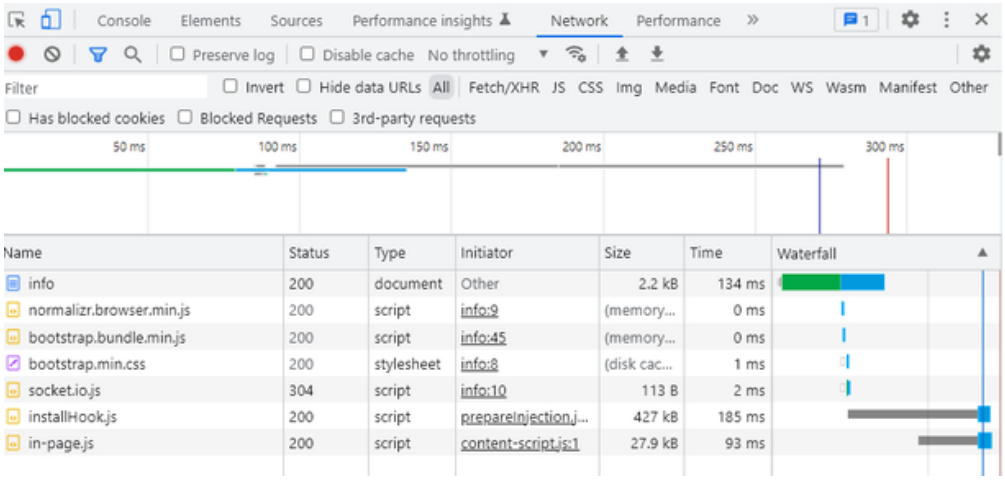
Desafío: Loggers, gzip y análisis de performance

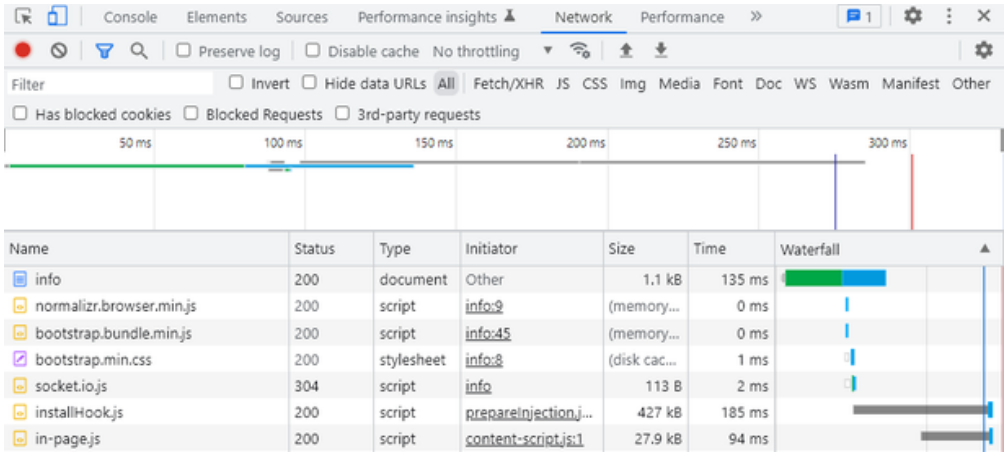
Busato Gabriel

compresión:

