.5% | 99% | Avg | Stdev | Max |
|---------|-------|-------|--------|--------|-----------|----------|--------|
| Latency | 93 ms | 99 ms | 176 ms | 232 ms | 109.08 ms | 26.84 ms | 362 ms |

| Stat | 1% | 2.5% | 50% | 97.5% | Avg | Stdev | Min |
|-----------|---------|---------|---------|---------|--------|--------|---------|
| Req/Sec | 400 | 400 | 996 | 1083 | 909.9 | 163.84 | 400 |
| Bytes/Sec | 1.18 MB | 1.18 MB | 2.95 MB | 3.21 MB | 2.7 MB | 486 kB | 1.18 MB |

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

18k requests in 20.05s, 53.9 MB read
```

- Resultados con *console.log*

```
> npm run test

> coderhouse_backend@0.0.0 test
> node benchmark.js

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

| Stat | 2.5% | 50% | 97.5% | 99% | Avg | Stdev | Max |
|---------|--------|--------|--------|--------|----------|----------|--------|
| Latency | 101 ms | 111 ms | 196 ms | 254 ms | 121.4 ms | 29.68 ms | 372 ms |

| Stat | 1% | 2.5% | 50% | 97.5% | Avg | Stdev | Min |
|-----------|---------|---------|---------|---------|---------|--------|---------|
| Req/Sec | 357 | 357 | 869 | 964 | 819.35 | 155.66 | 357 |
| Bytes/Sec | 1.06 MB | 1.06 MB | 2.58 MB | 2.86 MB | 2.43 MB | 462 kB | 1.06 MB |

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

16k requests in 20.04s, 48.6 MB read
```

Conclusión:

En base a los resultados de los test de rendimiento, se llega a la conclusión de que el *loggear* la información hace que se demanden más recursos para la ejecución del servidor, en comparación con no *loggearla*. Como recomendación y pasos a seguir, se aconseja solamente *loggear* la información necesaria, ya que si se hace de manera muy detallada, se debe considerar que el rendimiento de nuestro servidor empeorará.