Desafío 16

Loggers, Gzip y análisis de performance

Nancy Martinez | Programación Backend- Comisión 32115 | Febrero 2023

# Consigna 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.

Luego implementar loggueo (con alguna librería vista en clase) que registre lo siguiente:

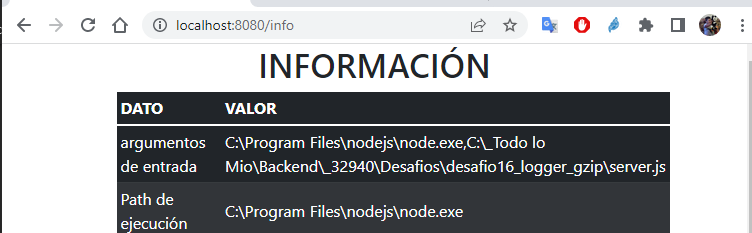
* Ruta y método de todas las peticiones recibidas por el servidor (info)
* Ruta y método de las peticiones a rutas inexistentes en el servidor (warning)
* Errores lanzados por las apis de mensajes y productos, únicamente (error)

Considerar el siguiente criterio:

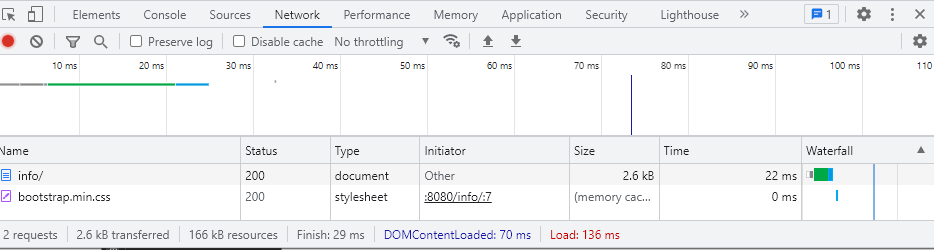
* Loggear todos los niveles a consola (info, warning y error)
* Registrar sólo los logs de warning a un archivo llamada warn.log
* Enviar sólo los logs de error a un archivo llamada error.log

## Resultados:

Ejecución de info de manera habitual: Se observan 2.6KB transferidos.



Ejecución Normal. 1



Ejecución normal. 2

Ejecución de info con compresión Gzip: se observan 129 B.

Captura de pantalla de un celular

Descripción generada automáticamente

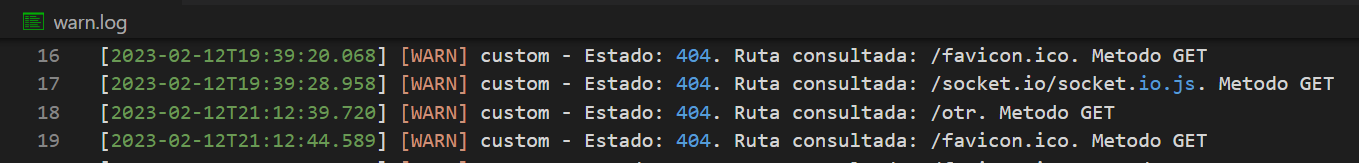
Ejecución con compresión Gzip 1

Tabla

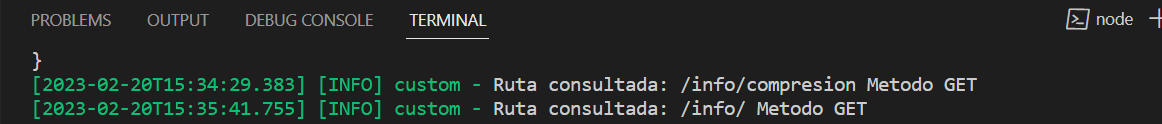
Descripción generada automáticamente

Ejecución con compresión Gzip 2

Logs en archivos:



Logs en consola:



# Consigna 2

Luego, realizar el análisis completo de performance del servidor con el que venimos trabajando.

Vamos a trabajar sobre la ruta '/info', en modo fork, agregando ó extrayendo un console.log de la información colectada antes de devolverla al cliente. Además desactivaremos el child\_process de la ruta '/randoms'

Para ambas condiciones (con o sin console.log) en la ruta '/info' OBTENER:

1) 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.

## Ejecucion 1:

**Terminal1:** node --prof server.js

**Terminal2**: artillery quick --count 50 --num 20 --output infoConConsoleLog.txt 'http://localhost:8080/info'

**Finalmente:** node --prof-process isolate-conConsole-v8.log > resultProfConConsoleLog.txt

Artillery: infoConConsolelog en panntalla

Metrics for period to: 01:15:30(-0300) (width: 3.934s)

--------------------------------------

http.codes.200: ................................................................ 603

http.request\_rate: ............................................................. 167/sec

http.requests: ................................................................. 653

http.response\_time:

min: ......................................................................... 181

max: ......................................................................... 339

median: ...................................................................... 257.3

p95: ......................................................................... 314.2

p99: ......................................................................... 327.1

http.responses: ................................................................ 603

vusers.created: ................................................................ 50

vusers.created\_by\_name.0: ...................................................... 50

