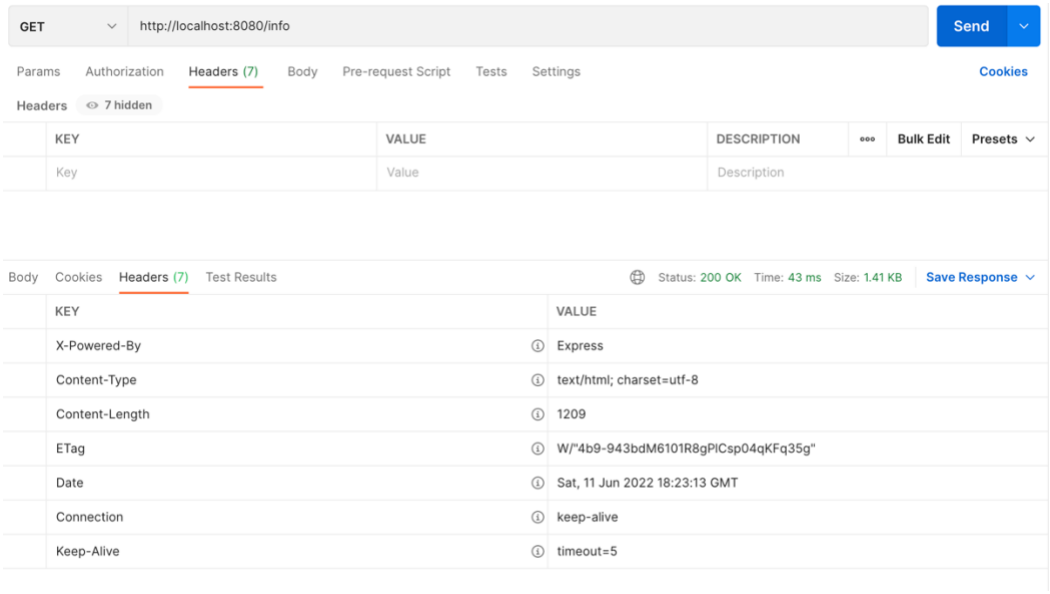
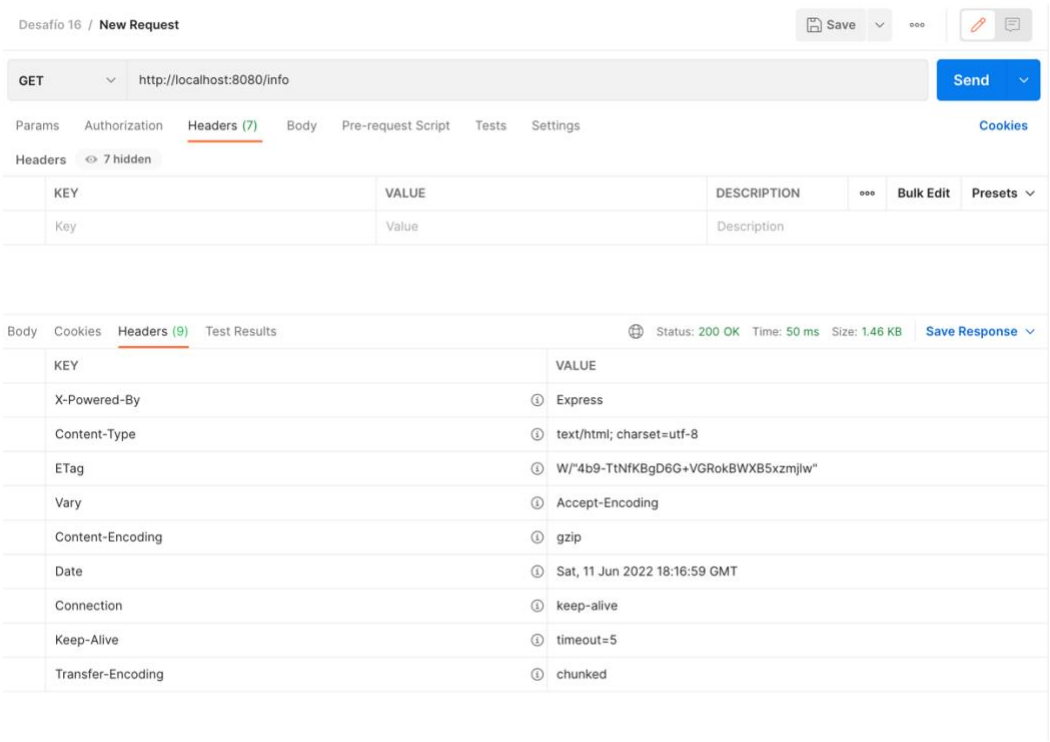
Reporte Desafío 16

