Análisis de performance del servidor

- Perfilamiento del servidor:

  con console.log

```
[Summary]:
  ticks  total  nonlib   name
     95    4.5%    4.9%  JavaScript
   1810   86.7%   93.6%  C++
     62    3.0%    3.2%  GC
    155    7.4%          Shared libraries
     28    1.3%          Unaccounted
```

  sin console.log

```
[Summary]:
  ticks  total  nonlib   name
     52    2.8%    3.1%  JavaScript
   1601   87.2%   94.9%  C++
     93    5.1%    5.5%  GC
    148    8.1%          Shared libraries
     34    1.9%          Unaccounted
```

Se puede notar que en el caso de “con console.log” el proceso tiene más ticks aunque no parece que haya una diferencia muy grande entre los dos.

- Test de carga con Artillery:

  con console.log

```
http.request_rate: ... 436/sec
http.requests: ....... 1000
http.response_time:
  min: ............... 0
  max: ............... 26
  median: ............ 2
```

  sin console.log

```
http.request_rate: ......... 500/sec
http.requests: ............. 1000
http.response_time:
  min: ...................... 0
  max: ...................... 14
  median: ................... 1
```

Aquí podemos ver que “sin console.log” saca ventaja al tener más request por segundo y además un tiempo de respuesta máximo de casi la mitad que su contraparte.

- Con Autocannon:

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

con console.log

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	41 ms	56 ms	108 ms	123 ms	60.81 ms	17.42 ms	198 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	1311	1311	1585	1831	1629.6	138.36	1311
Bytes/Sec	722 kB	722 kB	873 kB	1.01 MB	898 kB	76.3 kB	722 kB

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

sin console.log

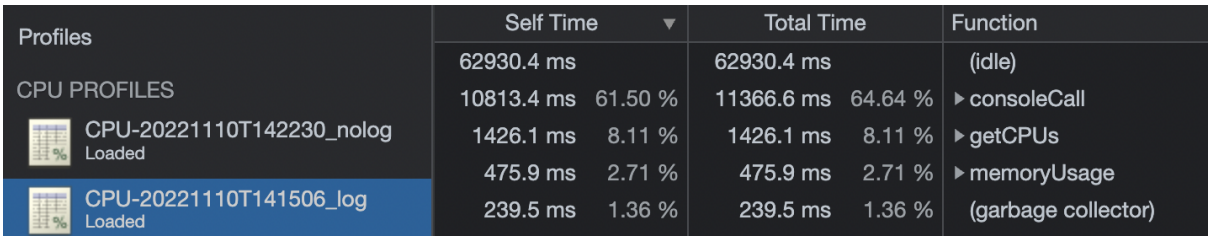
Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	29 ms	36 ms	64 ms	72 ms	38.4 ms	9.2 ms	121 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	1783	1783	2681	2827	2568.4	256.65	1783
Bytes/Sec	983 kB	983 kB	1.48 MB	1.56 MB	1.42 MB	141 kB	982 kB

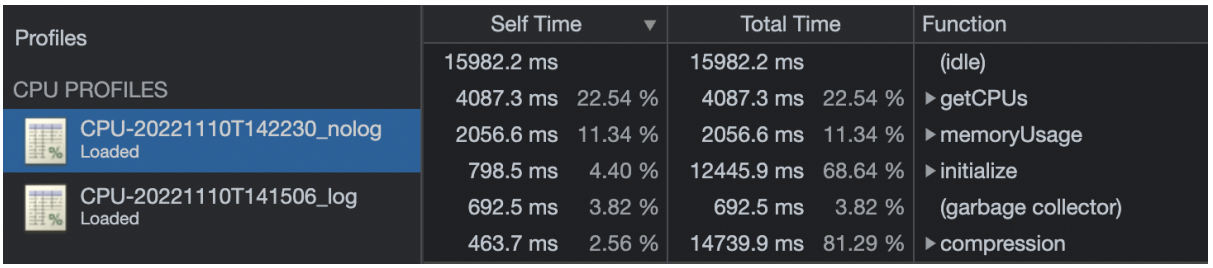
También podemos ver que donde no hay console.log la latencia es menor y las request por segundo son mucho mayores.

