Desafío Clase 32 Logs, debug, profiling

Alumno: Valentin Garcia Devrient

Coderhouse – Diciembre 2022

Comisión 29020

Verificar sobre la ruta /info con y sin compresión, la diferencia de cantidad de bytes devueltos en un caso y otro

Test de performance sobre la ruta /info, con y sin console.log

Perfilamiento del servidor, realizando el test con --prof de node.js, analizando los resultados obtenidos luego de procesarlos con --prof-process. Request realizadas utilizando **Artillery**

Con console log

[Summary]:

ticks	total	nonlib	name
158	1.0%	98.1%	JavaScript
0	0.0%	0.0%	C++
226	1.5%	140.4%	GC
15349	99.0%		Shared libraries
3	0.0%		Unaccounted

Sin console log [Summary]:

ticks	total	nonlib	name
156	1.5%	99.4%	JavaScript
0	0.0%	0.0%	C++
218	2.2%	138.9%	GC
9948	98.4%		Shared libraries
1	0.0%		Unaccounted

Pruebas utilizando **Autocannon** en línea de comandos, emulando 100 conexiones concurrentes realizadas en un tiempo de 20 segundos.

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

Con console log:

```
79                  });
80      0.1 ms      app.get("/info", (0, compression_1.default)(), (req, res) => {
81      3.9 ms          const { argv, execPath, platform, version, pid, memoryUsage, cwd } = process;
82     22.9 ms          console.log(`Info en ${process.pid} y ${process.argv}`);
83                      const { rss } = memoryUsage();
84     22.4 ms          console.log(argv, execPath, platform, version, pid, rss, cwd(), os.cpus().length);
85     12.4 ms          res.render("info", {
86                          layout: "info",
87                          argv,
88                          execPath,
89                          platform,
90      0.1 ms              version,
91                          pid,
92                          rss,
93      1.0 ms              currentDir: cwd(),
94      4.3 ms              cpus: os.cpus().length,
95                      });
96      0.1 ms      });
97                  // LOGIN
```

```
PS D:\Mati\CoderHouse\Back\coder-back\clase-32\desafio-entregable> node ./src/benchmark.js
Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8080/info
50 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	246 ms	280 ms	375 ms	394 ms	289.3 ms	34.19 ms	451 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	100	100	172	200	171.4	24.14	100
Bytes/Sec	216 kB	216 kB	371 kB	432 kB	370 kB	52.1 kB	216 kB

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

3k requests in 20.04s, 7.39 MB read
```

Sin console log:

```
80   0.2 ms   app.get("/info", (0, compression_1.default)(), (req, res) => {
81   2.5 ms       const { argv, execPath, platform, version, pid, memoryUsage, cwd } = process;
82  34.2 ms       console.log(`Info en ${process.pid} y ${process.argv}`);
83   5.8 ms       const { rss } = memoryUsage();
84                /*console.log(
85                  argv,
86                  execPath,
87                  platform,
88                  version,
89                  pid,
90                  rss,
91                  cwd(),
92                  os.cpus().length
93                );*/
94  10.4 ms       res.render("info", {
95                    layout: "info",
96   0.1 ms           argv,
97   0.1 ms           execPath,
98   0.1 ms           platform,
99                    version,
100                   pid,
101  0.1 ms           rss,
102  1.3 ms           currentDir: cwd(),
103  6.5 ms           cpus: os.cpus().length,
104               });
```

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

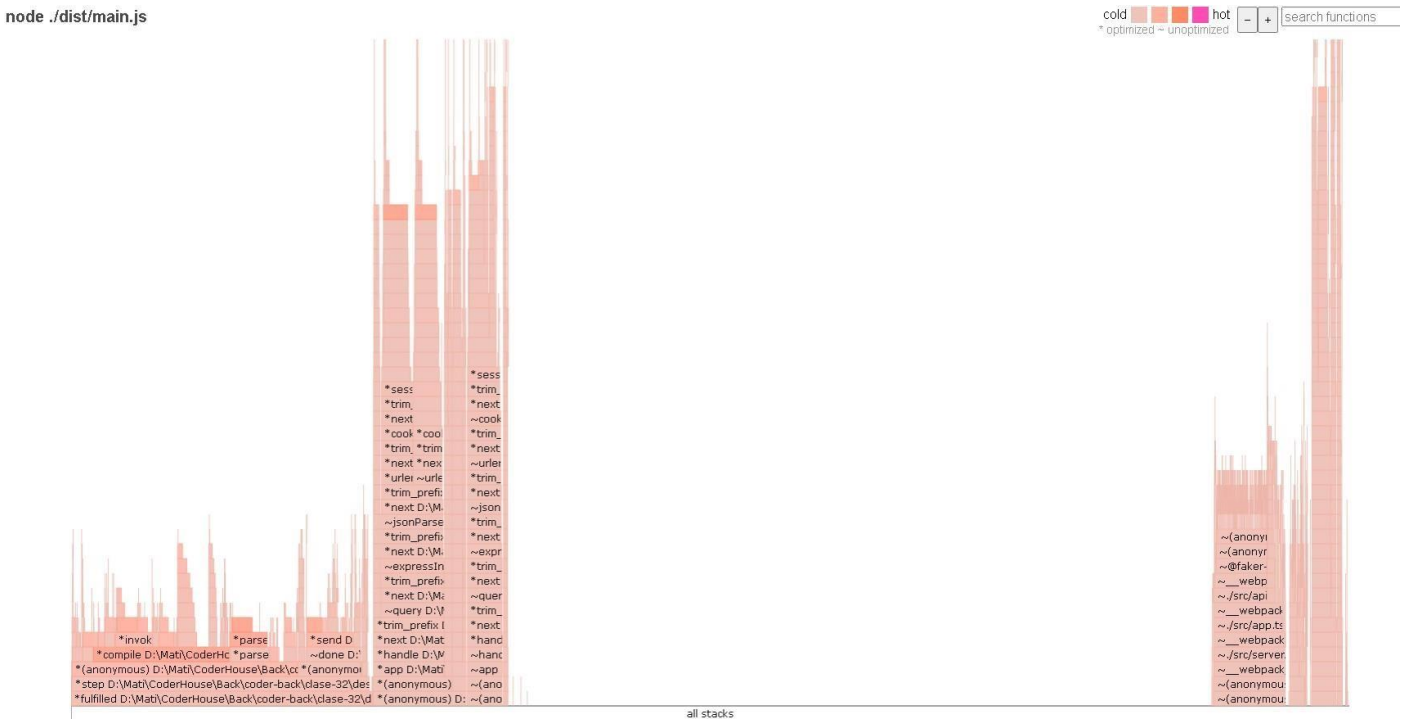
Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	160 ms	183 ms	263 ms	281 ms	191.69 ms	28.79 ms	394 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	150	150	261	287	259.3	27.47	150
Bytes/Sec	324 kB	324 kB	563 kB	620 kB	559 kB	59.2 kB	324 kB