1. Incorporar al proyecto de servidor de trabajo la compresión gzip. Verificar sobre la ruta /info con y sin compresión, la diferencia de cantidad de bytes devueltos en un caso y otro.

Respuesta sin compresión



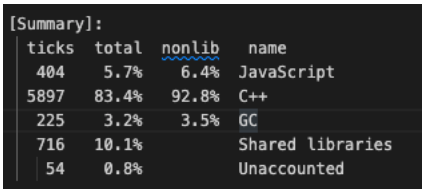
Respuesta utilizando compresión



Conclusión: Se observa una disminución de 0,05 KB en el tamaño de la respuesta del método info al utilizar compression.

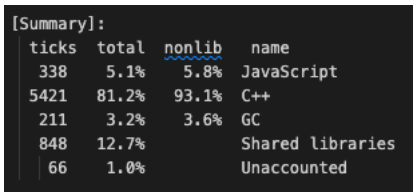
2) El perfilamiento del servidor, realizando el test con --prof de node.js. Analizar los resultados obtenidos luego de procesarlos con --prof-process. Utilizaremos como test de carga Artillery en línea de comandos, emulando 50 conexiones concurrentes con 20 request por cada una. Extraer un reporte con los resultados en archivo de texto.

Resultado método info sin console log:



```
[Summary]:
  ticks  total  nonlib   name
    404   5.7%    6.4%  JavaScript
   5897  83.4%   92.8%  C++
    225   3.2%    3.5%  GC
    716  10.1%          Shared libraries
     54   0.8%          Unaccounted
```

Resultado método info con console log:

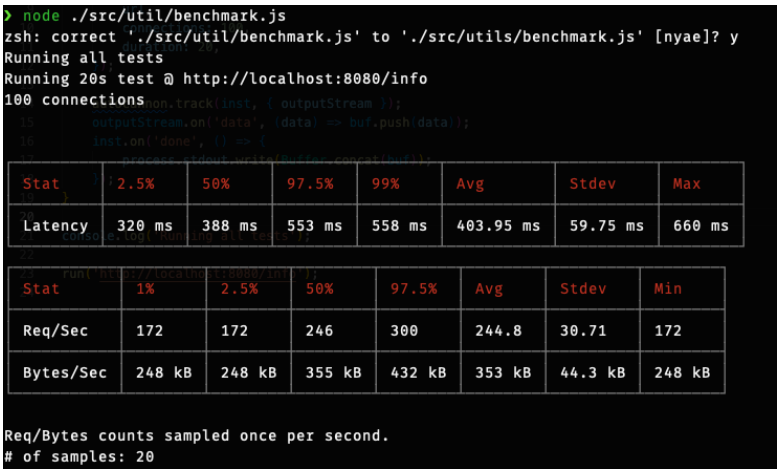


```
[Summary]:
  ticks  total  nonlib   name
    338   5.1%    5.8%  JavaScript
   5421  81.2%   93.1%  C++
    211   3.2%    3.6%  GC
    848  12.7%          Shared libraries
     66   1.0%          Unaccounted
```

Conclusión: Se observa que el método info que realiza el console.log tiene un mayor número de ticks que la versión que no realiza el console.log. Inicialmente se esperaba que se diera la situación contraria dado en la segunda configuración el código ejecuta una operación adicional.