Sin comprimir: 2.2kb



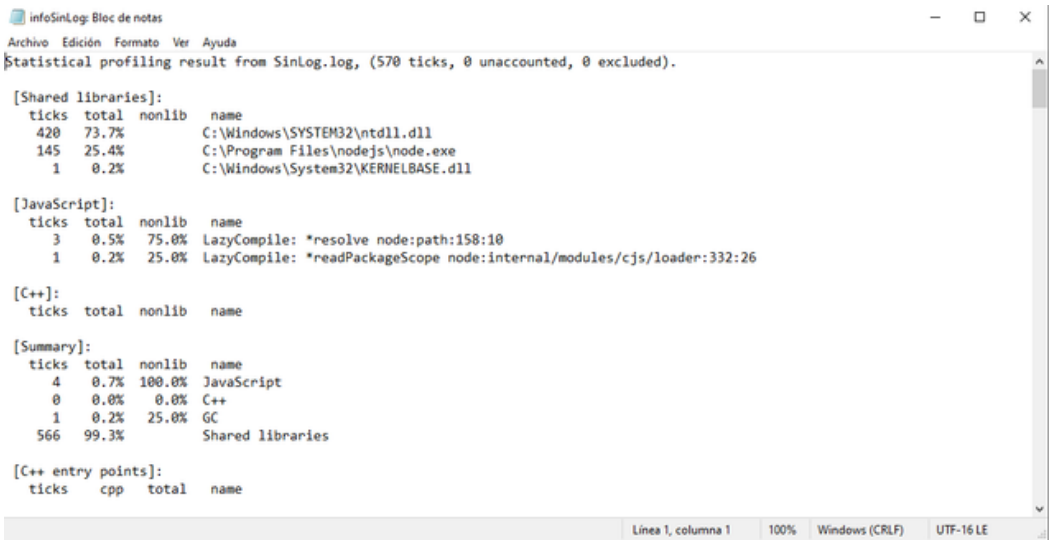
Con compresión: 1.1kb



Con compresión la carga de "/info" es un 50% mas liviana para el servidor .

--Prof:

SIN log: 570 ticks



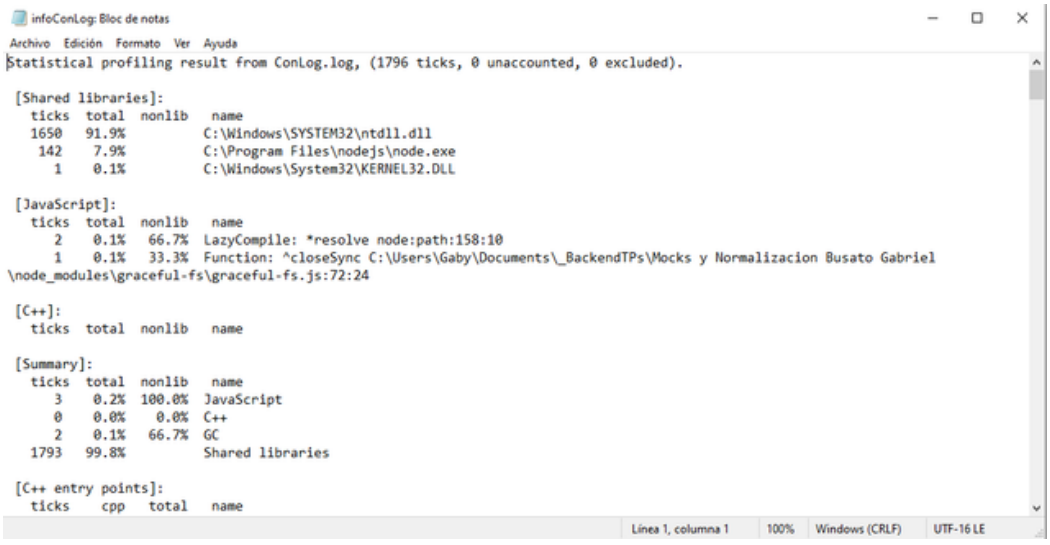
infoSinLog: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
Statistical profiling result from SinLog.log, (570 ticks, 0 unaccounted, 0 excluded).

 [Shared libraries]:
   ticks  total  nonlib   name
     420   73.7%          C:\Windows\SYSTEM32\ntdll.dll
     145   25.4%          C:\Program Files\nodejs\node.exe
       1    0.2%          C:\Windows\System32\KERNELBASE.dll

 [JavaScript]:
   ticks  total  nonlib   name
       3    0.5%   75.0%  LazyCompile: *resolve node:path:158:10
       1    0.2%   25.0%  LazyCompile: *readPackageScope node:internal/modules/cjs/loader:332:26

 [C++]:
   ticks  total  nonlib   name

 [Summary]:
   ticks  total  nonlib   name
       4    0.7%  100.0%  JavaScript
       0    0.0%    0.0%  C++
       1    0.2%   25.0%  GC
     566   99.3%          Shared libraries

