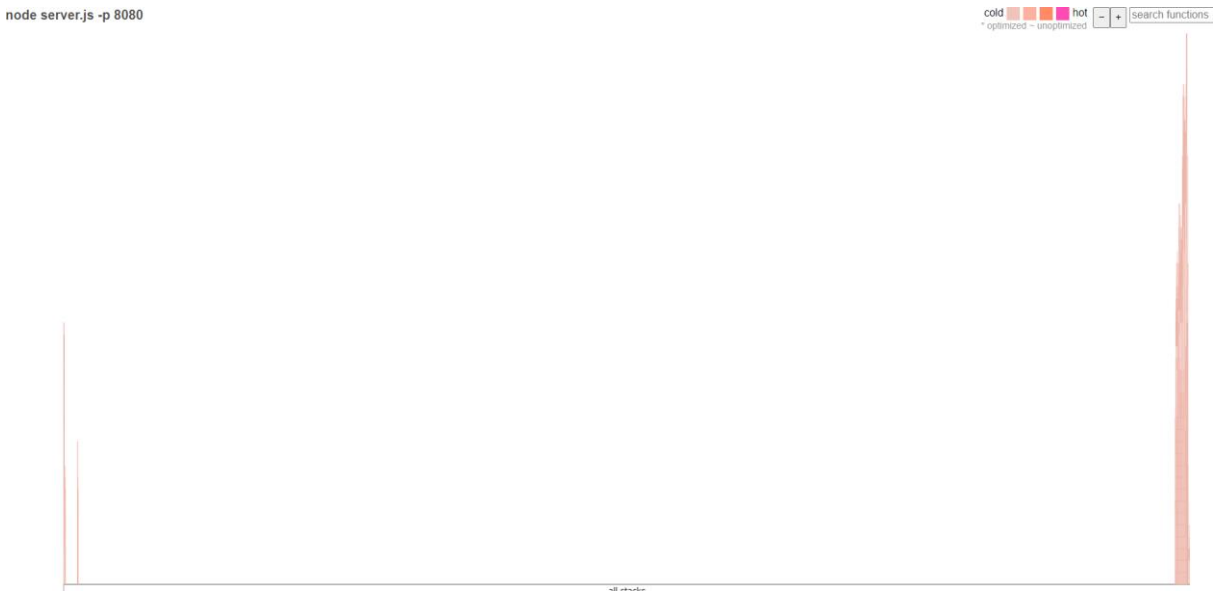
DESAFÍO 14 - Logs, debug, profiling

node --Inspect server.js
npm run test // script autocannon

20958.8 ms		20958.8 ms		(idle)
3117.3 ms	16.46 %	6796.8 ms	35.89 %	▼consoleCall
2125.1 ms	11.22 %	5188.3 ms	27.39 %	▼getInfoBloq
2125.1 ms	11.22 %	5188.3 ms	27.39 %	▶handle
992.2 ms	5.24 %	1608.4 ms	8.49 %	▶(anonymous)
3094.8 ms	16.34 %	3094.8 ms	16.34 %	▶writeUtf8String
905.4 ms	4.78 %	905.4 ms	4.78 %	▶getCPUs
875.9 ms	4.62 %	875.9 ms	4.62 %	▶stat
594.4 ms	3.14 %	594.4 ms	3.14 %	▶open
441.4 ms	2.33 %	441.4 ms	2.33 %	▶access
420.7 ms	2.22 %	980.0 ms	5.17 %	▶compile
340.7 ms	1.80 %	340.7 ms	1.80 %	(garbage collector)
229.7 ms	1.21 %	229.7 ms	1.21 %	(program)
222.9 ms	1.18 %	222.9 ms	1.18 %	▶close
221.2 ms	1.17 %	221.2 ms	1.17 %	▶writev
211.5 ms	1.12 %	211.5 ms	1.12 %	▶read
210.1 ms	1.11 %	13789.9 ms	72.81 %	▶authenticate
209.1 ms	1.10 %	14929.4 ms	78.83 %	▶compression
196.4 ms	1.04 %	335.7 ms	1.77 %	▶scanLine
160.5 ms	0.85 %	160.5 ms	0.85 %	▶Hash
156.6 ms	0.83 %	156.6 ms	0.83 %	▶fstat
126.8 ms	0.67 %	203.2 ms	1.07 %	▶nextTick
117.9 ms	0.62 %	117.9 ms	0.62 %	▶_addOutput
117.7 ms	0.62 %	75404.6 ms	398.13 %	▶next
111.6 ms	0.59 %	88618.6 ms	467.90 %	▶handle
110.9 ms	0.59 %	169.2 ms	0.89 %	▶createRegex
107.2 ms	0.57 %	14648.2 ms	77.34 %	▶session
105.9 ms	0.56 %	323.3 ms	1.71 %	▶hash
97.4 ms	0.51 %	1914.4 ms	10.11 %	▶send
93.1 ms	0.49 %	101.4 ms	0.54 %	▶exports.shallowCopy
92.0 ms	0.49 %	2706.9 ms	14.29 %	▶(anonymous)
84.2 ms	0.44 %	137.7 ms	0.73 %	▶resolve
76.9 ms	0.41 %	407.8 ms	2.15 %	▶(anonymous)
76.4 ms	0.40 %	76.4 ms	0.40 %	▶run
76.3 ms	0.40 %	439.3 ms	2.32 %	▶store.generate
76.2 ms	0.40 %	76.2 ms	0.40 %	▶setWriteHeadHeaders
74.7 ms	0.39 %	126.8 ms	0.67 %	▶writeHead
70.1 ms	0.37 %	231.2 ms	1.22 %	▶Hash
69.3 ms	0.37 %	9604.7 ms	50.71 %	▶getInfoBloq
67.0 ms	0.35 %	67.0 ms	0.35 %	▶getColorDepth
61.4 ms	0.32 %	1274.4 ms	6.73 %	▶compile
60.5 ms	0.32 %	75.6 ms	0.40 %	▶asString
59.2 ms	0.31 %	59.2 ms	0.31 %	▶normalizeString
58.8 ms	0.31 %	59.1 ms	0.31 %	▶onHeaders
58.7 ms	0.31 %	1976.3 ms	10.43 %	▶done

Se puede ver cuánto tardó el consoleCall por el getInfoBloq (que tiene el console.log(datos))



En la anterior imagen se puede ver como en el primer pico de la izquierda, se muestra el profiling de la ruta “/info”, la cual no tiene funciones bloqueantes. Y en el segundo pico, de la ruta “/info-bloq” (que si tiene un console.log síncrono) a la derecha, se notan procesos más “largos” en el sentido de que el pico es más “grueso” y no tan fino como el primero. Se puede distinguir entonces que la ruta no-bloqueante es más rápida al no tener funciones síncronas bloqueantes, y en cambio la ruta bloqueante tiene que esperar a que termine esa función síncrona para poder continuar. En este caso una sola función síncrona que dura pocos milisegundos no perjudica la velocidad pero si se fueran a amontonar muchas y muchas peticiones podrían llegar a representar un problema a resolver en un futuro.