3) Luego utilizaremos Autocannon en línea de comandos, emulando 100 conexiones concurrentes realizadas en un tiempo de 20 segundos. Extraer un reporte con los resultados (puede ser un print screen de la consola)

Resultado método info sin console log:



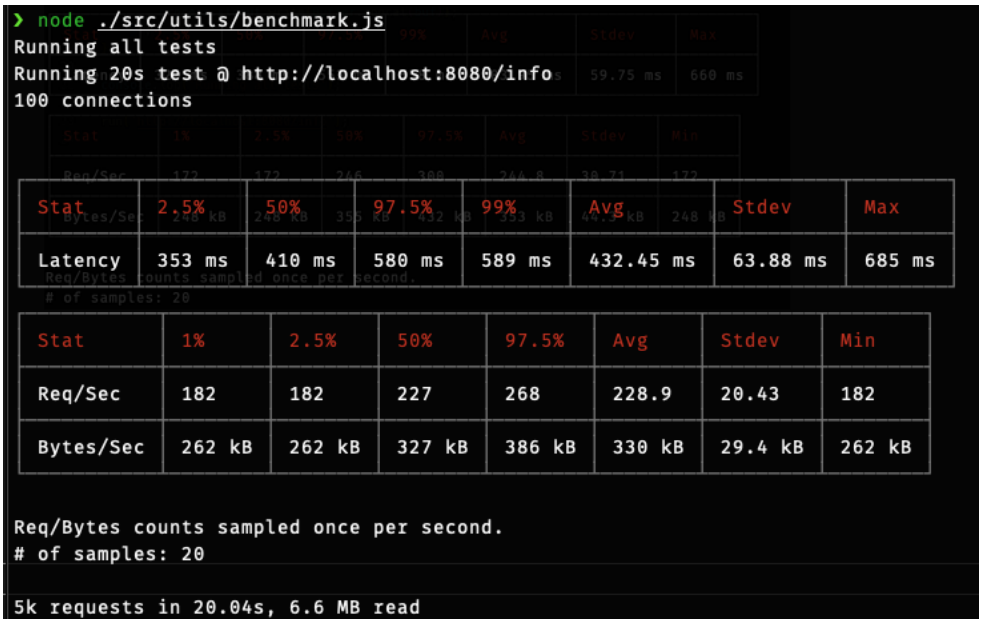
```
> node ./src/util/benchmark.js
zsh: correct './src/util/benchmark.js' to './src/utils/benchmark.js' [nyae]? y
Running all tests
Running 20s test @ http://localhost:8080/info
100 connections

| Stat    | 2.5%   | 50%    | 97.5%  | 99%    | Avg       | Stdev    | Max    |
|---------|--------|--------|--------|--------|-----------|----------|--------|
| Latency | 320 ms | 388 ms | 553 ms | 558 ms | 403.95 ms | 59.75 ms | 660 ms |

| Stat      | 1%     | 2.5%   | 50%    | 97.5%  | Avg    | Stdev   | Min    |
|-----------|--------|--------|--------|--------|--------|---------|--------|
| Req/Sec   | 172    | 172    | 246    | 300    | 244.8  | 30.71   | 172    |
| Bytes/Sec | 248 kB | 248 kB | 355 kB | 432 kB | 353 kB | 44.3 kB | 248 kB |

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

Resultado método info con console log:



```
> node ./src/utils/benchmark.js
Running all tests
Running 20s test @ http://localhost:8080/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	353 ms	410 ms	580 ms	589 ms	432.45 ms	63.88 ms	685 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	182	182	227	268	228.9	20.43	182
Bytes/Sec	262 kB	262 kB	327 kB	386 kB	330 kB	29.4 kB	262 kB

