

Ruta /info con y sin compresión

Sin compresión

Nombre	Estado	Tipo	Iniciador	Tamaño	Hora	Cascada
info	200	document	Otros	9.7 kB	15 ms	

Con compresión

Nombre	Estado	Tipo	Iniciador	Tamaño	Hora	Cascada
info	200	document	Otros	620 B	7 ms	

Perfilamiento con `–prof` (se adjuntan archivos en `/performace`)

(el `console.log()` se agrega en la línea 28 de `/routes/infoRouter.js`)

1

En el `profConConsole.txt` `node.exe` ocupa 6.1% de los ticks

En el `profSinConsole.txt` `node.exe` ocupa 5.3% de los ticks

En `artilleryConConsole.txt` se observan 390 req./seg. y respuesta promedio 55.2ms

En `artillerySinConsole.txt` se observan 674 req./seg y respuesta promedio de 24.8ms

AUTOCANON

Con `console.log()`

```
Tabare@DESKTOP-1SLHONS MINGW64 ~/Desktop/Full_Stack/Backend_40845/desafio13 (main)
$ node ../node_modules/autocannon/autocannon -c 100 -d 20 http://localhost:8080
Running 20s test @ http://localhost:8080
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	118 ms	168 ms	204 ms	211 ms	167.28 ms	18.98 ms	257 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	467	467	598	651	595.71	44.06	467
Bytes/Sec	333 kB	333 kB	426 kB	465 kB	425 kB	31.6 kB	333 kB

Req/Bytes counts sampled once per second.
of samples: 20
12k requests in 20.00s, 8.5 MB read

Sin `console.log()`

```
Tabare@DESKTOP-1SLHONS MINGW64 ~/Desktop/Full_Stack/Backend_40845/desafio13 (main)
$ node ../node_modules/autocannon/autocannon -c 100 -d 20 http://localhost:8080/info
Running 20s test @ http://localhost:8080/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	59 ms	65 ms	124 ms	138 ms	74.79 ms	19.96 ms	165 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	921	921	1355	1485	1328.9	148.14	921
Bytes/Sec	657 kB	657 kB	968 kB	1.06 MB	949 kB	186 kB	657 kB

Req/Bytes counts sampled once per second.
of samples: 20
27k requests in 20.08s, 19 MB read

2 Autocanon - INSPECT

Inspect

Con console.log()

Perfiles	Gruesa (de abajo hacia arriba)	Tiempo individual	Tiempo total	Función
PERFILES DE CPU	13321.7 ms	13321.7 ms	(idle)	
Perfil 1	982.8 ms 36.78 %	1549.7 ms 58.01 %	▶ consoleCall	
	445.8 ms 16.69 %	445.8 ms 16.69 %	▶ writeUtf8String	
	76.7 ms 2.87 %	76.7 ms 2.87 %	▶ writev	
	52.4 ms 1.96 %	52.4 ms 1.96 %	▶ (program)	
	49.3 ms 1.84 %	49.3 ms 1.84 %	▶ (garbage collector)	
	31.1 ms 1.16 %	31.1 ms 1.16 %	▶ getColorDepth	
	27.1 ms 1.01 %	2196.6 ms 82.22 %	▶ session	
	26.9 ms 1.01 %	63.5 ms 2.38 %	▶ hash	
	23.2 ms 0.87 %	23.2 ms 0.87 %	▶ hash	
	22.9 ms 0.86 %	2021.1 ms 75.65 %	▶ (anónimo)	
	20.1 ms 0.75 %	387.0 ms 14.49 %	▶ send	
	17.6 ms 0.66 %	40.9 ms 1.53 %	▶ Hash	
	17.1 ms 0.64 %	6736.0 ms 252.13 %	▶ next	
	16.3 ms 0.61 %	29.7 ms 1.11 %	▶ nextTick	
	15.2 ms 0.57 %	15.3 ms 0.57 %	▶ getHeader	
	14.9 ms 0.56 %	8721.7 ms 326.46 %	▶ handle	
	12.4 ms 0.46 %	19.5 ms 0.73 %	▶ writeHead	
	12.1 ms 0.45 %	12.8 ms 0.48 %	▶ parse	
	11.8 ms 0.44 %	14.3 ms 0.53 %	▶ asString	
	11.4 ms 0.43 %	61.2 ms 2.29 %	▶ store.generate	
	11.0 ms 0.41 %	26.2 ms 0.98 %	▶ afterWriteDispatched	
	10.5 ms 0.39 %	10.5 ms 0.39 %	▶ setWriteHeadHeaders	
	10.4 ms 0.39 %	10.4 ms 0.39 %	▶ run	
	10.4 ms 0.39 %	208.5 ms 7.80 %	▶ end	
	10.4 ms 0.39 %	10.4 ms 0.39 %	▶ stat	
	10.1 ms 0.38 %	58.0 ms 2.17 %	▶ resOnFinish	
	10.1 ms 0.38 %	565.6 ms 21.17 %	▶ log	
	9.7 ms 0.36 %	585.4 ms 21.91 %	▶ Socket_writeGeneric	
	9.4 ms 0.35 %	13.5 ms 0.50 %	▶ set maxAge	
	9.4 ms 0.35 %	96.2 ms 3.60 %	▶ expressInit	
	9.4 ms 0.35 %	17.0 ms 0.63 %	▶ setHeader	
	9.3 ms 0.35 %	43.7 ms 1.64 %	▶ value	
	9.0 ms 0.34 %	9.0 ms 0.34 %	▶ memoryUsage	
	8.9 ms 0.33 %	118.2 ms 4.42 %	▶ processTicksAndRejections	
	8.5 ms 0.32 %	2257.2 ms 84.49 %	▶ callbackTrampoline	