- Perfilamiento del servidor con modo –inspect:

  con console.log



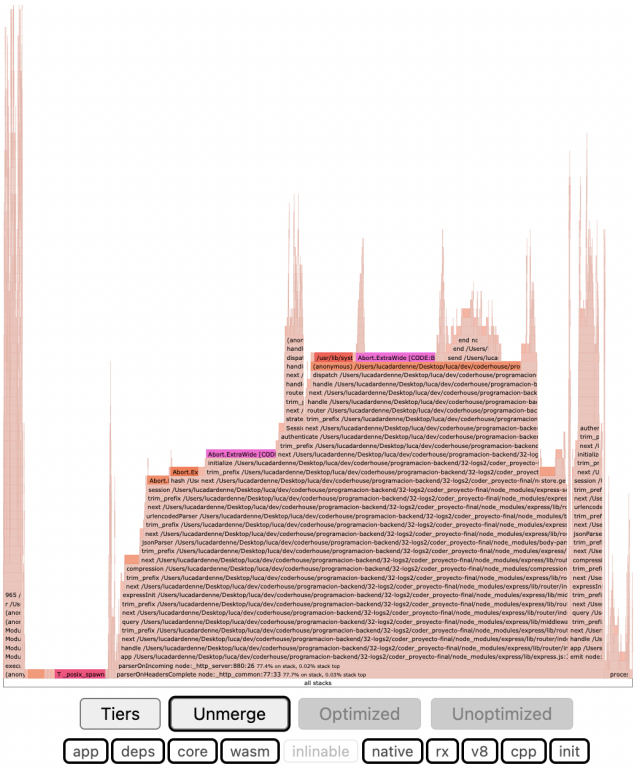
  sin console.log



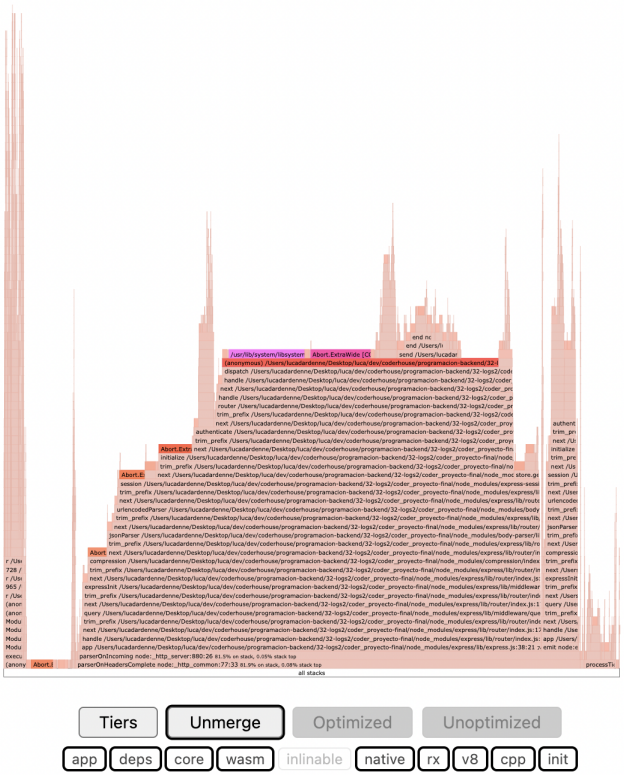
En los dos casos podemos notar que la función *getCPUs* se ejecuta durante un gran periodo de tiempo del programa, pero en el caso de “con console.log” *consoleCall* es -con mucha diferencia- la que mayor tiempo toma.

- Diagramas de flama con 0x y Autocannon

  con console.log



  sin console.log



  En este las dos pruebas lucen muy similares y es difícil marcar diferencias claras.

Conclusión

A partir de los datos obtenidos en las pruebas realizadas notamos que el efecto que produce la utilización de la función *log* del objeto *console* en un ámbito de producción es considerable y afecta al tiempo de respuesta y la administración de recursos de un servidor. Aunque no podemos negar su utilidad, su uso es ampliamente recomendable mientras una aplicación esté en fase de desarrollo y con fines de depuración de código.

IMPORTANTE: todos los archivos que se muestran en esta presentación se encuentran en la carpeta "performance" del repositorio de github