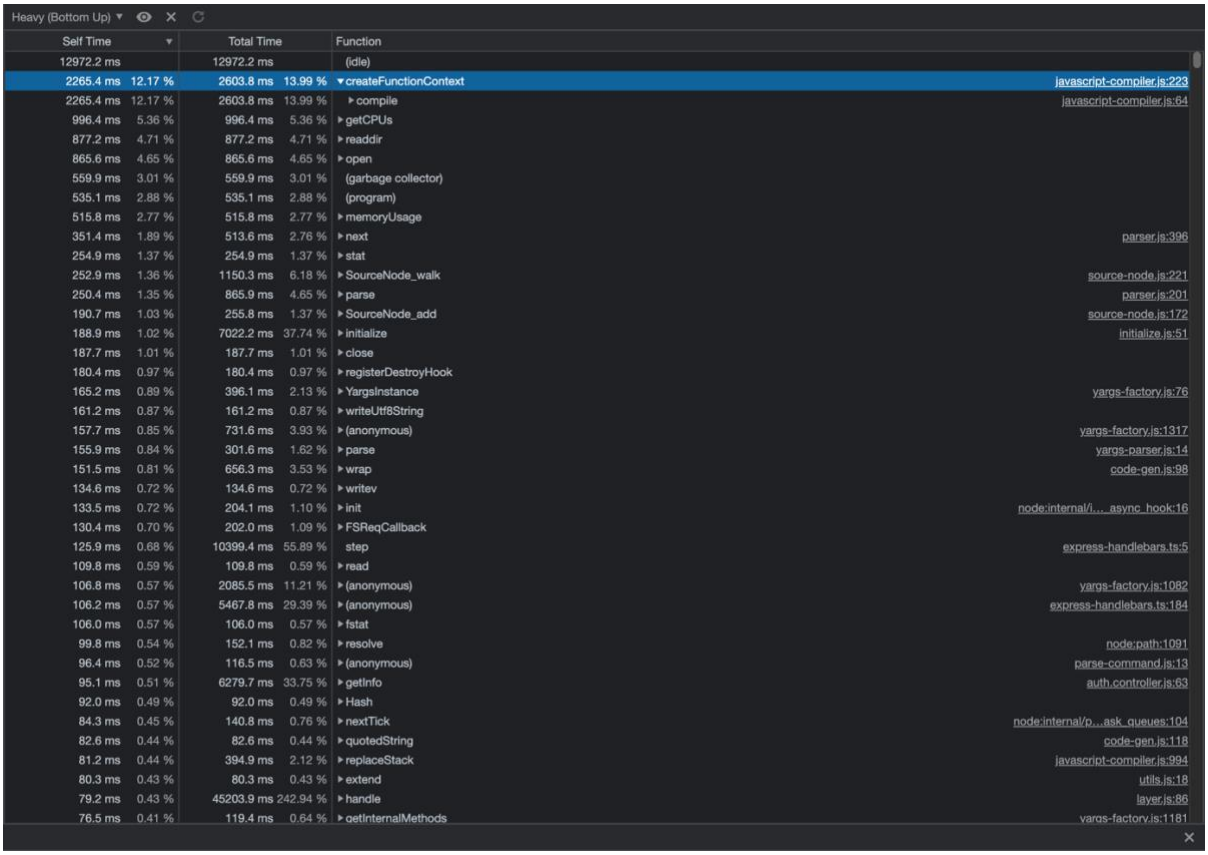
```
Req/Bytes counts sampled once per second.
# of samples: 20

5k requests in 20.04s, 6.6 MB read
```