 [C++ entry points]:
   ticks    cpp   total   name
```

Línea 1, columna 1 | 100% | Windows (CRLF) | UTF-16 LE

CON log: 1796 ticks



infoConLog: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
Statistical profiling result from ConLog.log, (1796 ticks, 0 unaccounted, 0 excluded).

 [Shared libraries]:
   ticks  total  nonlib   name
    1650   91.9%          C:\Windows\SYSTEM32\ntdll.dll
     142    7.9%          C:\Program Files\nodejs\node.exe
       1    0.1%          C:\Windows\System32\KERNEL32.DLL

 [JavaScript]:
   ticks  total  nonlib   name
       2    0.1%   66.7%  LazyCompile: *resolve node:path:158:10
       1    0.1%   33.3%  Function: ^closeSync C:\Users\Gaby\Documents\_BackendTPs\Mocks y Normalizacion Busato Gabriel
\node_modules\graceful-fs\graceful-fs.js:72:24

 [C++]:
   ticks  total  nonlib   name

 [Summary]:
   ticks  total  nonlib   name
       3    0.2%  100.0%  JavaScript
       0    0.0%    0.0%  C++
       2    0.1%   66.7%  GC
    1793   99.8%          Shared libraries

 [C++ entry points]:
   ticks    cpp   total   name
```

Línea 1, columna 1 | 100% | Windows (CRLF) | UTF-16 LE

Se aprecia que sin el console.log baja sustancialmente la cantidad de pedidos al servidor.

Artillery

SIN log media: 788.5

```
Metrics for period to: 15:13:00(-0300) (width: 1.599s)
-------------------------------------

errors.ECONNREFUSED: ..................................................... 187
http.codes.200: .......................................................... 813
http.request_rate: ....................................................... 990/sec
http.requests: ........................................................... 990
http.response_time:
  min: ................................................................... 29
  max: ................................................................... 929
  median: ................................................................ 788.5
  p95: ................................................................... 907
  p99: ................................................................... 925.4
http.responses: .......................................................... 813
vusers.created: .......................................................... 990
vusers.created_by_name.0: ................................................ 990
vusers.failed: ........................................................... 187
```

CON log media: 820.7

```
Metrics for period to: 15:11:50(-0300) (width: 1.59s)
-------------------------------------

errors.ECONNREFUSED: ..................................................... 234
http.codes.200: .......................................................... 766
http.request_rate: ....................................................... 990/sec
http.requests: ........................................................... 1000
http.response_time:
  min: ................................................................... 102
  max: ................................................................... 939
  median: ................................................................ 820.7
  p95: ................................................................... 925.4
  p99: ................................................................... 944
http.responses: .......................................................... 766
vusers.created: .......................................................... 1000
vusers.created_by_name.0: ................................................ 1000
vusers.failed: ........................................................... 234
```

Sin el log, la media de respuesta del servidor es mas rapida por 40 mls.

Autocannon:

Sin Log

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	500	500	923	998	900.4	104.67	500
Bytes/Sec	1.11 MB	1.11 MB	2.05 MB	2.22 MB	2 MB	233 kB	1.11 MB