Consola

MINGW64/c/Users/...

2001 mensajes

2000 mensajes de usuario

No hay errores

Process Id	11992				
Numero de procesadores	C:\Users\Tabare\Desktop\Full_Stack\Backend_40845\desafio13\src\main.js				
[2023-03-10T13:57:04.102] [INFO] default - Ruta: /info, metodo: GET					
<table border="1"><tr><th>Descripcion</th><th>Valor</th></tr><tr><td>Sist</td><td></td></tr></table>	Descripcion	Valor	Sist		
Descripcion	Valor				
Sist					

Sin console.log()

Perfiles	Gruesa (de abajo hacia arriba)	Tiempo individual	Tiempo total	Función
PERFILES DE CPU	30589.7 ms	30589.7 ms	(idle)	
Perfil 1	309.1 ms 23.88 %	447.3 ms 34.56 %	▶ consoleCall	
	90.7 ms 7.01 %	90.7 ms 7.01 %	▶ writeUtf8String	
	48.3 ms 3.73 %	48.3 ms 3.73 %	▶ writev	
	42.6 ms 3.29 %	42.6 ms 3.29 %	▶ (program)	
	32.1 ms 2.48 %	32.1 ms 2.48 %	▶ (garbage collector)	
	18.4 ms 1.42 %	909.3 ms 70.25 %	▶ session	
	17.6 ms 1.36 %	46.1 ms 3.56 %	▶ hash	
	15.6 ms 1.21 %	2899.1 ms 223.96 %	▶ next	
	15.3 ms 1.18 %	25.4 ms 1.96 %	▶ hash	
	14.4 ms 1.11 %	269.2 ms 20.80 %	▶ send	
	13.2 ms 1.02 %	783.3 ms 60.51 %	▶ (anónimo)	
	12.8 ms 0.99 %	23.4 ms 1.81 %	▶ nextTick	
	12.3 ms 0.95 %	12.3 ms 0.95 %	▶ getColorDepth	
	11.3 ms 0.85 %	18.6 ms 1.44 %	▶ writeHead	
	10.8 ms 0.84 %	3651.9 ms 282.11 %	▶ handle	
	10.0 ms 0.78 %	10.0 ms 0.78 %	▶ Hash	
	9.7 ms 0.75 %	20.4 ms 1.57 %	▶ writev	
	9.5 ms 0.74 %	83.8 ms 6.48 %	▶ expressInit	
	9.5 ms 0.74 %	12.3 ms 0.95 %	▶ asString	
	8.5 ms 0.65 %	14.9 ms 1.15 %	▶ deserializeObject	
	7.9 ms 0.61 %	13.1 ms 1.01 %	▶ emitHook	
	7.9 ms 0.61 %	7.9 ms 0.61 %	▶ stat	
	7.7 ms 0.59 %	46.3 ms 3.58 %	▶ store.generate	
	7.4 ms 0.57 %	39.9 ms 3.08 %	▶ resOnFinish	
	7.2 ms 0.56 %	127.4 ms 10.61 %	▶ callbackTrampoline	

Consola

MINGW64/c/Users/...