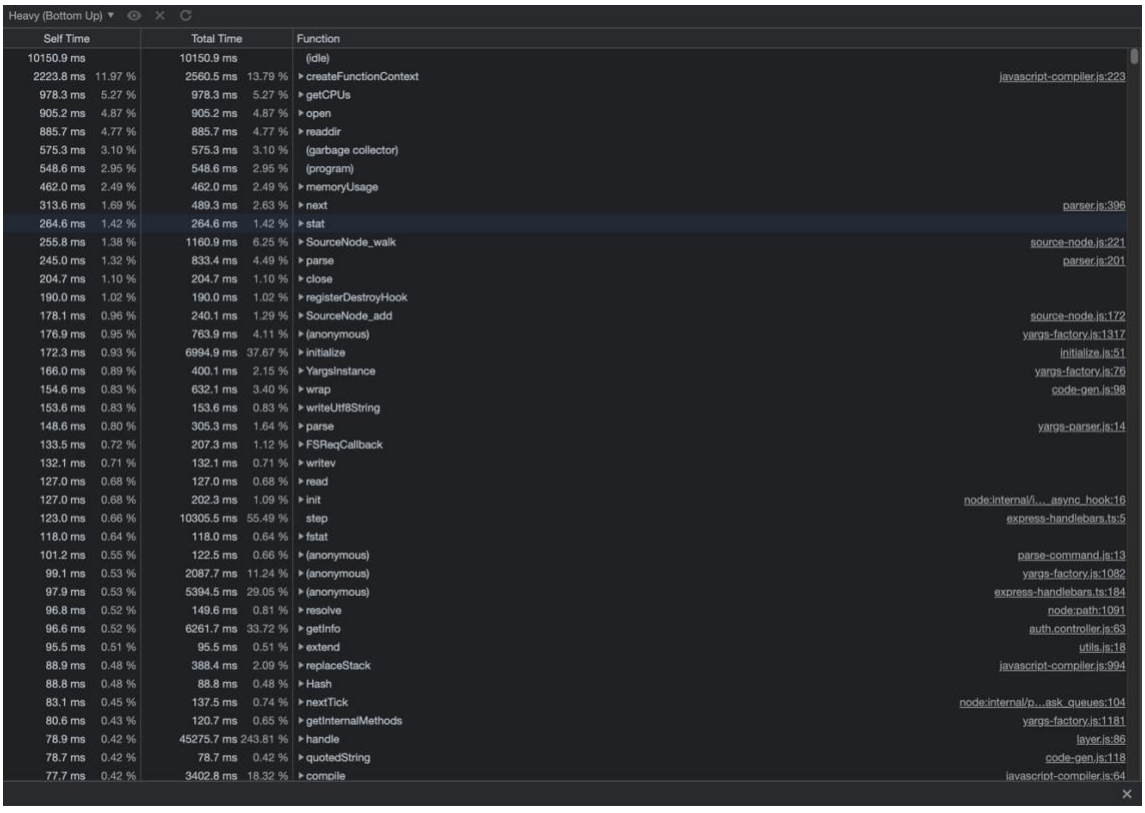
Conclusión: Se observa que el método info que tiene la operación adicional console.log puede atender menos requests por segundos y puede trasmitir más información por segundo en línea con lo que se esperaba. Así también tiene una mayor latencia como se esperaba.

4) El perfilamiento del servidor con el modo inspector de node.js --inspect. Revisar el tiempo de los procesos menos performantes sobre el archivo fuente de inspección.

Resultado método info sin console log:



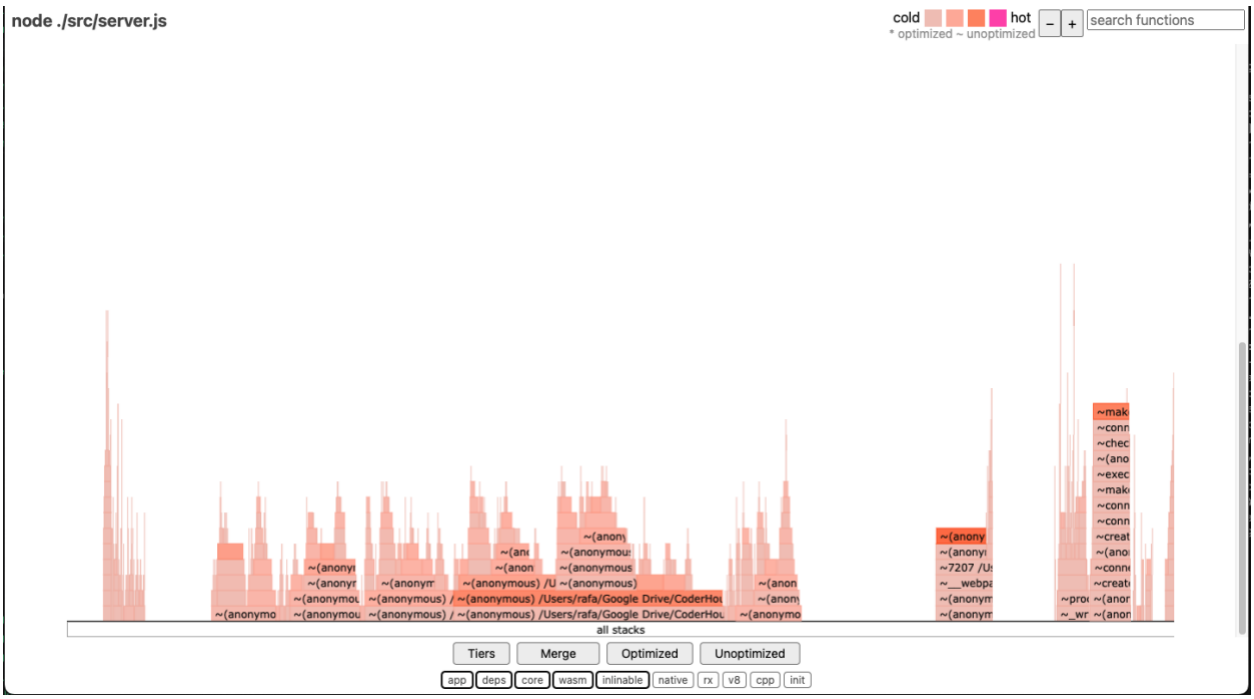
Resultado método info con console log:



