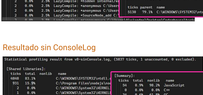
Se debia realizar el analisis de performance sobre la ruta /info del proyecto en modo fork. Agregado o extrayendo un console.log antes de devolver la informacion al cliente.

V8
prof-process

1- Utilizando el test con –prof de node.js y aplicando Artillery con el siguiente comando: artillery quick -count 20 -n 50 "http://localhost:8080/info?
Se deben realizar 50 conexiones concurrentes con 20 request cada una.

Resultado con ConsoleLog

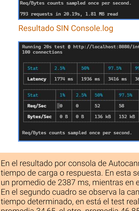


Resultado sin ConsoleLog

Desde este profiling se observa una mayor cantidad de Ticks de procesador en la version de console.log, mientras que en la version sin console.log una cantidad inferior.

Artillery

Resultado sin ConsoleLog



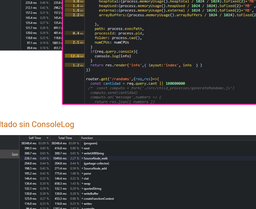
Resultado con ConsoleLog

En el profiling resultante de Artillery se puede medir el tiempo de respuesta en cada version, en este se observa una media en la version sin ConsoleLog de 382 segundos mientras que en la otra el tiempo de respuesta promedia los 449 segundos.

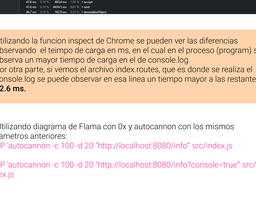
Autocannon

2- Utilizando Autocannon en linea de comando con 100 conexiones y en un tiempo de 20 segundos. autocannon -c 100 -d 20 "http://localhost:8080/info"

Resultado con Console.log



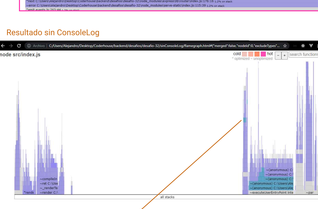
Resultado SIN Console.log



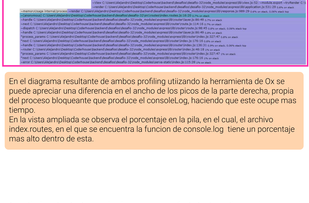
En el resultado por consola de Autocannon se observa la latencia, es decir el tiempo de carga o respuesta. En esta se ve en los test realizado con consoleLog un promedio de 2387 ms, mientras en el otro de 2032.38 ms.
En el segundo cuadro se observa la cantidad de peticiones atendidas en un tiempo determinado, en esta el realizado sobre la ruta info con console.log promedia 34,65, el otro promedia 46.85.
En el ultimo item se muestra la cantidad de bytes devueltos, en el test de console.log se observa 90.7kb de promedio mientras que en el otro 123 kb, una cantidad superior.

2- Utilizando Autocannon en linea de comando con 100 conexiones y en un tiempo de 20 segundos con el siguiente argumento con node en modo **inspect** : autocannon -c 100 -d 20 "http://localhost:8080/info"

Resultado con ConsoleLog



Resultado sin ConsoleLog



Utilizando la funcion inspect de Chrome se pueden ver las diferencias observando el tiempo de carga en ms, en el cual en el proceso (program) se observa un mayor tiempo de carga en el de console.log.
Por otra parte, si vemos el archivo index.routes, que es donde se realiza el console.log se puede observar en esa linea un tiempo mayor a las restantes de **12.6 ms.**

3- Utilizando diagrama de Flama con 0x y autocannon con los mismos parametros anteriores:
0x -P 'autocannon -c 100 -d 20 "http://localhost:8080/info"' src/index.js

0x -P 'autocannon -c 100 -d 20 "http://localhost:8080/info?console=true"' src/index.js

Resultado con ConsoleLog

Vista Ampliada

Resultado sin ConsoleLog

Vista Ampliada

En el diagrama resultante de ambos profiling utizando la herramienta de Ox se puede apreciar una diferencia en el ancho de los picos de la parte derecha, propia del proceso bloqueante que produce el consoleLog, haciendo que este ocupe mas tiempo.
En la vista ampliada se observa el porcentaje en la pila, en el cual, el archivo index.routes, en el que se encuentra la funcion de console.log tiene un porcentaje mas alto dentro de esta.

A partir de estos test se pudo observar como afecta tanto en tiempo de procesador, como en tiempo de respuesta el incluir un proceso bloqueante como lo es el consoleLog, en todos los datos obtenidos se puede ver una diferencia.