1001 mensajes

1000 mensajes de usuario

No hay errores

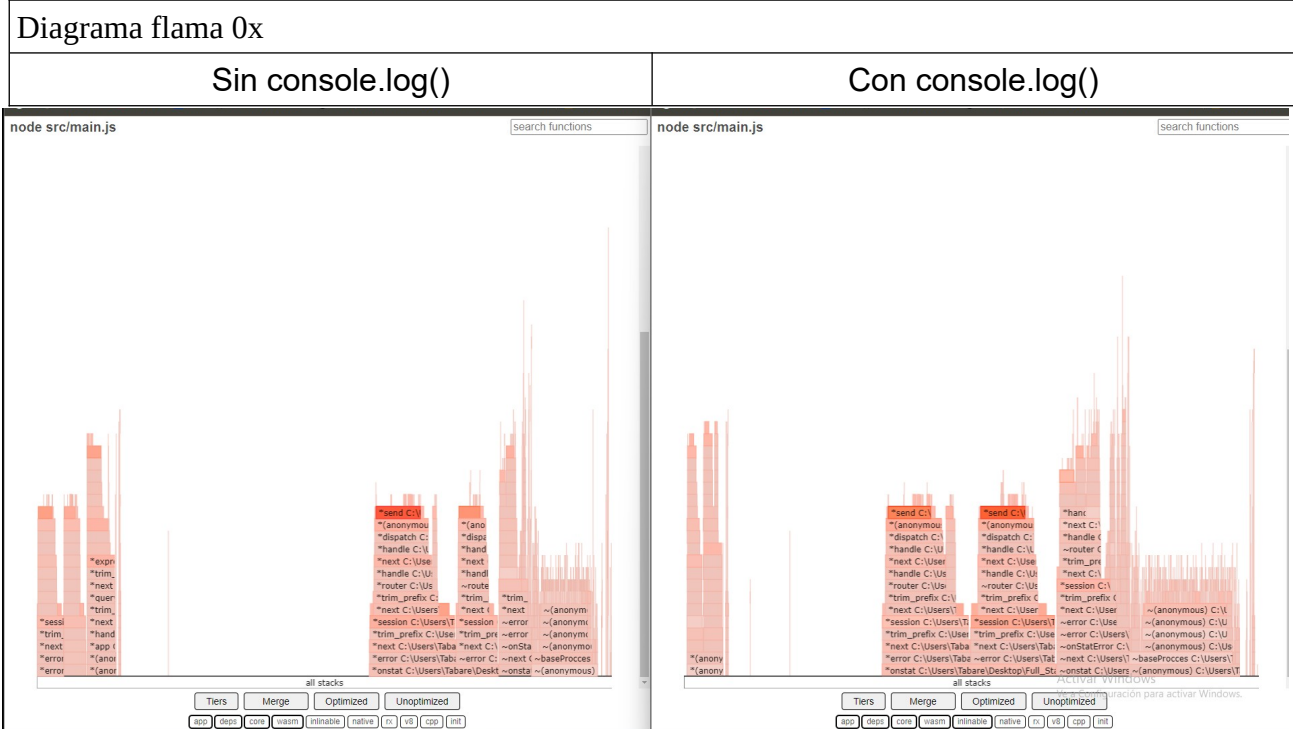
1 advertencia

1000 informaciones

No hay mensajes con verbosidad

[2023-03-10T13:53:08.472] [INFO] default - Ruta: /info, metodo: GET	
[2023-03-10T13:53:08.476] [INFO] default - Ruta: /info, metodo: GET	
[2023-03-10T13:53:08.479] [INFO] default - Ruta: /info, metodo: GET	
[2023-03-10T13:53:08.486] [INFO] default - Ruta: /info, metodo: GET	
[2023-03-10T13:53:08.489] [INFO] default - Ruta: /info, metodo: GET	
[2023-03-10T13:53:08.492] [INFO] default - Ruta: /info, metodo: GET	

3 Autocanon - 0x



CONCLUSIÓN

Después de realizar múltiples pruebas con las herramientas de Artillery y Autocannon, y analizar los resultados con las herramientas `–prof`, el paquete `0x` e `inspect` en el navegador, se ha podido evidenciar una mejora significativa en el tiempo de respuesta y la cantidad de solicitudes realizadas al quitar la función `console.log()` de nuestro código. Estos resultados sugieren que la eliminación de esta función puede tener un impacto positivo en el rendimiento de nuestras aplicaciones.