Puntos del desafío

Consigna parte 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.

- Prueba del endpoint sin compresión: http://localhost:8080/info
- Prueba del endpoint con gzip: http://localhost:8080/info/gzip

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)

Consigna parte 2:

---

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.

PRUEBAS CON ARTILLERY EN MODO FORK Y CLUSTER:

- Al endpoint info: localhost:8080/info
- 50 conexiones concurrentes con 20 peticiones cada una.
- Reportes en los archivos: result_fork.txt y result_cluster.txt

node server.js -p 8080 -m FORK

artillery quick -c 50 -n 20 "http://localhost:8080/info" > result_fork.txt

result_fork.txt U

challenge14 > result_fork.txt

```
Running scenarios...
Phase started: unnamed (index: 0, duration: 1s) 01:06:22(-0500)

Phase completed: unnamed (index: 0, duration: 1s) 01:06:23(-0500)

All VUs finished. Total time: 5 seconds

--------------------------------
Summary report @ 01:06:26(-0500)
--------------------------------

http.codes.200: ..................................................................... 1000
http.request_rate: .................................................................. 171/sec
http.requests: ...................................................................... 1000
http.response_time:
  min: .............................................................................. 4
  max: .............................................................................. 330
  median: ........................................................................... 98.5
  p95: .............................................................................. 198.4
  p99: .............................................................................. 214.9
http.responses: ..................................................................... 1000
vusers.completed: ................................................................... 50
vusers.created: ..................................................................... 50
vusers.created_by_name.0: ........................................................... 50
vusers.failed: ...................................................................... 0
vusers.session_length:
  min: .............................................................................. 1051
  max: .............................................................................. 2306
  median: ........................................................................... 2101.1
  p95: .............................................................................. 2276.1
  p99: .............................................................................. 2276.1

```

node server.js -p 8080 -m CLUSTER

artillery quick -c 50 -n 20 "http://localhost:8080/info" > result_cluster.txt

result_cluster.txt U

challenge14 > result_cluster.txt

```
Running scenarios...
Phase started: unnamed (index: 0, duration: 1s) 01:05:46(-0500)

Phase completed: unnamed (index: 0, duration: 1s) 01:05:47(-0500)

All VUs finished. Total time: 10 seconds

--------------------------------
Summary report @ 01:05:55(-0500)
--------------------------------

http.codes.200: ..................................................................... 1000
http.request_rate: .................................................................. 89/sec
http.requests: ...................................................................... 1000
http.response_time:
  min: .............................................................................. 2
  max: .............................................................................. 696
  median: ........................................................................... 85.6
  p95: .............................................................................. 361.5
  p99: .............................................................................. 608
http.responses: ..................................................................... 1000
vusers.completed: ................................................................... 50
vusers.created: ..................................................................... 50
vusers.created_by_name.0: ........................................................... 50
vusers.failed: ...................................................................... 0
vusers.session_length:
  min: .............................................................................. 1078.8
  max: .............................................................................. 2849.6
  median: ........................................................................... 2515.5
  p95: .............................................................................. 2725
  p99: .............................................................................. 2780
```

PRUEBAS SOBRE PROFILING:

- Manejado en consola.

1. Ejecutamos el server y realizamos test con artillery

```
node --prof server.js
artillery quick -c 50 -n 20 "http://localhost:8080/info" > artillery_slow.txt

node --prof-process slow-v8.log > prof_slow.txt
```

```
Statistical profiling result from slow-v8.log, (991 ticks, 0 unaccounted, 0 excluded).

 [Shared libraries]:
   ticks  total  nonlib   name
     745   75.2%          C:\WINDOWS\SYSTEM32\ntdll.dll
     236   23.8%          C:\Program Files\nodejs\node.exe

 [JavaScript]:
   ticks  total  nonlib   name
       4    0.4%   40.0%  LazyCompile: *resolve node:path:158:10
       1    0.1%   10.0%  LazyCompile: *readPackageScope node:internal/modules/cjs/loader:321:26
       1    0.1%   10.0%  LazyCompile: *nextPart node:fs:2386:31
       1    0.1%   10.0%  LazyCompile: *compileFunction node:vm:308:25
       1    0.1%   10.0%  Function: ^realpathSync node:fs:2410:22
       1    0.1%   10.0%  Function: ^nativeModuleRequire node:internal/bootstrap/loaders:332:29
       1    0.1%   10.0%  Function: ^getOptions node:internal/fs/utils:314:20

 [C++]:
   ticks  total  nonlib   name

 [Summary]:
   ticks  total  nonlib   name
```

