LOGGERS, GZIP y ANÁLISIS DE PERFORMANCE

GZIP

Se agregó “const longResponseProof = "Hola que tal".repeat(1000);” para visualizar diferencias, obteniendo:

a) Ruta “/info” sin compression:
   Size: 13.2 kB

b) Ruta “/info” con compression:
   Size: 864 B

LOGGERS

Se usó la librería pino

ANÁLISIS DE PERFORMANCE

ARTILLERY

Se agregó “const longResponseProof = "Hola que tal".repeat(1000);” para visualizar diferencias, obteniendo:

a) Ruta “/info-nodebug” sin console.log (longResponseProof):

   Pasos (artillery):

   - Se inicia servidor : node --prof src/server.js
   - Se usa comando: artillery quick --count 20 -n 50 http://127.0.0.1:8080/info-nodebug > result_info-nodebug.txt
   - Se decodifican archivos: node.exe --prof-process

info-nodebug-v8.log > result_info-nodebug.txt

Pasos (inspect):

- Se inicia servidor : node --inspect src/server.js
- En el navegador: chrome://inspect → DevTools → profiler → start
- En consola se usa comando: artillery quick --count 20 -n 50 http://127.0.0.1:8080/info-nodebug > result_info-nodebug.txt

b) Ruta "/info-debug" con console.log (longResponseProof):

Pasos (artillery):

- Se inicia servidor : node --prof src/server.js
- Se usa comando: artillery quick --count 20 -n 50 http://127.0.0.1:8080/info-debug > result_info-debug.txt
- Se decodifican archivos: node.exe --prof-process info-debug-v8.log > result_info-debug.txt

Pasos (inspect):

- Se inicia servidor : node --inspect src/server.js
- En el navegador: chrome://inspect → DevTools → profiler → start
- En consola se usa comando: artillery quick --count 20 -n 50 http://127.0.0.1:8080/info-debug > result_info-debug.txt

CONCLUSIÓN:

Los tiempos de procesos son mayores en la ruta /info-debug debido al console.log que es sincrónico y bloqueante (se observa en el número de ticks en cada archivo descargado, siendo más altos en la ruta /info-debug)

AUTOCANNON Y 0x

Resultados obtenidos con autocannon:

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

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	816 ms	1021 ms	2531 ms	2538 ms	1259.07 ms	463.06 ms	2823 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	0	0	97	135	78.8	36.9	3
Bytes/Sec	0 B	0 B	107 kB	149 kB	86.8 kB	40.7 kB	3.3 kB

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

2k requests in 20.38s, 1.74 MB read
```

/info-nodebug

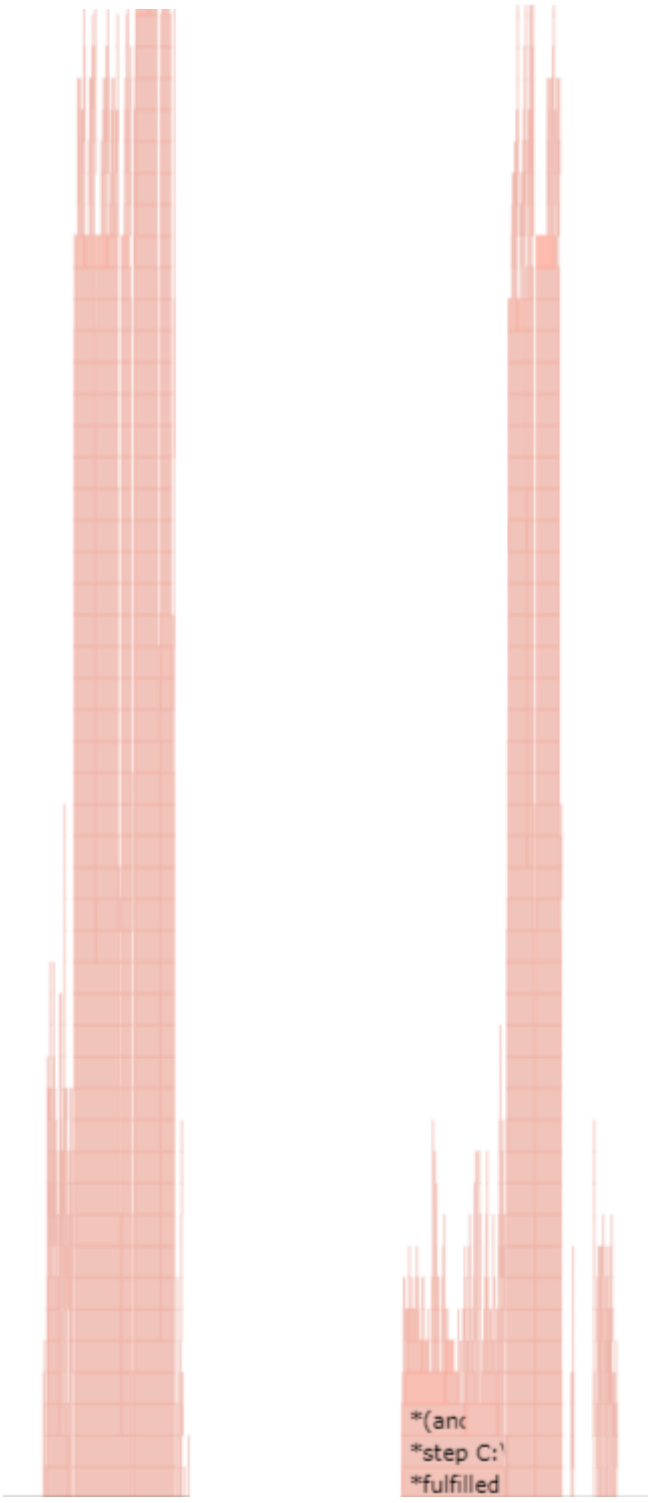
```
Running 20s test @ http://localhost:8080/info-debug
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	922 ms	1022 ms	2101 ms	2506 ms	1245.3 ms	378.29 ms	2976 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	0	0	99	114	79.7	35.6	2
Bytes/Sec	0 B	0 B	1.3 MB	1.5 MB	1.05 MB	468 kB	26.3 kB

/info-debug

Resultados obtenidos con 0x:



Izquierda: /info-nodebug
Derecha: /info-debug

CONCLUSIÓN:

El diagrama de flama expresa los procesos bloqueantes de forma horizontal, lo que significa que duran mucho más tiempo en el stack. En este caso, se ve que el proceso bloqueante (/info-debug) tiene más procesos horizontales que el proceso no bloqueante (/info-nodebug) , por lo que se evidencia su mayor duración en el stack.