--------------------------------------

Metrics for period to: 01:15:40(-0300) (width: 1.854s)

--------------------------------------

http.codes.200: ................................................................ 397

http.request\_rate: ............................................................. 191/sec

http.requests: ................................................................. 347

http.response\_time:

min: ......................................................................... 31

max: ......................................................................... 352

median: ...................................................................... 242.3

p95: ......................................................................... 327.1

p99: ......................................................................... 333.7

http.responses: ................................................................ 397

vusers.completed: .............................................................. 50

vusers.failed: ................................................................. 0

vusers.session\_length:

min: ......................................................................... 4828

max: ......................................................................... 5597.2

median: ...................................................................... 5168

p95: ......................................................................... 5487.5

p99: ......................................................................... 5487.5

All VUs finished. Total time: 7 seconds

--------------------------------

Summary report @ 01:15:33(-0300)

--------------------------------

http.codes.200: ................................................................ 1000

http.request\_rate: ............................................................. 179/sec

http.requests: ................................................................. 1000

http.response\_time:

min: ......................................................................... 31

max: ......................................................................... 352

median: ...................................................................... 257.3

p95: ......................................................................... 320.6

p99: ......................................................................... 333.7

http.responses: ................................................................ 1000

vusers.completed: .............................................................. 50

vusers.created: ................................................................ 50

vusers.created\_by\_name.0: ...................................................... 50

vusers.failed: ................................................................. 0

vusers.session\_length:

min: ......................................................................... 4828

max: ......................................................................... 5597.2

median: ...................................................................... 5168

p95: ......................................................................... 5487.5

p99: ......................................................................... 5487.5

Log file: infoConConsoleLog.txt

Prof : resultProfConConsolelog.txt

Texto

Descripción generada automáticamente

…

Texto

Descripción generada automáticamente

## Ejecucion 2:

**Terminal1:** node --prof server.js

**Terminal2**: artillery quick --count 50 --num 20 --output infoSinConsoleLog.txt 'http://localhost:8080/info'

**Finalmente:** node --prof-process isolate-sinConsole-v8.log > resultProfSinConsoleLog.txt

Artillery: infoSinConsoleLog en pantalla

--------------------------------------

Metrics for period to: 01:54:10(-0300) (width: 3.745s)

--------------------------------------

http.codes.200: ................................................................ 1000

http.request\_rate: ............................................................. 269/sec

http.requests: ................................................................. 1000

http.response\_time:

min: ......................................................................... 8

max: ......................................................................... 277

median: ...................................................................... 156

p95: ......................................................................... 190.6

p99: ......................................................................... 252.2

http.responses: ................................................................ 1000

vusers.completed: .............................................................. 50

vusers.created: ................................................................ 50

vusers.created\_by\_name.0: ...................................................... 50

vusers.failed: ................................................................. 0

vusers.session\_length:

min: ......................................................................... 2765

max: ......................................................................... 3584.6

median: ...................................................................... 3134.5

p95: ......................................................................... 3534.1

p99: ......................................................................... 3534.1

All VUs finished. Total time: 5 seconds

--------------------------------

Summary report @ 01:54:06(-0300)

--------------------------------

http.codes.200: ................................................................ 1000

http.request\_rate: ............................................................. 269/sec

http.requests: ................................................................. 1000

http.response\_time:

min: ......................................................................... 8

max: ......................................................................... 277

median: ...................................................................... 156

p95: ......................................................................... 190.6

p99: ......................................................................... 252.2

http.responses: ................................................................ 1000

vusers.completed: .............................................................. 50

vusers.created: ................................................................ 50

vusers.created\_by\_name.0: ...................................................... 50

vusers.failed: ................................................................. 0

vusers.session\_length:

min: ......................................................................... 2765

max: ......................................................................... 3584.6

median: ...................................................................... 3134.5

p95: ......................................................................... 3534.1

p99: ......................................................................... 3534.1

Log file: infosinConsoleLog.txt

Prof: resultProfSinConsoleLog.txt

Una captura de pantalla de una computadora

Descripción generada automáticamente

…

Texto

Descripción generada automáticamente

### Conclusión de la comparación:

Se observan llamadas a librerias de interpretacion de expresiones regulares en la ejecución con consoleLog únicamente. No se observan mayores ventajas en la cantidad de ticks.

Sin embargo desde el Artillery se observan 7 seg en total de tardanza en la prueba con logs contra 5 seg de tardanza para la prueg sin Logs. También diferencia en los tiempos mínimos y máximos de respuesta.

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: El autocannon no lo pude hacer funcionar

2) 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.

## Ejecucion

**Terminal1:** node --inspect server.js

**Crhome:** chrome://inspect/ --> Open dedicated DevTools for Node

**Terminal2**: artillery quick --count 50 --num 20 --output infoConConsoleLog.txt 'http://localhost:8080/info'

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteTexto

Descripción generada automáticamente con confianza media

### Conclusión de la ejecución

El mayor tiempo global se consume con las escrituras. En particular las escrituras que se encontraron a log, son mayores a lo estipulado.

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

Resultado: El autocannon no lo pude hacer funcionar