Análisis

Pruebas realizadas con artillery 50 conexiones con 20 peticiones cada una:

- A la ruta /infoLog con compresión y sin compresión. Con el console log.
- A la ruta /infoNoLog con compresión y sin compresión. Sin console log.

Para /infoLog con compresión en comparación a sin compresión hay una diferencia muy grande en cuanto al rendimiento.

1. Ambos tuvieron 1000 peticiones de status 200
2. Requests Rate con compresión es mucho mayor que sin compresión. 266/sec para el primero y 221/sec para el segundo.
3. Las velocidades de respuesta fueron más rápidas que las de sin compresión. 18 de mínima contra 5 de mínima
4. La compresión obtuvo 2295 ticks 99.4% y la no-Compresión 3362 99.6% que muestra la diferencia de consumo de recursos.

La misma lógica de análisis aplica para la ruta de noLog siendo que noLog sin compresión tomo 1708 ticks (mucho menos unidades de procesamiento) y 4686 para compresión y sin console Log.

Node inspect

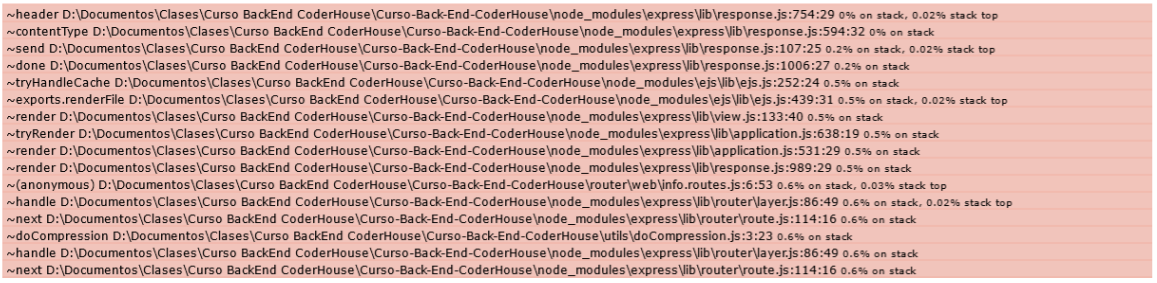
```
                  infoRoute.get("/infoLog/:compress?", doCompression, (req, res) => {
0.3 ms              const info = {
1.4 ms                directorio: process.cwd(),
0.8 ms                memory: process.memoryUsage(),
0.3 ms                processId: process.pid,
0.1 ms                node: process.version,
0.3 ms                os: process.platform,
0.2 ms                CPUs,
0.5 ms                inputArgs: process.argv,
0.3 ms                path: process.execPath,
                    };
5.0 ms              console.log(info);
22.7 ms             res.status(200).render("info.ejs", {info});
                  });

                  infoRoute.get("/infoNoLog/:compress?", doCompression, (req, res) => {
0.5 ms              const info = {
0.8 ms                directorio: process.cwd(),
1.1 ms                memory: process.memoryUsage(),
0.2 ms                processId: process.pid,
0.1 ms                node: process.version,
                      os: process.platform,
                      CPUs,
0.3 ms                inputArgs: process.argv,
0.2 ms                path: process.execPath,
                    };

4.0 ms              res.status(200).render("info.ejs", {info});
                  });
```

La prueba que se realizó fue entre compresión y no compresión con log y sin log. Lo principal acá es ver que el rendimiento para enviar la petición al usuario con una compresión de por medio toma más del 100% de tiempo. 22.7ms para compresión y 4.0ms para la no compresión. Para ambos casos los puntos críticos son el envío de información al usuario.

FlameGraph y autocannon



```
~header D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\node_modules\express\lib\response.js:754:29 0% on stack, 0.02% stack top
~contentType D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\node_modules\express\lib\response.js:594:32 0% on stack
~send D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\node_modules\express\lib\response.js:107:25 0.2% on stack, 0.02% stack top
~done D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\node_modules\express\lib\response.js:1006:27 0.2% on stack
~tryHandleCache D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\node_modules\ejs\lib\ejs.js:252:24 0.5% on stack
~exports.renderFile D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\node_modules\ejs\lib\ejs.js:439:31 0.5% on stack, 0.02% stack top
~render D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\node_modules\express\lib\view.js:133:40 0.5% on stack
~tryRender D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\node_modules\express\lib\application.js:638:19 0.5% on stack
~render D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\node_modules\express\lib\application.js:531:29 0.5% on stack
~render D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\node_modules\express\lib\response.js:989:29 0.5% on stack
~(anonymous) D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\router\web\info.routes.js:6:53 0.6% on stack, 0.03% stack top
~handle D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\node_modules\express\lib\router\layer.js:86:49 0.6% on stack, 0.02% stack top
~next D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\node_modules\express\lib\router\route.js:114:16 0.6% on stack
~doCompression D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\utils\doCompression.js:3:23 0.6% on stack
~handle D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\node_modules\express\lib\router\layer.js:86:49 0.6% on stack
~next D:\Documentos\Clases\Curso BackEnd CoderHouse\Curso-Back-End-CoderHouse\node_modules\express\lib\router\route.js:114:16 0.6% on stack
```

Como en el otro test, se ve visualmente que la tarea bloqueante más pesada para el servidor es el envío de la respuesta

```
Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8080/infoLog/?compress=true
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	152 ms	259 ms	419 ms	435 ms	266.2 ms	67.96 ms	458 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	202	202	380	582	373.8	88.16	202
Bytes/Sec	403 kB	403 kB	758 kB	1.16 MB	745 kB	176 kB	403 kB

```
8k requests in 20.07s, 14.9 MB read
Running 20s test @ http://localhost:8080/infoLog/
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	152 ms	264 ms	444 ms	464 ms	267.68 ms	70.27 ms	487 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	204	204	374	584	370.2	94.9	204
Bytes/Sec	407 kB	407 kB	745 kB	1.16 MB	738 kB	189 kB	407 kB

No hay mucha diferencia haciendo los test con auto cannon para compresión y sin compresión. Una diferencia de 2 request mas sin compresión.