- Usando la devtools de node

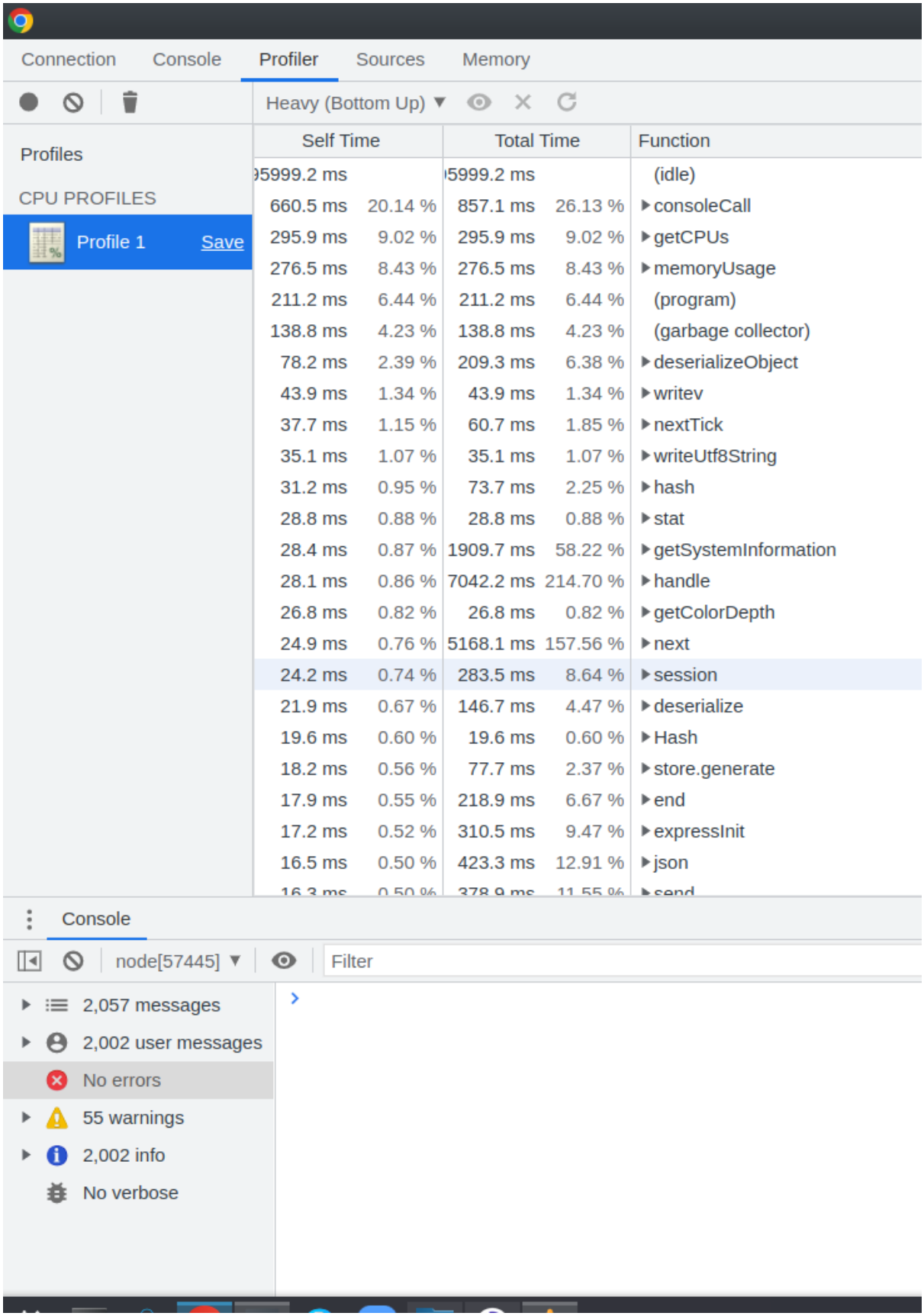
1. Ejecutamos el server en modo inspect

```
node --inspect server.js
```

2. En el navegador abrimos las devtool de node con: chrome://inspect

3. Realizamos test con artillery

```
artillery quick -c 50 -n 20 "http://localhost:8080/info" > artillery_slow.txt
```



Consigna parte 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

PRUEBAS CON AUTOCANNON

- Al endpoint http://localhost:8080/info

1. Código de test
2. Ejecutar npm run start.
3. ejecutar el npm run test.

```
> node benchmark.js

Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8080/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	167 ms	227 ms	371 ms	382 ms	239.89 ms	58.81 ms	453 ms

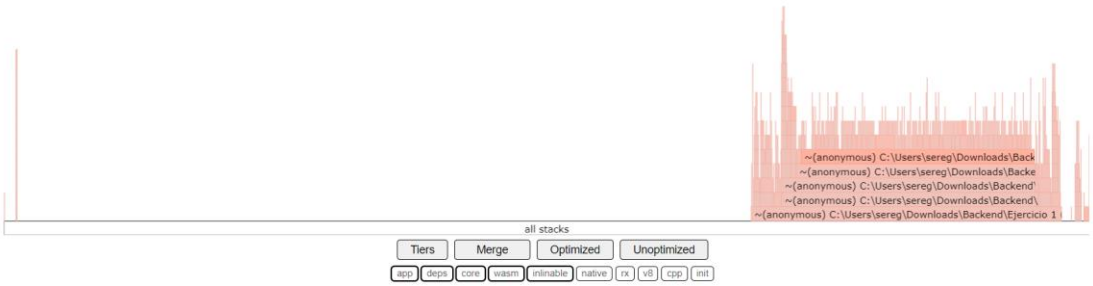
Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	306	306	413	555	414.65	77.62	306
Bytes/Sec	208 kB	208 kB	281 kB	377 kB	282 kB	52.7 kB	208 kB

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

8k requests in 20.06s, 5.63 MB read
```

node server.js -p 8080

cold hot
* optimized ~ unoptimized
- +
search functions



Tiers | Merge | Optimized | Unoptimized

app | deps | core | wasm | inlinable | native | rx | v8 | cpp | init