

Entrega 16 - Coderhouse

Compresión:

```
/info sin comprimir: 621B  
/info con compresión: 640B
```

Esto se debe a que gzip tiene un tamaño mínimo de archivo a la hora de comprimir y en archivos tan pequeños puede llevar a este caso. Si lo enviado en /info fuera de 1KB, por ejemplo, sí que se notaría la compresión.

Performance con --prof de node.js y artillery:

Con console.log:

```
[Shared libraries]:  
  ticks  total  nonlib   name  
  4412   65.0%           C:\WINDOWS\SYSTEM32\ntdll.dll  
  2290   33.7%           C:\Program Files\nodejs\node.exe  
    3    0.0%           C:\WINDOWS\System32\KERNELBASE.dll  
    2    0.0%           C:\WINDOWS\System32\WS2_32.dll  
    2    0.0%           C:\WINDOWS\System32\KERNEL32.DLL
```

Sin console.log:

```
[Shared libraries]:  
  ticks  total  nonlib   name  
  4074   68.7%           C:\WINDOWS\SYSTEM32\ntdll.dll  
  1792   30.2%           C:\Program Files\nodejs\node.exe  
    2    0.0%           C:\WINDOWS\System32\KERNELBASE.dll
```

Se puede apreciar que, con tan sólo un console.log el uso de node ha sido un 3,5% mayor.

En las carpetas Diagrama y DiagramaConsole se pueden observar los gráficos y, bajo mi punto de vista, destaca el proceso compression en ambos.



Diagrama de consola

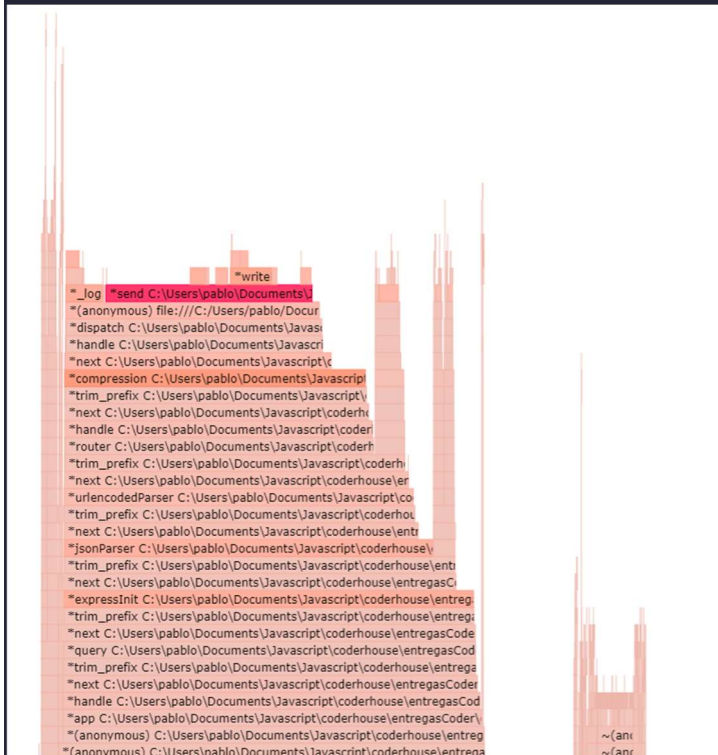


Diagrama sin consola

En ambos además se observa que el proceso "send" es un punto crítico, pero que el proceso es más corto sin la consola.

Usando el insect y las tolos de Chrome, vemos el perfilado es mucho más claro y la consola consume mucho más

Self Time	Total Time	Function
12099.7 ms	12099.7 ms	(idle)
8937.5 ms 46.25 %	15184.6 ms 78.58 %	▶ consoleCall
5640.3 ms 29.19 %	5640.3 ms 29.19 %	▶ writeUtf8String
343.0 ms 1.77 %	343.0 ms 1.77 %	▶ (garbage collector)
275.1 ms 1.42 %	275.1 ms 1.42 %	▶ writev
215.5 ms 1.12 %	215.5 ms 1.12 %	▶ getColorsDepth
161.0 ms 0.83 %	161.0 ms 0.83 %	▶ (program)
113.1 ms 0.59 %	17533.1 ms 90.74 %	▶ compression
106.6 ms 0.55 %	12303.5 ms 63.56 %	▶ (anonymous)
103.6 ms 0.54 %	180.7 ms 0.93 %	▶ nextTick
88.1 ms 0.46 %	1722.6 ms 8.91 %	▶ send
81.0 ms 0.42 %	92.2 ms 0.48 %	▶ asString
68.6 ms 0.35 %	119.1 ms 0.62 %	▶ writeHead
68.0 ms 0.35 %	22348.9 ms 69.89 %	▶ next
67.6 ms 0.35 %	40298.3 ms 728.35 %	▶ handle
67.2 ms 0.35 %	75.4 ms 0.39 %	▶ parse
64.7 ms 0.33 %	64.7 ms 0.33 %	▶ setWriteHeader

Sin la consola:

Self Time	Total Time	Function
23170.8 ms	23170.8 ms	(idle)
10151.9 ms 52.27 %	12770.0 ms 65.75 %	▶ consoleCall
2012.3 ms 10.36 %	2012.3 ms 10.36 %	▶ writeUtf8String
579.4 ms 2.98 %	579.4 ms 2.98 %	▶ (garbage collector)
469.5 ms 2.42 %	469.5 ms 2.42 %	▶ writev
266.7 ms 1.37 %	266.7 ms 1.37 %	▶ (program)
211.1 ms 1.09 %	211.1 ms 1.09 %	▶ getColorsDepth
189.6 ms 0.98 %	16483.5 ms 64.87 %	▶ compression
163.7 ms 0.84 %	288.5 ms 1.49 %	▶ nextTick
151.1 ms 0.78 %	2728.3 ms 14.05 %	▶ send
145.9 ms 0.75 %	165.8 ms 0.85 %	▶ asString
132.1 ms 0.68 %	218.1 ms 1.12 %	▶ writeHead
129.6 ms 0.67 %	16933.9 ms 87.19 %	▶ expressInit
114.7 ms 0.59 %	16789.4 ms 60.13 %	▶ next
113.1 ms 0.58 %	16105.5 ms 62.92 %	▶ (anonymous)
112.9 ms 0.58 %	112.9 ms 0.58 %	▶ setWriteHeader
106.6 ms 0.55 %	125.4 ms 0.65 %	▶ parse
95.0 ms 0.49 %	32664.1 ms 683.14 %	▶ handle
86.3 ms 0.44 %	1063.3 ms 5.47 %	▶ end
77.2 ms 0.40 %	90.2 ms 0.46 %	▶ storeHeader
74.8 ms 0.39 %	74.8 ms 0.39 %	▶ Hash
73.1 ms 0.38 %	159.9 ms 0.82 %	▶ header

Es decir, con consola la función consoleCall usa un 78,58% del tiempo y sin console.log, nos vamos a un 65,75%.