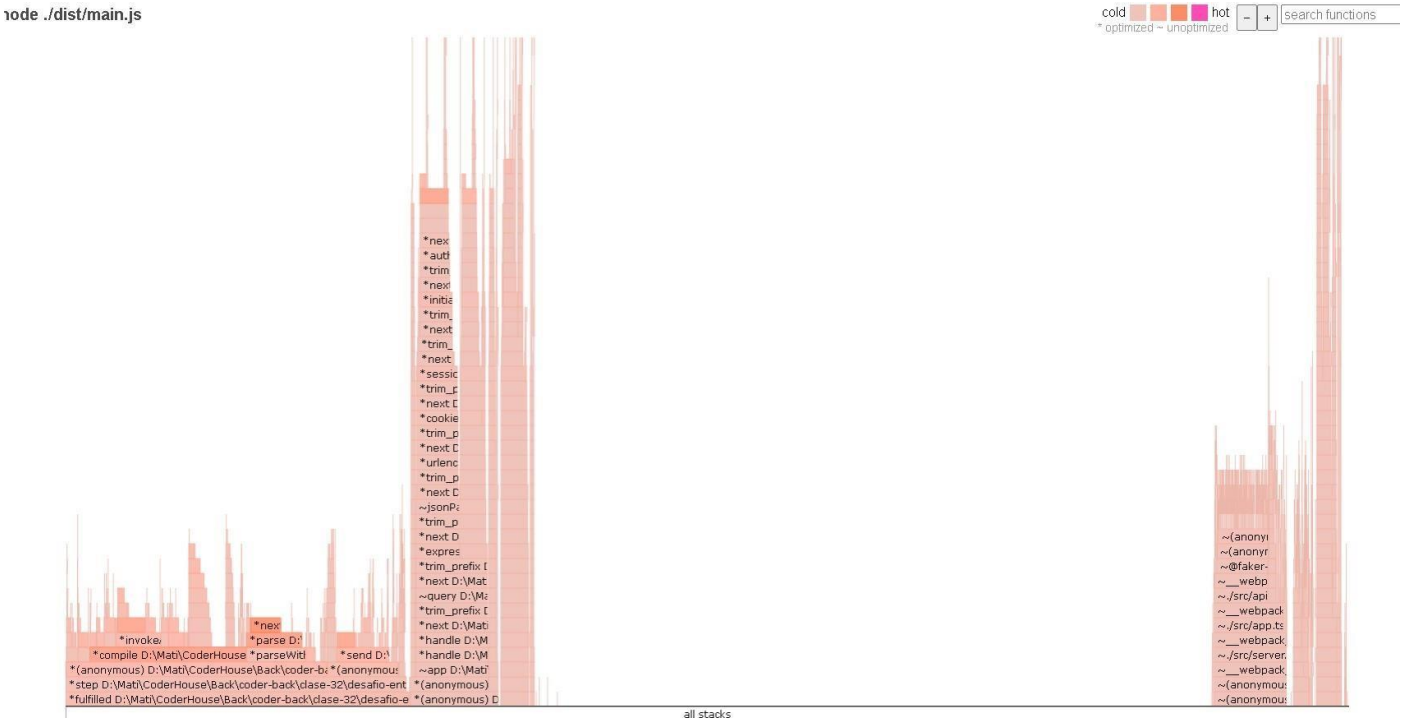
```
Req/Bytes counts sampled once per second.
# of samples: 20

5k requests in 20.05s, 11.2 MB read
```

Diagrama de flama con 0x, emulando la carga con Autocannon:

Con console log:



Sin console log:



Conclusión

Luego de haber realizado pruebas de rendimiento con distintas herramientas podemos concluir que:

- La herramienta Gzip nos genera buena compresión en el tamaño de las respuestas que da nuestro servidor, pero hay que usarla con cuidado ya que no está recomendada para servidores en producción con tráfico elevado
- Es importante hacer las pruebas de rendimiento en el mismo entorno y condiciones para que los resultados no se vean afectados. De está manera vamos a poder compararlos mejor y sacar conclusiones al respecto.
- Las operaciones síncronas y los console log afectan bastante al rendimiento del servidor. Es importante evitarlas siempre que se pueda y utilizar alguna librería para loguear errores en vez de hacerlo de esta manera.