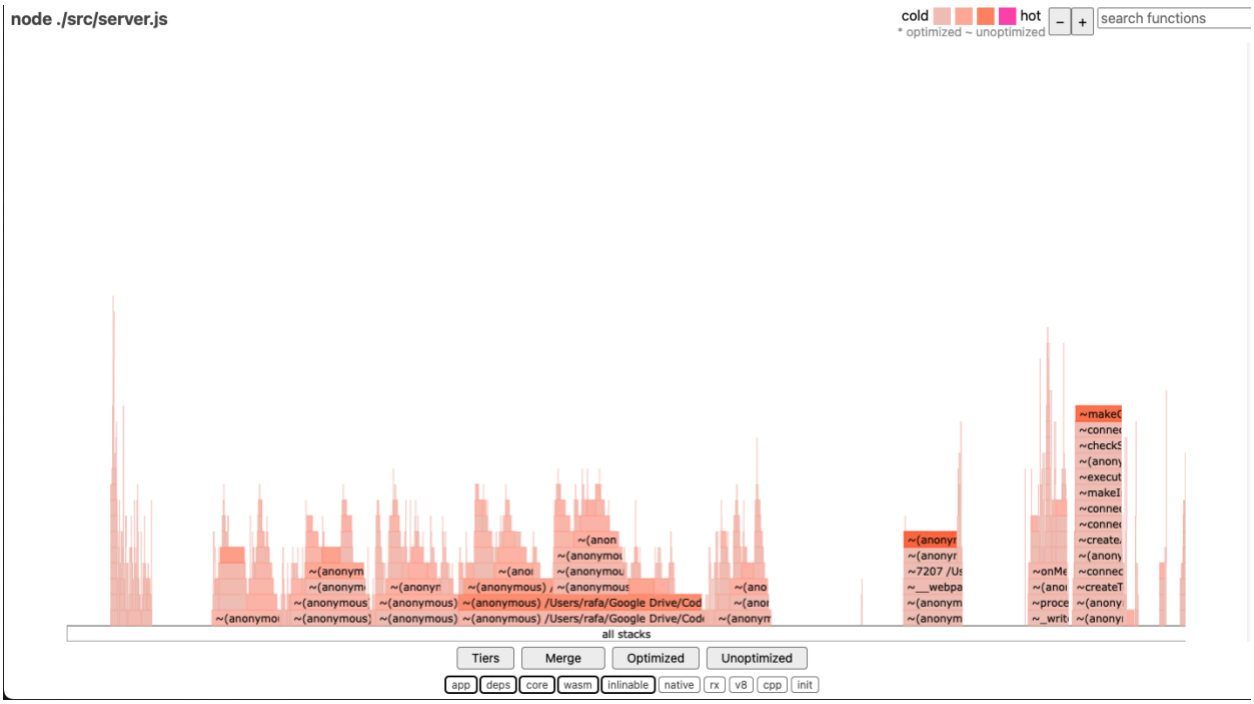
Conclusión: Se observa que el código que más demora en ejecutar lo hace en menos tiempo en el caso donde se imprime por consola el resultado del método info. No se observa/identifica dentro del listado del este profile un tiempo significativo a la impresión del resultado en el segundo caso.

5) El diagrama de flama con 0x, emulando la carga con Autocannon con los mismos parámetros anteriores.

Resultado método info sin console log:



Resultado método info con console log:



Conclusión: Luego de una inspección visual no se observan mayores diferencias entre el código que imprime en consola el resultado del método info con el que no lo hace. Se hubiese esperado ver alguna diferencia en la cantidad de peaks o la duración del stack total.