Consigna

Vamos a trabajar con la ruta “/info” en modo “fork”, agregando o 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

   Se hizo un informe de artillery tanto con console log como sin él, los reportes de artillery resultaron en que en promedio el servidor corriendo con console.log tardó alrededor de una media de 60ms demás.
   En el prof-process de console.log, aparece tener 400 ticks menos que el prof-process sin console.log
2) 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 de pantalla de la consola)
   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

   En la carpeta “/profiling/autocannon” se dejan imágenes de la respuesta con console.log y sin console.log

A continuación se muestra los procesos ejecutados con console.log

```
 0.1 ms  router.get("/info", (req, res) => {
             try {
 7.0 ms          logger.info(`Se accedio a la ruta ${req.originalUrl} con el metodo ${req.method}`)
11.4 ms          console.log({
 2.8 ms              cpus: os.cpus().length,
 3.2 ms              argv: process.argv.slice(2),
 0.5 ms              platform: process.platform,
                     version: process.version,
 2.1 ms              rss: process.memoryUsage(),
 0.4 ms              cwd: process.cwd(),
 0.1 ms              pe: process.execPath,
 1.1 ms              pid: process.pid,
                 });
23.5 ms          res.json({
 2.3 ms              cpus: os.cpus().length,
 5.5 ms              argv: process.argv.slice(2),
 0.8 ms              platform: process.platform,
 0.5 ms              version: process.version,
 0.7 ms              rss: process.memoryUsage(),
 0.7 ms              cwd: process.cwd(),
 0.3 ms              pe: process.execPath,
 0.7 ms              pid: process.pid,
                 })
             } catch (error) {
                 logger.error(`${error.message}`)
                 next(error)
             }
         })
```

Y a continuación el que no tiene console.log

```
15          router.get("/info", (req, res) => {
16              try {
17   5.7 ms         logger.info(`Se accedio a la ruta ${req.originalUrl} con el metodo ${req.method}`)
18                  // console.log({
19                  //     cpus: os.cpus().length,
20                  //     argv: process.argv.slice(2),
21                  //     platform: process.platform,
22                  //     version: process.version,
23                  //     rss: process.memoryUsage(),
24                  //     cwd: process.cwd(),
25                  //     pe: process.execPath,
26                  //     pid: process.pid,
27                  // });
28  17.2 ms         res.json({
29   4.3 ms             cpus: os.cpus().length,
30   4.0 ms             argv: process.argv.slice(2),
31   0.3 ms             platform: process.platform,
32                      version: process.version,
33   2.8 ms             rss: process.memoryUsage(),
34   0.6 ms             cwd: process.cwd(),
35   0.7 ms             pe: process.execPath,
36   0.3 ms             pid: process.pid,
37                  })
38              } catch (error) {
39                  logger.error(`${error.message}`)
40   0.1 ms         next(error)
41              }
42          })
```

Podemos ver una diferencia de más de 15ms de performance

3) El diagrama de flama con 0x, emulando la carga con autocannon con los mismos parámetros anteriores.
   En la carpeta “/profiling/x0” se dejan los diagramas de flama

   En el diagrama podemos ver que los procesos de nuestro proyecto son picos y si son mesetas no son pronunciadas

Conclusión

Podemos confirmar que nuestro proyecto cumple en aspectos de performance, y que cualquier línea de código puede influir en la performance de nuestro proyecto, por esa misma razón hay que considerar que líneas de código realmente se necesitan ejecutar