Desafío 16 - Reporte de performance de un servidor

Alejandro Portaluppi

El objetivo actual es aprender a extraer informes que nos permitan realizar un análisis de performance de un servidor. En particular se extraerán dos informes de la ruta /api/randoms: uno iniciando el servidor en modo fork y otro en modo cluster.

- Modo fork:

Iniciaremos el servidor "a modo de profiling", es decir, que va a estar en constante escucha de cada movimiento que se haga en el servidor.

Ejecutamos en la terminal "node --prof src/app.js" (se inicia el servidor en modo fork por defecto). A partir de este momento se levanta un archivo isole.

Para realizar el test de carga vamos a usar el módulo Artillery con el siguiente comando en otra terminal:

"artillery quick --count 50 -n 20 http://localhost:8080/api/randoms?cant=20000 > test_artillery_fork.txt"

Este código lo que hace es simular 50 usuarios, donde cada uno va a realizar 20 peticiones a la URL solicitada. Los resultados los devuelve en el archivo test_artillery_fork.txt

En dicha URL se realiza un cálculo que requiere mucho esfuerzo computacional: devuelve un objeto que contiene como claves a los números desde el 1 hasta el 1000, y como valores a la cantidad de veces que salió cada uno, considerando "cant" números naturales randoms entre 1 y 1000.

Cuando Artillery haya terminado debemos finalizar el proceso de node con CTRL+C para que poder manipular los datos del archivo isole, cuyo nombre le cambio a "test_profiling_fork.log" para mayor comodidad.

Ejecutamos "node --prof-process test_profiling_fork.log > test_profiling_fork.txt" para que se genere un reporte final en formato txt.

- Modo cluster:

Los pasos a seguir son muy similares.

Ejecutamos en la terminal "node --prof src/app.js --mode CLUSTER" para iniciar el servidor en modo cluster y al mismo tiempo generar varios archivos isole (la cantidad depende de la computadora). En este caso es más de uno ya que estamos en modo cluster y por lo tanto hay varios procesos abiertos.

Al igual que antes, ejecutamos en la terminal:

artillery quick --count 50 -n 20 http://localhost:8080/api/randoms?cant=20000 > test_artillery_cluster.txt

Con el debido cuidado de colocarle un nombre distinto al txt

Finalizamos los procesos y cambiamos los nombres de los archivos isole a "test_profiling_clusterX.log", donde X representa un numero distinto por cada isole. En mi caso son 5, por lo tanto, van desde "test_profiling_cluster1.log" hasta "test_profiling_cluster5.log".

Ejecutamos "node --prof-process test_profiling_clusterX.log > test_profiling_clusterX.txt" varias veces, uno para cada valor de X, para que se generen los reportes finales en formato txt.

Lo que se puede hacer es comparar el archivo

- test_artillery_fork.txt con el test_artillery_cluster.txt

por un lado, y

- test_profiling_fork.txt con los test_profiling_clusterX.txt

por otro. Por ejemplo, en el primer caso de mis archivos se puede observar que las peticiones por segundo son más altas en modo cluster, tal como puede esperarse.

Dos comandos adicionales

"artillery quick --count 50 -n 20 http://localhost:8080/api/randoms?cant=20000 --output test_artillery_fork.json"

Para ver los resultados en formato json. Luego puedes ejecutar

"arrtillery report test_artillery_fork.json -o test_artillery_fork.html"

saca un reporte a partir del json y devuelve el resultado en un output HTML, para visualizar los resultados en una página.