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

18k requests in 20.09s, 40.1 MB read
PS C:\Users\Gaby\Documents\_BackendTPs\Mocks y Normalizacion Busato Gabriel>
```

Con Log

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	90 ms	105 ms	149 ms	163 ms	109.78 ms	17.23 ms	276 ms

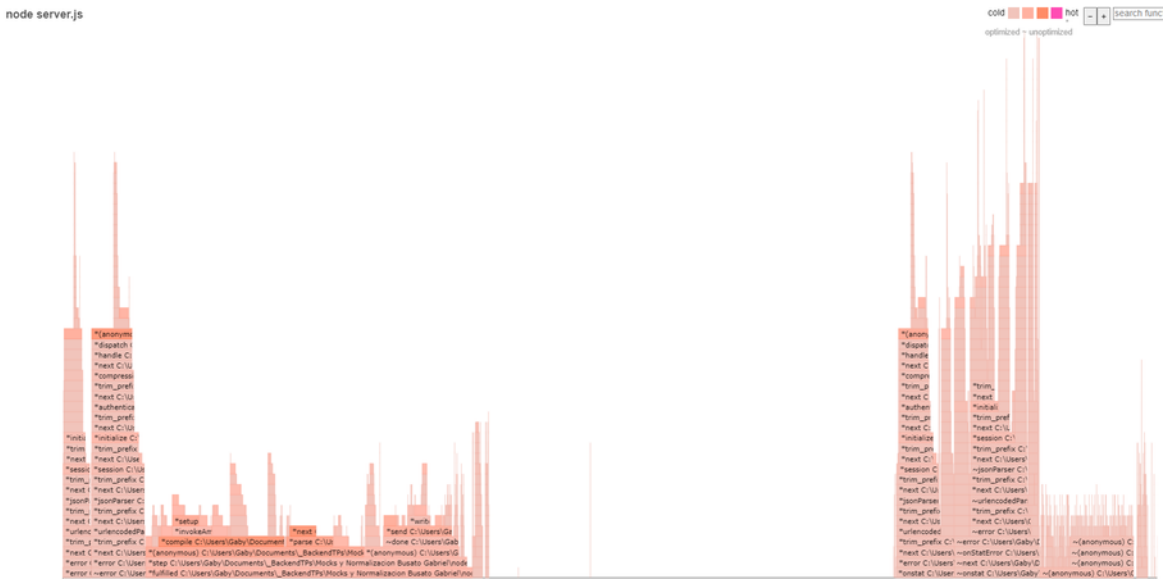
Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	528	528	909	994	906.75	104.66	528
Bytes/Sec	1.17 MB	1.17 MB	2.02 MB	2.21 MB	2.02 MB	233 kB	1.17 MB

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

18k requests in 20.09s, 40.3 MB read
PS C:\Users\Gaby\Documents\_BackendTPs\Mocks y Normalizacion Busato Gabriel>
```

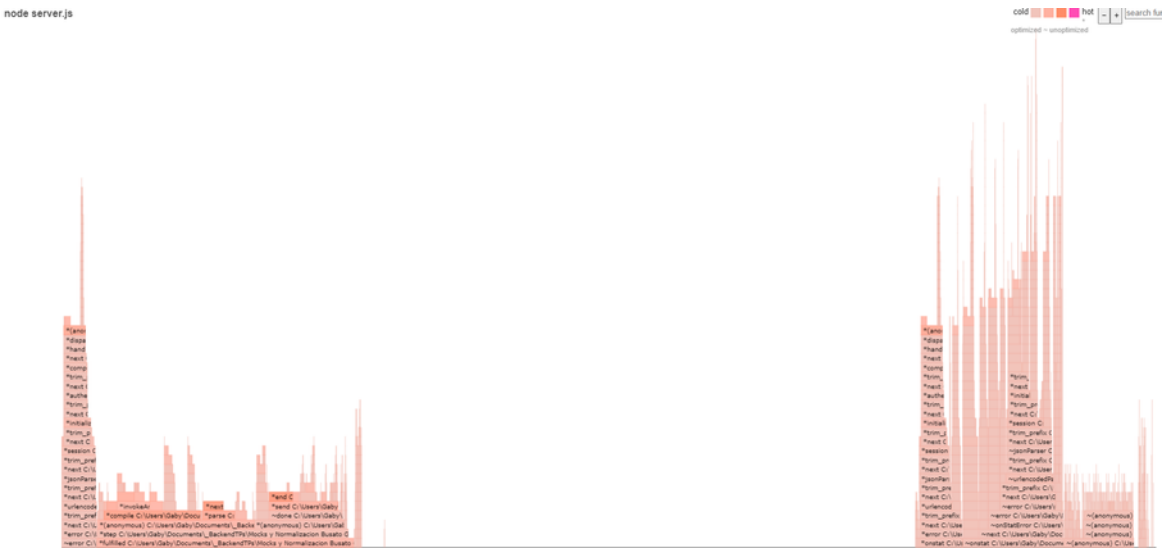
Se aprecia una marcada suba de uso de datos teniendo el console.log() en el codigo.

0x:

Sin Log



Con Log



Como conclusion final, se nota de manera marcada, que incluso en un programa tan simple como este el uso de procesos bloqueantes genera un letargo en el resto de los procesos y por lo tal ha de ser evitado en el ambito profesional.